Advancing Visual Analytics Using Dimensionality Reduction

By

Takanori Fujiwara

Dissertation

Submitted in partial satisfaction of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Office of Graduate Studies

of the

University of California

Davis

Approved:

Dr. Kwan-Liu Ma, Chair

Dr. Zhaojun Bai

Dr. S. Felix Wu

Committee in Charge

2021

Copyright © 2021 by
Takanori Fujiwara
All rights reserved.

Contents

Abstract

Advancing Visual Analytics Using Dimensionality Reduction

High-dimensional data analysis is a major target application for visualization. When analyzing high-dimensional data, dimensionality reduction (DR) plays a pivotal role as it uncovers the intrinsic features of the data. Current DR methods, however, provide little support in terms of the interpretability in their results, usability with respect to interactive visualizations, and flexibility for handling various data types. This dissertation investigates the problem stated above and presents a set of novel DR methods coupled with interactive visualization, in which four different topics are addressed: (1) interpretation of DR results, (2) analysis of time-dependent multivariate data, (3) design of multivariate streaming data visualization, and (4) introduction of a new approach to comparative network analysis. Illustrative case studies demonstrate the new capabilities greatly enhance the collective power of visual analytics and DR.

Acknowledgments

This dissertation is dedicated to many important individuals who helped my research voyage, where I struggled to find my research philosophy, directions, ideas, and much more. First, I would like to thank my research advisor, Professor Kwan-Liu Ma who always guided me through this journey. Because of his great advice, I was able to explore various possible directions and was motivated to keep going even when I had a hard time. I want to thank all my dissertation proposal and dissertation committee members, Professor Zhaojun Bai, Professor S. Felix Wu, Professor Charan Ranganath, and Professor Jian Zhao. Many of the works in this dissertation have been greatly influenced by them.

Also, I would like to thank all of my collaborators. I was very fortunate to have the opportunity to collaborate with many researchers in various domains. From my collaboration with over 50 researchers, I gained fantastic experiences and was able to build my research fundamentals. I especially want to say special thanks to Professor Preeti Malakar, Professor Khairi Reda, Professor Jian Zhao, and Dr. Francine Chen for being the best mentors for me.

I also want to thank the labmates of VIDI at UC Davis, especially Dr. Jia-Kai Chou, Dr. Tarik Crnovrsanin, Oh-Hyun Kwon, Sandra Bae, Suraj Kesavan, and Tyson Neuroth. The discussions with them were always engaging and serendipitously provided me hints that led to new research ideas. In addition, I would like to thank all my close friends, Hyeyoung Lim, Tzu-Ping Liu, and Nicole Liu. They enriched my life in Davis and always cleared my mind from my worries. Finally, I would like to thank Akiko, my parents, and family, including my small friends Hattchi, Kukku, and Toto.

With everyone's support, I was able to continue pursuing my dream as a researcher.

Chapter 1
Introduction

Visual analytics of high-dimensional data is a major research topic in the visualization community [158, 159]. Various visualization methods (e.g., the parallel coordinates [117], scatterplot matrices [106], and star coordinates [127]) have been introduced to visually present high-dimensional information in the given space (typically 2D on a computer screen) that viewers can perceive and interpret. Besides these visualization methods, dimensionality reduction (DR) is a widely used approach to examining high-dimensional data by computing a 2D projection of the data, which effectively provides a visual overview of the relationships across the high-dimensional data points [181, 198].

The strength of DR methods is their ability to uncover the similarity between data points as spatial proximity. In DR results, by referring to the "similarity $\approx$ proximity" [245] relationship, we can intuitively find meaningful patterns, such as clusters and outliers. Many fields of study, including biology [113], social science [224], and machine learning [187], aptly require analyzing high-dimensional data and thus rely on DR methods.

Despite the benefits of DR, current DR methods suffer from several critical problems when applied to visual analytics. The first problem is the **lack of interpretability**. This problem particularly occurs with nonlinear DR methods, such as t-SNE [233] and UMAP [170]. These methods are becoming more frequently used due to their ability to uncover the intrinsic structure of large, complex high-dimensional data. However, unlike linear DR methods, such as principal component analysis (PCA), many of the nonlinear DR methods used for visualizations do not provide a parametric mapping between the original high-dimensional space and the projected low-dimensional space [231]. Therefore, it is difficult to understand how these DR methods have depicted

meaningful patterns (e.g., clusters) in the lower-dimensional representations.

Another problem is **insufficient usability** with interactive visualizations. Most of the DR methods are developed within the statistics or machine learning communities [52, 234] and, consequently, these methods carry little considerations of utilizing them in interactive visualizations. For example, when interactively adjusting a hyperparameter or updating data points used for a DR method, the generated results can vary significantly from the previous ones. This makes it difficult to keep track of the changes through the interactions.

Lastly, existing DR methods have **limited flexibility** in supporting certain data types and analyses. Currently available DR methods used for 2D or 3D visualizations [52,234] can only be applied to data that can be formed into a feature matrix (i.e., a matrix of instances $\times$ features), such as a single-time-point multivariate data. Consequently, for multivariate time-series data or network data, for example, both of which are increasingly important to model various phenomena, the existing DR methods cannot be directly used for analysis since these data types are conventionally represented as a dynamic feature matrix (i.e., a matrix of timestamps $\times$ instances $\times$ features) and as an adjacency matrix (i.e., a matrix of instances $\times$ instances), respectively.

This dissertation focuses on enhancing and developing DR methods as well as their associated interactive visualizations, each of which addresses one or a combination of the aforementioned three challenges.

1.1 Background

The following subsections describe the terminologies used in this dissertation and the previous works in visual analytics using dimensionality reduction.

1.1.1 Dimensionality Reduction (DR)

Dimensionality reduction (or representation learning) is the transformation that aims to process a high-dimensional dataset $\mathbf{X}$ represented as an $n \times d$ matrix (n: the number of data points, d: the number of dimensions) into a lower-dimensional dataset $\mathbf{Y}$ represented as an $n \times d'$ matrix (d': the number of dimensions where $d' < d$) while

maximally preserving certain information, such as the structural characteristics of the high-dimensional dataset [234]. DR methods can be categorized as either linear or nonlinear DR. A brief introduction to each type of DR is provided below.

1.1.1.1 Linear Dimensionality Reduction

Linear dimensionality reduction can be defined as DR that produces a linear transformation matrix $\mathbf{W}$ (or projection matrix) represented as a $d \times d'$ matrix [52]. By using $\mathbf{W}$, a lower-dimensional dataset $\mathbf{Y}$ can be obtained with $\mathbf{Y} = \mathbf{XW}$. By solving a different optimization problem, each linear DR method obtains a projection matrix $\mathbf{W}$. For example, PCA [114, 125] maximizes data variance captured in a lower-dimensional dataset $\mathbf{Y}$. Linear discriminant analysis [118], with a class label for each data point, maximizes inter-class variance while minimizing intra-class variance.

1.1.1.2 Nonlinear Dimensionality Reduction

Nonlinear dimensionality reduction aims to capture nonlinear structure of a high-dimensional dataset $\mathbf{X}$ onto a lower-dimensional dataset $\mathbf{Y}$. While some of the nonlinear DR methods provide a parametric mapping from $\mathbf{X}$ to $\mathbf{Y}$ (e.g., kernel PCA [200]), other methods such as t-SNE [233], LargeVis [219], and UMAP [170] do not. The latter methods first generate a neighbor graph where each edge weight between nodes (i.e., data points) represents a dissimilarity of the corresponding nodes; then the methods maximally preserve local neighborhoods for each data point. These methods have the advantage of preserving local neighborhood relationships—an important capability when visually looking for clusters and outliers—and are commonly used for visualizations.

1.1.2 Visual Analytics

A definition of visual analysis by Keim et al. [128] states that "visual analytics combines automated analysis techniques with interactive visualizations for an effective understanding, reasoning and decision making on the basis of very large and complex data sets." Visual analytics can be described as an analysis approach that connects advanced analysis methods and visualizations with effective interactions. Therefore, in visual analytics, how we choose or develop the best method for each component

(i.e., the analysis method, visualization, and interaction) and how to combine them effectively is critical. While there are various domains where researchers have approached this challenge [214], this dissertation specifically focuses on visual analytics using dimensionality reduction.

1.1.2.1 Visual Analytics Using Dimensionality Reduction

Liu et al. [159] and Sacha et al. [198] provide a complete overview of visual analytics using DR. To analyze high-dimensional data, most works utilize existing DR methods as they are. However, as stated earlier, there are the three challenges that need to be addressed to effectively use DR for visual analytics.

Interpretation of DR Results. A common approach used to support interpretation of DR results is to visualize statistical charts (e.g., bar charts and boxplots) for each feature of the user-selected groups of data points [59, 142, 167, 183, 209]. However, this approach is not scalable when there is a large number of features (e.g., 10 features). To address this scalability issue, a few identify the representative features of each group of data points [124, 227]. These works referred to each group's principal components computed by PCA. However, PCA only identifies the representative features within each group and does not consider which features make one group unique when compared with respect to the other groups. More detailed discussion can be found in Sec. 2.1.

Improvement of Usability of DR Methods for Visualizations. Most DR methods, including PCA, multidimensional scaling (MDS), and t-SNE, often produce drastically different projections based on a hyperparameter or incorporated data points. To mitigate this problem, several works enhanced existing DR methods. For example, to analyze high-dimensional data changing over time, Rauber et al. [188] developed Dynamic t-SNE. Unlike t-SNE, Dynamic t-SNE offers a controllable trade-off on how much temporal coherence between projections of consecutive time points is strictly kept and how much neighborhood relationships are precisely preserved in the t-SNE results. Several works [84, 119, 226] also improved the stability of PCA and MDS projections when updating the projections with new data points by adjusting the axes of projections before and after the updates.

Advanced data analysis support. Many different visual analytics methods and systems are developed for multivariate time-series data, as described in the survey by Bach et al. [17]. To utilize existing DR methods for analyzing multivariate time-series data, one simple and common strategy is to apply DR to a feature matrix at each time point and then show temporal changes with either animation or juxtaposition. However, when there are many time points, this strategy produces a large amount of DR results, which makes it difficult to find important changes (e.g., the emergence of outliers).

Several works used DR for network comparison. For example, Fujiwara et al. [74] compared a larger number of brain networks by providing an overview with MDS. Similarly, to analyze a dynamic network, van den Elzen et al. [230] and Bach et al. [20] showed the similarities of networks across time by utilizing DR. These methods assume that all networks contain the same set of nodes and the correspondence of nodes across the networks is known. However, this assumption is a critical limitation as networks collected from different resources typically do not have known node-correspondence.

1.2 Content Overview

This dissertation is organized into different chapters, each of which addresses the challenges in visual analytics using DR.

Chapter 2 introduces an analysis method, called ccPCA (contrasting clusters in PCA), which supports interpreting results generated by any DR methods, especially for characterizing clusters that appear in DR results. By utilizing contrastive learning [4, 263], ccPCA first computes each feature's relative contributions to the contrast between one cluster and the other clusters and then identifies the essential features to characterize each cluster. Additionally, this chapter presents an interactive analysis system that utilizes ccPCA with a scalable visualization of each cluster's feature contributions. A version [76] of the research in Chapter 2 was published in IEEE Transactions on Visualization and Computer Graphics.

Chapter 3 describes a DR framework, MulTiDR, that processes and provides a

comprehensive overview of time-dependent multivariate data. Time-dependent multivariate data is represented as a 3D array of instances, time points, and attributes. To produce a DR result from such data, we employ DR in two steps. The first DR step reduces the three axes of the array to two, and the second DR step visualizes the data in a lower-dimensional space. In addition, by coupling ccPCA introduced in Chapter 2 with interactive visualizations, MulTiDR enhances analysts' ability to interpret the DR results. A version [78] of the research in Chapter 3 was published in IEEE Transactions on Visualization and Computer Graphics.

Chapter 4 presents an incremental DR method in order to analyze streaming multidimensional data. Unlike existing incremental methods, the incremental method in Chapter 4 is designed with the consideration of usability with respect to visualizations. This method can provide lower-dimensional representations with coherent data positions between the previous and updated time points. Furthermore, this method has the capability of handling varying data dimensions. With these functionalities, the analyst can visually identify time-varying patterns, such as anomalies and forming clusters. A version [75] of the research in Chapter 4 was published in IEEE Transactions on Visualization and Computer Graphics.

Chapter 5 introduces a new network analysis approach called contrastive network representation learning (cNRL). cNRL embeds network nodes into a lower-dimensional space that reveals unique characteristics of one network by contrasting with another. Within this approach, we also design a method, named i-cNRL, which offers interpretability in the learned results, allowing the capability to understand which specific patterns are found in one network but not the other. A version [81] of the research in Chapter 5 was preprinted in arXiv.

Chapter 6 presents a visual analytics framework, ContraNA, that enhances the usability of i-cNRL with an interactive visualization interface. ContraNA helps analyze the uniqueness of one network relative to another by relating i-cNRL's embedding results to the network structures as well as explaining the learned features from i-cNRL. We evaluate ContraNA's usability through a controlled user study with network

comparison tasks. A version [80] of the research in Chapter 6 was presented at IEEE Conference on Visual Analytics Science and Technology (VAST) 2020.

Lastly, Chapter 7 concludes this dissertation with a summary of the main contributions and discusses the implications to future visualization and dimensionality reduction research.

Chapter 2
Supporting Analysis of Dimensionality Reduction Results

According to the recent surveys [34, 181], analyzing a DR result involves the following tasks: (1) identifying clusters in the DR result, (2) understanding the characteristics of the clusters, and (3) comparing the clusters with predefined classes of data points. In the case that the DR result has interpretable axes, such as the dimensions generated by principal components analysis (PCA) [114, 125], understanding the characteristics of each axis and comparing the axis with the original dimensions (or features) are also included as part of the analysis tasks.

Among the aforementioned tasks, the main task sequence is first identifying clusters and then understanding their characteristics [34]. Many automatic methods (e.g., density-based clustering methods [14, 38, 68, 139]) have been introduced to identify clusters (the first task). Also, interactive visualization has shown promise to aid both generation of DR results and identification of clusters [181, 198]. DR incorporating interactive visualization can involve human-in-the-loop to evaluate the DR results as well as to adjust DR's parameters [67, 79] or input features [120, 241] based on analysts' interests and their domain knowledge. Also, interactive selection of data points over the visualized DR results allows the analysts to manually adjust cluster members assigned by the automatic methods [245]. However, methods to assist the second task have still not been well studied, especially in the case that the data has many features. Reviewing the original feature values is essential to understanding each cluster's characteristics. To support this task, many existing visual analytics systems [59, 142, 167, 183, 209] employ basic statistical plots, such as histograms and parallel coordinates, for inspecting each feature of the selected clusters. However, because these visualizations render all of the

features' values, they are limited in handling a large number of features. In addition, even if we were able to show all the features, it could be very time-consuming to find the common patterns within each cluster or find the differences among the clusters by individually referring to the values for each of the many associated features.

To address these problems, this chapter introduces an analysis method that highlights those essential features for understanding characteristics of each cluster in a DR result. This method adopts contrastive learning [263], a new emerging analysis approach for high-dimensional data. Contrastive learning aims to discover "*patterns that are specific to, or enriched in, one dataset relative to another*" [2]. Among the contrastive learning methods, contrastive principal component analysis (cPCA) [2, 3, 87] is specifically chosen and enhanced for visual analysis. A new usage of cPCA, called ccPCA (contrasting clusters in PCA), can measure each feature's relative contribution to each cluster's *contrast* to the others. By referring to these relative contributions, users can easily focus on the features they should review in detail. I describe the strengths of using ccPCA with both numerical formulas and concrete examples. In addition, because cPCA requires parameter tuning to obtain a useful result, an automatic selection method is developed to find the best parameter value.

Moreover, a heatmap-based visualization is introduced to show all the features' contributions of each cluster. By employing hierarchical clustering and matrix reordering, the visualization helps the user find where clusters have similar features' contributions or how the features have similar contributions within or across clusters. Additionally, with these methods, a scalable visualization can be provided, which can handle the case of analyzing many features (e.g., 100 features or more). I build an interactive visual analytics system using ccPCA and a heatmap-based visualization. The effectiveness of our methods and system are demonstrated with case studies using several publicly available datasets.

2.1 Related Work

The relevant works to this chapter are categorized in (1) visualization for exploring DR results and (2) discriminant analysis and contrastive learning.

2.1.1 Visualization for Exploring DR Results

Various visualizations have been developed to assist analysis tasks for a DR result [63, 134, 143, 145, 158, 159]. Here focuses on describing the works that support the aforementioned main task sequence (i.e., identifying clusters and understanding clusters' characteristics). Stahnke et al. [209] developed visualizations to help understand multidimensional-scaling (MDS) [222] results. To support a feature comparison of clusters in the MDS result, their visualization allows the user to manually select clusters and then it depicts the selected clusters' density plots for each of the features. Similarly, for a cluster comparison in the DR results, other works [59, 142, 167, 183] visualized statistical charts (e.g., bar charts and boxplots) of the features for each manually or automatically selected cluster. However, because the approaches in the above works [59,142,167,183,209] depict the statistical chart for each feature, they are not scalable when there is a large number of features (e.g., 10 features). Broeksema et al. [36] took further steps to provide a summary of the DR results. They developed visualizations to help understand patterns that appeared in multiple correspondence analysis (MCA) [1], which is a similar DR method as PCA for categorical data. They visualized each data point's salient feature value extracted with MCA as a colored Voronoi cell around each projected point in the MCA result. This linking of the DR result and the salient features helps the user interpret the DR result. Similarly, Joia et al. [124] linked the DR result and the information of features into one plot. In addition to an automatic selection of clusters, they obtained representative features for each cluster by using PCA. Afterward, they visualized these features' names as a word cloud within each clustered region instead of showing the projected points. Turkay et al. [227] also used PCA to obtain the representative features in the MDS result.

Among the mentioned studies, the works by Joia et al. [124] and Turkay et al. [227] are most related to the work in this chapter in terms of identifying the representative

features for each cluster. To identify such features, both methods refer to each cluster's principal components (PCs) computed by PCA (and the correlation between the features and PCs). Even though they applied PCA within each cluster, the computed PCs might capture only the global tendency in the dataset. For example, all clusters may have similar or even the same PCs. Also, their methods cannot find features that highly contribute to the differentiation or *contrast* between one cluster and the others. It is important to provide features that make each cluster's characteristics unique.

2.1.2 Discriminant Analysis and Contrastive Learning

Discriminant analysis, including linear discriminant analysis (LDA) [118], quadratic discriminant analysis (QDA) [171], and mixture discriminant analysis (MDA) [109], is a supervised learning method used for classification and DR. Discriminant analysis methods use labeled data points as a learning set and construct a classifier to distinguish each class as much as possible [118]. For example, LDA finds new dimensions (or components) which provide good separations between each class. Note that while both PCA and LDA can be categorized as linear DR methods, PCA is an unsupervised method and finds dimensions that maximize the variance of the input data points.

As similar to PCA, we can obtain the contribution of each original dimension (or feature) to each component constructed by LDA. Therefore, for visual analytics, LDA has been utilized to inform the features which have an important role to distinguish clusters. For example, Wang et al. [241] developed linear discriminative star coordinates (LDSC). LDSC shows each feature's contribution to distinguishing a cluster from each other as a length of a corresponding axis of the star coordinates [127]. To obtain a better-clustered result, the user can use these axes as interfaces to discard the less contributed features or change the weight of the features used for clustering.

While discriminant analysis is used for discriminating the data points based on their classes, contrastive learning [263] focuses on finding patterns that contrast one dataset with another. For example, contrastive PCA (cPCA) [2, 3, 87] is the extended version of PCA for contrastive learning. cPCA takes two different datasets (i.e., target and background), and then identifies the directions (or contrastive principal components)

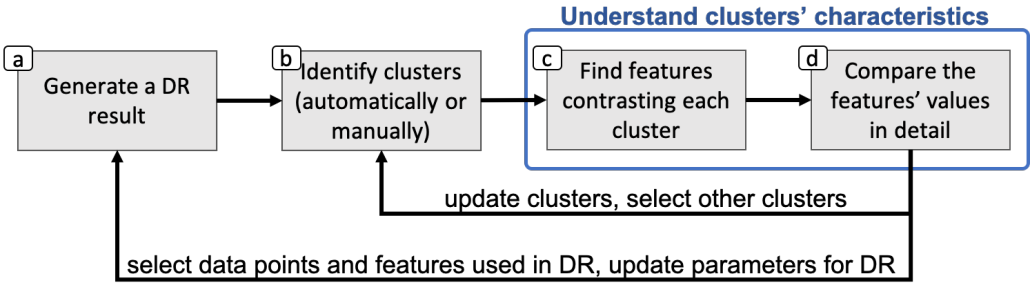


Figure 2.1: The analysis workflow.

that have a higher variance in the target dataset when compared to the background dataset. Projection of the target dataset with these contrastive principal components provides the patterns which are uniquely found only in the target dataset. In addition to cPCA, several extended methods for contrastive learning have been developed (e.g., contrastive versions of latent Dirichlet allocation [263], hidden Markov models [263], regressions [87], multivariate singular spectrum analysis [62], and variational autoencoders [4]).

To the best of my knowledge, this chapter provides the first research using a contrastive learning method, specifically cPCA, for interactive visual analytics. The major advantages of using cPCA instead of PCA or LDA are demonstrated in Sec. 2.3.

2.2 Workflow and an Analysis Example

This section first defines a workflow for analyzing high dimensional data using DR, and then provides an analysis example to motivate my work.

2.2.1 Analysis Workflow

Fig. 2.1 shows an analysis workflow using ccPCA. It starts from (a) applying a DR method (e.g., MDS, PCA, or t-SNE [233]) on high-dimensional data. Then, the task is (b) to identify clusters in the DR result by applying a clustering method (e.g., k-means [107], DBSCAN [68], or spectral clustering [180]) or selecting clusters manually. Afterward, the task is to understand the clusters' characteristics. This task has two steps. The first step is (c) finding features (or dimensions) that have a high contribution to contrasting each cluster with the others. For this step, we utilize cPCA [2, 3, 87], as described in

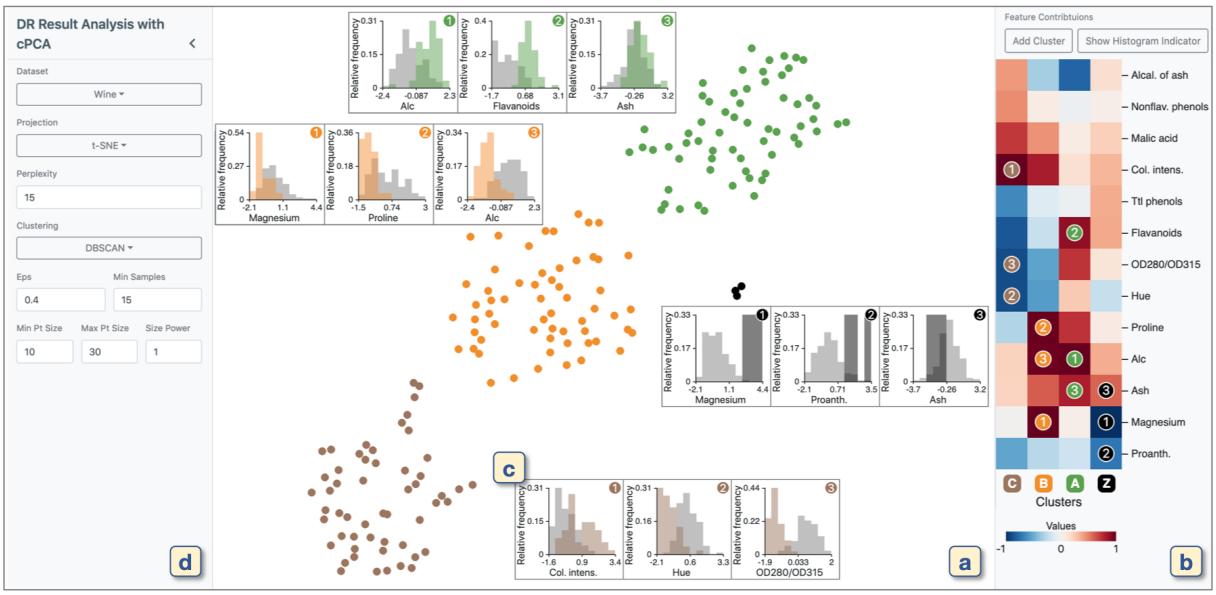


Figure 2.2: A screenshot of the prototype system. The dimensionality reduction (DR) view (a) visualizes a result after DR and clustering. The feature contributions view (b) shows the measures of each feature's contribution to *contrasting* each cluster with the others. The feature values of the selected cells in (b) are visualized as histograms, as shown in (c). In (d), we can change the settings for the analysis methods and visualizations.

Sec. 2.3. The second step is (d) reviewing the detailed differences of values of the highly contributed features between each corresponding cluster and the other data points. Existing methods are used for DR and clustering while I introduce new methods for the last two steps. With the last two steps, we can obtain an understanding of which and how features contribute to the uniqueness of each cluster. After understanding the selected clusters' characteristics, as indicated with the arrows from (d) to (a) and (b), the user can update the DR result or clusters by selecting a subset of the data points based on their interest, changing the parameters of the algorithms, etc.

2.2.2 An Analysis Example

We analyze the Wine Recognition dataset from UCI Machine Learning Repository [65] while following the workflow shown in Fig. 2.1. The dataset includes 178 data points (wines) with 13 features (e.g., alcohol, color intensity, and flavanoids). First, to generate a DR result, t-SNE [233] is used for all of the data points. Then, to detect clusters, DBSCAN [68] is applied to the DR result. As shown in Fig. 2.2-a, three clusters are

identified and colored with green, orange, and brown. The black data points are outliers or noise points labeled by DBSCAN. To understand the characteristics of the wines in each cluster, the system immediately applies our cPCA-based analysis method for each detected cluster. Now, the features' contributions to contrasting each cluster are obtained. The measures of contributions are visualized with a blue-to-red divergent colormap, as indicated in Fig. 2.2-b. As the absolute value of the measure approaches 1, the corresponding feature has a higher contribution. Finally, for each cluster, histograms of values of the three features that have the highest contributions are visualized. The results are shown in Fig. 2.2-c. The histograms for each target cluster are colored with its respective cluster color, while the others are colored gray. The y-axis shows relative frequency and its maximum limit is set to the maximum relative frequency of each pair of the histograms.

Based on the result shown in Fig. 2.2, we can easily perceive each cluster's characteristics. For example, the green cluster has higher alcohol percentage ('Alc') and flavanoids when compared to the others. The orange cluster has lower magnesium, proline, and alcohol percentage. Also, the brown cluster has lower OD280/OD315 (i.e., low dilution degree), lower hue, and higher color intensity. The black outliers have higher magnesium and proanthocyanidins ('Proanth').

Even though this analysis example uses relatively a small number of features and clusters, finding these results is not a trivial task without the suggestions of highly contributed features. For example, in Fig. 2.3, similar to the work by Kwon et al. [142], each cluster's feature values with parallel coordinates [117] are visualized. Without using ccPCA, to find the same results, the user would need to review all the features of each cluster one by one. This is not only time-consuming but also introduces a possibility of overlooking important characteristics.

2.3 Methodology

As demonstrated in Sec. 2.2.2, when a dataset has many features, even only around ten, reviewing the values for each feature becomes tedious. Finding features which contrast

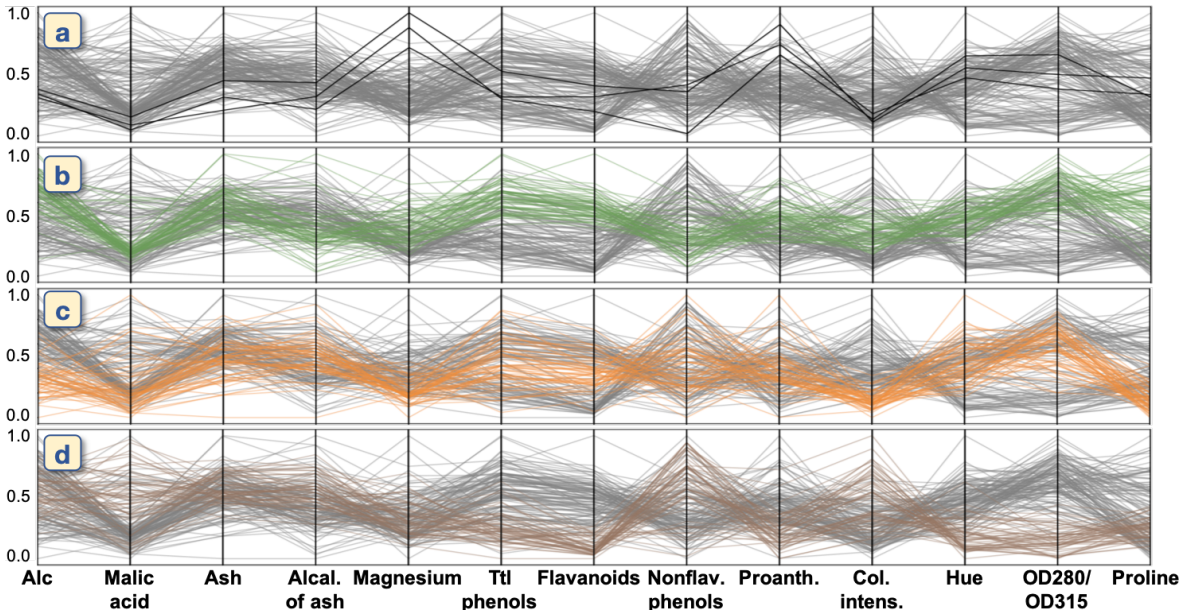


Figure 2.3: Parallel coordinates showing all features in the Wine Recognition dataset. The corresponding polylines for the wines are highlighted with (a) black, (b) green, (c) orange, and (d) brown clusters. It is difficult to discern the essential features from this visualization.

each cluster with the other data points is the core analysis of my approach. To do this, we utilize cPCA [2, 3] and its linearity to obtain the features' contributions (FCs) to the contrast.

There is a clear advantage of using cPCA over PCA [114, 125] and LDA [118], both of which are linear DR methods. PCA has been used to find the representative features within the selected data points [124, 227]. However, as shown in the examples of Fig. 2.4(middle), while PCA is useful to find variations within each cluster, it cannot consider the differences between one cluster and the others. This consideration is important to find the unique characteristics in the target cluster. On the other hand, LDA focuses only on distinguishing the target cluster from the others. Thus, as shown in Fig. 2.4(left), LDA would judge whether a feature has a high contribution to distinguishing the target cluster even in the case where the feature has little variance in the target cluster and zero variance in the others. This could frequently happen especially when the number of features is large. A new cPCA-based method, ccPCA, finds the features which are well-balanced in terms of variety (similar to PCA) and separation (similar to LDA). Also, this balance can be controlled with the contrast parameter, as

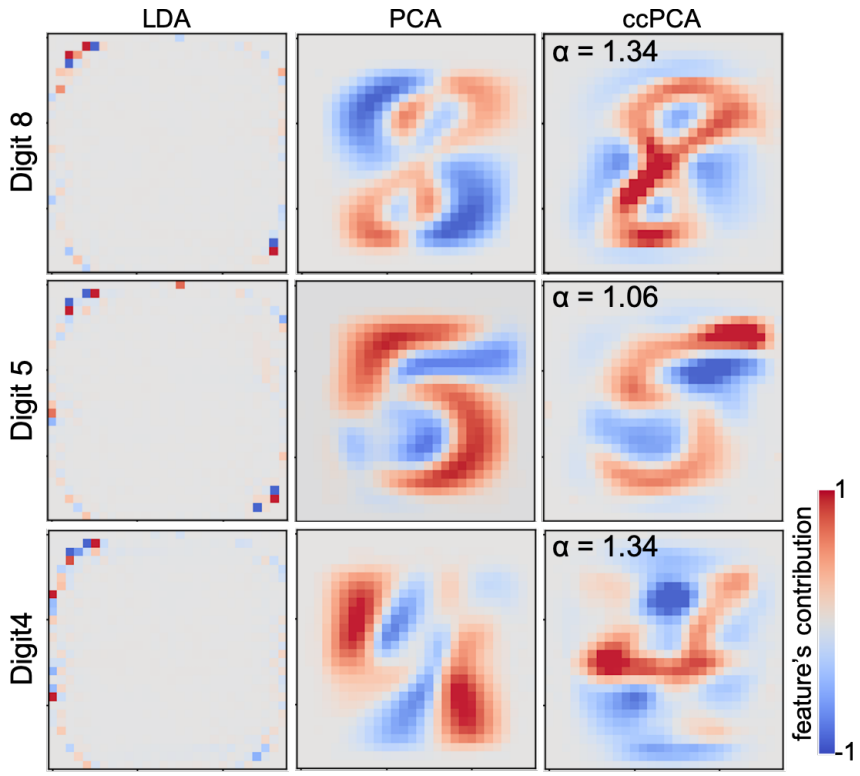


Figure 2.4: Comparison of features' relative contributions of MNIST digits. LDA, PCA, and ccPCA are compared. All of these methods can calculate the features' relative contributions to the first component by respectively referring to either LDA's loadings, PC loadings, or cPC loadings described in Sec. 2.3.3. Each loading is scaled in the range from -1 to 1 by dividing the maximum absolute value of the loadings. The scaled loading for each pixel is visualized with a blue-to-red colormap. For LDA, classification between the target digit and the others is performed. The LDA results, placed on the left column, show that the outside pixels have high contributions. We can consider that LDA tries to distinguish each target digit from the others by referring to the pixels that are less frequently used in the other digits. PCA is applied to each target cluster in the same manner as the works by Joia et al. [124] and Turkay et al. [227]. We can see that the PCA results show variations of the strokes when drawing each digit. The cPCA results are obtained from ccPCA with the automatic selection of α (refer to Sec. 2.3.2). When compared with PCA, the cPCA results clearly show the strokes contrasting the target digit with the others. For example, for Digit 5, the pixels on the upper right have high contributions, as indicated in dark red. When only drawing Digit 5, we tend to use these pixels, and thus, we can see that cPCA captures Digit 5's characteristics. Similarly, for Digit 4, we can see that there are dark red pixels around the middle left.

described in Sec. 2.3.2.3.

2.3.1 Contrastive PCA (cPCA)

This subsection provides a brief introduction to cPCA, which is utilized to find features contrasting a target cluster with the other data points. cPCA is developed for "the

setting where we have multiple datasets and are interested in discovering patterns that are specific to, or enriched in, one dataset relative to another" [2]. For instance, from the examples provided by Abid and Zhang et al. [2], when we have a medical dataset X of diseased patients, we would want to find trends and variations of the disease's influence. If we apply the classical PCA to X, the first principal component would only present the diseased patients' demographic variations [86], instead of showing the variation of the disease's effects. However, if there is another medical dataset Y of healthy patients, cPCA can utilize the fact that Y could have similar demographic variations as X, and no variations related to the disease. By taking X and Y as the target and background datasets, respectively, cPCA can find the directions (or components) in which X has high variance but Y has low variance.

2.3.1.1 Description of the Algorithm

Here describes how cPCA obtains such directions by using the target and background datasets. Let $X = \{\mathbf{x}_i\}_{i=1}^{n}$ be the target dataset and $Y = \{\mathbf{y}_i\}_{i=1}^{m}$ be the background dataset where $\mathbf{x}_i, \mathbf{y}_i \in \mathbb{R}^d$, n and m are the numbers of data points, and d is the number of dimensions (or features). Similar to the classical PCA, for the first step, cPCA applies centering to each dimension of X and Y and then obtains their corresponding empirical covariance matrices $\mathbf{C_X}$ and $\mathbf{C_Y}$. Let $\mathbf{v}$ be any unit vector of d dimensions.

Then, with a given direction $\mathbf{v}$, the variances for the target and background datasets can be written as: $\lambda_X(\mathbf{v}) \stackrel{\text{def}}{=} \mathbf{v}^\mathsf{T}\mathbf{C_X}\mathbf{v}$, $\lambda_Y(\mathbf{v}) \stackrel{\text{def}}{=} \mathbf{v}^\mathsf{T}\mathbf{C_Y}\mathbf{v}$. Now, the optimization that finds a direction $\mathbf{v}^*$ where X has high variance but Y has low variance can be written as:

$$\mathbf{v}^* = \underset{\mathbf{v}}{\operatorname{argmax}}\ \lambda_X(\mathbf{v}) - \alpha\lambda_Y(\mathbf{v}) = \underset{\mathbf{v}}{\operatorname{argmax}}\ \mathbf{v}^\mathsf{T}(\mathbf{C_X} - \alpha\mathbf{C_Y})\mathbf{v} \tag{2.1}$$

where α is a contrast parameter ($0 \leq \alpha \leq \infty$). The details of α are described in Sec. 2.3.1.2. From Eq. 2.1, we can see that $\mathbf{v}^*$ corresponds to the first eigenvector of the matrix $\mathbf{C} \stackrel{\text{def}}{=} (\mathbf{C_X} - \alpha\mathbf{C_Y})$. The eigenvectors of $\mathbf{C}$ can be calculated with eigenvalue decomposition (EVD). These computed eigenvectors are called contrastive principal components (cPCs) and are orthogonal to each other. Similar to the classical PCA, by using these cPCs (typically two cPCs), we can plot the DR result of X. An example from the work by Abid and Zhang et al. [2] is shown in Fig. 2.5.

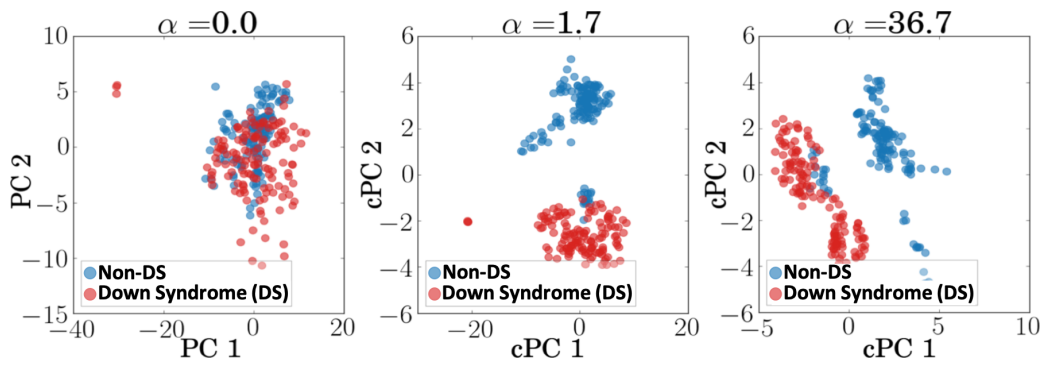


Figure 2.5: cPCA results of the Mice Protein Expression dataset [112] provided by Abid and Zhang et al. [2]. A different contrast parameter α value is used for each result. When $\alpha = 0$, cPCA generates the same result when applying PCA to the target dataset. In this result, we cannot see clear differences between down syndrome (DS) and non-DS mice, indicated with red and blue points, respectively. While clear differences between DS and non-DS start to appear when $\alpha = 1.7$, we can see that DS is further separated into two groups when $\alpha = 36.7$. More examples can be found in the work by Abid and Zhang et al. [2, 3].

2.3.1.2 The Contrast Parameter and Semi-Automatic Selection

The contrast parameter α controls the trade-off between having high target variance and low background variance. When $\alpha = 0$, cPCs will only maximize the variance of the target dataset. These cPCs are the same as the principal components (PCs) of the target dataset when computed with the classical PCA. As α increases, cPCs will become more optimal directions that reduce the variance of the background dataset. Fig. 2.5 shows the example from the work by Abid and Zhang et al. [2] with different α values.

As shown in Fig. 2.5, the selection of α has a strong impact on the DR result. Thus, Abid and Zhang et al. [2,3] introduced an algorithm suggesting multiple α values. Their algorithm calculates a set of cPCs for each of the multiple values of α (with 40 values as their default), and the α values are logarithmically spaced in a certain range (the default is between 0.1 and 1000). Then, the similarity between each pair of the different cPCs, each obtained with a different α value, is measured by calculating the product of the cosine of the principal angles. Afterward, based on the user's input p (the number of values of α to suggest), the algorithm finds p clusters from the similarities with spectral clustering [180]. Finally, the algorithm returns p values of α which correspond to the medoids of the clusters. From the suggested p values, the algorithm returns a set of DR results. By referring to this set, the user can choose their preferred α value.

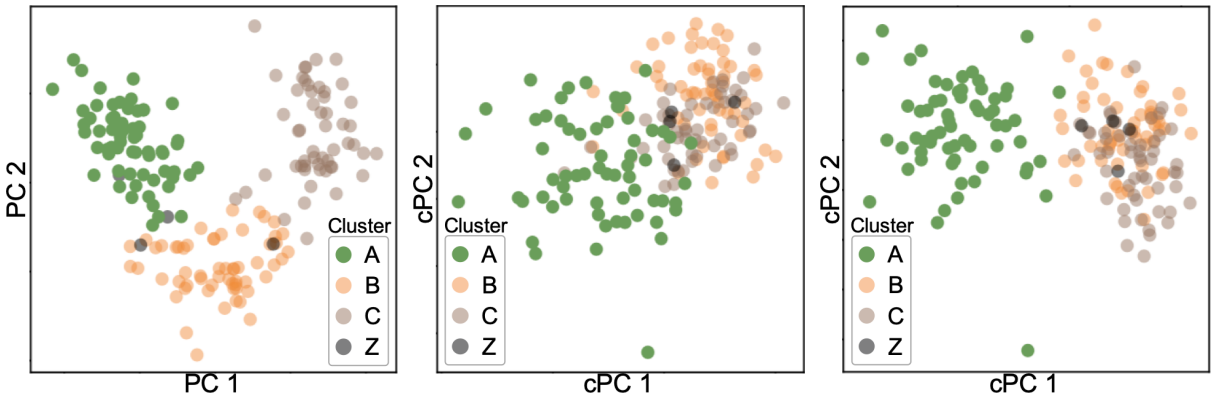
2.3.2 Finding the Direction that Contrasts a Target Cluster

As described above, cPCA discovers patterns that are specific to, or enriched in, the target dataset relative to the background dataset. cPCA is originally designed for the situation where the patterns the user wants to identify are included within the target dataset X, while the background dataset Y contains the structure the user wants to remove from the target dataset. Therefore, the provided examples for $\{X, Y\}$ by the authors of cPCA [2, 3] are {'diseased subjects', 'control group subjects'}, {'patients after treatment', 'patients before treatment'}, {'images mixed with interests and noises', 'images only including noises'}, etc.

In our case, we want to find the directions (i.e., cPCs) which contrast one cluster with the other data points. If we follow the examples of X and Y as stated above, X can be the target cluster and Y can be the other data points. However, in this case, cPCA will find cPCs that only enrich the variations specific to the target cluster. For example, when the target cluster includes diseased subjects and the other data points correspond to healthy subjects, cPCA will find enriched variations within the diseased subjects (e.g., differences among multiple diseases), but will not consider the differences between diseased and healthy subjects.

To utilize cPCA for finding the directions contrasting a target cluster with the others, I introduce a novel usage of cPCA, named ccPCA. Instead of using the target cluster as the target dataset X and the other data points as the background dataset Y, ccPCA uses the entire dataset as X and the data points other than the target cluster as Y. With this approach, we can find the directions that contrast the target cluster. As described in the following subsections, ccPCA has the strengths in regards to two aspects: (1) an implicit extension of the contrast parameter α and (2) a proper setting of the centroid. The DR results shown in Fig. 2.6 provide a comparison of the classical PCA, the original usage of cPCA (i.e., using only the target dataset as X), and ccPCA.

Let $E = \{\mathbf{e}_i\}_{i=1}^{s}$ be the entire dataset and $K = \{\mathbf{k}_i\}_{i=1}^{t}$ be the target cluster ($K \subset E$, $\mathbf{e}_i, \mathbf{k}_i \in \mathbb{R}^d$, s and t are the numbers of data points). Then, we denote $R = \{\mathbf{r}_i\}_{i=1}^{u}$ as the difference of the two sets K and E (i.e., $R = E \setminus K$ and $u = s - t$). With these notations,



(a) PCA (b) cPCA ($\alpha = 2.15$) (c) ccPCA ($\alpha = 4.38$)

Figure 2.6: The DR results of the Wine Recognition dataset. The cluster labels generated in Sec. 2.2.2 are used. Here, PCA, cPCA, and ccPCA are used to find the (c)PC contrasting the green cluster. In (a), the classical PCA is applied to the entire dataset. Though there is a separation of the green cluster when using the first and second PCs, there are overlaps of the green and orange clusters when only using the first PC (PC 1). In (b), the data points in the green cluster are used as the target dataset and the other data points are used as the background dataset. α value is selected from the suggestions using the semi-automatic selection in Sec. 2.3.1.2. We cannot see a clear separation of the green cluster from the others. In (c), ccPCA uses the entire data points instead of only the green cluster as the target dataset. α value is selected with the automatic selection method in Sec. 2.3.2.3. We can see a better separation when compared to that of (a) and (b) even when using only the first cPC (cPC 1).

we can say that ccPCA uses E and R as the target X and background Y datasets, respectively.

2.3.2.1 An Implicit Extension of the Contrast Parameter

To provide a simple and clear explanation, here assumes the centering effects to the datasets E, K, and R are all the same (i.e., E, K, and R have the same mean value for each feature). After centering the target dataset E and the background dataset R, cPCA obtains their corresponding empirical covariance matrices $\mathbf{C_E}$ and $\mathbf{C_R}$. Then, cPCA calculates cPCs by performing EVD to $\mathbf{C_E} - \alpha\mathbf{C_R}$. Let $\mathbf{C_K}$ be the empirical covariance matrix of the target cluster K after centering. Because $\mathbf{C_K} = \sum_{i=1}^{t} \mathbf{k}_i\mathbf{k}_i^\mathsf{T}/t$, $\mathbf{C_R} = \sum_{i=1}^{u} \mathbf{r}_i\mathbf{r}_i^\mathsf{T}/u$, $E = K \sqcup R$, and $s = u + t$, $\mathbf{C_E}$ can be represented as $\mathbf{C_E} = (t\mathbf{C_K} + u\mathbf{C_R})/s$. With this,

$\mathbf{C_E} - \alpha\mathbf{C_R}$ can be rewritten as:

$$\mathbf{C_E} - \alpha\mathbf{C_R} = (t\mathbf{C_K} + u\mathbf{C_R})/s - \alpha\mathbf{C_R} \tag{2.2}$$

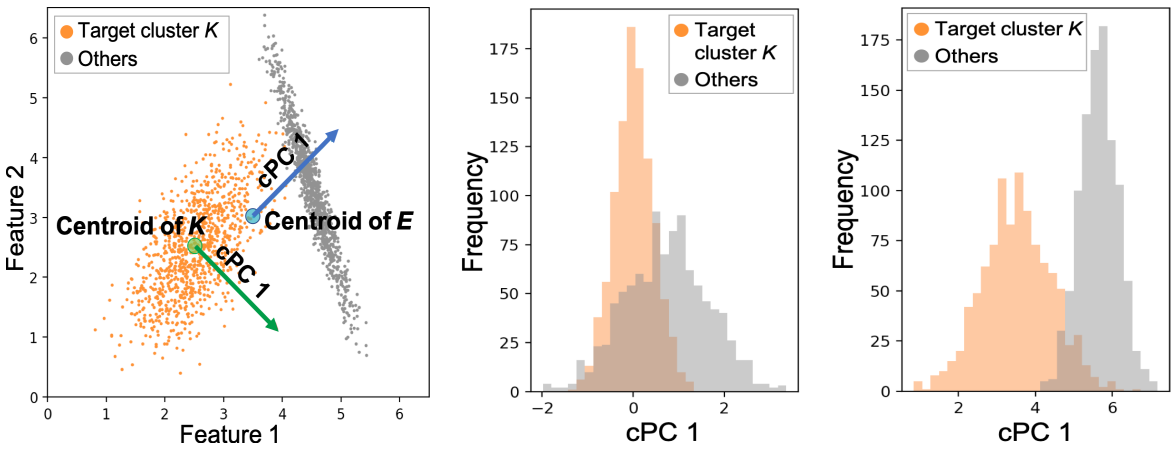
$$= \frac{t}{s}\left(\mathbf{C_K} - \frac{(s\alpha - u)}{t}\mathbf{C_R}\right) = \frac{t}{s}\left(\mathbf{C_K} - \beta\mathbf{C_R}\right) \tag{2.3}$$

where $\beta = (s\alpha - u)/t$. Because $0 \leq \alpha \leq \infty$, $-u/t \leq \beta \leq \infty$. Note that if we use K and R as the target and background datasets, respectively, cPCA performs EVD to $\mathbf{C_K} - \alpha\mathbf{C_R}$. Therefore, a fundamental difference between the cases of using E (i.e., the entire dataset) and using K (i.e., only the target cluster) as the target dataset for cPCA is the difference between α and β.

While α only takes a non-negative value, β can be a negative value. When $\beta = -u/t$, cPCA selects the directions that maximize the variance of the entire dataset E, and hence reduces to PCA applied on E. As β increases to 0, cPCA provides more weight to the target cluster K than the others R to select the directions. When $\beta = 0$, cPCA selects the directions that maximize the variance of the target cluster K, and hence reduces to PCA applied on K. Then, as β increases from 0 to ∞, the directions from cPCA will become more optimal to reduce the variance of the others R. While Eq. 2.3 with $\beta \geq 0$ has a capability to find the same directions with $\mathbf{C_K} - \alpha\mathbf{C_R}$, ccPCA also searches the directions that consider the differences between the target cluster K and the others R by using the range $\beta < 0$.

2.3.2.2 The Centering of the Target Dataset

ccPCA not only implicitly extends the searching range of α of $\mathbf{C_K} - \alpha\mathbf{C_R}$, but it also uses a proper centroid of the dataset. The centering (i.e., the mean subtraction for each feature) in cPCA is used for translating the dataset to its centroid. When using K as the target dataset, the centroid is calculated from only the target cluster K. In contrast, ccPCA uses E as the target dataset, and the centroid is calculated from all the data points. Fig. 2.7 shows an example of the two methods of calculating the centroid and the first cPC in each case. For the same reason as the classical PCA, the centering should be applied to the entire dataset in our case. This is to ensure that the first cPC is the direction of the maximum variance, which contrasts the differences between the target



(a) Centroids and cPCs (b) Histogram along the green cPC 1 (c) Histogram along the blue cPC 1

Figure 2.7: A comparison of centering effects to the cPCA results. For this example, two sets of data points from different 2D Gaussian distributions are generated. In (a), the green circle and arrow show the centroid and the first cPC when using the target cluster K as the target dataset and the others as the background dataset. The blue circle and arrow are the centroid and the first cPC when using the entire dataset E as the target dataset. From the DR results, as shown in (b) and (c), ccPCA, using the entire dataset E as the target dataset, generates a better separation between the target cluster and the others.

cluster and the others.

2.3.2.3 Automatic Selection of the Best Contrast Parameter

The selection of the contrast parameter α is the remaining procedure. Even though we can use the existing semi-automatic selection of α in Sec. 2.3.1.2, selecting the best alpha from the multiple suggested options is tedious when analyzing multiple clusters. Thus, I introduce a method for an automatic selection of the best α for our usage. The pseudocode of this method is shown in Algo. 2.1. To understand the characteristics of the cluster, we should find the first cPC which not only (1) shows a clear separation between the target cluster from the others, but also (2) maintains the variability in the target cluster well (i.e., a high variance within the target cluster). Similar to the classical PCA, the second condition tries to preserve the target clusters' original structure. Without the second condition, when using a large α, cPCA may preferentially select features where the target cluster only has subtle variability, but the other data points have no variability (i.e., zero variance). This example can be seen in

Algorithm 2.1 Our usage of cPCA with automatic selection of α

Inputs: datasets of the target cluster and the others $K = \{\mathbf{k}_i\}_{i=1}^{t}$, $R = \{\mathbf{r}_i\}_{i=1}^{u}$; list of possible $\{\alpha_i\}_{i=1}^{q}$; a threshold ratio of the variance γ.

Obtain the concatenated dataset $E = K \sqcup R = \{\mathbf{e}_i\}_{i=1}^{s}$
Apply centering to E and R
Calculate the empirical covariance matrices $\mathbf{C_E}$ and $\mathbf{C_R}$ from E and R
Perform EVD on $\mathbf{C} = \mathbf{C_E} - \alpha_1 \mathbf{C_R}$
Obtain 1D DR results K' and R' by projecting K and R with the top eigenvector of $\mathbf{C}$ (the first cPC).
Calculate the distance $D(\alpha_1)$ and variance $V(\alpha_1)$ with K' and R'
best_$\alpha = \alpha_1$, best_$D = D(\alpha_1)$, $V_{\alpha_1} = V(\alpha_1)$, best_$K' = K'$, best_$R' = R'$
for all $i = 2, 3, \cdots, q$ **do in parallel**
    Perform EVD on $\mathbf{C} = \mathbf{C_E} - \alpha_i \mathbf{C_R}$
    Obtain 1D DR results K' and R' by projecting K and R with the first cPC
    Calculate the distance $D(\alpha_i)$ and variance $V(\alpha_i)$ with K' and R'
    if $D(\alpha_i) >$ best_D and $V(\alpha_i) \geq \gamma V_{\alpha_1}$ **then**
        best_$\alpha = \alpha_i$, best_$D = D(\alpha_i)$, best_$K' = K'$, best_$R' = R'$
return best_α, best_K', best_R'

the far right of Fig. 2.8.

Similar to the semi-automatic selection in Sec. 2.3.1.2, ccPCA's automatic selection lists multiple candidates of α (ccPCA's default is also 40 values). These candidates consist of 0 and a set of logarithmically spaced values given a certain range (ccPCA's default also ranges from 0.1 to 1000). Here we denote these alphas as $\{\alpha_i\}_{i=1}^{q}$ (q is the number of candidate values for the best α) and assume $\{\alpha_i\}$ is sorted by ascending order (i.e., $\alpha_1 = 0$). Then the method selects a value that obtains the best separation while having enough variance in the target cluster K.

To measure the separation between the target cluster and the others along the first cPC, the histogram intersection (HI) [216] is used, which can measure the overlaps of the histograms of the two sets. While there are many different (dis)similarity measures between two probability distributions, such as the Kullback-Leibler divergence [141], HI is chosen for its robustness to outliers and low computational cost. Let $H_A = \{h_{A_j}\}_{j=1}^{b}$, $H_B = \{h_{B_j}\}_{j=1}^{b}$ be the histograms of two given sets of real numbers A and B where b is

the number of bins, h_{A_j} and h_{B_j} are the numbers of data points in the j-th bin of A and B, respectively. Both H_A and H_B have the same bins. The bin-width is decided by using Scott's normal reference rule [202] from the set of real numbers obtained by combining A and B. The HI of the two sets A and B is defined as: $I(A,B) = \sum_{j=1}^{b} \min(h_{A_j}, h_{B_j})$. Let K'_i and R'_i be the data points of 1D DR results of K and R with the first cPC corresponding to the i-th candidate α value (i.e., α_i), respectively. Then, we can calculate the measurement of separation with the inverse HI (i.e., $I(K'_i, R'_i)^{-1}$) for each α_i. We refer $I(K'_i, R'_i)^{-1}$ as the discrepancy score $D(\alpha_i)$.

For the variance of K'_i, to handle the scaling differences in each DR result, first, the method applies the min-max scaling to K'_i with the minimum and maximum values of $K'_i \sqcup R'_i$. Then, the method calculates the variance of the scaled K'_i. I denote this variance of K'_i as $V(\alpha_i)$.

With the measures of $D(\alpha_i)$ and $V(\alpha_i)$, the automatic selection method selects the best alpha with:

$$\underset{\alpha_i \in \{\alpha_1, \ldots, \alpha_q\}}{\operatorname{argmax}} \; D(\alpha_i) \;\; s.t. \;\; V(\alpha_i) \geq \gamma V(\alpha_1) \tag{2.4}$$

where γ ($\gamma \geq 0$) is a ratio that controls the threshold of the variance $V(\alpha_i)$. Note that $V(\alpha_1)$ is the variance of K'_1 of the cPCA result with $\alpha = 0$, which will be the same result when applying the classical PCA to the entire dataset E. While the method allows the user to select any non-negative value for γ, $\gamma = 0.5$ is used as the default to ensure that $V(\alpha_i)$ has at least a half of $V(\alpha_1)$. Fig. 2.8 shows the cPCA results with different α values. The automatic α selection chooses $\alpha = 1.06$ in this case. More comprehensive experimental results with various datasets and α values can be found in the online supplementary materials listed in Appendix A.

In summary, the original cPCA is enhanced as ccPCA by using Eq. 2.1 with $X = E$ and $Y = E \setminus K$ and by selecting α as the solution to Eq. 2.4.

Parallel calculation of the best contrast parameter: The original semi-automatic selection of the contrast parameter [2, 3] calculates cPCA for each $\alpha_i \in \{\alpha_i\}_{i=1}^{q}$ in serial [1] ($q = 40$ by default). Because the calculation of cPCA for each α_i is independent of each

[1] The original cPCA implementation: `https://github.com/abidlabs/contrastive`

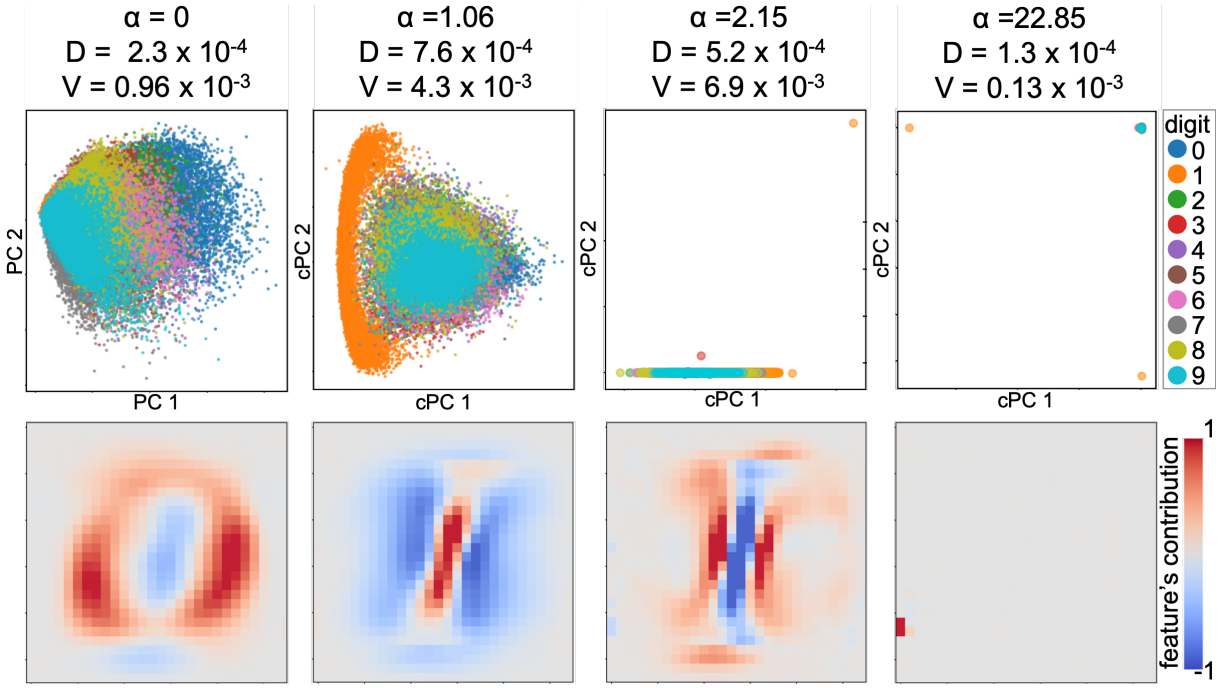


Figure 2.8: The DR results with the first and second cPCs (top row) and the features' (i.e., pixels') relative contributions (bottom row) of the MNIST dataset [148] with different α values (refer to Sec. 2.3.3 about the features' relative contributions). Here, we try to contrast Digit 1 with the other digits. We can see that when $\alpha = 0$ (reduced to applying the classical PCA for all digits), cPCA does not separate Digit 1, and the features' contributions do not show any useful information to understand the characteristics of Digit 1. On the other hand, when $\alpha = 22.85$, while some of Digit 1 (e.g., points placed on the top left) are well separated, the variance V is small. Also, from the features' contributions, we can see that only a few pixels in the lower left have high contributions. This is expected because these pixels are rarely used when drawing Digit 1. The result with $\alpha = 1.06$ produces the best discrepancy score D and a large variance V. This α will be selected by Eq. 2.4. Also, we can see that cPCA highlights the pixels around the center, which are typically used for drawing Digit 1.

other, in order to achieve faster computation, ccPCA uses multi-threads and calculates each cPCA result, $D(\alpha_i)$, and $V(\alpha_i)$ in parallel. The comparison of the completion time of the original cPCA and the implementation with and without parallelization is available on the online supplementary materials (Appendix A).

2.3.3 Features' Relative Contributions to the First cPC

By using cPCA, with the automatically selected α, we can now obtain the direction (i.e., the first cPC) that contrasts the target cluster. Next, we determine how strongly each feature of the target cluster contributes to this direction. Similar to the classical PCA, by using the top eigenvalue λ^* and the corresponding eigenvector $\mathbf{v}^*$ (i.e., the

first cPC) of the matrix $\mathbf{C_E} - \alpha\mathbf{C_R}$, the relative contributions can be calculated with: $\mathbf{w}^* = \sqrt{\lambda^*}\mathbf{v}^*$ where $\mathbf{w}^* = \{w_i^*\}_{i=1}^d$ $(-1 \leq w_i^* \leq 1)$. Analogous to the classical PCA, I call $\mathbf{w}^*$ the cPC loadings of the first cPC. As $|w_i|$ approaches 1, the i-th feature has a stronger contribution (or correlation) to the first cPC. Based on this value, we can decide which features we should review to understand the target cluster. Fig. 2.4 shows an example of the features' contributions and comparisons with the results from LDA and PCA. Comprehensive comparisons of LDA and PCA, using multiple datasets, can be found in the online supplementary materials (Appendix A). As shown in Fig. 2.4, signed cPC loadings can clearly differentiate features whose positive centered values contribute to the negative or positive direction of the first cPC by using blue and red, respectively. This is as opposed to taking the absolute value of the signed cPC loadings.

2.4 Visual Analytics System

To demonstrate the methods of analyzing real-world datasets, I develop a prototype system that supports the analysis workflow shown in Fig. 2.1. A major portion of the system's functionality and a video of an interaction demonstration are available in the online site (Appendix A).

2.4.1 Dimensionality Reduction View

The dimensionality reduction (DR) view, as shown in Fig. 2.2-a, is used for the first two processes: generating a DR result and identifying clusters. In this view, first, the user can visualize a 2D DR result of a high-dimensional dataset. t-SNE [233] (specifically, Barnes-Hut t-SNE implementation [232]) is employed as a DR method because t-SNE can effectively depict the local structure of the dataset, and thus, it is useful to visually identify the clusters within the dataset. From the settings in Fig. 2.2-d, the user can adjust the perplexity parameter of t-SNE, which controls a balance of the effects from local and global structures of the dataset. While a larger perplexity will preserve more of the distance relationship in the global structure, a smaller perplexity will focus on more preserving the distance relationship among a small number of neighbors.

After obtaining the DR result with t-SNE, the user can identify clusters automatically

or manually. As a default, the automatic clustering method will be immediately applied to the obtained DR result. As part of the automatic method, the system supports DBSCAN [68] because the density-based clustering algorithm is able to identify clusters with arbitrary shapes [184], which are often generated from DR. The user can change the parameters required for DBSCAN from the settings in Fig. 2.2-d. The categorical color of each point in the DR result is assigned to the clustering label obtained from DBSCAN. The color black, in particular, is used to represent outliers or noise points labeled by DBSCAN. For a manual selection of a cluster, the system supports a rectangle selection. The user can select data points by drawing a rectangle with mouse dragging in the DR result. Also, the user can add additional data points or unselect data points by using different selection modes provided in the system. From these interactions, the user can create a new cluster consisting of the selected points by clicking the "Add Cluster" button placed at the top of Fig. 2.2-b. The system also supports basic view-level interactions, such as zooming and panning.

2.4.2 Features' Contributions View

The two remaining processes (i.e., finding features contrasting each cluster and comparing the features' values in detail) are performed with the features' contributions (FCs) view shown in Fig. 2.2-b. In the FCs view, the FCs contrasting each cluster described in Sec. 2.3.3 are visualized as a heatmap. While each row name shows the corresponding feature, each column name shows the cluster label ('Z' is used to represent the outliers, noise points, or both). Also, to indicate the corresponding cluster in the DR view, the background of each column name is colored with the corresponding color. Each cluster's FCs are scaled in the range from -1 to 1 by dividing each FC by the maximum absolute value of the FCs (e.g., the original range from -0.1 to 0.5 will be changed to the range from -0.2 to 1.0). Then, the scaled FCs is encoded with a blue-to-red colormap. The next subsections describe the algorithm organizing the heatmap.

2.4.2.1 Optimal Sign Flipping of cPCs and FCs

Similar to the classical PCA, cPCA has the "sign ambiguity" problem [35, 75, 121]. Because of this problem, arbitrary sign flipping in each (c)PC occurs when performing

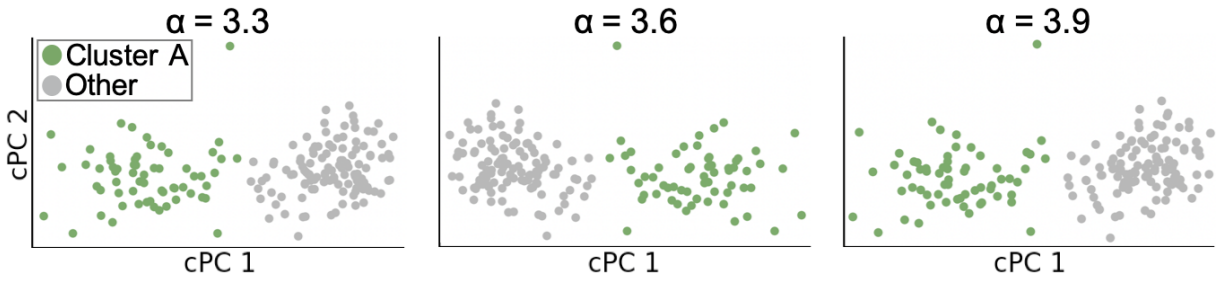


Figure 2.9: Sign flipping of cPCs. The cPCA results of the Wine Recognition dataset used in Sec. 2.2.2 are generated with different α values. Sign flipping occurs between $\alpha = 3.3$ and $\alpha = 3.6$; $\alpha = 3.6$ and $\alpha = 3.9$.

EVD. An example of sign flipping in cPCA is shown in Fig. 2.9. Sign ambiguity affects the comparison of the FCs among the clusters. Each cluster might have the opposite direction of the first cPC only due to this sign ambiguity problem. In this case, the FCs also have opposite signs, and thus, it is difficult to judge whether these clusters have similar patterns in the FCs or not.

To solve this problem as much as possible, I introduce a method to optimally reduce unnecessary sign flipping. Let $\mathbf{v}_i^*$ and $\mathbf{v}_j^*$ be the first cPCs of i-th and j-th clusters, respectively. We can measure how the directions $\mathbf{v}_i^*$ and $\mathbf{v}_j^*$ are similar with the cosine similarity $\mathrm{sim}(i,j) = \mathbf{v}_i^* \cdot \mathbf{v}_j^* / (\|\mathbf{v}_i^*\| \|\mathbf{v}_j^*\|)$. $\mathbf{v}_i^*$ and $\mathbf{v}_j^*$ have the same direction when $\mathrm{sim}(i,j) = 1$, while $\mathbf{v}_i^*$ and $\mathbf{v}_j^*$ have opposite directions when $\mathrm{sim}(i,j) = -1$. Ideally, by flipping the signs of the first cPCs of some clusters, we want to ensure that all of the clusters' first cPCs face the same side (i.e., $\mathrm{sim}(i,j) \geq 0 \quad \forall i,j$). However, the sign flipping to a certain cluster affects all cosine similarities related to this particular cluster. Thus, in many cases, it is theoretically impossible to obtain the result stated above. However, alternatively, we can maximize the sum of all $\mathrm{sim}(i,j)$ with sign flipping. This optimization can be written as:

$$\underset{\varphi=\{\varphi_i,\ldots,\varphi_l\}}{\mathrm{argmax}} \sum_{i=1}^{l} \sum_{j=1,j\neq i}^{l} \frac{(\varphi_i \mathbf{v}_i^*) \cdot (\varphi_j \mathbf{v}_j^*)}{\|\mathbf{v}_i^*\| \|\mathbf{v}_j^*\|} = \sum_{i=1}^{l} \sum_{j=1,j\neq i}^{l} \varphi_i \varphi_j \mathrm{sim}(i,j)$$
$$s.t.\ \varphi_i, \varphi_j \in \{-1, 1\} \tag{2.5}$$

where l is the number of clusters and φ is a set of signs.

To solve Eq. 2.5, I use a heuristic approach. φ is initialized with $\varphi = \{1, 1, \ldots, 1\}$.

We can expect that there is a higher chance to obtain a better result if we start to flip the sign where i-th cluster has the largest negative value in the sum of the similarities ($\sum_{j=1}^{l} \varphi_i \varphi_j \text{sim}(i,j)$). Therefore, the approach here first checks whether sign flipping to the first cPC of such a cluster provides a better result in the objective function of Eq. 2.5. If so, we flip its first cPC's sign. Then, we repeatedly apply this procedure until $\sum_{j=1}^{l} \varphi_i \varphi_j \text{sim}(i,j) \geq 0$ for all $i \in \{1,2,\ldots,l\}$ is satisfied or all clusters have been checked. Afterward, based on the optimized set φ, we allocate the new signs to respective cPC and FCs for each cluster.

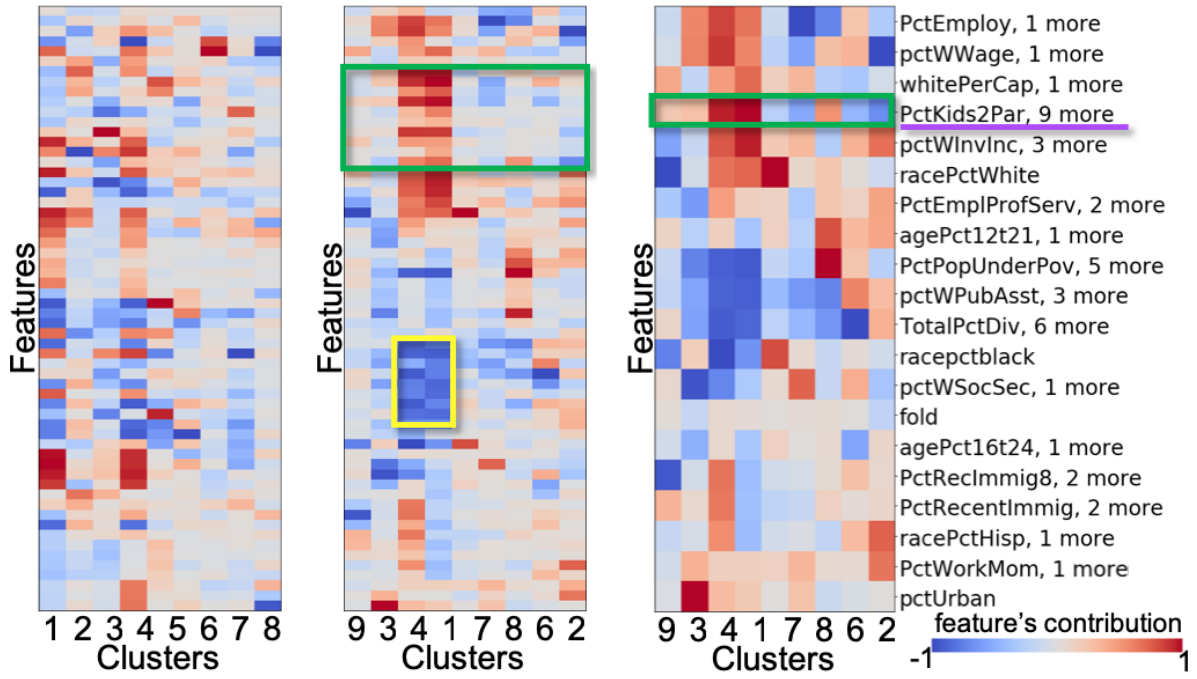
2.4.2.2 Ordering of Features and Clusters

The FCs view can be used for finding not only the heatmap cells which have high FCs, but also the clusters which have similar FC patterns; the features which have similar FCs within and/or among clusters. The case when the clusters have similar FCs implies that these clusters are contrasted due to the same features, but they have different distributions in their features' values. When the features have similar FCs, by reviewing the distributions of one of these features' values, we can expect that the other features may also have similar distributions.

To help find these patterns, the system applies reordering of the features (i.e, rows) and clusters (i.e., columns) based on the FCs. Ordering choice is important since this affects how easily we can find patterns in a heatmap [27]. I use a hierarchical clustering, specifically the complete-linkage method [177], with the optimal-leaf-ordering [24]. Recent survey [27] reported that this combination tends to produce a coherent and quality result to help reveal patterns. Fig. 2.10-a, b show the results before and after the reordering. From Fig. 2.10-b, we can easily see a group of similar FCs.

2.4.2.3 Scalable Visualization

When the number of features is large (e.g., 100 or more), the heatmap-based visualization would have a scalability issue. Moreover, in this case, many features could have high FCs, and as a result, it would still be difficult to decide which features we should review in detail. To solve this issue, I introduce an aggregation method, utilizing the hierarchical clustering result obtained through the reordering method.



(a) Original (b) Reordered (c) Aggregated

Figure 2.10: Reordering and aggregation of the FCs. (a) shows the original FCs. There are 8 clusters and 60 features. (b) shows the reordered FCs in both rows (i.e., features) and columns (i.e., clusters). With (b), we can see a group of similar FCs (e.g., the features are indicated with a yellow rectangle). In (c), the 60 feature clusters are aggregated into 20 rows. For example, the ten features indicated with the green rectangle in (b) is aggregated into one row indicated with the green rectangle in (c).

When the number of features is larger than threshold δ ($\delta = 40$ is used by default), the method obtains δ clusters from the features by referring to the hierarchical clustering result. Then, the method aggregates the FCs into one representative value: the mean or the maximum absolute value. As a default, the method takes the maximum absolute value to show the most prominent feature. Fig. 2.10-c shows an example of the aggregation. Additionally, to provide a representative name for each aggregated feature, the method chooses the name based on which FC has the maximum absolute value. With this name, the method also shows how many features are aggregated in each row, as shown with a purple underline on the right side of Fig. 2.10-c ('PctKids2Par, *9 more*').

2.4.3 Interactions between Views

From DR View: When the user updates the clusters with the clustering method in the DR view, the FCs view updates the heatmap with the reordering (and aggregation) method(s). When the user adds a new cluster manually, the FCs view updates the heatmap with the new cluster.

From FCs View: The FCs view can be used as an interface to compare the details of the features' values within/across features or clusters. When the user places the mouse over a certain heatmap cell, the system shows a popup window of the histograms of feature values of the corresponding cluster and the others (e.g., Fig. 2.2-c and Fig. 2.14-b). The selected cluster's histogram is colored with a categorical color representing its cluster label, while the gray color is used for the other data points' histogram. When hovering over a certain (representative) feature name, the system shows a value of the (representative) feature as the size of each data point in the DR view (e.g., Fig. 2.12-a and Fig. 2.13-a).

Moreover, when hovering over a certain cluster label, the system highlights the corresponding cluster in the DR view. In addition, with the popup window, the system visualizes the histograms of 1D DR results of the cluster and the others. From these histograms, the user can grasp how well the cluster is contrasted with the other data points. Additionally, the system shows the histograms of three (representative) feature values that have the highest absolute FCs. These histograms are useful to understand each cluster's characteristics quickly.

Also, to make the comparison within/across features or clusters easier, the system allows the user to prevent the histograms from disappearing with a mouse-click. The clicked histograms can also be moved with mouse-dragging. The corresponding heatmap cell for each histogram is annotated with a gray line and a pair of numbers shown in the heatmap cell and the histogram (e.g., Fig. 2.2 and Fig. 2.14-b). The gray line can be turned on or off by clicking the "Show/Hide Histogram Indicator" placed at the top of the FCs view.

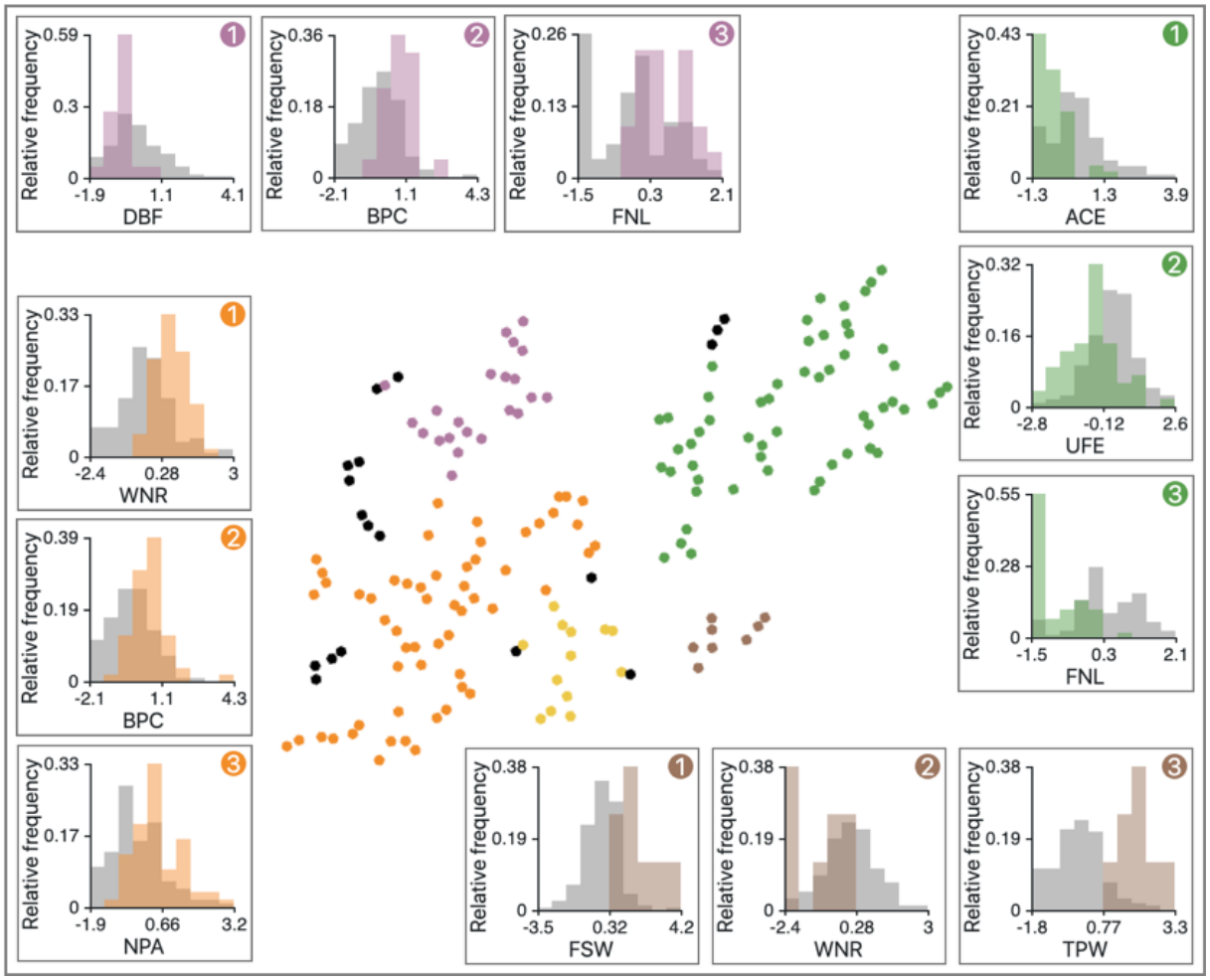


Figure 2.11: An analysis result of female players from the Tennis Major Tournament Match Statistics dataset.

2.4.4 Implementation

The system is developed as a web application. To achieve fast calculation, the methods described in Sec. 2.3 are implemented with C++ and Eigen library [94] for linear algebraic calculations. Python bindings are provided for the C++ implementation. The source code is available online (Appendix A). The back-end of the system uses Python with the stated bindings. The front-end visualization is implemented with a combination of Elm[2], HTML5, JavaScript, WebGL, and D3 [33]. While D3 is used for the FCs view, WebGL is used to render the data points efficiently for the DR view. WebSocket is used to communicate between the front- and back-ends.

[2]Elm: `https://elm-lang.org/`. Accessed: 2019-3-7.

2.5 Case Studies

The effectiveness of ccPCA and the visual analytics system has been shown by analyzing the Wine Recognition [65] and MNIST [148] datasets in the previous sections. This section presents three additional case studies with publicly available datasets. For each case study, the corresponding dataset is preprocessed to clean up missing values in the data or extract useful information for the analysis. All the preprocessed datasets are available online (Appendix A).

2.5.1 Study 1: Tennis Major Tournament Match Statistics

This study analyzes the Tennis Major Tournament Match Statistics dataset from the UCI Machine Learning Repository [65]. This dataset contains the match statistics for both females and males at four major tennis tournaments in 2013. The statistics include first serve won by each player, double faults committed by each player, etc. From this dataset, female players' mean values for each statistic across all tournaments are obtained. The obtained dataset consists of 174 data points (tennis players) and 13 features (statistics).

Similar to the analysis of Sec. 2.2.2, the DR result with t-SNE, clusters with DBSCAN, and FCs with ccPCA are generated. Then, to analyze each cluster's characteristics, the histograms of the top-3 contributed features are visualized. The result is shown in Fig. 2.11.

From Fig. 2.11, we can see that each cluster has a different playing style. For example, the purple cluster tends to have low 'DBF' (double faults committed by player), high 'BPC' (break points created by player), and high 'FNL' (final number of games won by player). This indicates that these players had fewer mistakes in their serves and performed well when they were the receiver, and as a result, they won more games. Similarly, the orange cluster has high 'WNR' (winners earned by player) and 'NPA' (net points attempted by player). These statistics will tend to be higher when a player tries to obtain points aggressively during a rally. On the other hand, the brown cluster has low 'WNR' but high 'FSW' (first serve won by player) and 'TPW' (total points won by player). Therefore, we can say that these players tend to obtain more points with their

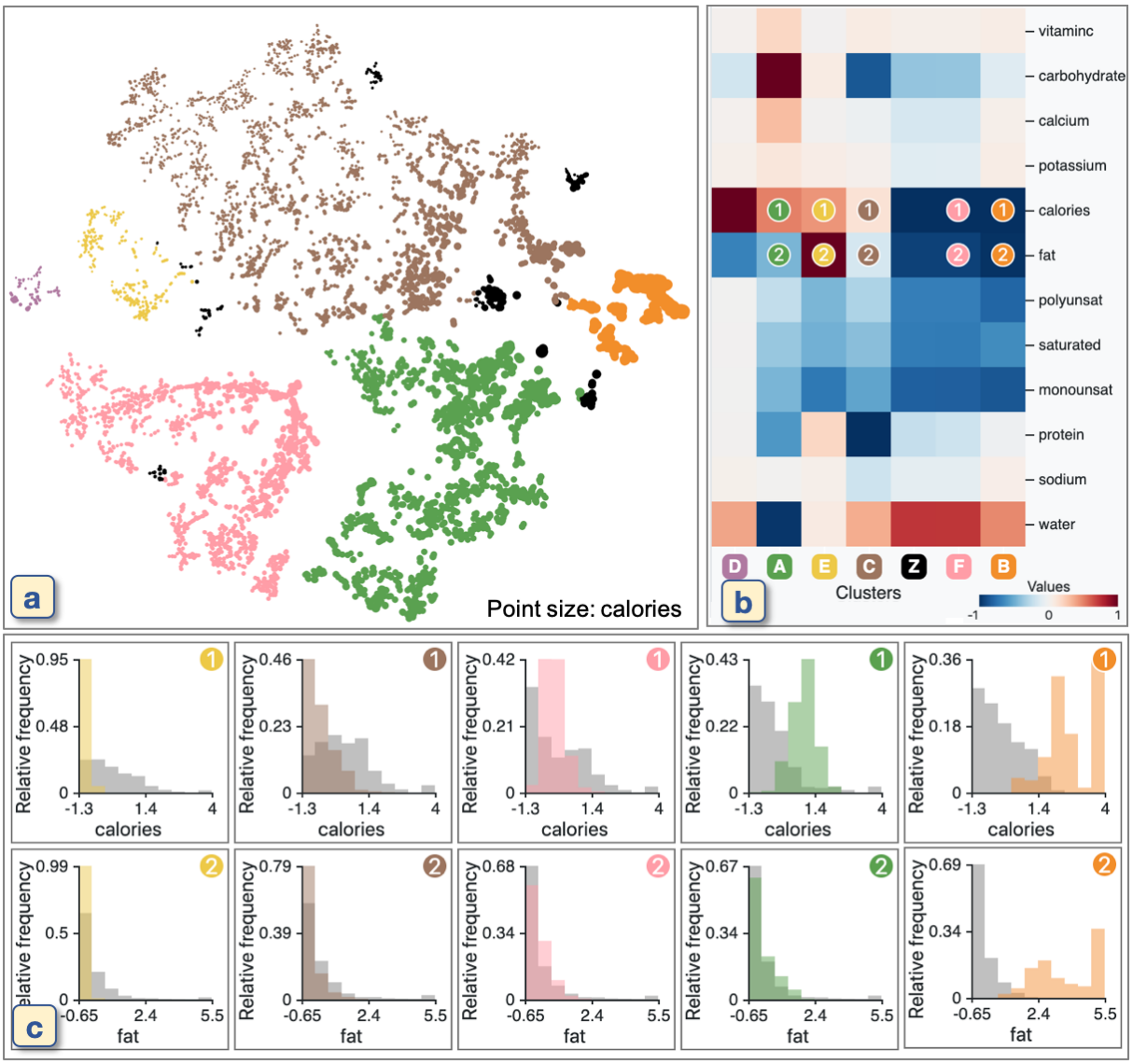


Figure 2.12: A result of the Nutrient dataset. (a) shows the result after applying t-SNE and DBSCAN. A point's color and size show the clustering label and the value of 'calories', respectively. (b) shows the FCs of each cluster. (c) shows the histograms of the selected cells in (b), as indicated with the colored numbers in both (b) and (c).

serves.

2.5.2 Study 2: Food and Nutrient

This study analyzes the Nutrient dataset in the USDA Food Composition Databases [221] as an analysis example with a large number of data points. The version available online[3] is used for this study. This dataset consists of the nutrient content for each food.

[3]Nutrient Explorer: `http://bl.ocks.org/syntagmatic/raw/3150059/`, Accessed: 2019-3-7.

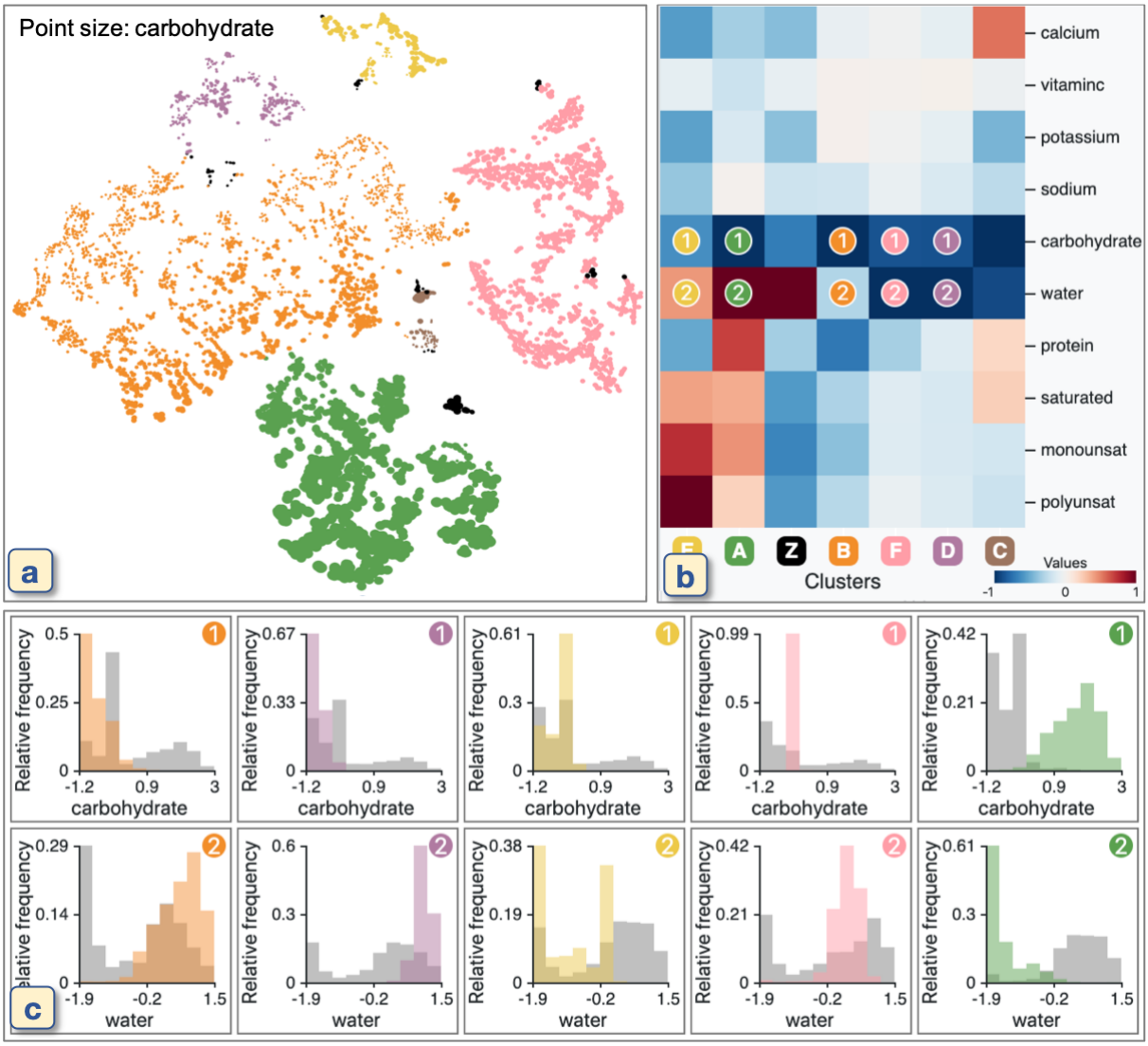


Figure 2.13: The result after filtering out the 'calories' and 'fat' features from the Nutrient dataset. In (a), a point size represents the value of 'carbohydrate'. The histograms of the selected cells in (b) are shown in (c).

The dataset has 7,637 data points (foods) and 14 features (nutrients).

This dataset has 12,507 missing values and this is 11.7% of all the values. Since this high percentage of missing values could affect an analysis result [5], the dataset is preprocessed to reduce this ratio to less than 5%. In this preprocessing step, features where more than 40% of the values are missing are removed. Also, data points where more than 40% of the feature values are missing are removed. Afterward, 7,499 data points and 12 features remain and there are 4,447 missing values (4.9% of all the values).

Then, the missing values are replaced with the mean of each corresponding feature.

The result after using t-SNE, DBSCAN, and ccPCA is shown in Fig. 2.12. As shown in Fig. 2.12-b, we can see that all clusters except for the brown cluster have high FCs in 'calories', 'fat', or both. When comparing the histograms of 'calories' and 'fat' for each cluster, as shown in Fig. 2.12-c, each cluster, in fact, has different distributions in 'calories' and 'fat'. For example, while the yellow cluster tends to have low calories and fat, the orange cluster tends to have high values for both.

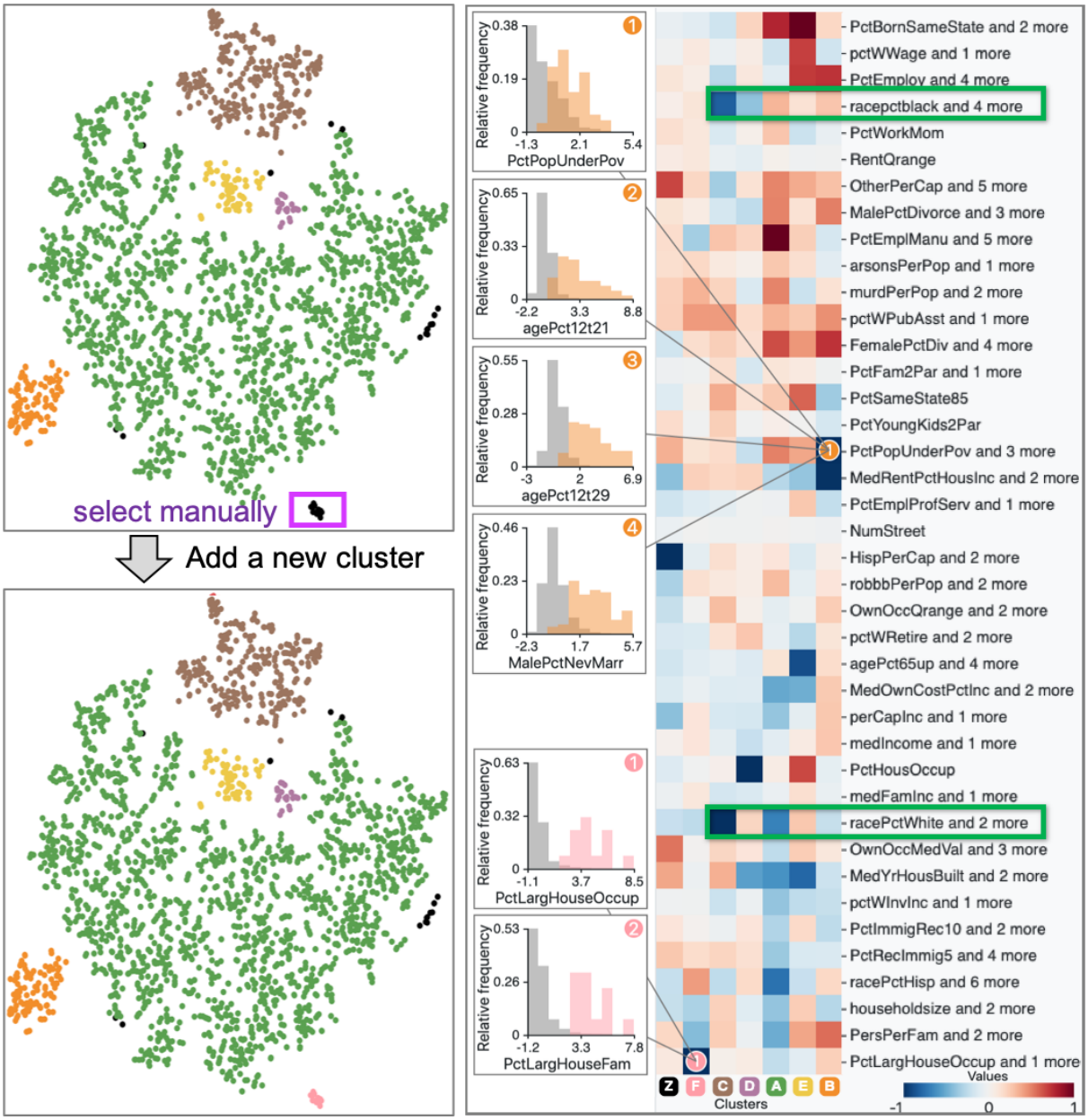
We have understood the main characteristics of each cluster. However, the effects of the two specific features ('calories' and 'fat') are too dominant. As a result, we cannot find any other interesting patterns. The dataset is preprocessed to filter out these two features and generate a new result with new cluster labels, as shown in Fig. 2.13. At this time, from Fig. 2.13-b, we can see that most of the clusters are contrasted by mainly 'water', 'carbohydrate', or both. For example, the purple and orange clusters placed in the upper left of Fig. 2.13-a have fewer carbohydrates and more water when compared with the pink and green clusters, as shown in Fig. 2.13-c. These two examples show that the FCs are useful to know which features have a dominant effect on cluster forming in the DR result.

2.5.3 Study 3: Communities and Crime

As an example with a large number of features, this study analyzes the Communities and Crime dataset [189] from the UCI Machine Learning Repository [65]. This dataset consists of both socio-economic and crime statistics (e.g., the median family income and the number of murders) for each community. The dataset contains 2,215 data points (communities) and 143 features (statistics) after excluding identifiers (e.g., county codes).

Because this dataset has many missing values (42,147 values, 13.3% of all the values), as similar to Sec. 2.5.2, the features where more than 80% of the values are missing are removed. The dataset now has 121 features and only 963 missing values (0.4% of all the values). The missing values are replaced with the mean of each corresponding feature.

Fig. 2.14-a (top) shows the result after DR and clustering. As indicated with the



(a) DR result (b) Aggregated FCs and histograms

Figure 2.14: The results for the Communities and Crime dataset. The top of (a) shows the result with t-SNE and DBSCAN. In the bottom of (a), the pink cluster which was not identified by DBSCAN is manually added. (b) shows 40 aggregated features from 121 features. Also, some of the histograms of the original features are visualized at the left of (b).

purple rectangle, an additional cluster is manually selected as a pink cluster. Then, the FCs are obtained, as shown in Fig. 2.14-b. Because there are many features, the system has aggregated them into 40 features using the aggregation method described

in Sec. 2.4.2.3. From Fig. 2.14-b, we can say that the small clusters (yellow, purple, brown, orange, and pink) are separated from the green cluster due to race, house size, etc.—not due to the criminal statistics. For instance, as indicated with the green rectangles, the brown cluster has high FCs in race percentages of African Americans and Caucasians ('racepctblack' and 'racePctWhite'). Also, the pink cluster has high FC in 'PctLargHouseOccup' (percentage of all occupied households that are large).

The histograms of the features aggregated to the 'PctLargHouseOccup and 1 more' are visualized, as shown in the lower left of Fig. 2.14-b. We can see that both 'PctLargHouseOccup' and 'PctLargHouseFam' (percentage of family households that are large) have similar distribution patterns. These patterns can be found because the aggregation method is performed after applying the optimal sign flipping and ordering described in Sec. 2.4.2. The aggregation method is able to provide a scalable visualization and help the user analyze many features. Another example for 'PctPopUnderPov and 3 more' of the orange cluster is shown in the upper left of Fig. 2.14-b. All 'PctPopUnderPov' (percentage of people under the poverty level), 'agePct12t21' (percentage of population that is 12–21), 'agePct12t29' (percentage of population that is 12–29), and 'MalePctNevMarr' (percentage of males who have never married) tend to have a higher value in comparison to that of others.

2.6 Discussion

Generality of ccPCA. The method in this chapter utilizes cPCA [2, 3] to find features contrasting the target cluster. I discuss the reason why this approach is chosen instead of analyzing how the DR method generates clusters. If possible, the latter approach would be effective because the cluster formation is a result of the DR method. However, many of the nonlinear DR methods used for visualization (e.g., t-SNE [233], LargeVis [219], and UMAP [170]) generate an irreversible low-dimensional projection of the original data structure. These methods do not have a parametric mapping between the original and projected dimensions; therefore, it is difficult to provide information about how these DR methods affect cluster forming. ccPCA provides flexibility for analyzing

results from any type of DR methods.

ccPCA is introduced to understand the characteristics of the clusters identified in the DR result. ccPCA can also be used in other situations. For example, even though using DR before clustering is a common approach [245,250], ccPCA can support visual analytics of clusters that are obtained from the clustering methods without going through the DR step. This would be helpful to understand clusters' characteristics and to analyze the quality of the clustering methods without any effects derived from DR (e.g., distortion in the projection space). Another example is applying ccPCA to labeled data. ccPCA can identify the essential features to contrast a labeled group from the others. Therefore, ccPCA would be useful to understand the characteristics of each group and could help design classifiers based on the gained knowledge. The prototype system demonstrated in this chapter can support these types of analysis by changing the parts related to steps (a) and (b) in Fig. 2.1, such as the DR view and clustering algorithms.

Advantages of using cPCA. In Sec. 2.3, I have already discussed the advantages of using cPCA when compared with using PCA and LDA. It is also possible to compute the discrepancy score D introduced in Sec. 2.3.2.3 for each original feature without using ccPCA and then use the score as the feature contribution. However, this approach has a similar problem with LDA because the obtained score only shows the separation and does not take into account the variety (i.e., variance) for each feature.

Another potential option is using the two-group differential statistics methods [168], such as two-sample t-test, Wilcoxon signed-rank test, and Mann-Whitney U test, to find features that have differences between the target cluster and the others. Unlike LDA, PCA, or cPCA, these methods cannot produce a quantitative measure for analyzing the FCs to the contrast of the cluster. More importantly, these statistical methods are designed to test whether there is a difference in a certain statistic (typically mean) between two clusters. Therefore, these methods are not suitable for performing exploratory analysis on clusters when we do not know their characteristics beforehand.

Limitations. Since cPCA is used, we will need to address its limitations in terms of

time and space complexity for a large-scale problem. Similar to the classical PCA, cPCA computes the covariance matrices and then performs EVD. For a fixed α, it has the same time and space complexity with PCA, which are $O(d^2n + d^3)$ and $O(d^2)$, respectively, where n is the number of data points and d is the number of features. Thus, cPCA can achieve fast computation for a dataset that has a large n, but not for a dataset with a large d (the experimental results are included in the online supplementary materials (Appendix A). For PCA, incremental algorithms [182,193,244] have been developed to solve this issue. For example, the algorithm introduced by Ross et al. [193] has the time and space complexity of $O(dm^2)$ and $O(d(k+m))$, respectively, where m is the number of data points used in each batch, and k is the number of principal components. I thus plan to develop an incremental version of cPCA next.

2.7 Summary

Dimensionality reduction is widely used to analyze high-dimensional data for pattern discovery and real-world problem-solving. The work in this chapter makes a tangible contribution to interpreting and understanding DR results by introducing a visual analytics method that capitalizes on contrastive learning. Using a scalable visualization, the method directs the user to the essential features within the data. The work, thus, further enhances the usability of DR methods.

Chapter 3
Multivariate Time-Series Data Analysis

Analysis of multivariate time-series data is becoming increasingly important to study various phenomena in the real world. For example, analyzing electronic health records (EHRs) that contain temporal changes of individuals' various medical measures (e.g., blood pressure and heart rate) for cohort studies can help clinical researchers develop healthcare plans [96, 116, 150]. Many other analysis examples can be found in other domains, such as diagnosis of the performance of parallel computing systems [77, 95, 175], fault detection of factory assembly lines [247, 249, 262], and optimization of transportation systems [54, 156]. As seen in the emergence of the Internet of Things, the growing capability and use of sensing devices improves the granularity, quality, and accessibility of multivariate time-series data [60, 215]; at the same time, the increase of the data size and dimensionality makes analysis tasks more challenging [7].

To effectively analyze and visualize large, high-dimensional data, dimensionality reduction (DR) methods are often used [159, 198] because of their ability to provide a succinct overview of such complex data. Currently available DR methods designed for 2D or 3D visualization [52, 234] can be only applied to data that can be formed into a 2D matrix, such as single-time-point multivariate data (matrix rows: instances, columns: variables), univariate time-series data (rows: instances, columns: time points), and multivariate time-series with a single instance (rows: time points, columns: variables). When multivariate time-series data consists of multiple instances, the data is often represented as a third-order tensor (or 3D array); consequently, we either cannot directly apply some DR methods such as principal component analysis to the data or we do not know how to properly prepare a distance matrix as an input for other DR methods (e.g., t-SNE [233]).

A common approach to the above problem is slicing the 3D array and then applying a DR method to each resulting slice [17]. For instance, when slicing along a temporal direction, where each slice represents a matrix of instances and variables, we can visualize a set of DR results with animation or small multiples. However, when a sliced direction has high dimensionality (e.g., 100 time points), the analyst must examine a large amount of DR results and can easily overlook important patterns (e.g., the emerge of outliers).

To support effective analysis of multivariate time-series data, this chapter introduces a visual analytics framework, MulTiDR, which employs a two-step DR to generate an overview of the data and supports interpreting the DR results with ccPCA described in Chapter 2 and interactive visualization. Particularly, in the first step of DR, MulTiDR compresses and converts a third-order tensor into a matrix, and then, in the second step, it projects high-dimensional data points into a lower-dimensional space. Similar to the existing DR methods, the two-step DR result shows similarities of instances, variables, or time points and enables visual identification of essential patterns, such as clusters and outliers. When compared with ordinary DR, the two-step DR result is derived from two different directions (e.g., variables and time points) and could be more difficult to understand why specific patterns appear. Thus, to support the analysis of the two-step DR, ccPCA is integrated to identify essential aspects for the analysts to review and interpret in detail with interactive visualization. I demonstrate the effectiveness of MulTiDR for multivariate time-series analysis with multiple case studies using real-world datasets and also make qualitative comparisons of MulTiDR with other potential DR methods.

3.1 Background and Related Work

This section provides a brief description of third-order tensors and discusses relevant works.

3.1.1 Third-Order Tensors

A third-order tensor is a 3D array (note that first- and second-order tensors correspond to vectors and matrices, respectively). Each axis of a tensor is called *mode*. When a third-order tensor represents multivariate time-series data, the three modes correspond to time points, instances (or samples), and variables (or attributes). As the main analysis target, the description in this chapter focuses on third-order tensors of multivariate time-series data; however, the framework, MulTiDR, is designed to be able to deal with the other types of third-order tensors.

The notations used in this chapter follow the conventions in the literature [136]. Here we denote scalars, vectors, matrices, and third-order tensors with lowercase (e.g., x), boldface lowercase (e.g., $\mathbf{x}$), boldface uppercase (e.g., $\mathbf{X}$), and boldface Euler script (e.g., $\boldsymbol{\mathcal{X}}$) letters, respectively. We use indices $t = 1, \ldots, T$, $n = 1, \ldots, N$, and $d = 1, \ldots, D$ for time points, instances, and variables, respectively. Here T, N, and D are lengths of modes of time points, instances, and variables, respectively (i.e., a third-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{T \times N \times D}$).

3.1.2 Related Work

This chapter's work relates to visual analytics of third-order tensors. A third-order tensor commonly found in the visualization field is a "generalized" space-time cube. A generalized space-time cube represents a 2D visualization space that changes over time (e.g., temporal geospatial visualizations and animated 2D scatterplots). Bach et al. [16,17] provided a comprehensive survey of visualizations of generalized space-time cubes. They also provided a categorization of visualization strategies. The strategies include 3D rendering (i.e., render a cube as it is), time cutting (i.e., extracting a 2D snapshot at a particular time point), time flattening (i.e., collapsing temporal changes into a single 2D image), time juxtaposing (i.e., arranging multiple 2D snapshots as small multiples), space cutting (i.e., extracting a planar cut in one direction of the 2D space), and among others. One strategy that the survey did not discuss in detail is dimensionality reduction (DR), which can be considered as a special form of flattening. Since MulTiDR also employs DR, here we focus on discussing the works using DR to

visualize third-order tensors.

Similar to this chapter's work, a target application of many of the existing works is visualizing multivariate time-series data. One simple strategy is applying DR to a matrix of instances and variables at each time point and then showing temporal changes with animation or juxtaposition. To support such a visualization, the researchers developed dynamic DR methods that provide coherent node positions between consecutive time points, such as the time-based least square projection [9] and Dynamic t-SNE [188]. However, finding useful patterns, such as outliers or similar time points, is difficult when relying on animation or juxtaposition.

Another common strategy is applying DR based on a dissimilarity of each time point's matrix. This strategy generates an overview of the (dis)similarities of time points. For example, Bach et al. [20] computed the dissimilarity of each pair of 2D images at different time points with a certain distance measure, such as a Euclidean distance; then applied multidimensional scaling (MDS) based on their dissimilarities. In the MDS result, a 2D image at each time point is visualized as a dot. To covey the time information, they connected dots of two consecutive time points and colored them according to time. Several researchers also used a similar approach to provide a visual summary of dynamic network data [74, 230, 237]. On the other hand, Jäckle et al. [119] visualized an MDS result in a 1D axis and used another axis to represent time. Since MDS may produce unnecessary rotation in the result, they reduced the rotations by flipping the y-coordinates based on their positions in the previous time point. Muelder et al. [175] also took a similar approach but they used a graph layout algorithm as a DR method instead of MDS. However, all the approaches above have several problems. For example, when two modes in each matrix slice have different types (e.g., instances and variables), the DR result might not capture any useful patterns because each mode is mixed together when computing dissimilarities (refer to Sec. 3.5 for concrete examples). Also, because each dot in the DR result represents a matrix, it is difficult to identify which instances or variables highly relate to a certain pattern appeared in the DR result (e.g., clusters) and, consequently, the result has low interpretability.

While the visualizations above focus on showing the time points' similarities, some works are to overview the instance similarities over time. For example, Fujiwara et al. [77] used time-series distance measures, such as dynamic-time warping, to obtain the similarity of each instance's changes in a variable value across time. Afterward, for each variable, they applied MDS or t-SNE to the computed similarities and then juxtaposed the DR results for different variables. Kesavan et al. [130] extended the same approach for streaming high-dimensional data. In contrast to MulTiDR, these approaches handle only one variable in each DR result.

Recently, visualization researchers have started to use tensor decompositions [136, 164] to analyze or simplify third-order tensors. The two most popular tensor decompositions are canonical decomposition (or CP decomposition) [40, 104] and the Tucker decomposition [225]. CP decomposition expresses a tensor as the sum of a finite number of rank-one tensors (i.e., tensor-to-vector decomposition). On the other hand, the Tucker decomposition can be considered as a high-order version of PCA and decomposes a tensor to a core tensor and a matrix along each mode (i.e., tensor-to-tensor decomposition). For example, TPFlow [156] introduces a similar method to CP decomposition, which finds the best slice of a space-time cube, where some meaningful patterns likely exist. Voila [39] uses the Tucker decomposition to detect anomalies from a space-time cube. Also, TTHRESH [21] utilizes the Tucker decomposition to compress volume data into a smaller file size. While these works extract important features or elements from third-order tensors, MulTiDR generates an overview from a third-order tensor for visual identification of patterns, such as clusters and outliers, and provides interpretability in the DR result.

3.2 Algorithm Architecture

Fig. 3.1 shows a general architecture of the back-end of MulTiDR. MulTiDR provides two major functionalities: (a) two-step DR to project a third-order tensor onto a low-dimensional space and (b) generation of essential information for interpreting the two-step DR results.

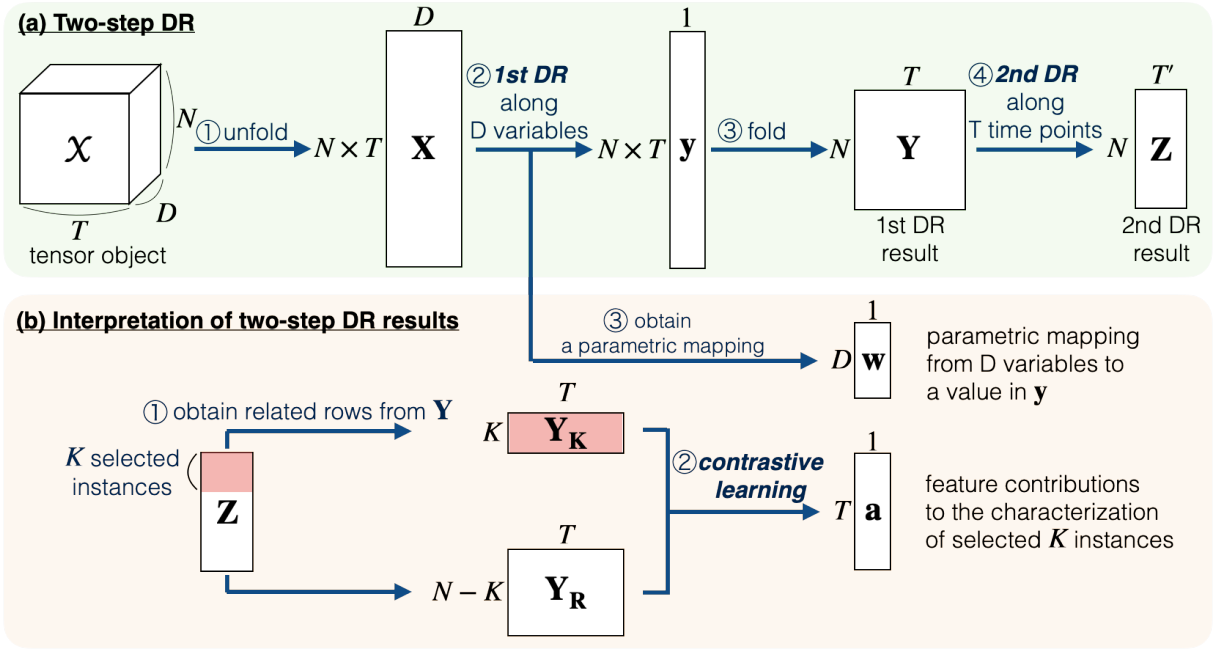


Figure 3.1: General architecture of MulTiDR back-end. This figure demonstrates a case when showing instance similarities based on their temporal changes of variable values.

3.2.1 Two-Step DR

This subsection describes how MulTiDR achieves the projection of a third-order tensor. To make an explanation concrete and concise, here we use a case shown in Fig. 3.1-a. The descriptions below related to T time points, N instances, and D variables are interchangeable between themselves.

The first step of DR is to compress a third-order tensor $\mathcal{X} \in \mathbb{R}^{T \times N \times D}$ into a matrix $\mathbf{Y} \in \mathbb{R}^{N \times T}$, where certain information (e.g., variances) of variables is preserved as much as possible. To achieve this, we first apply *tensor unfolding* [136] along a variable mode (Fig. 3.1-a①), which reshapes $\mathcal{X}$ to a matrix $\mathbf{X}$ of $(N \times T)$ rows and D columns by arranging all vectors (or often called *fibers*) of D length obtained through the slicing of $\mathcal{X}$ along both time and instance modes. Afterward, a DR method is applied to $\mathbf{X}$ and reduces D dimensions to 1 dimension (Fig. 3.1-a②). Now, $\mathcal{X}$ is compressed into a vector $\mathbf{y}$ of length $(N \times T)$. Based on $\mathbf{y}$'s indexes correspond to the time and instance directions of $\mathcal{X}$, we can fold $\mathbf{y}$ into a matrix $\mathbf{Y} \in \mathbb{R}^{N \times T}$ (Fig. 3.1-a③). Because the main purpose of the first DR is preserving the information of variables, a linear DR method

that can be used for data compression is suitable. For example, while we can use PCA to preserve the variances of variables, linear discriminant analysis (LDA) is also a potential option if the analyst wants to preserve differences between $\mathcal{X}$ and another third-order tensor. Also, the linearity of DR is important to provide interpretability, as described in Sec. 3.2.2..

The second step of DR is to visualize $\mathbf{Y}$ in a lower-dimensional space. For this step (Fig. 3.1-a④), based on the analysis purpose, we can simply select any DR method that can be applied to a matrix, such as PCA and t-SNE [233]. Through this step, $\mathbf{Y} \in \mathbb{R}^{N \times T}$ can be represented as $\mathbf{Z} \in \mathbb{R}^{N \times T'}$ ($T' < T$, typically $T' \in \{1, 2, 3\}$).

Instead of using the two-step DR above, similar to Unfold PCA [132, 133], another potential approach is unfolding $\mathcal{X}$ to a matrix of N rows and $(D \times T)$ columns and then apply DR in order to reduce dimensions of $(D \times T)$ to a lower number of dimensions. When compared with this approach, the two-step DR has the main advantage in handling different modes (e.g., variables and time points) with clear distinction, and this benefits both identification and interpretation of patterns in $\mathcal{X}$. For example, when the analyst wants to review the similarities of N instances mainly based on patterns seen along a time mode but not a variable mode, they can use the process shown in Fig. 3.1-a. Also, as described in the next subsection, the interpretation of the DR results becomes more straightforward because, for example, we can understand which time points highly contribute to the characteristics of a cluster seen in the DR result. More detailed comparisons are provided in Sec. 3.5.

There are six different combinations to generate the two-step DR result $\mathbf{Z}$ based on which modes are selected as the first and second DR targets: (1st DR target mode, 2nd DR target mode) = {(time, instance), (time, variable), (instance, time), (instance, variable), (variable, time), (variable, instance)}. The analyst can choose a preferable combination from these based on their analysis interest. For example, when selecting (variable, time), a two-step DR result shows instance similarities mainly based on temporal behaviors while considering distribution differences in variables. On the other hand, a selection of (variable, instance) generates time points' similarities based

on instances' states (i.e., values of the compressed variables each instance has) at each time point.

3.2.2 Supporting Interpretability

When analyzing the DR result, as discussed in Chapter 2, we often want to identify clusters from the DR result and understand the characteristics of the clusters [34, 76, 181]. Similar to the existing DR methods, identification of clusters can be visually performed on the two-step DR result (i.e., finding a set of points placed closely to each other). However, when compared with the case of applying ordinary DR methods to a matrix, understanding the cluster's characteristics from a two-step DR result is more complicated. Therefore, I provide algorithmic support for this task.

As shown in Fig. 3.1-b, MulTiDR provides two different pieces of information for the interpretability: $\mathbf{a}$, feature contributions of T time points to a cluster's characteristics, and $\mathbf{w}$, a parametric mapping used to compress D variables into one dimension.

To obtain feature contributions $\mathbf{a}$, we follow the approach in Chapter 2. As shown in Fig. 3.1-b①, MulTiDR first takes K instance indices related to a target cluster and then, from $\mathbf{Y}$, extracts $\mathbf{Y_K} \in \mathbb{R}^{K \times T}$, a submatrix corresponding to these K instances, and $\mathbf{Y_R} \in \mathbb{R}^{(N-K) \times T}$, the rest of $\mathbf{Y}$ (i.e., $\mathbf{Y_R} = \mathbf{Y} \setminus \mathbf{Y_K}$). From inputs $\mathbf{Y_K}$ and $\mathbf{Y_R}$, contrastive learning (CL) generates $\mathbf{a} \in \mathbb{R}^T$ (Fig. 3.1-b②), which shows how strongly each time point contributes to the uniqueness of a target cluster with respect to the others. By referring to $\mathbf{a}$, the analyst knows which time points in $\mathbf{Y}$ they should review to understand the target cluster's characteristics.

However, each cell of $\mathbf{Y}$ represents the compressed variable from D to 1 dimension. To understand the cluster's characteristics, we also need to know how the compressed variable is derived from the original D variables. To do so, we can refer to a parametric mapping vector $\mathbf{w} \in \mathbb{R}^D$ (Fig. 3.1-b③), which is usually provided by DR methods for data compression (e.g., PCA). $\mathbf{w}$ consists of a weight for each of D variables, which is used to project D variable values to one compressed value.

3.2.3 Implementation Example

As described above, the back-end architecture of MulTiDR provides flexibility in the selection of the first DR, second DR, and CL. Here describes a representative implementation example, which is used through the rest of the chapter. For the first DR, PCA is selected because it is most popularly used for data compression when applying machine learning methods, including DR methods. UMAP [170] is chosen as the second DR because of its effectiveness to find patterns from nonlinear relationships. Also, unlike the other nonlinear DR methods (e.g., t-SNE [233]), UMAP is suitable for capturing both local and global topological structures of the data. Because of this ability, UMAP is effective in finding patterns from both small- and large-scale data while t-SNE and many other nonlinear DR methods are not suitable for small-scale data (e.g., data with 50 instances). Lastly, for the purpose of understanding the characteristics of clusters, currently, ccPCA introduced in Chapter 2 is the only available option; thus ccPCA is used as a CL method.

3.3 MulTiDR Visual Interface

MulTiDR provides a visual interface to support interactive analysis of the two-step DR results together with the information that helps the interpretation of the results. As shown in Fig. 3.2, MulTiDR visual interface consists of five coordinated views: (a) a two-step DR (TDR) view, (b) a supplemental information (SI) view, (c) a feature contribution (FC) view, (d) a histogram comparison (HC) view, and (e) a projection mapping (PM) view.

Fig. 3.3 shows an analysis workflow with MulTiDR visual interface. This is an extended workflow from those for high-dimensional data analysis introduced in Chapter 2 (refer to Fig. 2.1) for multivariate time-series analysis. After obtaining a two-step DR result $\mathbf{Z}$, feature contributions $\mathbf{a}$, and a parametric mapping $\mathbf{w}$, the two-step DR result is visualized in Fig. 3.2-a. The analyst can first visually identify clusters from the DR result (Fig. 3.3-A) and then analyze each cluster. When points in the DR result have the auxiliary information (e.g., the location information of instances), the analyst can

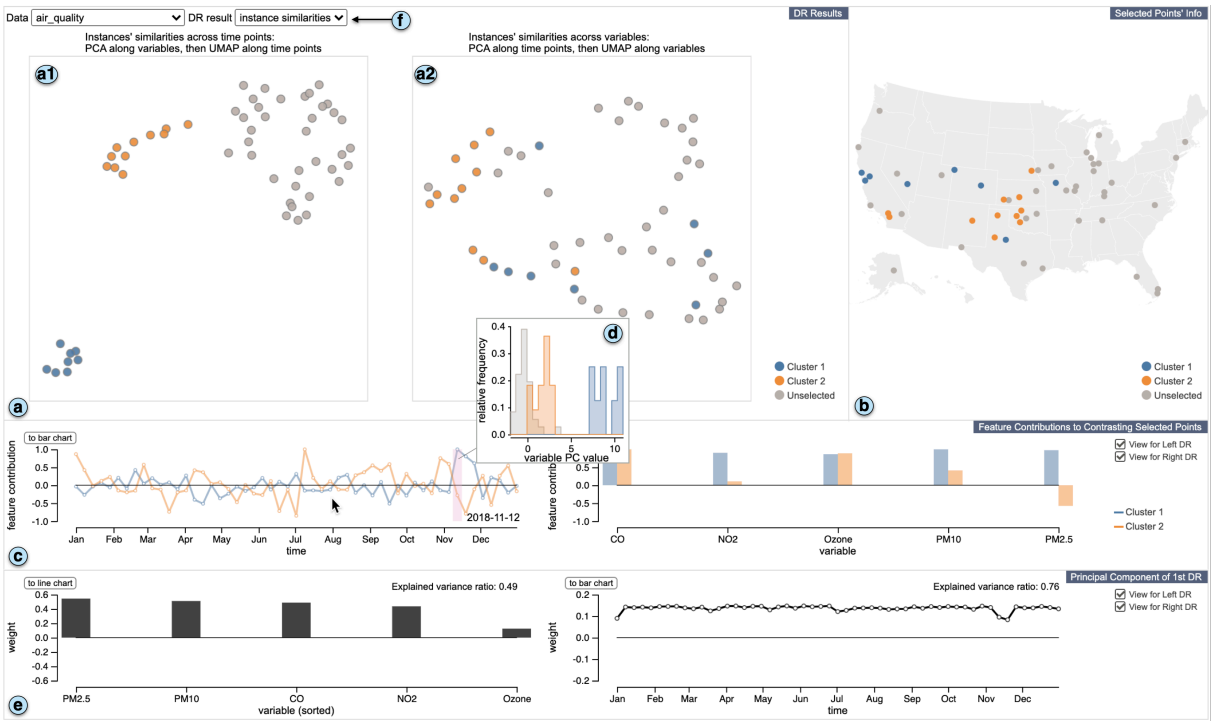


Figure 3.2: A screenshot of MulTiDR visual interface. Here the system visualizes the AirData [229], air quality data at outdoor monitors across the US, collected in 2018. (a) A two-step DR (TDR) view draws the DR results obtained through the two-step DR. (b) A supplemental information (SI) view supports understanding selected points in the TDR view with the auxiliary information. (c) A feature contribution (FC) view visualizes features (either instances, variable, or time points) and their contributions to characteristics of each of selected clusters. (d) A histogram comparison (HC) view shows the feature values in the first DR result $\mathbf{Y}$ of the selected element in (c). (e) A parametric mapping (PM) view depicts parametric mappings generated in the first DR, specifically the mappings to the first principal component in this example. (f) The analyst can select a type of the DR results.

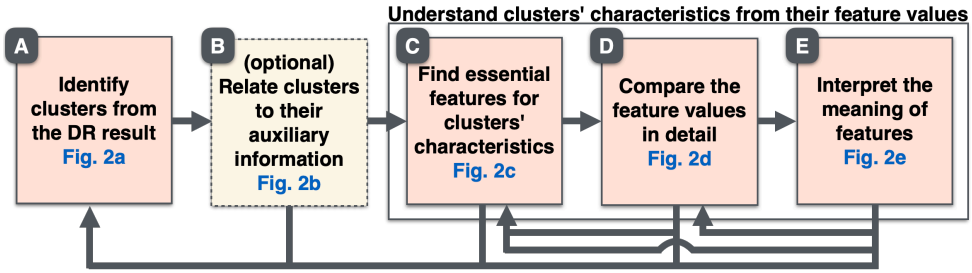


Figure 3.3: Multivariate time-series analysis workflow with MulTiDR visual interface.

(B) relate the identified clusters to such information, as shown in Fig. 3.2-b. Afterward, they can move forward to steps (C, D, E), where the information of feature contributions and parametric mapping is used to understand the clusters' characteristics. With Fig. 3.2-c, the analyst can start with (C) finding which features (i.e., columns in $\mathbf{Y}$) highly

contribute to characterizing each cluster. For each of the highly contributed features, by using Fig. 3.2-d, the analyst can (D) compare the differences of feature value distributions among clusters. Since the features are obtained through the compression with the first DR, the analyst also (E) interprets the meaning of the features by reviewing the parametric mapping information provided in Fig. 3.2-e. As indicated with the arrows in Fig. 3.3, the above steps often drive a continuous analysis loop in order to identify other clusters, select other features of interest, or examine findings obtained in the other view.

The rest of the section describes each view of MulTiDR with a concrete analysis example using the US weekly air quality data in 2018 [229], which consists of 53 weeks, 55 counties, and 5 different air quality measures (i.e., $T = 53$, $N = 55$, $D = 5$). A demonstration video of the interface is available at the online site listed in Appendix A.

3.3.1 Visualization of Two-Step DR Results

The TDR view (Fig. 3.2-a) visualizes the results obtained through the two-step DR as scatterplots. As described in Sec. 3.2.1, the two-step DR can generate six different results from a multivariate time-series dataset based on target modes of the first and second DR (e.g., the first DR along a variable mode and the second DR along a time mode). From these results, the analyst can select which mode's similarities they want to show from a drop-down menu at Fig. 3.2-f. For example, in Fig. 3.2-a, instance similarities are selected. Consequently, as described at the top of each of scatterplots (a1 and a2), the TDR view shows two results that are obtained by applying the first and second DR along (variable, time) and (time, variable) at the left and right, respectively.

From the results, the analyst can visually identify clusters and manually select them by using a lasso selection. Selected points are labeled as one cluster and color-coded with a categorical color. For example, in Fig. 3.2-a1, the analyst has first selected Cluster 1 (blue) and then Cluster 2 (orange). In addition to the lasso selection, TDR view also supports fundamental interactions, such as zooming and panning. After the selection, all other views update their visualizations. From Fig. 3.2-a1 and a2, we can see that although the orange points in a1 are also relatively placed closely to each other in a2,

the clusters in a1 tend to be more mixed with each other in a2. This indicates that instances (i.e., counties) in these clusters generally have similar temporal patterns in their representative variable values (i.e., representative air quality measure); however, they tend to have different patterns in variable values (e.g., some of the blue points may have high variable values but the others do not). An analysis example utilizing both of the two different DR results is demonstrated in Sec. 3.4.3.

3.3.2 Visualization of Related Contexts

After identifying clusters, we often want to understand what kind of points are included in each cluster and why they are clustered by the two-step DR.

The SI view (Fig. 3.2-b) is designed for the former task. The SI view visualizes the auxiliary information of the selected points in the TDR view if available. In Fig. 3.2-b, the location information of the selected counties is visualized, where the blue and orange clusters tend to be seen in more west and center, respectively. MulTiDR provides a set of predefined visualizations and selects one from them based on which mode and dataset need to be visualized. For example, when showing the information for a time mode of the air quality data, MulTiDR shows a calendar-based visualization to convey the seasonal patterns. While the SI view shows the location information for an instance mode of a geospatial dataset, when analyzing a network dataset, the SI view can provide a node-link diagram. Sec. 3.4 demonstrates examples.

3.3.3 Visualization of Feature Contributions and Values

In the next step, the analyst can analyze the clusters' characteristics with the FC view (Fig. 3.2-c) and HC view (Fig. 3.2-d).

The FC view shows feature contributions **a** for each of DR results in the TDR view (the left and right plots in Fig. 3.2-c correspond to Fig. 3.2-a1 and a2, respectively). In default, line charts are employed for feature contributions of time points, while bar charts are used for those of instances or variables. However, the analyst can switch line and bar charts by clicking the button placed at the top of each of y-axes (e.g., "to bar chart" at the left side of the Fig. 3.2-c). Also, with the checkboxes placed at

the far right in Fig. 3.2-c, the analyst can select showing only one of the plots to use more screen space. Since feature contributions are obtained for each cluster, they are visualized with the corresponding cluster color. MulTiDR scales feature contributions between $[-1, 1]$ by dividing each set of feature contributions by their maximum absolute value. Closer to either 1 or -1 indicates higher contributions to the characterization of a cluster. The meaning of the sign is discussed in Sec. 3.3.3.1. For features that have high contributions, each cluster likely has different distributions from the other points.

To compare value distributions of the selected feature, as shown in Fig. 3.2-d, the HC view shows relative frequency histograms of selected clusters (e.g., blue and orange) and unselected points (gray) with the corresponding colors. The x-axis of the histograms represents feature values (i.e., cell values in $\mathbf{Y}$). The y-axis shows the relative frequency—the ratio of the number of items in each bin to the total number of items across all bins—within each group and its maximum limit is set to the maximum relative frequency among the histograms. From the result in Fig. 3.2-d, at the selected week highlighted with pink (i.e., a week of November-12th, 2018), the blue cluster tends to have much higher feature values than the others.

3.3.3.1 Sign Adjustment of Feature Contributions

ccPCA, which is used as a default CL method in MulTiDR, produces signed feature contributions (FCs). Signed FCs have a strength of differentiating features of having lower and having higher values within a selected cluster. For example, when looking at Cluster 2 (orange) in Fig. 3.4-a, where the absolute FCs are shown, both time points ③ and ④ have the relatively high FCs; however, as shown in Fig. 3.4-d, while Cluster 2 tends to have high values at ③, it has low values at ④. On the other hand, the signed FCs shown in Fig. 3.4-b indicates the difference of time points ③ and ④ (③: negative sign, ④: positive sign).

Despite the usefulness of signed FCs, similar to ordinary PCA, the *sign ambiguity* problem [75,76,226] in ccPCA limits their interpretability. That is, the signs are arbitrarily selected and thus they do not reflect whether features contribute to having higher or lower values than others. For example, in Fig. 3.4-b, although both Clusters 1 and

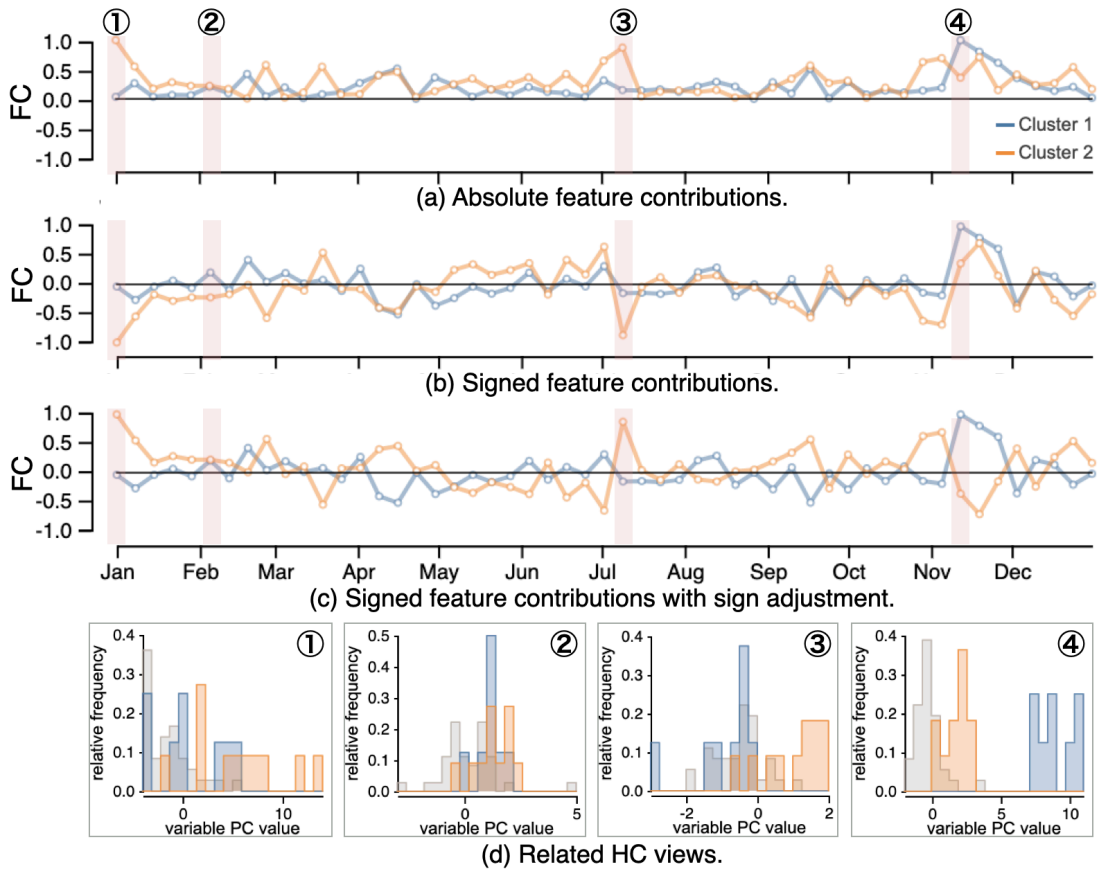


Figure 3.4: Comparison of visualizations of feature contributions (FCs): (a) absolute FCs, (b) signed FCs, and (c) signed FCs with sign adjustment. (d) shows the HC views corresponding to the four selected time points ①–④ in (a), (b), and (c).

2 have similar line shapes and strong positive FCs at ④, as shown in Fig. 3.4-d, at this time point, Cluster 1 (blue) tends to have high values while Cluster 2 (orange) tends to have low values. Therefore, using the signed FCs directly produced by ccPCA might mislead the analyst (e.g., they consider Clusters 1 and 2 are similar from Fig. 3.4-b).

To solve the above problem, I introduce a sign adjustment algorithm that optimally matches the directions of sign and value distributions (i.e., when a sign is positive, a cluster tends to have higher feature values than others, and vice versa). First, for each feature, the algorithm computes Pearson's correlation coefficient r ($-1 \leq r \leq 1$) between all points' cluster memberships (i.e., 0: points are non-members, 1: points are members of the selected cluster) and their feature values. When r is closer to 1, members of the cluster more likely have a higher feature value than non-members. On the other hand,

closer to −1, higher possibility to have a lower feature value. Here we denote a set of r for all features as a vector **r**. Next, the algorithm computes a score of agreement s between correlation coefficients **r** and signed FCs **a** by taking their dot product (i.e., $s = \mathbf{r} \cdot \mathbf{a}$). s increases when an element of **r** and the corresponding element of **a** have the same signs, while s decreases when they have the opposite signs. Also, the magnitudes of elements of **r** and **a** can be considered as weights to decide how much s should increase or decrease. As a result, s becomes a higher positive value when each pair of elements of **r** and **a** has higher magnitudes of r and FC with the same signs. When $s < 0$, **r** and **a** disagree with each other; thus, the algorithm flips signs of all elements in **a**.

The result after applying the sign adjustment is shown in Fig. 3.4-c. Now, we can see that Cluster 1 and 2 have clearly different patterns in FCs. For example, while Cluster 1 has a strong positive FC at ④, Cluster 2 has strong positive FCs at ① and ③ and a strong negative FC at ④. Also, by referring to the HC views in Fig. 3.4-d, these differences well represent differences in the distributions of feature values. For instance, at ①, Cluster 2 tends to have high feature values but low feature values at ②–④, while Cluster 1 has high feature values at ④. With the sign-adjusted FCs, to understand the differences between clusters, the analyst can mainly focus on reviewing features that have much different contributions between clusters.

Note that Sec. 2.4.2.1 in Chapter 2 also presents a sign adjustment algorithm. The algorithm in Chapter 2 is to deal with the inconsistency of signs of FCs across clusters, while the algorithm in this chapter focuses on matching the directions of sign and value distributions for each cluster to ensure that a cluster has high feature values when its feature has a strong positive FC.

From the result shown in Fig. 3.2-c(left), now we know Cluster 1 tends to have high feature values around the middle of November but low feature values around the middle of April. Also, we can see that Cluster 2 tends to have the opposite patterns from Cluster 1.

3.3.4 Visualization of Parametric Mappings

The last analysis step is to understand the meaning of features obtained after the first DR of the two-step DR (i.e., columns in $\mathbf{Y}$). To support this task, the PM view (Fig. 3.2-e) visualizes a vector of parametric mapping $\mathbf{w}$ for each of the DR results as either line or bar chart, as similar to the FC view. Note that $\mathbf{w}$ is common across all points, and thus there is all lines or bars are colored in black. Also, using texts, at the top-right corner of each plot in the PM view, the quality of the first DR (e.g., *explained variance ratio* provided by many linear DR methods such as PCA and LDA) is informed. From Fig. 3.2-e(left), we can see that the feature values are generated with similar weights for all measures except for "Ozone". Therefore, we can interpret the feature values in Fig. 3.2-d are close to the mean of "PM2.5", "PM10", "CO", and "NO2".

3.3.5 Implementation

MulTiDR is developed as a web application. For the back-end of MulTiDR, including the algorithms described in Sec. 3.2, the sign adjustment algorithm in Sec. 3.3.3.1, and the generation of histogram information for the HC view, Python is used to integrate all the existing implementations, such as UMAP [170] and ccPCA [76]. The front-end visual interface is implemented with a combination of HTML5, JavaScript, D3 [33], and WebGL. WebSocket is used to communicate between the front-end and back-end.

3.4 Case Studies

The effectiveness of MulTiDR has been shown through the analysis of the air quality data in US [229]. Here we further analyze the same data from different aspects. Additionally, this section demonstrates three additional case studies, including analyses of a body sensing dataset, a dynamic social network, and supercomputer's hardware logs. For each study, each dataset is preprocessed to deal with its missing values or extract useful information for the analysis. All the processed datasets except for the supercomputer's hardware logs (due to its confidentiality) and parameters used for each DR result are available in the online site listed in Appendix A.

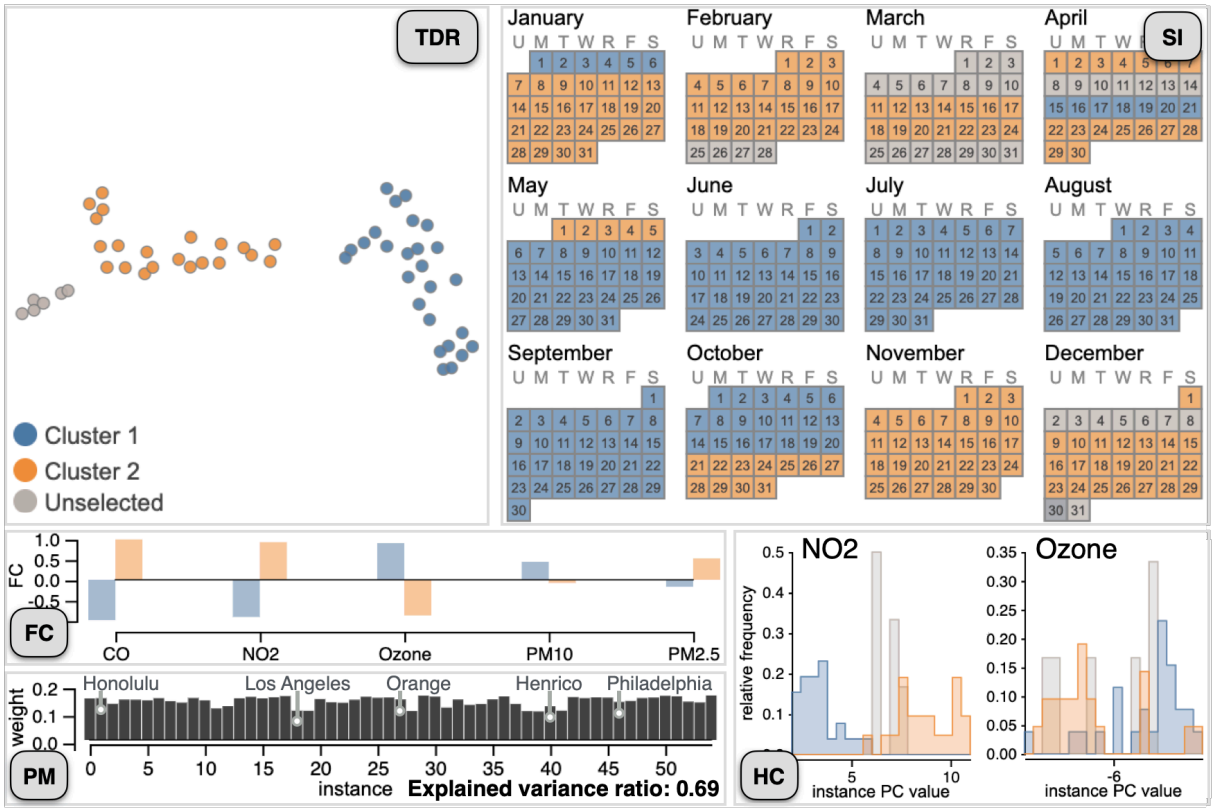


Figure 3.5: Case study 1. (TDR) shows the similarity of each week's air quality measures. (SI, FC, PM) are the SI, FC, PM views after selecting Clusters 1 and 2 in the TDR view, respectively. (HC) shows the HC views when selecting "NO2" and "Ozone" form (FC).

3.4.1 Study 1: Analysis of US Air Quality Data

Analysis of Weekly Patterns of Air Quality Measures. In Sec. 3.3, we have analyzed the clusters of instances (i.e., US counties) selected in Fig. 3.2-a1. This study analyzes the similarities of time points (weeks in 2018) based on their values of air quality measures. For this task, we apply the two-step DR using PCA along an instance mode (i.e., counties) and then UMAP along a variable mode (i.e., air quality measures). The generated results are shown in Fig. 3.5.

From the TDR view shown in Fig. 3.5-TDR, we select several clearly separated points (i.e., weeks) as clusters. For this data, the SI view provides a calendar-based visualization and indicates the corresponding weeks for each cluster (Fig. 3.5-SI). We notice that while the blue cluster generally relates to the weeks from May to the middle of October, the orange cluster consists of the weeks from the late fall to the early spring.

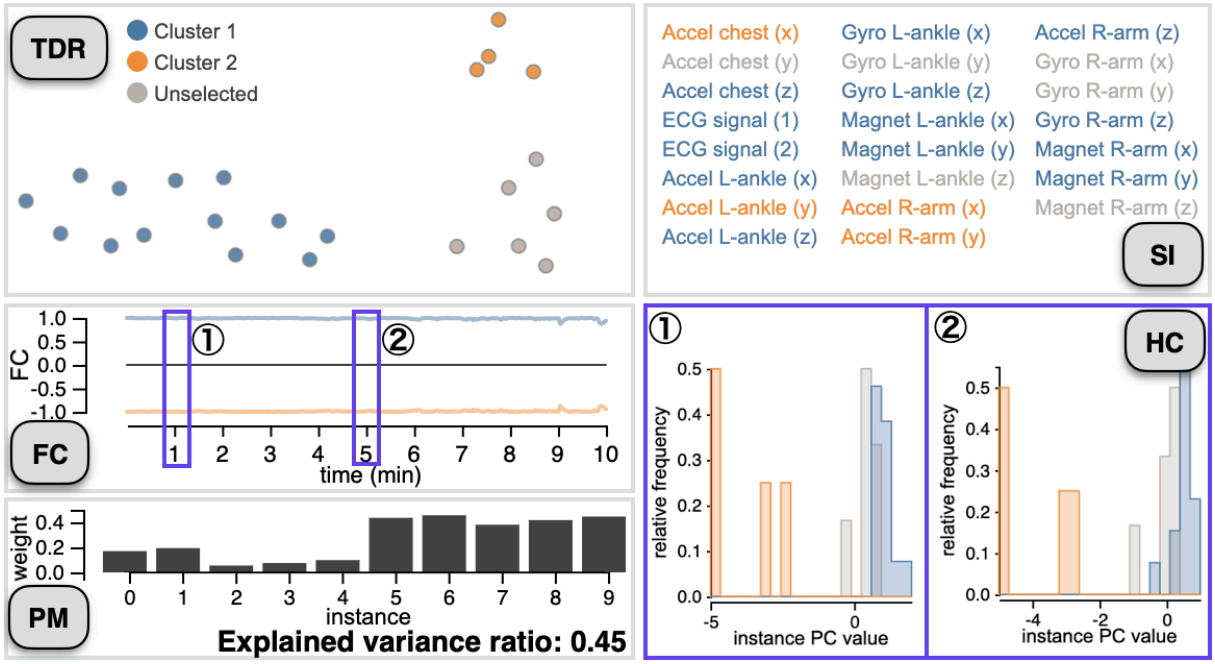


Figure 3.6: Case study 2-1. (TDR) shows similarities of physical activity measurements based on their temporal behaviors. (FC, PM) are the FC and PM views after selecting two clusters from (TDR). (HC) shows the HC views after selecting two different timestamps from (FC).

To understand the differences of each cluster, we refer to the FC view in Fig. 3.5-FC. The two clusters have quite different FCs, and thus seem to have different feature values as well. For example, in the histograms of "NO2", as shown in Fig. 3.5-HC, Clusters 1 (blue) and 2 (orange) tends to have low and high feature values when compared with others, respectively. On the other hand, in the histograms of "Ozone", we can see the opposite distributions. From the PM view (Fig. 3.5-PM), we can see that several counties (e.g., Honolulu) have slightly higher weights than others when generating the feature values.

In general, we can conclude that the air quality data has seasonal changes, such as "Ozone" has higher values around the summer (blue weeks) when compared with around the winter (orange weeks).

3.4.2 Study 2: Analysis of MHEALTH (Mobile Health) dataset

This case study analyzes the MHEALTH (Mobile HEALTH) [22, 23] dataset. This dataset consists of physical recordings of motion and vital signs for ten volunteers

while performing twelve physical activities. Sensors are placed on the subjects' chest, wrist, and ankle. The measurements taken from sensors include movement experienced by different body parts, such as acceleration with the magnitude for each of X-, Y- and Z-directions. The sensor modalities are recorded at a sampling rate of 50 Hz. The dataset contains points that represent bursts of highly active minutes.

Study 2-1: Categorization of Physical Activity Measurements. As shown in Fig. 3.6, from the TDR view (Fig. 3.6-TDR), where variables' (i.e., measurements') similarities are shown by applying PCA and UMAP along instance and time modes, respectively, we select Cluster 1–2 (blue and orange). The SI view in Fig. 3.6-SI lists all measurements related to each cluster as texts.

Afterward, we review the related information with the FC, PM and HC views (Fig. 3.6-FC, PM, HC). From Fig. 3.6-FC, we can see that, across time, blue and orange clusters have strong positive and negative FCs, respectively. To further investigate, we select several timestamps and review the corresponding histograms. Fig. 3.6-HC shows the histograms at two examples of the selected timestamps (① and ②). From the results shown in Fig. 3.6-HC, all the measures within each cluster tend to have close feature values (e.g., in ①, all orange points have the low feature values). By looking at Fig. 3.6-PM, PCA seems to generate the feature values with higher weights for Subjects 5–9 when compared to Subjects 0–4 with an explained variance of 0.45.

From the observations above, we can say that all the measures in Cluster 2, which includes the accelerations of the chest (X-direction), the left-ankle (Y-direction), and the right-arm (X- and Y-directions) have similar value distributions for each timestamp. The same applies to the measures in Cluster 1.

Study 2-2: Classification of Temporal Patterns among Subjects. Next, this study analyzes the similarities of time points in the duration of activity measurement (10 minutes). During the measurement, the subjects performed an activity set, including standing still, walking, running, etc. We apply the two-step DR using PCA along a variable mode and then UMAP along an instance mode. The generated results are shown in Fig. 3.7. From the TDR view shown in Fig. 3.7-TDR, we select multiple clearly

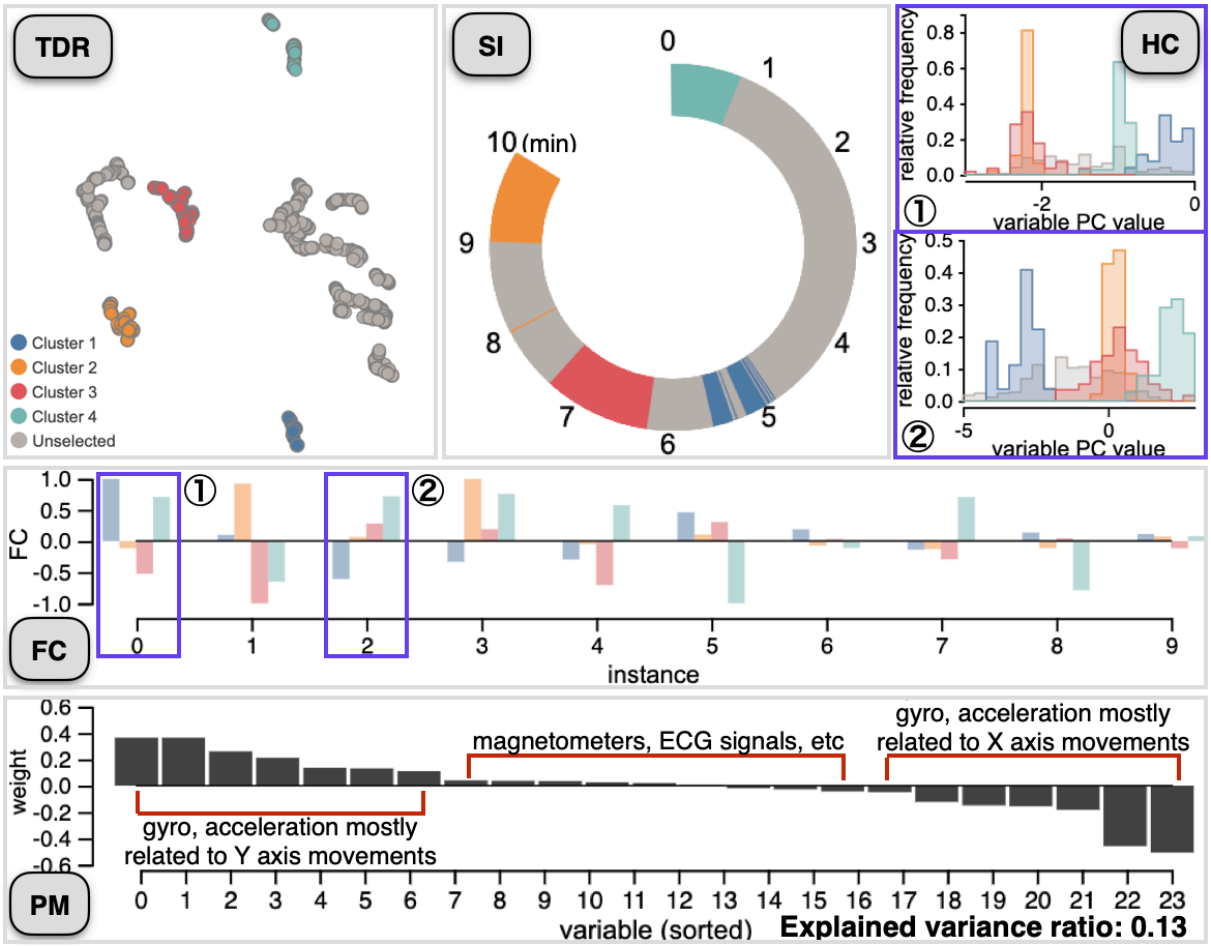


Figure 3.7: Case study 2-2. (TDR) shows similarities of timestamps based on subjects' activities. (SI) visualizes the corresponding time information with a circular layout. (FC, PM) are the FC and PM views after selecting four clusters from (TDR). (HC) shows the HC views after selecting two different instances from (FC).

separated clusters. In the SI view (Fig. 3.7-SI), we see that each cluster is gathered together with a range of about 1 minute. Since the subjects were asked to perform each activity with a duration of approximately 1 minute for collecting data, we can expect that each cluster well represents each of the activities.

To understand the differences of each cluster, we refer to the other views (Fig. 3.7-FC, HC, and PM). All clusters have quite different FCs in Fig. 3.7-FC. Also, by referring to the HC views, as the examples in Fig. 3.7-HC show, each subject tends to have quite different feature values. For example, while Subject 0 (annotated with ①) tends to have high feature values during the activity corresponding to Cluster 1 (blue), Subject

2 (annotated with ②) tends to have low feature values for Cluster 1. From the PM view, we understand that the feature values mainly relate to the measures of gyroscopes and accelerations but not magnetometers or ECG signals. More specifically, the measures related to Y-directions (e.g., "gyro R-forearm (y)", "gyro L-ankle (y)", "acceleration chest (y)") tend to have positive weights, while the measures of X-direction have negative weights.

Therefore, we can conclude that, in general, the two-step DR successfully separates time points related to a specific activity based on the differences of each activity in the measures of X- and Y-directions.

3.4.3 Study 3: Analysis of Dynamic Contact Networks

This study provides an analysis example of dynamic networks, using a dataset of contacts between high school students in Marseilles, France [72]. This dataset contains network links, which represent the students' face-to-face contacts collected with 20-second intervals for several days. Temporal snapshots are constructed from this dynamic network by aggregating contacts within a time window of 5–9 AM, 10 AM–12 PM, 1–3 PM, 4–6 PM, or after 6 PM for each day. This procedure generated 30 snapshots (i.e., networks) of 180 students (i.e., network nodes) with the mean of 193 contacts (i.e., network links). To extract features of network nodes, DeepGL is applied [194], a network representation learning method that produces features consisting of node attributes (e.g., gender), network centralities (e.g., degree centrality [179]), network measures (e.g., k-core number [179]), and those of statistical values of neighbors (e.g., the mean degree centrality of 1-hop neighbors). As a result, a tensor of $T = 30$, $N = 180$, and $D = 10$ is obtained.

Study 3-1: Categorization of Students Based on Node Features. Fig. 3.8-TDR shows a result generated by the two-step DR using PCA along a time mode and UMAP along a variable mode (i.e., the dots in the TDR view represent instances). The result contains five distinct clusters and we select four of them. The resultant visualizations are shown in Fig. 3.8-SI, HC, FC, and PM. Here, the SI view draws an overall network constructed using a time window of the entire measurement period. From Fig. 3.8-SI, we notice

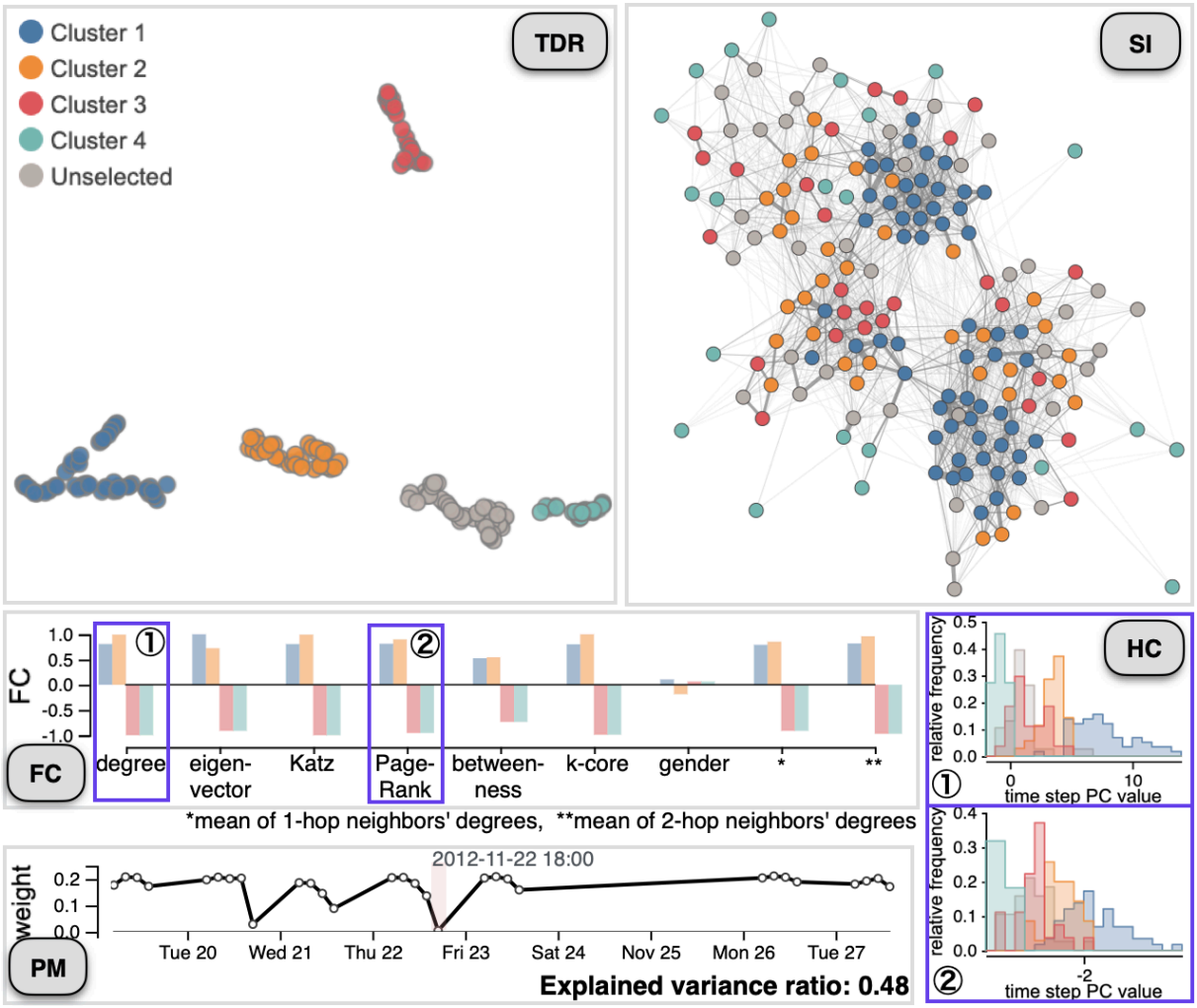


Figure 3.8: Case study 3-1. (TDR) shows similarities of the students based on their node features obtained with DeepGL [194]. (SI) draws a node-link diagram of the entire contact network. (FC, PM) are the FC and PM views after selecting four clusters from (TDR). (HC) shows the HC views after selecting two different node features from (FC).

that the blue nodes (i.e., students) in Cluster 1 can be seen in the strongly connected regions. In contrast, the teal nodes in Cluster 4 have only a small number of links to the others.

To further understand each cluster's characteristics, we review the FC view (Fig. 3.8-FC). We can see that, except for "gender", generally Clusters 1 and 2 have strong positive FCs, while Clusters 3 and 4 have strong negative FCs. In the HC views (Fig. 3.8-HC), which show the histograms of ① degree and ② PageRank, Clusters 1 and 2 tend to have higher values than Clusters 3 and 4. Especially, Cluster 1 has much higher values

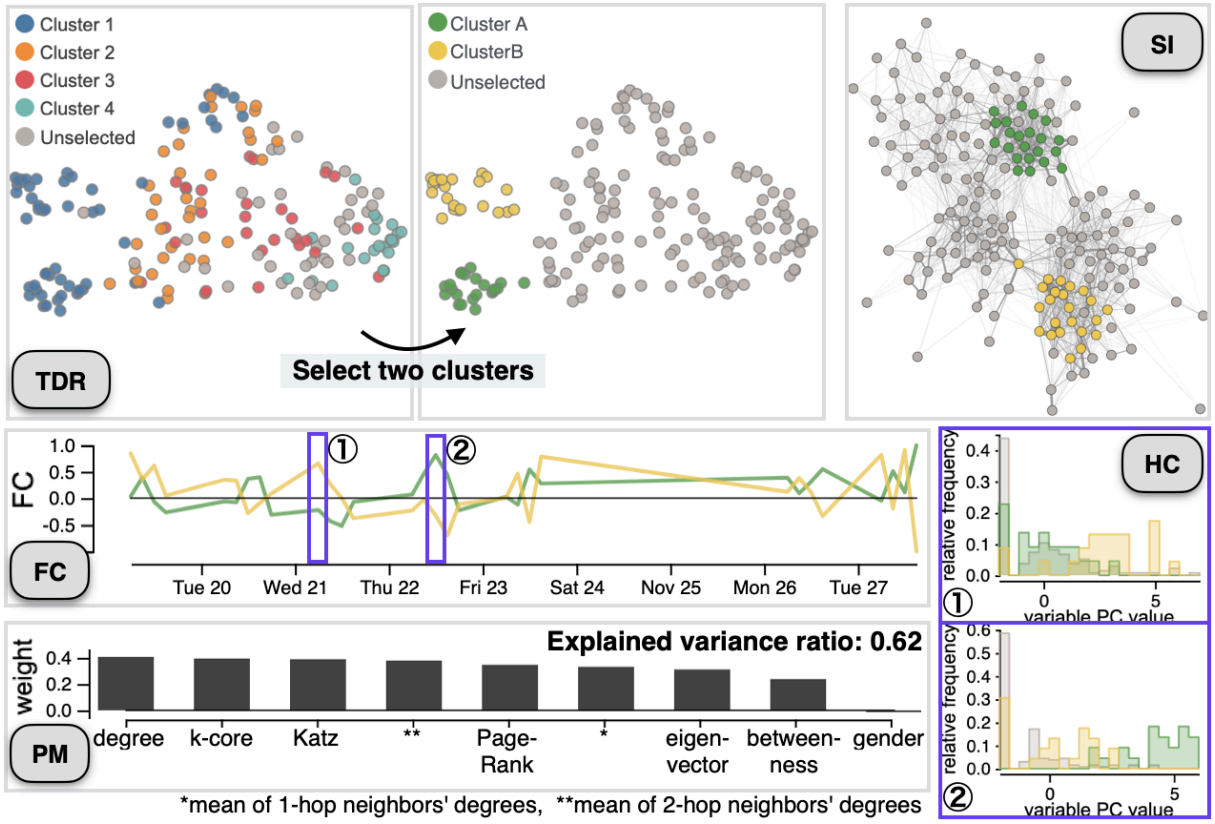


Figure 3.9: Case study 3-2. (TDR) shows similarities of the students based on their temporal behaviors. From the result at the left, where the colors correspond to the selection in Fig. 3.8-TDR, we select Clusters A and B at the right. (SI, FC, PM) are the SI, FC, and PM views after selecting the two clusters. (HC) shows the HC views after selecting two different timestamps from (FC).

than the others. From the PM view (Fig. 3.8-PM), we can see that the features in the FC view are generated by using large weights around mornings whereas evenings have close to zero weights (e.g., 6 PM on November 22nd, as highlighted).

With the above analysis, we can conclude that during school hours, the students in Cluster 1 played a central role in communications among students as they have high values for various network centralities.

Study 3-2: Categorization of Students Based on Temporal Communication Patterns. Together with the results in Study 3-1, we further review the instance similarities obtained by applying PCA along a variable mode and UMAP along a time mode. Fig. 3.9-TDR(left) shows the two-step DR result colored based on the selection in Fig. 3.8-TDR. We can see that each of the currently selected clusters is generally arranged from

left to the right. However, most students in Cluster 1 are separated into two distinct clusters in the far left. We select the two clusters as Clusters A (green) and B (yellow), as presented in Fig. 3.9-TDR(right). In the SI view (Fig. 3.9-SI), these two clusters are clearly separated into the two strongly connected regions at the top and bottom. The FC view (Fig. 3.9-SI) shows that the two clusters have different patterns in FCs across time. We select two clear peaks, ① (Wednesday morning) and ② (Thursday noon), to see their value distribution differences with the HC view (Fig. 3.9-HC). We can see that Clusters A and B tend to have high feature values at ② and ①, respectively. By looking at the PM view (Fig. 3.9-PM), the feature values represent the network centralities and measures but not the gender.

From the above observations, we can say that the students in Cluster 1 (the central role of the communications) can be further categorized into two different groups, Clusters A and B, which had active communications at the different time periods.

These insights found in Studies 3-1 and 3-2 are not revealed by the visual analytics performed on the same data by van den Elzen et al. [230]. Their approach is designed only for comparison of network structures of temporal snapshots (i.e., network-level comparison), and cannot find (dis)similarities of students (i.e., node-level comparison). van den Elzen et al.'s study showed that student communications stayed similar during the nighttime even on different days, while they had more variance in the network structure during the daytime. MulTiDR also implies this finding in Fig. 3.8-PM, where the nighttime has much lower weights due to their low variance. In addition, van den Elzen et al. found that the communication network had a distinct structure at each day and slowly changed across time of each day, which can be expected from the statistics calculated by the authors who collected this communication network data [72]. On the other hand, the analysis in this study using MulTiDR focuses on the categorization of students based on their roles and behaviors in the communications, resulting in the new insights into this network data.

3.4.4 Study 4: Analysis of Supercomputer Hardware Logs

This study analyzes hardware logs obtained from a supercomputer. Supercomputers are required to have high robustness and reliability to continuously run large-scale computations. Analyzing their hardware logs is fundamental to revealing and understanding abnormal hardware behaviors (e.g., extreme increases of CPU temperatures), which can lead to hardware failures or errors [95, 206].

Here, we specifically review the K computer's [172] hardware logs on January-12th, 2017. The logs were obtained from 864 compute racks, where 1,163 different measures (e.g., CPU temperatures, circuit voltages, and cooling fan spin speeds) are collected every 5 minutes (i.e., 1,440 timestamps in a day). Therefore, the logs can be represented as a $T \times N \times D$ tensor where $T = 1,440$, $N = 864$, and $D = 1,163$ (in total, more than 1.4 billion elements). This case study demonstrates how MulTiDR helps the analyst identify and characterize outliers from an extremely large-scale dataset.

Study 4-1: Identification and Characterization of Outlier Racks. As a first analysis, we identify racks that have unusual temporal behaviors. To achieve this, we apply the two-step DR with PCA along a variable mode and then UMAP along a time mode (i.e., the dots in the TDR view represent instances). The visualized DR result is shown in Fig. 3.10-TDR. We can see that while there is a large cluster that contains many racks (the gray points placed at the right side), some racks form small distinct clusters from the main cluster. We select three of these small clusters (Clusters 1–3) in Fig. 3.10-TDR. Because these outlier clusters could relate to a specific physical location (e.g., a parallel application is often allocated to run in a specified location), we refer to the SI view (Fig. 3.10-SI), where the physical coordinates of racks are visualized; however, these clusters seem not to fit such a case.

To understand the clusters' characteristics, we analyze the results with the FC, HC, and PM views (Fig. 3.10-FC, HC, PM). In Fig. 3.10-FC, we first see that, across time, Clusters 1, 2, and 3 generally have moderate, strong negative, and strong positive FCs, respectively. From Fig. 3.10-FC, we select Timestamp ① as a sample timestamp following this general pattern and two timestamps (② and ③) that have a unique shift

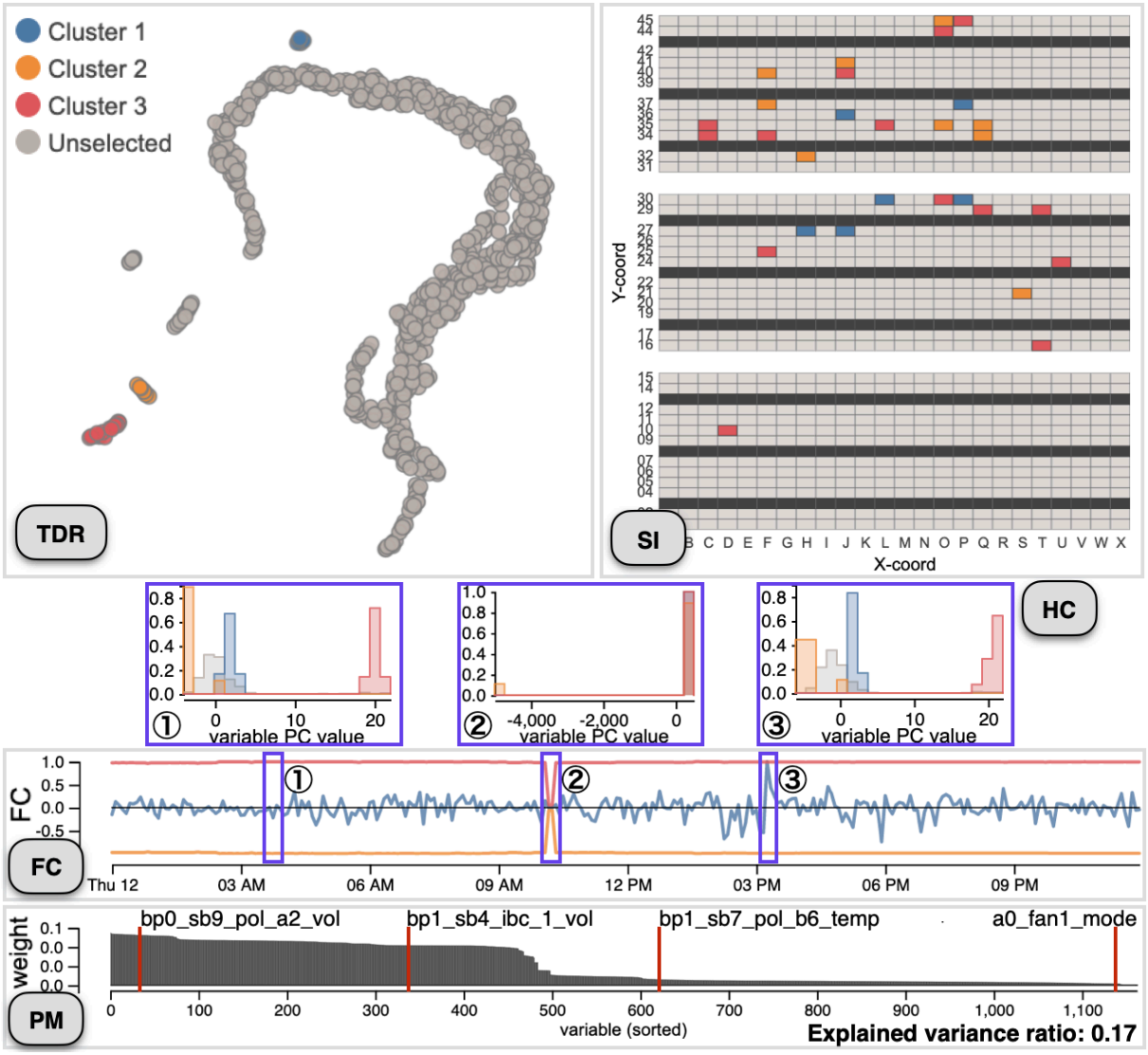


Figure 3.10: Case study 4-1. (TDR) shows similarities of racks based on their temporal behaviors. (SI) visualizes the racks' physical coordinates in the K computer. (FC, PM) are the FC and PM views after selecting three outliers from (TDR). (HC) shows the HC views after selecting three different timestamps from (FC).

of the FCs. By looking at the HC view of Timestamp ①, we observe that Cluster 3 (red) has much higher feature values than the others, while Clusters 1 (blue) and 2 (orange) have slightly higher and lower feature values than the main cluster (gray), respectively. However, at Timestamp ②, all the racks have similar feature values. At Timestamp ③, when compared with Timestamp ①, Cluster 1 has slightly fewer overlaps with the gray bins. Next, we review the PM view (Fig. 3.10-PM), where 1,163 measures'

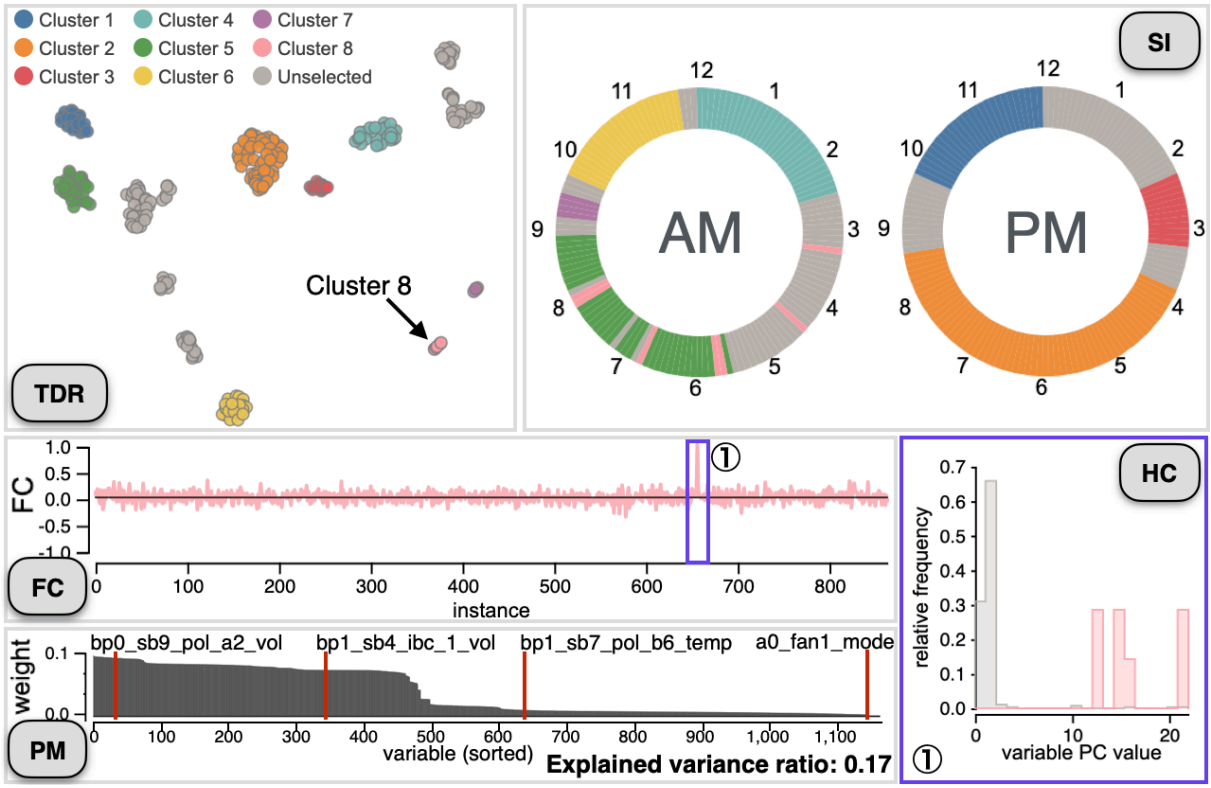


Figure 3.11: Case study 4-2. (TDR) shows similarities of timestamps based on behaviors of racks at the corresponding time. (SI) informs the selected timestamps with a clock-based visualization. More information of Cluster 8 is shown in the FC, PM, and HC views (FC, PM, HC).

weights are shown, and notice that only the first 500 measures have weights not close to zero. By showing these measures' names by hovering a mouse, we know that the first 500 measures are related to the voltages ("vol") but not the others, including the temperatures ("temp") and fan information ("fan_mode").

Therefore, we can conclude that, across time except for around 10 AM, the racks in Cluster 3 had extremely high voltages while Cluster 1 and 2 had slightly higher and lower voltages than most of the racks.

Study 4-2: Identification and Characterization of Outlier Timestamps. Next, we identify timestamps, at which racks had different behaviors from a usual state, by reviewing the two-step DR result generated by applying PCA along a variable mode and then UMAP along an instance mode (i.e., the dots in the TDR view represent timestamps). The visualized results are shown in Fig. 3.11.

From Fig. 3.11-TDR, we select several distinct timestamp clusters (Clusters 1–8).

In the SI view (Fig. 3.11-SI), where the corresponding timestamps are shown with a clock-based visualization, we can see most clusters relate to the specific time range (e.g., timestamps in Cluster 3 are seen from about 2 to 3 PM). We can consider that each of these clusters corresponds to the duration when performing an allocated job. However, we can see that Cluster 8 is separated in several short time ranges in AM. Since this pattern might relate to the abnormal behavior, we further review Cluster 8 by using the FC, PM, and HC views. From Fig. 3.11-FC, we can see that Cluster 8 has a strong positive FC for one instance (i.e., rack), as annotated with ①. By looking at the HC view, we can see that, for this instance, the timestamps belonging to Cluster 8 have a much higher feature value than the other timestamps (note: here the gray bins include all the timestamps except for those in Cluster 8). Since the parametric mapping is the same as Study 4-1, we can say that these feature values mainly represent multiple voltage measures. Therefore, Cluster 8 is considered as outlier timestamps by the two-step DR because one specific rack had extremely high voltages at the corresponding timestamps.

3.5 Qualitative Comparison

The two-step DR in MulTiDR employs data compression with DR to produce a matrix from a third-order tensor. Instead, as discussed in Sec. 3.2.1, we can simply apply tensor unfolding along one mode to generate a matrix and then perform DR on such a matrix (e.g., applying DR on a matrix of N rows and $(D \times T)$ columns to visualize instance similarities). Another option is computing statistical measures, such as mean values, when generating a matrix from a third-order tensor. MulTiDR contains this approach if we consider the computation of statistical measures as one of DR methods that generates a representative value. Here, we compare three different methods above and discuss the advantages of the two-step DR.

More specifically, we compare two different implementations of the two-step DR, (1) using PCA for the first DR and UMAP for the second DR (I call this method *PCA & UMAP*) and (2) using the mean computation for the first DR and UMAP for the second

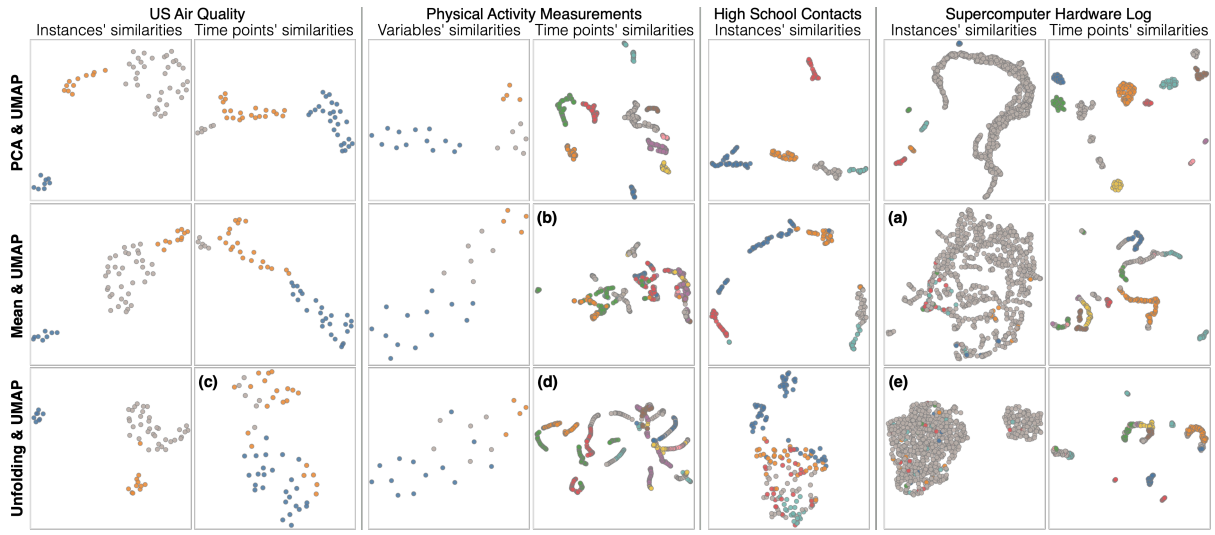


Figure 3.12: Visual comparison of the DR results. Colors represent clusters selected in the result of PCA & UMAP.

DR (*Mean & UMAP*), and (3) the unfolding approach (i.e., without the first DR step) using UMAP as a DR method (*Unfolding & UMAP*). We apply these methods to the datasets analyzed in Sec. 3.3 and 3.4.

Fig. 3.12 shows the DR results. Here, several distinct clusters are manually selected from the results of PCA & UMAP and then color-code the corresponding points in the other views based on the cluster information. In general, some of the findings described in Sec. 3.4 from the results of PCA & UMAP cannot be uncovered by either using Mean & UMAP or Unfolding & UMAP.

PCA & UMAP vs Mean & UMAP. Mean & UMAP generates similar results with PCA & UMAP when PCA & UMAP generates a projection mapping consisting of almost uniform weights (e.g., the results for the US air quality dataset); however, for the other cases, Mean & UMAP does not show several meaningful clusters and outliers or does not clearly discern them from the other points. A concrete example can be seen in the instances' similarities of the supercomputer hardware log in Fig. 3.12-a, where the result of Mean & UMAP mainly shows a single cluster and does not reveal the outlier clusters found with PCA & UMAP. Also, unlike PCA & UMAP, the result in Fig. 3.12-b does not provide a clear separation of time points that are related to different physical activities. When a target mode of the first DR has significant differences in variances

for each index (i.e., a variable, an instance, or a time point), PCA & UMAP can preserve more variety along the mode, and thus PCA & UMAP would produce more useful results. However, again, the two-step DR does not restrict a DR method used for the first DR and allows the analyst to select a preferable compression/feature selection method, including the mean computation, PCA, LDA, etc.

PCA & UMAP vs Unfolding & UMAP. Unfolding & UMAP has quite different results from the ones with PCA & UMAP and seems to fail to find several clusters and outliers. For example, in the time points' similarities of the US air quality dataset (Fig. 3.12-c), while PCA & UMAP shows the clusters that represent the seasonal air quality change (as described in Sec. 3.4.1), Unfolding & UMAP does not clearly display such clusters. Moreover, similar to Mean & UMAP, the result in Fig. 3.12-d does not clearly discern different physical activities. Also, Fig. 3.12-e does not uncover the outlier clusters seen in the result with PCA & UMAP. This limitation of Unfolding & UMAP relates to the fact that Unfolding & UMAP mixes two different modes together and, as a result, it cannot discover the patterns highly related to a specific mode. Another major drawback of Unfolding & UMAP is that it makes the characterization of clusters more difficult because of the complexity of features in the FC view, where each feature represents a mix of two modes (e.g., variables $\times$ instances), and the massiveness of the number of features (e.g., the supercomputer log dataset of $D = 1,163$ and $N = 864$ generates $D \times N = 1,004,832$ features).

3.6 Discussion

MulTiDR has been evaluated with the case studies and qualitative comparison. The qualitative comparison has demonstrated the strength of the two-step DR when compared with the other approaches. This section provides an additional discussion from different aspects.

Limitations of Visual Scalability. MulTiDR's visual interface overlays multiple charts in the FC view and the HC view to make comparison of different clusters' FCs and feature value distributions easier. However, when many clusters are selected (e.g., ten

clusters), these visualizations could cause too many overlaps and clutters. To deal with such a situation, we can provide a visual comparison using small multiples and allow the analyst to select either overlays or small multiples based on their preference.

Also, when each mode has many dimensions, it becomes difficult to grasp what kind of dimensions has high FCs and weights from the FC and PM views, respectively. This is especially problematic when x-axis of the FC or PM view represents a variable mode because it often consists of variables that have different types of measures (e.g., network routers' temperatures, voltages, sent, and received packets). For this issue, before visualizing the information of a variable mode, we can consider applying aggregation based on their similarities or available external information (e.g., a class of measures, such as physical loads, including temperatures and voltages, and network loads on routers, including sent and receive packets).

Limitations of the Two-Step DR. The two-step DR is mainly limited by the first DR step. Because this step compresses a target mode into 1D, when the mode has many dimensions (e.g., 1,000 variables in a variable mode), a large amount of information could be lost. However, at the same time, the analyst can check how much of the information is preserved by referring to a quality measure provided by each DR method, such as explained variance ratio in PCA and LDA. When the quality is extremely low (e.g., explained variance ratio is smaller than 0.01), the analyst can consider selecting a subset of dimensions for their analysis. In addition, to inform the second DR's quality, I plan to incorporate several model-agnostic quality measures [151, 234] and visualizations [74, 242] in the future.

Generality of the Two-Step DR. MulTiDR's back-end algorithms described in Sec. 3.2 are used to obtain and understand a low-dimensional representation of multivariate time-series data. However, these algorithms can be applied to other types of data that can be formed into a third-order tensor. For example, even when analyzing single-time-point multivariate data, the analyst may want to separate variables into two different modes, such as patients' demographics (e.g., ages) and their medical tests (e.g., blood pressures) for an analysis of medical datasets. In such a usage, MulTiDR's algorithms

can help the analyst avoid mixing the influences from two different types of variables on the DR result.

Additional Enhancement for Time-Series Analysis. This chapter has demonstrated the effectiveness of MulTiDR using PCA, UMAP, and ccPCA as the first DR, second DR, and CL methods, respectively. We also can use representation learning methods that focus on time-series analysis. For example, instead of PCA, when applying the first DR along a time mode, we are able to use functional PCA [239], which aims to extract representative temporal patterns, or use multivariate singular spectrum analysis (MSSA) [108], which is suitable to find outlier time points. Also, we can design a CL method that is similar to ccPCA by extending a contrastive version of MSSA [62]. Once it becomes available, we can replace ccPCA with such a method.

Visualizations also can be enhanced for time-series analysis. For example, to convey the temporal order of time points in the TDR view's scatterplots, we can couple the TDR view with the existing visualization methods described in Sec. 3.1.2, such as methods developed by Bach et al. [20] and van den Elzen et al. [230].

3.7 Summary

This chapter has introduced a visual analytics framework, MulTiDR, which enables us to derive and interpret low-dimensional representations of multivariate time-series data by employing a two-step DR and contrastive learning together with interactive visualization. As demonstrated with case studies, MulTiDR has abilities of identifying and characterizing clusters and outliers from complex datasets. Therefore, MulTiDR provides a new effective approach to demanding tasks of analyzing multivariate time-series data.

Chapter 4
Streaming Data Analysis

In many streaming data applications, including social media text-analysis [32], traffic flow monitoring [39], financial fraud detection [243], computer network screening [15,248], and assembly lines performance diagnostics [249], the data is often multi-dimensional. For these datasets, utilizing effective visualizations is crucial for performing timely analysis. However, applying dimensionality reduction (DR) to continuously updating dataset is not a trivial task due to the following challenges: (1) the computation time needs to keep up with the data-collection rate, (2) the viewer's mental map needs to be preserved, and (3) the potentially non-uniform number of dimensions for each data point needs to be handled.

The computational cost is the primary concern when using DR for streaming data. As new data keeps coming in, the time for calculating and updating positions of data points must be fast enough to keep the visualization up-to-date. This becomes particularly difficult when the number of data records and/or the number of dimensions is large.

Another challenge is how to preserve a viewer's mental map while continuously updating the visualization from DR results. Most of the well-known DR methods, such as PCA [125], MDS [222], and t-SNE [233], originally designed their approach for a static setting. As a result, each time DR is directly applied to streaming data, the projected data points' positions could look drastically different from the positions obtained at the previous time step. This would, therefore, easily interrupt the viewer's analysis process and be too difficult to maintain a mental map.

The last, and perhaps, the most challenging problem in employing DR for streaming data analysis is caused by the non-uniform number of dimensions. In scenarios where

an analysis is based on multiple data sources, some of the data points could have missing features if those features have not been recorded yet. For example, when monitoring a product assembly line, we may measure the time it takes for products to pass through the work stations that assemble the products, and then use the measured time as features for further analysis. However, at any given time point, some products might have already been assembled (i.e., the full set of features is collected), while the others are still being processed at one of the work stations (i.e., at least one of the features is missing). In such cases, ordinary DR methods are not directly applicable as they cannot handle data records with a variant number of dimensions.

To address the challenges mentioned above, this chapter introduces a method for visualizing DR results for streaming data. While there are many different types of DR methods, this chapter focuses on PCA because of its popularity for visualization [198]. To reduce the amount of computation needed at each iteration, incremental PCA [193] is employed, which calculates the new results by using the results obtained from the previous step as a base and then updates them according to the newly added information. However, the traditional incremental PCA will still rearrange data points' positions at each successive time point. We would still then run into the problem of disturbing the viewer's mental map. In my method, I, therefore, apply a geometric transformation, specifically the Procrustes transformation [31], to make the transition of each data point's position easier to follow. In addition, the animated transition of data points between subsequent time points is used to reduce the viewer's cognitive load.

To handle data records with a non-uniform number of dimensions, I introduce a position estimation method. It estimates where the positions of data points with an incomplete number of features would be in the PCA result of the other data points which would have the full set of features. I also provide a mechanism to measure the uncertainty introduced by the estimation method. By visually presenting uncertainty information, viewers can assess the trustworthiness of the displayed result. This will help them make better decisions as well as adjust their hypotheses during the

exploration and observation stages of the visualization.

To present the efficiency of the introduced methods, performance testing is conducted. The result shows that the calculation time of the methods meets the requirement of supporting real-time applications. Furthermore, a prototype system integrating the introduced methods is developed to demonstrate their effectiveness with in-depth analysis of real-world datasets. Two case studies showcase how the method can be used for visually detecting potential anomalies and finding forming clusters from streaming data.

4.1 Related Work

This section discusses the relevant works in streaming data visualization and dimensionality reduction methods.

4.1.1 Streaming Data Visualization

Visualizing streaming data for effective analytics is an important research topic. Dasgupta et al. [57] provided a comprehensive survey of streaming data visualization and its challenges. One main challenge is that the visualization needs to be constantly updated with incoming data. This introduces two major concerns: (1) cognitive load and (2) computational cost.

Krstajic and Keim [140] summarized the problems related to the cognitive load. They compared the occurring changes from streaming data in well-known visualizations, such as scatterplots and streamgraphs, and summarized the potential loss of context from its effects. For instance, if a new data value is outside of the current axis range(s) of a scatterplot, we would need to decide whether to update the axis range(s) or not. In the case that we decide to make an update, the viewer's mental map then may be lost at the same time. On the contrary, if no update is applied, we run into the issue of information loss.

As for overcoming the issue of computational cost, various incremental methods have been introduced [49, 161, 218]. Tanahashi et al. [218] extended the storyline generation algorithm for streaming data. To reduce both cognitive load and calculation

cost, they utilized the previous steps' storylines to decide the new data points' layout. Crnovrsanin et al. [49] developed the incremental graph layout based on FM^3 [98]—a fast force-directed layout algorithm. To achieve faster calculation, they applied a GPU acceleration to FM^3. Also, they designed the initialization, merging, and refinement steps of the graph layout to maintain the viewer's mental map. In addition, they used animation to provide smooth transitions from the previous to the current graph layout. To support text stream analysis, Liu et al. [161] introduced a streaming tree cut algorithm to detect the incoming topics in time. Also, their streamgraph visualization with a river metaphor can depict topics at different level-of-details to explore both global patterns from the accumulated results and local details from the new topics.

Gansner et al. [84] also worked on visualizing streaming text. They visualized topic relationships from the text data using a node-link diagram with a map metaphor [83], which can show clusters of texts clearly. To keep the viewer's mental map when updating the graph layout, they utilized MDS [222] as a graph layout algorithm. When calculating the new positions with MDS, their algorithm uses the previous nodes' positions as the initial positions to obtain a result that better maintains the mental map. Also, the algorithm applies the Procrustes transformation [31] to reduce the positional changes caused by rotation and scaling between the successive MDS results. Similarly, Cheng et al. [43] used MDS for showing an overview of similarities between temporal behaviors in streaming multivariate data. In addition, they introduced the concept of sliding MDS, which visualizes temporal changes in the similarities between selected points as line paths.

The works by Gansner et al. [84] and Cheng et al. [43] are closely related to the work in this chapter, in which we all utilize DR methods to visualize the relationships between the streaming data points. Both of the existing works employed MDS as their DR methods. However, using MDS makes it difficult to incrementally update node positions based on new data points, as it requires a recalculation of all node positions every time a new data point appears (e.g., MDS needs several seconds to project 1,000 data points [251]). This scalability issue is particularly prominent when handling a

large data size or if there is a frequent need to update the data. My approach solves this scalability issue by using an incremental DR method. My approach also takes further steps to preserve the mental map by (1) minimizing the changes between current and incoming layouts and (2) using animation to smoothen the transition between the layouts.

4.1.2 Dimensionality Reduction (DR) Methods

As described in Sec. 4.1.1, one of the purposes of applying DR is to summarize time-series and/or multivariate data, including streaming data [43]. For example, to identify anomalies from sensor networks, Steiger et al. [211] produced an overview of the sensors' behaviors. They used time-series similarity measures and then plotted the similarities with MDS. This method focuses on the comparison of each sensor's value over time. In contrast, some visualizations calculate the similarity of the state of all data points at each time point, and then show their temporal differences. For example, Bach et al. [20] visualized the similarity of multivariate data between each time point by using MDS. van den Elzen et al. [230] also applied similar methods. Rauber et al. [188] developed Dynamic t-SNE to compare the DR result for each time step. Dynamic t-SNE offers a controllable trade-off between how much temporal coherence is strictly kept and how much neighborhood relationships are precisely preserved in the t-SNE results. Jäckle et al. [119] introduced Temporal MDS Plots. They used x- and y-coordinates to represent time and MDS similarity, respectively. Also, they reduced the unnecessary rotation in the MDS results by flipping the y-coordinates based on their positions in the previous time point.

Even though the stated existing works used DR methods for summarizing time-series data and addressed the issue of preserving a mental map, they still run into the issue of dealing with new data points due to the calculation cost. This issue should be addressed for streaming data visualization. How to incorporate new data points to the existing result is one of the open problems in DR [213]. Incremental DR methods have been developed to reduce the computation cost at each time point by updating the result incrementally. For example, methods like incremental PCA [182, 193, 244], incremental

Isomap [147], and incremental local linear embedding (LLE) [138] are categorized as such.

In progressive visual analytics [176, 212, 228], researchers have started to apply incremental DR methods. The main idea of progressive visual analytics is to provide useful intermediate results within a reasonable latency when the computational cost for an entire calculation is too high. Being able to produce usable results with a latency restriction is a common requirement for streaming data visualizations. For instance, Pezzotti et al. [185] developed Approximated t-SNE (A-tSNE). Compared with t-SNE, A-tSNE stores the neighborhood information for each data point and only utilizes this information to refine the layout. Therefore, updating the layout in A-tSNE can work on each data point and its neighbors. By utilizing this characteristic, they also achieved an incremental update of the layout when adding or deleting points. While A-tSNE has been developed for progressive visual analytics, this is useful for streaming data visualization as well, as shown in their case study. However, A-tSNE does not consider the mental-map preservation, and the added or deleted points would drastically affect the other data points' positions. Turkay et al. [226] used incremental PCA [193] in their system to generate an overview of multidimensional data within a second. They also employed an animated transition since the (incremental) PCA generates arbitrary rotations and flips in the plotted results at each iteration. The animated transition acts as a cognitive support that helps the user understand the incrementally updated PCA results.

The two works [185, 226] are the most related works to the work in this chapter. However, I approach a new problem where the streaming data has a different length of dimensions between each data point. In addition, when compared with the incremental PCA, I improve the incremental PCA in terms of preserving a mental map by using both position adjustment and animated transitions together.

4.2 Methodology

As mentioned, our goals are to effectively manage computational costs, preserve the viewer's mental map, and cope with data records with different numbers of dimensions. To meet these goals, several design considerations are made for extending an existing incremental DR method. The resulting methodology is presented in this section. To better illustrate the work, the online supplementary materials (Appendix A) provide animations corresponding to Fig. 4.1, 4.3, and 4.5. The source code for a major portion of the methods is also available online.

4.2.1 Incremental PCA

Incremental DR methods incrementally update the lower-dimensional representations as new data points arrive [213]. Because the update only considers a small subset of the entire dataset, both computational complexity and memory usage can be reduced. Incremental PCA [182,193,244] is employed because of PCA's popularity in the visualization community [198].

Among incremental PCA algorithms, the model by Ross et al. [193], an extension of the Sequential Karhunen-Loeve algorithm [154], is selected. To apply the model, several parameters need to be pre-determined: d, the number of dimensions that a data point has, n, the number of data points processed so far, m, the number of accumulated new data points for the next update (the model requires $m \geq 2$), and k, the number of principal components to use. One of the main advantages of utilizing this model is its relatively low computation and space complexity. The time and space complexities of an ordinary PCA [125] are $O(d^2(n+m)+d^3)$ and $O(d^2)$, respectively. In contrast, the model by Ross et al. reduces the time and space complexity to $O(dm^2)$ and $O(d(k+m))$, respectively. This is because, based on only the partial singular value decomposition (SVD) with m new data points, the model incrementally updates the SVD for all data points, which is required for PCA. Because we usually have a fairly small m value in streaming data applications, the computational cost can be scaled down significantly.

There are other benefits in applying Ross et al.'s model. Unlike other incremental PCA methods (e.g., [100,154]), this model constantly updates the sample mean, which is

subsequently used for updating the eigenbasis of PCA. As a result, utilizing the model does not require setting up a learning phase, which addresses two common issues in handling streaming data: (1) we do not need to wait until a certain amount of data is accumulated to perform an update; (2) we always have an updated sample mean for incorporating new incoming data.

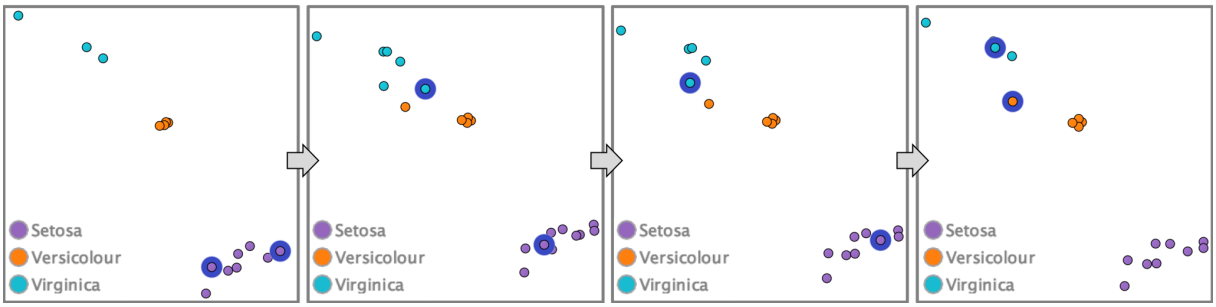
Furthermore, in the model, we can set a "forgetting factor", denoted f, which provides a way to reduce the contributions of past observations (existing data points) to the latest result. The value of f ranges from 0 to 1, where $f = 1$ means no past results will be forgotten. Whereas, when $f < 1$, the contributions of past observations are gradually decreased as new data points are obtained. The effective size of the observation history (the number of observations that will affect the PCA result) equals to $m/(1-f)$ [193]. For example, when $f = 0.998$ and $m = 2$, only the most recent $1,000$ observations are effective. By utilizing f, we can support both incremental addition of new data points and incremental deletion of past observations. Once the number of observations reaches the effective size, the effects of the past observations on the PCA calculation are ignored. As a result, we can choose to either keep or delete the past observations based on the user's need.

In summary, the model by Ross et al. is selected because of the following reasons: (1) the computational efficiency to incrementally update principal components ($O(dm^2)$ time complexity), (2) the capability to update the sample mean during the incremental updates, and (3) the flexibility to change the contributions from past observations. Especially, the last two are this model's unique strengths when compared with other models of incremental PCA. While the sample mean update enables to generate principal components closer to the exact solutions produced by ordinary PCA, the forgetting factor allows analysts to adjust the model based on their analysis focus (e.g., considering only recent 1,000 observations to compute principal components).

4.2.2 Preserving the Viewer's Mental Map

The results directly derived from the incremental PCA would have an arbitrary rotation and/or flipping of data points at subsequent time steps. Fig. 4.1-a shows an example

(a) Without the geometric transformation



(b) With the geometric transformation

Figure 4.1: Comparison of the incremental PCA results for the Iris dataset (a) without and (b) with the geometric transformation. The point colors represent the Iris species. For each step, two points, highlighted in blue, are added to the result. In (a), a noticeable rotation and flipping can be seen. In (b), the plotted result is stable across all steps.

demonstrating this issue using the Iris dataset [12, 70]. If this issue is not handled properly, it is difficult for the viewer to follow the updates in the visualization as the mental map can easily get lost during the analysis. My solution is to minimize the moving distance of the same set of data points between two subsequent time steps by applying a geometric transformation and then using animations for smoother transitions.

The PCA's flipping issue is known as the "sign ambiguity" problem and some possible solutions for visualizations have been proposed by Bro et al. [35], Jeong et al. [121], and Turkay et al. [226]. However, these methods do not consider the issue of arbitrary rotation of data points. To address both the flipping and arbitrary rotation, the Procrustes transformation [8, 92, 201] is applied. The Procrustes transformation is used to find the best overlap between two sets of positions (i.e., the previous and current PCA results in our case) by using only translation, uniform scaling, rotation, reflection,

or a combination of these transformations. The objective function to find the geometric transformation for the best overlap can be written as:

$$\text{Minimize } \|c(\mathbf{P'} + \mathbf{v}\boldsymbol{\tau}^\mathsf{T})\mathbf{R} - \mathbf{P}\|^2 \tag{4.1}$$

where $\mathbf{P}$ and $\mathbf{P'}$ are $(n \times k)$ matrices that contain the first k principal component values of n data points for the previous and current PCA results, respectively. n is the number of data points found in both the previous and current PCA results. $\boldsymbol{\tau}$ is a $(k \times 1)$ vector which translates data points of $\mathbf{P'}$ with $\mathbf{v}$, while $\mathbf{v} = (1\ 1\ \ldots 1)^\mathsf{T}$ is a $(n \times 1)$ vector. c represents the uniform scale factor ($c \in \mathbb{R}$). $\mathbf{R}$ is a $(k \times k)$ orthogonal rotation matrix, which handles rotation and reflection.

The Procrustes transformation starts by translating $\mathbf{P'}$ so that the centroid of $\mathbf{P'}$ is placed at the centroid of $\mathbf{P}$. Let $\bar{\mathbf{p}}$ and $\bar{\mathbf{p}}'$ be $(k \times 1)$ vectors which represent the centroids of $\mathbf{P}$ and $\mathbf{P'}$, respectively. We can compute the translation vector $\boldsymbol{\tau} = \bar{\mathbf{p}} - \bar{\mathbf{p}}'$. Now, $(\mathbf{P'} + \mathbf{v}\boldsymbol{\tau}^\mathsf{T})$ represents the translated result. The next step is scaling $(\mathbf{P'} + \mathbf{v}\boldsymbol{\tau}^\mathsf{T})$ to eliminate the scaling differences from $\mathbf{P}$. This can be achieved by matching the root mean square distances of $\mathbf{P}$ and $(\mathbf{P'} + \mathbf{v}\boldsymbol{\tau}^\mathsf{T})$ from the centroid of $\mathbf{P}$. This scaling factor c can be calculated as $c = \|\mathbf{P} - \mathbf{v}\bar{\mathbf{p}}^\mathsf{T}\| / \|\mathbf{P'} - \mathbf{v}\bar{\mathbf{p}}'^\mathsf{T}\|$. Lastly, the Procrustes transformation computes $\mathbf{R}$ for optimal rotation and reflection. To obtain $\mathbf{R}$, singular-value decomposition (SVD) is performed on $c\mathbf{P}^\mathsf{T}(\mathbf{P'} + \mathbf{v}\boldsymbol{\tau}^\mathsf{T})$ (i.e., $c\mathbf{P}^\mathsf{T}(\mathbf{P'} + \mathbf{v}\boldsymbol{\tau}^\mathsf{T}) = \mathbf{U}\Sigma\mathbf{V}^\mathsf{T}$). Then, $\mathbf{R} = \mathbf{V}\mathbf{U}^\mathsf{T}$. Refer to the articles about the Procrustes transformation [8, 92, 201] for more information on why this $\mathbf{R}$ provides the optimal rotation and reflection.

Now with c, $\boldsymbol{\tau}$, and $\mathbf{R}$, we can transform the data points in the current PCA result to minimize their moving distance from the previous result. Fig. 4.1 shows a comparison between results with and without applying the transformation. We can see that the transformation reduces unnecessary changes across the time points. Note that the time complexity of the Procrustes transformation is $O(k^2 n + k^3)$. For visualization purpose, usually $k \leq 3$, and thus, this transformation is fast enough to handle streaming data.

Furthermore, the change of the data points is animated to maintain the coherence between each subsequent step. This animation utilizes the staged transitions from Bach et al. [19], which was originally developed for visualizing dynamic node-link diagrams.

The transitions consist of three stages: (1) fading-out the data points that need to be removed; (2) moving the remaining data points from their previous positions to their new positions; (3) fading-in the new incoming data points.

4.2.3 Position Estimation for Dealing with a Non-uniform Number of Dimensions

When the streaming data contains data points with a non-uniform number of dimensions, ordinary incremental PCA cannot be directly used. This subsection first describes the problem of applying incremental PCA for such a case and then presents an algorithm that addresses the issue.

Let D be the complete number of dimensions (or features) that each data point can contain. From the data stream, n past data points have already gathered the information from all dimensions ($d = D$). On the other hand, some m new data points could have an incomplete l number of dimensions, ranging from anywhere between $d = 1$ and $d = D$. Consider the following example: in an online transaction stream, if we assume that there are D steps to reach to the purchase checkout, we have stored the history of n users' time spent at each step. However, m new users just finished the l-th step ($1 \leq l \leq D$) and, thus, we only have access to their time information for the first l steps.

If we want to compare the m new data points to the n existing data points using a DR method, one common method is to fill in the unknown values with a derived value (e.g., the mean or median value from the n data points). Another alternative is to apply DR only to the first l dimensions. Each approach has its limitation in streaming data applications. The first method does not capture the characteristics of the data well (e.g., correlations) [64], while the second method requires a re-calculation of the PCA every time the value of l changes.

Fig. 4.2 shows the relationship between the data points and the results after applying the incremental PCA for each different number of dimensions. When m new points have l dimensions, we can obtain the PCA results up to the l dimensions. Because $l \leq D$, we can only apply PCA to the $(n + m)$ data points using l dimensions (the area within the orange outline in Fig. 4.2-a). Alternatively, if we want to apply PCA using

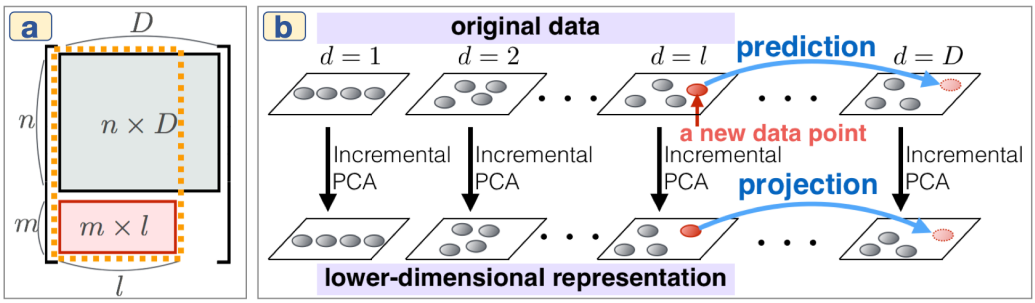


Figure 4.2: A relationship between variant dimensions and the incremental PCA results. (a) shows the data shape. While n stored data points have $d = D$ (the gray area), m new data points have l dimensions (the red area). (b) shows the relationships between the data and its lower-dimensional representation generated by the incremental PCA. Gray and red points represent the stored and new data points, respectively. The PCA can be applied to only the grey area or the area within the orange border in (a). Thus, we need to apply a prediction or projection to obtain the lower-dimensional representation for all $(n + m)$ data points where $d = D$.

the full dimension (i.e., $d = D$), we can only do so with the n data points (the gray area in Fig. 4.2-a).

There are two possible solutions for employing PCA to obtain the lower-dimensional representation for D dimensions with the $(n + m)$ data points, as indicated with the blue arrows in Fig. 4.2-b. The first method is to predict the values for the rest of the dimensions by using some machine learning or estimation methods [163] (e.g., linear regression). Then, we can apply incremental PCA to all $(n + m)$ data points. The second method is to project the m data points' positions from the PCA result of $d = l$ onto the PCA result of $d = D$. Compared to the first method, the second method executes in a simpler manner as it does not require choosing a proper model for a specific dataset, tuning the model used for a prediction-based method, or having an excessive computational cost.

I, therefore, use the second method and introduce a position estimation method. This method estimates where the positions of the new data points would be in the PCA result of $d = D$ by utilizing the distances between the new data points and the existing points (which already have the full dimension information $d = D$) in the PCA result of $d = l$. The estimation method proceeds in the following manner: first, the incremental PCA is applied for $d = l$; then, the positions of m new data points are projected to the PCA result of $d = D$, such that the projection maximally preserves

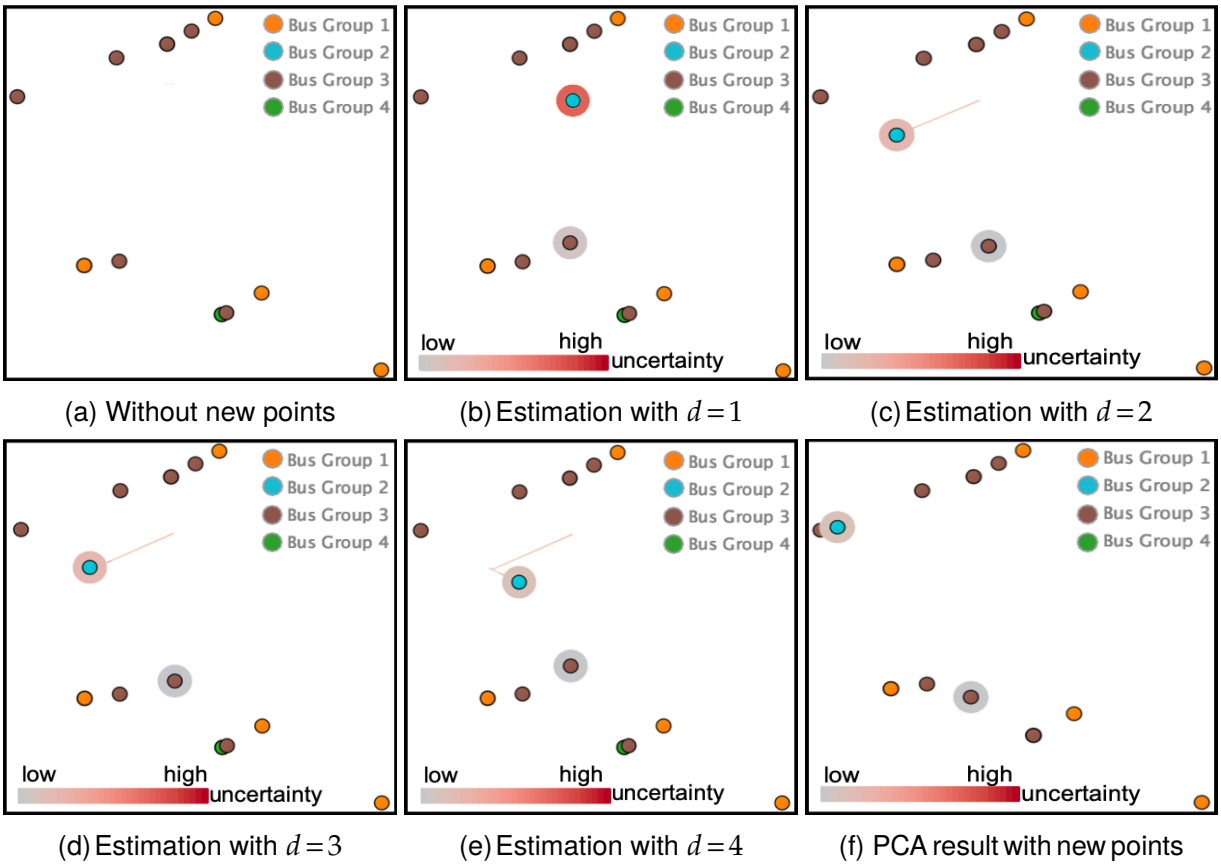
the distance relationships between the new and existing data points in the PCA result of $d = l$. This idea is based on the assumption that a new data point will likely have a similar relationship with the other data points in the remaining dimensions. The objective function for this optimization problem can be written as:

$$\underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{n} \left(s_{ui} - \alpha s'_{ui}\right)^2 = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{n} \left(s_{ui} - \alpha \left\|\mathbf{x} - \mathbf{q}_i\right\|\right)^2 \tag{4.2}$$

where θ consists of the parameters of α and $\mathbf{x}$ ($\alpha \in \mathbb{R}$, $\mathbf{x} \in \mathbb{R}^2$). s_{ui} and s'_{ui} are the distances from a new data point u to the i-th existing data point in the PCA results of l and D dimensions, respectively. $\mathbf{q}_i$ is the position of the i-th existing data point in the PCA result of $d = D$. $\mathbf{x}$ represents the estimated position of the new data point u in the PCA result of $d = D$ using this objective function. α is used for eliminating the scaling difference between each PCA result. The idea of adding data points to the DR result based on the distance relationships with the existing data points is similar to pivot-based MDS algorithms [126, 173] which target on reducing the computational cost.

A gradient descent algorithm [195] is applied to find the parameters θ in Eq. 4.2. Specifically, Adadelta [257] is used as this model can automatically adapt the learning rate for each parameter without providing a default value. After obtaining θ, the new data point u at the position $\mathbf{x}$ are placed in the PCA result of $d = D$. Since there are m new data points, this calculation is applied for each new point. Note that α may be a different value for each new point. I chose to apply Eq. 4.2 to each new point separately rather than finding the best common α for all new points, as the latter requires much more computations.

Once the new data points obtain the values of the additional dimensions (e.g., changing from l to $l+1$), the positions of the new data points will be updated by applying this method incrementally. Fig. 4.3 shows an example of the ongoing updates from the position estimation results. Same as Sec. 4.2.2, the transitions of the new points' positions are shown with the staged transitions.



(a) Without new points (b) Estimation with $d=1$ (c) Estimation with $d=2$

(d) Estimation with $d=3$ (e) Estimation with $d=4$ (f) PCA result with new points

Figure 4.3: Visualizations with the position estimation method. The bus transportation dataset from [223] is used for this visualization. Each point represents one bus and each color represents a bus group defined by "Block ID" in [223]. The time duration between each bus stop is used as a value for each dimension. Each bus passes through five stops and has four time durations ($D = 4$). The position transitions of two new buses are shown from (a) to (f). In (a), the PCA result of $d = D$ are shown without plotting the new buses. From (b) to (e), the two buses are plotted with the position estimation method where $d = 1$ to $d = D$. Then, the PCA result is updated in (f). The outer-ring color represents the uncertainty as described in Sec. 4.2.4. From (c) to (e), paths of the two buses are also visualized with the corresponding uncertainty colors. We can see that a bus with higher uncertainty in the top-left moves farther away from the final result in (f).

4.2.4 Visualizing Uncertainty of the Position Estimation

The position estimation method introduces two uncertainties. Both uncertainties represent how inaccurately the new point is projected onto the PCA result of $d = D$. A data point with higher uncertainty has a higher chance of moving drastically until its position is updated again with the next incremental PCA calculation (when $d = D$).

The first uncertainty is derived from the optimization using Eq. 4.2. The cost remaining after the optimization can indicate how the distance between each pair of

data points in the PCA result of $d = l$ is different from the one derived from the PCA result of $d = D$. This uncertainty is calculated in a range from 0 to 1. Similar with the "strain" in the classical MDS [222], the uncertainty U_{l_u} ($0 \leq U_{l_u} \leq 1$) for the new data point u with l dimensions can be calculated with:

$$U_{l_u} = \left(\frac{\sum_{i=1}^{n} \left(s_{ui} - \alpha s'_{ui} \right)^2}{\sum_{i=1}^{n} s_{ui}^2} \right)^{1/2} \tag{4.3}$$

The second uncertainty comes from the fact that a new data point does not have the values for all of the D dimensions when the position estimation method is applied (the new point has only l dimensions). Calculation of this uncertainty utilizes the principal component (PC) loading derived from the PCA. The PC loading represents the correlation between the original variables and the PCs. This can indicate how much variance each dimension contributes to each PC. The PC loading (w_{ij}) of j-th dimension to the i-th PC can be written down as: $w_{ij} = \sqrt{\lambda_i} h_{ij}$ where λ_i is the eigenvalue for the i-th PC and h_{ij} is the j-th element of the eigenvector $\mathbf{h}_i$ which corresponds to λ_i.

By using w_{ij}, the uncertainty V_l ($0 \leq V_l \leq 1$) for the new data points with l dimensions can be written down as:

$$V_l = 1 - \frac{1}{k} \sum_{i=1}^{k} \left(\frac{\sum_{j=1}^{j=l} |w_{ij}|}{\sum_{j=1}^{j=D} |w_{ij}|} \right) \tag{4.4}$$

Here, $\sum_{j=1}^{j=l} |w_{ij}| / \sum_{j=1}^{j=D} |w_{ij}|$ is the proportion of the sum of PC loading when we have l dimensions to the sum of PC loading for all dimensions. This means how much information of the i-th PC is already covered when we have l dimensions. By taking the average of these proportions for all the PCs that are utilized in the visualization (the first and second PCs when the result is in 2D), we can obtain a percentage that describes how much of the visualized information of the data the PCA result of l dimensions explains. Therefore, by subtracting this value from 1, V_l can indicate how much information has not been considered during the position estimation process. Note that V_l remains the same for all m new points, while U_{l_u} is different for each new data point.

To account for both uncertainties, we can compute a combined uncertainty W_{l_u} with

$W_{lu} = \beta U_{lu} + (1-\beta)V_l$ for each new data point u. The value of β ($0 \leq \beta \leq 1$) serves as a parameter for controlling the weight for either uncertainty. β can be defined manually or determined automatically (see the description in the following paragraph). The combined uncertainty W_{lu} is encoded with an outer-ring color for each plotted point using a red sequential colormap, as shown in Fig. 4.3. The saturated red outer-ring refers to a high uncertainty value. In addition, a path for each new data point's movement with gradient colors is drawn to represent the uncertainties at the corresponding source and target positions. This allows us to see the change of the data positions and uncertainties.

Selecting a proper value of β is not trivial because the user may not have a clear criterion to follow. Thus, I provide an automatic method to help users decide the value of β. Let σ_{ui} be the distance between a new data point u and an existing point i in the updated PCA result after u reaches $d = D$. The mean absolute error e_{l_u} for the estimated distance relationship of u when u has l dimensions of the data is:

$$e_{l_u} = \frac{1}{n}\sum_{i=1}^{n}\left|\sigma_{ui} - s'_{ui}\right| \tag{4.5}$$

W_{l_u} should be an indication of this future error e_{l_u}. Thus, we can assume that e_{l_u} is proportional to W_{l_u} (i.e., $e_{l_u} \propto W_{l_u}$). We calculate a proper β, as β adjusts the balance between U_{l_u} and V_l, to obtain this proportional relationship. From $e_{l_u} \propto W_{l_u}$, we obtain $e_{l_u} = \rho U_{l_u} + \phi V_l$ where $\rho = a\beta$ and $\phi = a(1-\beta)$ ($a \in \mathbb{R}$). Since β can be calculated with $\beta = \rho/(\rho+\phi)$, we want to obtain ρ and ϕ.

First, we utilize the fact that the estimated positions when $l = D$ have no uncertainty for V_l (i.e., $V_D = 0$). Then, we can consider that e_{Du} is proportional to U_{Du} (i.e., $e_{Du} \propto U_{Du}$). Second, for all dimensions $1 \leq l \leq D$, we approximate U_{lu} with $E[U]_u = \sum_{l=1}^{D} U_{lu}/D$ (i.e., $U_{lu} \simeq E[U]_u \forall l$). We then obtain a relationship of $e_{lu} - e_{Du} \propto V_l$. Then, we can approximate β with the following equations:

$$\rho = \frac{e_{Du}}{E[U]_u}, \quad \phi = \frac{\sum_{i=1}^{D}(e_{iu} - e_{Du})}{\sum_{i=1}^{D} V_i} \tag{4.6}$$

$$\beta = \frac{\rho}{\rho + \phi} \tag{4.7}$$

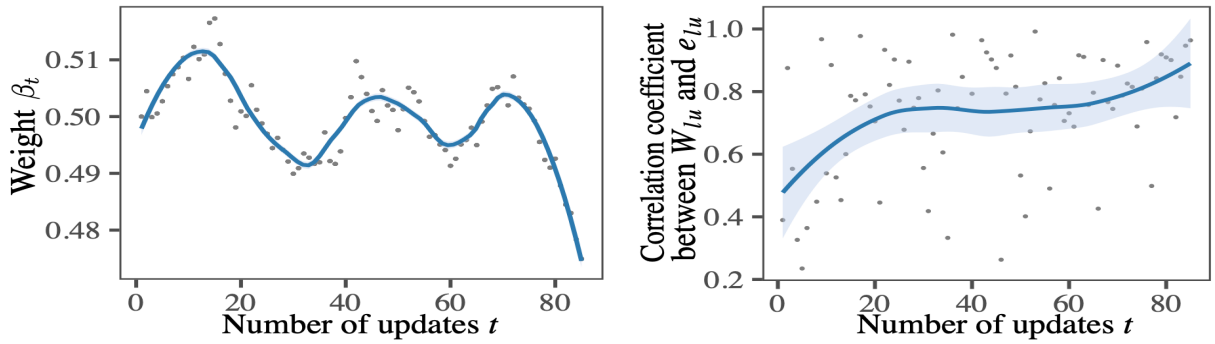


Figure 4.4: Results of the automatic selection of β. The same dataset as Fig. 4.3 is used. The left shows a scatterplot of the number of updates t and the weight β_t. The right is a scatterplot of the number of updates t and the Pearson correlation coefficient between the uncertainty W_{lu} and the error e_{lu}. The blue lines show the smoothed lines with LOESS [45], while the light blue areas represent the 95% confidence intervals.

However, this β cannot be calculated when we estimate the position for the new point u when we have only l dimensions since the value of σ_{ui} in Eq. 4.5 is still unknown at that time. Thus, we obtain the approximated β by calculating the gradient of β from the previous PCA result. Let β_t be the β for the PCA result after updating t times (i.e., after applying process a3 in Fig. 4.6 t times). By using the same update method as Adadelta [257], β_{t+1} can be estimated from β_t with:

$$g_t = \beta_t - \frac{\rho}{\rho + \phi}, \quad \Delta\beta_t = -\frac{RMS[\Delta\beta]_{t-1}}{RMS[g]_t} g_t \tag{4.8}$$

$$\beta_{t+1} = \beta_t + \Delta\beta_t \tag{4.9}$$

where $RMS[\cdot]$ is the root mean square and $\beta_0 = c$ is set as the initial parameter for β. By default, $c = 0.5$. This method automatically adjusts the weight β, as the PCA result is updated. As a result, the user can keep observing the uncertainty with well-balanced weights.

Fig. 4.4(left) shows an example of the automatically selected β. The same dataset as Fig. 4.3 (the bus transportation dataset from TransitFeeds [223]) is used. We observe that β keeps increasing to more than 0.5 when $t < 10$ (i.e., the uncertainty U_{lu} has more influence on the error e_{lu}), while we see the inverse relationship when $t \geq 70$ (i.e., β keeps decreasing from 0.5 implying that the uncertainty V_l has more influence on e_{lu}). Fig. 4.4(right) shows the transition of the Pearson correlation coefficient (PCC) between

the combined uncertainty W_{lu} and the error e_{lu} (more specifically, sets of W_{lu} and e_{lu} with $l=\{1,\cdots,D\}$ and $u=\{1,\cdots,m\}$ for each t). First, we can see that W_{lu} and e_{lu} have a positive association at each time point. Therefore, W_{lu} can well represent the uncertainty of the placement of each new data point. Also, the increase of PCC can be seen when $t < 20$. This indicates that the automatic update of β contributes to better obtainment of the uncertainty W_{lu}.

4.2.5 Automatic Tracking

Sec. 4.2.2 describes a method that helps the user follow the frequent changes that will occur in streaming data visualizations. There are two additional considerations that need to be taken into account. One is that the estimated position calculated from Sec. 4.2.3 can be outside of the range [140] of the PCA result. In this case, to avoid failing to inform the user of an important change, the visualization should update its ranges of axes or have an indicator to notify the user that there are points outside of the ranges. The second consideration relates to outliers when using linear DR methods, including PCA. When the data includes an outlier, DR methods will project the outlier to a position which is far away from other data points. For example, in Fig. 4.5-a, a purple point at the top-right and a green point at the bottom-left are two outliers. In this case, the user may be interested in only keeping track of the data points that are not outliers.

To address these issues, I provide an automatic tracking mechanism that allows the user to stay focused on the data points of their interest. Fig. 4.5 shows the process of the automatic tracking. First, the user indicates the data points of interest through some selection method, such as a lasso selection. For example, in Fig. 4.5-a, the user chooses the data points by lassoing and then selects the selected data points and incoming new data points as tracking targets from a dialog menu. Next, zooming and panning are applied to show the selected points in the center of the scaled window, as shown in Fig. 4.5-b. Zooming and panning will be applied again when the plotted result is updated by either new estimated positions, the addition of new points, or a recalculation of the PCA result (Fig. 4.5-c).

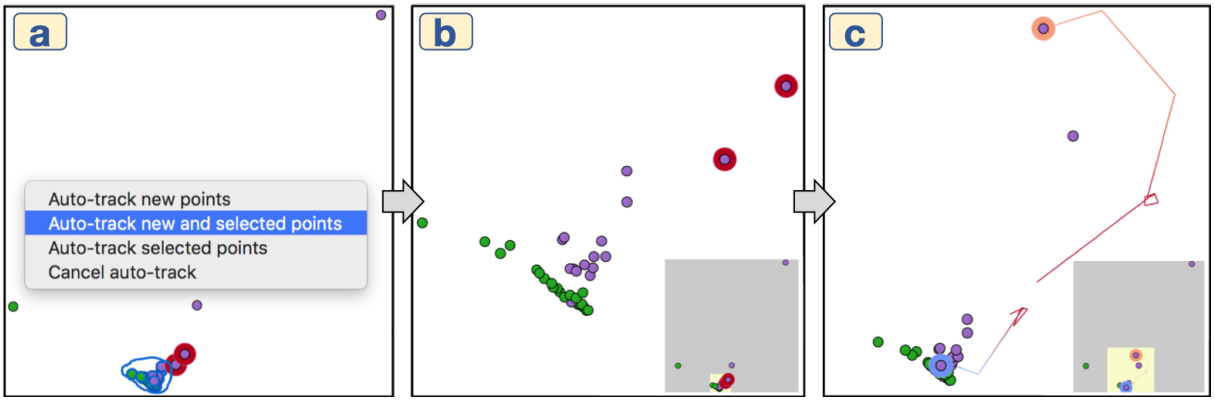


Figure 4.5: An example of the automatic tracking. In (a), since the outliers make the result sparse, points in the dense area are selected and then the function to track the selected points and new points is chosen from the dialog menu. Then, zooming and panning are automatically applied to focus on the tracking points in (b). When the positions of the tracking points are updated, the focus area is also automatically updated, as shown in (c).

However, when a large change occurs, it is difficult for a user to preserve their mental map. To help maintain the mental map, animated transitions are used for zooming and panning (referred to as the view-level transitions) in addition to the three staged animated transitions [19] (referred to as the visual-structure level transitions).

The animations are applied in the following order: panning, zooming in, removing, moving, and adding data points for the cases of zoom-in animations. The zoom-out animations follow the order of zooming out, panning, removing, moving, and adding data points. I have tested multiple alternative designs. First, I used the visual-structure level transitions and the view-level transitions in parallel. However, this caused many changes to happen simultaneously and the result was hard to follow. Another option was to use the visual-structure level transitions before the view-level transitions. In this case, actions, such as removing, moving, or adding data points, could happen outside of the axes ranges. As a result, there was a potential issue of failing to inform the changes to the user. The last consideration was the order of zoom and pan in the view-level transitions. When I first animated zoom and then pan, the visualization was zoomed into the unrelated area of the selected points. This also made it difficult to follow the changes. A similar result occurred when I first animated pan then zoom-out. This issue also happened when applying zooming and panning in parallel, similar to

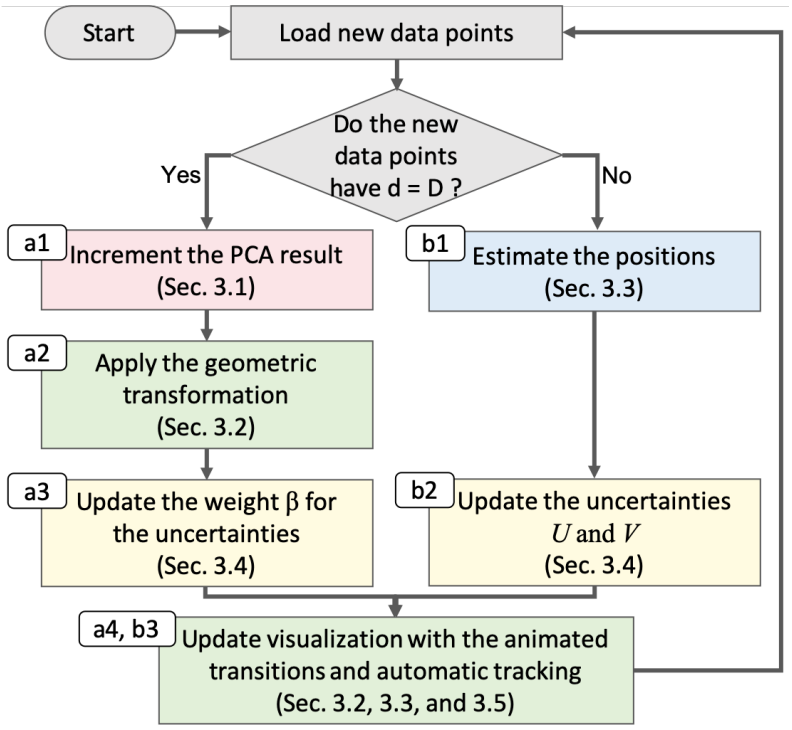


Figure 4.6: A flowchart of the streaming data visualization using incremental PCA. The red, green, blue, and yellow are process blocks that correspond to the methods for dealing with the computational cost, the viewer's mental map, and the non-uniform number of dimensions, the uncertainty visualization, respectively.

the work by van Wijk and Nuij [235]. Thus, I decided to employ different orders of steps based on whether it required zooming-in or zooming-out.

In addition, a mini-map is provided to help the user grasp which part of the plot they are looking at after panning and zooming. An example of visualization with animated transitions can be found online (Appendix A).

4.3 Performance Evaluation

The evaluation in this section demonstrates that the methods described in the previous section are fast enough for handling streaming data through an evaluation of computational performance for each method. As an experimental platform, iMac (Retina 5K, 27-inch, Late 2014) was used. It has 4 GHz Intel Core i7, 16 GB 1,600 MHz DDR3.

Fig. 4.6 shows the flowchart of the overall process starting from receiving the new data points to visualizing the results. There are two main flows on how to deal with new

Table 4.1: Completion time (in milliseconds) of each process in Fig. 4.6. Graphical results are also available online (Appendix A).

D	n	a1	a2	a3	b1	b2
10	100	0.011	0.006	0.003	0.914	0.002
10	1,000	0.014	0.010	0.016	4.417	0.002
10	10,000	0.067	0.091	0.144	42.357	0.002
100	100	0.029	0.004	0.022	0.900	0.110
100	1,000	0.072	0.010	0.160	4.410	0.110
100	10,000	0.949	0.085	1.578	42.338	0.110
1,000	100	0.198	0.004	0.222	0.908	8.618
1,000	1,000	0.962	0.011	1.652	4.410	8.574
1,000	10,000	24.410	0.085	15.291	42.335	8.580

data points based on whether they have the values for all D dimensions (processes a1–a4 in Fig. 4.6) or do not (b1–b3). Since the completion time of a4 and b3 mainly depends on the duration of animated transitions, the completion times of the other processes are only measured. To run the experiment, we use synthetic datasets containing data points with a different number of dimensions (D=10, 100, 1,000), and all values are randomly assigned in the $[-1:1]$ range. In addition, the frequency for updating the visualized results at every 2 new data points ($m = 2$) is used. The geometric transformation and position estimation method are applied for 2D points. The maximum number of iterations for running the Adadelta optimization (Eq. 4.2) is bounded at 1,000.

Tab. 4.1 shows the completion time for each method with different numbers of the pre-existing points (n=100, 1,000, 10,000). Each completion time is the average of ten executions. In Tab. 4.1, we can see that processes a1–a3 can be done in 40 ms even when we have 10,000 pre-existing points with 1,000 dimensions, while b1–b2 can be done in approximately 50 ms. Note that the completion time for process a1 increases as n increases even though incremental PCA's time complexity ($O(dm^2)$) does not relate to n. This is due to the projection step of $(n + m)$ data points using the first k principal components obtained from incremental PCA. These results show that the computational costs of the methods are low enough for supporting streaming data analysis with large numbers of data points and features in real-time.

4.4 Prototype System

A prototype system that integrates the methods described in Sec. 4.2 is developed. The prototype system has three views: the (a) DR, (b) parallel-coordinates (PCP), and (c) scatterplot-matrix (SM), as shown in Fig. 4.7. As the names indicate, the DR view shows the projection results from the incremental PCA, the PCP view displays the data points' values for each dimension with parallel coordinates, and the SM view presents the pairwise scatterplots between any of the two dimensions. While visualizing PCA results is effective in showing an overview of the streaming multivariate data, it neglects the detailed information of the data points. To supplement the DR view, the system incorporates the parallel coordinates which can show many dimensions of information all at once in a limited space and reveal the trend of the data points clearly [53]. However, the parallel coordinates are not suitable for analyzing the correlation between each pair of dimensions [53]. Thus, the system provides a scatterplot matrix for this type of analysis. To achieve fast calculation and rendering, C++ and OpenGL are used for visualization, Qt for the user interface, and Eigen [94] for linear algebraic calculations.

In the DR view, the system uses the point's color to indicate a user-defined grouping of the data points. For interactions, the system supports fundamental view operations and selection, such as zooming, panning, lasso selection, and filtering with linking to the other views. From a dialog menu, the user can also start to use the automatic tracker described in Sec. 4.2.5 with multiple options: track only incoming new data points, track only selected data points, or both.

In the PCP view, each data point is shown as a polyline. The vertical axis corresponds to each dimension and its y-coordinates reflect the data points' values of its corresponding dimension. The user can choose whether to scale the plotted values for each dimension within a 0-to-1 range or not. Each line color shows the corresponding group information defined in the DR view. To perform brushing & linking and filtering, the system provides a freeform selection for the parallel coordinates' lines. The user can also select which dimensions to be shown in the SM view by clicking the names of the dimensions placed at the bottom of the view. The selected dimensions are indicated

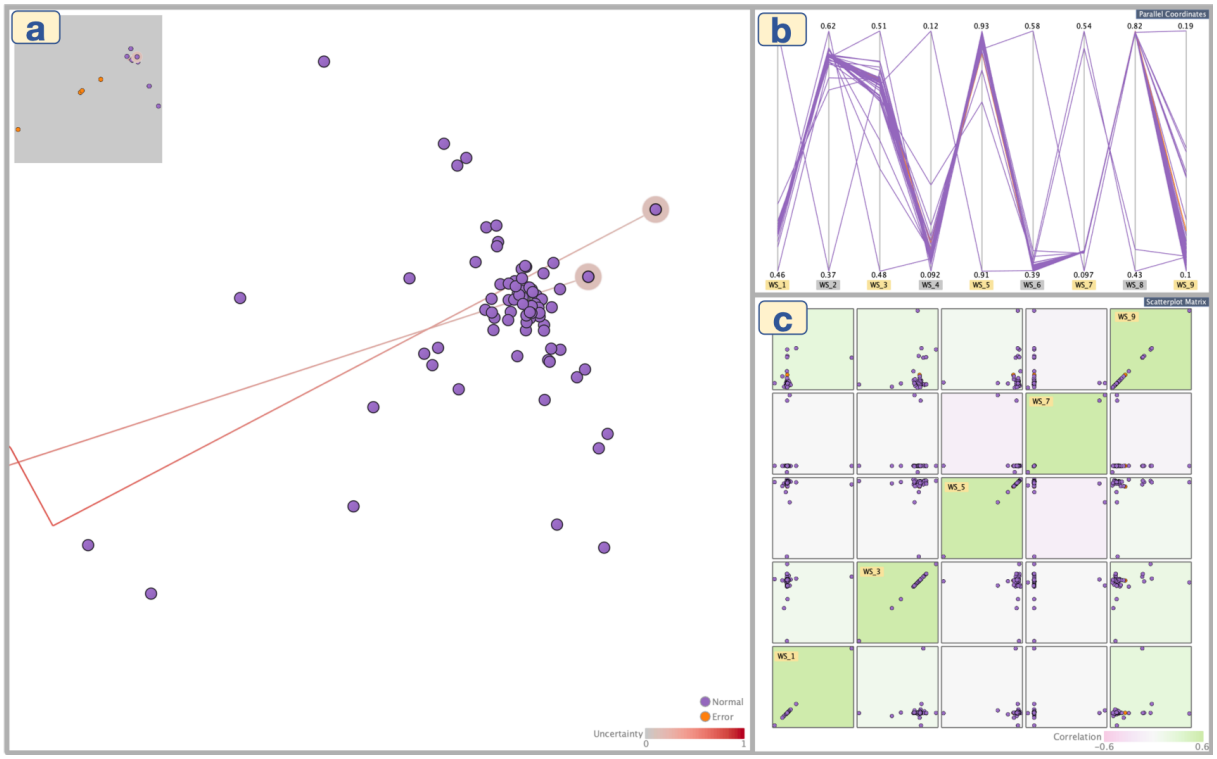


Figure 4.7: A prototype system consisting of three views: (a) the DR view, (b) the parallel-coordinates view, and (c) the scatterplot-matrix view.

in yellow.

The SM view shows the pair-wise scatterplots between any two of the selected dimensions. To show the Pearson correlation coefficient for each plot, the system uses a colored background with a pink-to-green colormap (pink: negative correlation, green: positive correlation). This view also supports lasso selection.

4.5 Case Studies

This section demonstrates the effectiveness of my incremental DR method for streaming multidimensional data with the prototype system. Analyses of two different types of time-series data show how the method is used for finding useful patterns, such as anomalies and clusters.

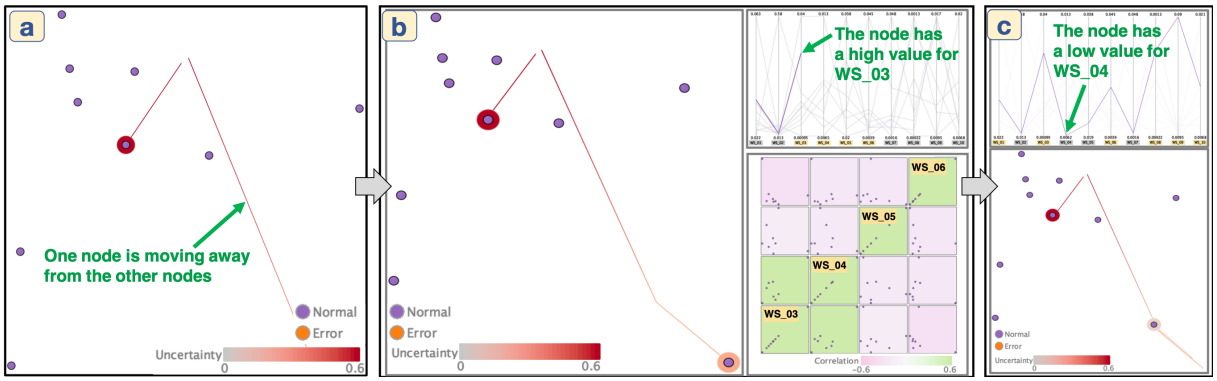


Figure 4.8: An example of visual detection of an anomaly node. In (a), one node moves far away from other nodes and goes out of view from the current visualized range. This suspicious behavior indicates that this node could be an anomaly. Thus, we follow the node with the automatic detection in (b). To see more details about this node, we visualize it in the PCP view. We also show the scatterplots of the current work station ("WS_03") and three immediate work stations ("WS_04", "WS_05", and "WS_06"). (c) shows the PCP and DR views after obtaining the entire set of values of the node.

4.5.1 Study 1: Visual Diagnosis of Assembly Line Performance

We analyze real-time tracking data of an assembly line in a smart factory [249]. The assembly line consists of a set of work stations. Each product part is moving from one station to the next. We use the status information sent from programmable logic controllers (PLCs) on the assembly lines when the parts arrive at the stations. Each part at a work station is set as a data point and its cycle time is set as the data point's features. The cycle time is calculated by subtracting the time range that a part finished the process of one station and has moved onto the next. There are 11 work stations in the selected subset of the assembly line. Therefore, if a part has finished passing through all 11 work stations, it has 10 features (cycle times). In addition, we use the fault code to categorize the group information, which is recorded by the PLCs when any error occurs during processing a part on a station.

We now show three examples of visual diagnosis of anomalies and errors from a subset of the assembly line data in a single day. The full dataset consists of 1,728 product parts (data points), 10 cycle times (features) from 11 work stations, and the fault code (i.e., error or no error). The median of all cycle times from all parts is approximately one minute.

The first example is shown in Fig. 4.8-a, where we notice that one node (a product

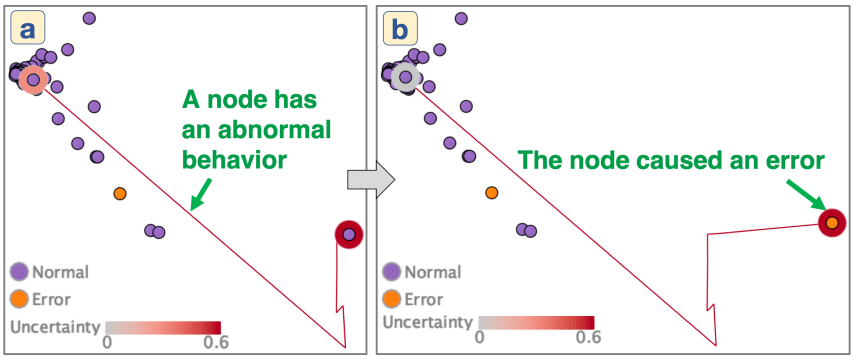


Figure 4.9: An example of a visual prediction of a future error. In (a), we can see that one node has abnormal behaviors. At last, the node causes an error (indicated by orange color) as shown in (b).

part) starts to move away from other nodes by looking at the path as indicated with a green arrow. Since this could be an anomaly, we start to track this node with the automatic tracking. As this node passes through more work stations, we find that this node keeps moving away from the other nodes, as shown in the DR view in Fig. 4.8-b. To review in more detail, we select this node and show its data values in each dimension in the PCP view in Fig. 4.8-b. We can see that this node has a high value for the work station "WS_03". Then, we look at the scatterplot matrices for the work stations from "WS_03" to "WS_06" (the lower-right of Fig. 4.8-b). We can see that "WS_03" has positive correlations with "WS_04", while it has negative correlations with "WS_05" and "WS_06". Therefore, if this node were to follow the same trend as the other nodes, we should expect that this node would have a high value for "WS_04" and low values for "WS_05" and "WS_06", respectively. However, as shown with the PCP view at the top of Fig. 4.8-c, the node holds a low value for "WS_04". Since the node behaves very differently from the other nodes, this foreshadows that this node will be an anomaly. As a result, as shown in the DR view at the bottom of Fig. 4.8-c, the node stays far away from the others. Despite the abnormal behavior, this does not cause an error during the process.

As a second example, we see a case where a visually detected anomaly node causes an error. In Fig. 4.9-a, we find that one node suddenly strays away from the others and continues to move farther away in successive steps. Similar to the previous example, this behavior indicates a high possibility of the node being an anomaly. As a matter of

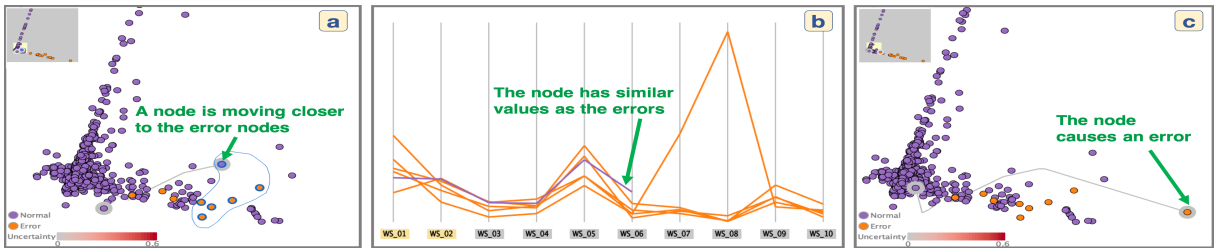


Figure 4.10: An example of a visual prediction of an error based on previous errors. In (a), we can see that one node, indicated with the green arrow, comes close to the error nodes colored with orange. From (a), we select this node and the error nodes with the lasso selection and visualized them as the parallel coordinates in (b). We can see that the values of the node (the purple polyline) follow closely to the values of the other nodes (the orange polylines). At two work stations after the state of (a) and (b), the node causes an error, as shown in (c).

fact, immediately after this step, the node causes an error, as shown in Fig. 4.9-b. This example demonstrates the functionality of the method: to visually identify a data point that could cause an unknown error.

The third example demonstrates how we use the method to foresee a future error by utilizing the known errors. As shown in Fig. 4.10-a, one node, as pointed by the green arrow, moves to a position where several error nodes (colored orange) reside. Since this behavior indicates that this node has a high possibility to cause the same error, we further investigate its relationships with those error nodes. We select the related nodes with a lasso selection in Fig. 4.10-a and visualize their values for each dimension with the PCP view, as shown in Fig. 4.10-b. From Fig. 4.10-b, we can see that the values of the node (represented with the purple polyline) have similar values with the error nodes (the orange polylines) up to the sixth work station. Given this observation, we predict that this node will cause an error in the near future. In fact, we find that this specific node causes an error after it has passed two more work stations, as shown in Fig. 4.10-c.

Through this case study, we find product parts that produce anomaly patterns and/or yield errors in the assembly line. We achieved this by applying the position estimation method on the product parts which have not passed all the work stations yet. This shows the usefulness of the method to perform real-time monitoring on time-series data for early anomaly detection and error reasoning.

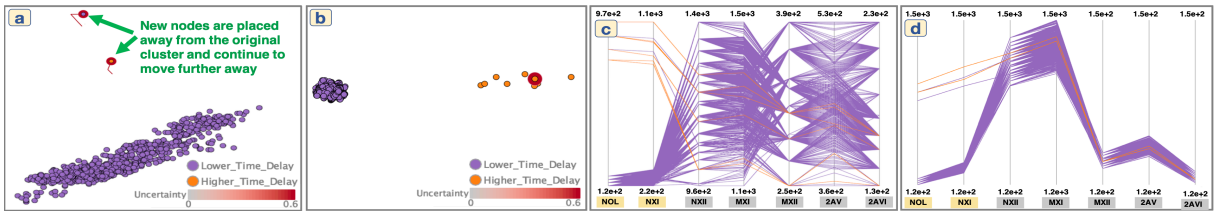


Figure 4.11: An example of visual detection of a new forming cluster which has delays in the MTA bus trips. (a) shows two new nodes moved away from the originally formed cluster. (b) shows a distinct cluster (mainly consisted of orange nodes) formed with more arrived data points. (c) shows the PCP view of the two clusters. We can see that the values of the node (the purple polylines) show a clear distinction to the values of the new nodes (the orange polylines). In (d), we show the PCP view with the original scales of the values.

4.5.2 Study 2: Bus Traffic Analysis

For the second case study, we use the tracking data from the Metropolitan Transportation Authority (MTA) and RTA (Regional Transportation Authority) at Nashville, United States [223]. Nashville MTA/RTA Stops and Routes are used in mapping programs, such as Google Transit. We use the arrival times of an MTA bus from one station to the next to calculate the transit time. Each data point in this dataset is a trip taken by each bus. The MTA dataset consists of many routes. For this case study, we analyze one route that runs through downtown Nashville. The dataset consists of approximately 1,500 data points (trips).

Fig. 4.11 shows the results of processing 800 points. We can see that the incremental PCA has split the data points into one large, distinct cluster (purple points) as seen in Fig. 4.11-a. The newly incoming data points (the purple and orange points) promptly deviate from the large cluster. As more incoming points are processed, we see a new cluster forming, which mainly consists of orange nodes, as shown in Fig. 4.11-b. To understand how the clusters are being split, we further analyze the data with the PCP view, as shown in Fig. 4.11-c. We find that the incoming orange nodes in the new cluster follow a different value on each bus stop when compared with the purple nodes. More specifically, we can see that these nodes have higher time delays for the first two stops ("NOL" and "NXI") when compared to the purple nodes. In Fig. 4.11-c, the parallel coordinates expand the values of each dimension to the minimum-maximum range on the y-direction. This makes judging more difficult on which cluster has higher delays

in total. Alternatively, we choose to show the original values in each dimension without expanding them, as shown in Fig. 4.11-d. This allows us to see that the new cluster does have a higher delay time in total. This example shows how we can discover and review newly emerging clusters with my method.

As we continue to process more data points, referring to Fig. 4.12, we observe that the data points with higher delays start to form more clusters. Fig. 4.12-a shows that there are six additional clusters being formed. Intuitively, we could assume that there is more correlation between the clusters that are closer to each other (i.e., the time delays are comparable). To understand which criteria leads to these separated clusters, we compare the values in each cluster with the PCP view. For example, in Fig. 4.12-b, we highlight the values of the clusters which mainly contain brown or cyan nodes by selecting from the DR view with the lasso selection. We can see that the brown lines ("Higher_Time_Delay_5") are mostly at the upper end of the figure when compared to the blue lines ("Higher_Time_Delay_2"). With further investigation for each cluster, we find the clusters that are farther apart show higher time delays. Note, for the examples above, the coloring of the nodes is used to make the explanations clear. The findings and patterns can be found even by using the provided selection and filtering methods instead of using these colors.

At this point, we know how the clusters are formed. This information could be used to gain some additional insight from the dataset based on the time of the day when the bus trip occurred. We now then visually group the data points based on the hour of the day, and the groups are categorized as "Morning"(5 AM–11 AM), "Afternoon"(11 AM–4 PM), "Evening"(4 PM–9 PM), and "Night"(9 PM–2 AM) trips. From Fig. 4.13, we notice that the highest delays (orange points in the middle right) occur in the "Afternoon" trips. In addition, the "Afternoon" trips' nodes can be found in all the clusters that have delays. On the other hand, from my general assumption, one might assume that the "Night" trips would incur no delays. However, this is not true, as seen in Fig. 4.13. Many of the "Night" trips are grouped into a cluster with delays (indicated with the green arrow in the middle). The time duration for the groups could be varied

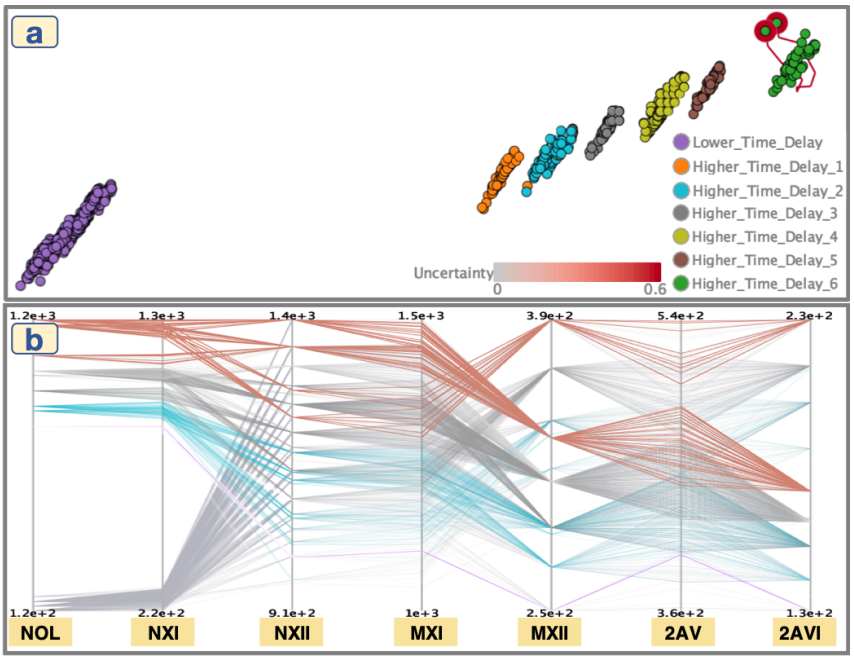


Figure 4.12: The bus trips after processing more data points from Fig. 4.11. (a) shows the additional clusters formed away from the original cluster at Fig. 4.11-a. (b) shows the PCP view of the two selected clusters "Higher_Time_Delay_2" (cyan) and "Higher_Time_Delay_5" (brown).

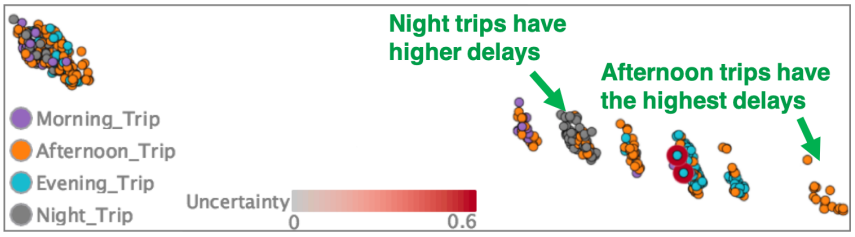


Figure 4.13: The visually grouped bus trips colored by the hour of the day that each trip occurred in.

and this will change the final result of Fig. 4.13. However, we can consider that this particular choice gives us a concrete idea of how the data is laid out in the final result.

4.6 Discussion

To preserve the viewer's mental map, the enhanced incremental PCA introduced in this chapter uses the Procrustes transformation consisting of translation, uniform scaling, rotating, and reflection. For the purpose of visualizing the data, the geometric transformations are not harmful because these transformations do not change the relative distance relationships among the data points. However, if the user wants to analyze the data based on the original PCA result, the algorithm can also provide enough informa-

tion to restore the transformed result back to the original result. This can be achieved by using c, $\boldsymbol{\tau}$, and $\mathbf{R}$ obtained with Eq. 4.1. I would also like to note that the Procrustes transformation can be used to reduce the total positional changes between any two sets of data points (e.g., MDS results [83] and node-link diagrams). This can help in the comparison of two different visualized results.

To deal with a non-uniform number of dimensions, the enhanced incremental PCA employs the position estimation method utilizing the distance relationships among the new and exiting data points in the PCA result. This approach is simple and generalizable, and thus can be applied to other incremental DR methods, such as an incremental MDS [246]. As described in Sec. 4.2.3, another potential option to handle a non-uniform number of dimensions is to predict the missing or unknown values using some machine learning approaches. Even though choosing a proper model for the prediction is challenging, it is worth pursuing as a following research. With its predictive capability, the method can then possibly be used for streaming data with missing values in arbitrary dimensions.

4.6.1 Limitations

The enhanced incremental PCA is developed with the model by Ross et al. [193], which requires at least two new data points to update the PCA result, as described in Sec. 4.2.1. When streaming data visualization requires frequent updates, this limitation is not a problem since, in most real-world scenarios, more than two new data points are constantly received. In cases where updates do not frequently occur, we have enough time for updating the PCA result, and thus we can use the ordinary PCA instead. Also, my method inherits the limitation of a linear DR method and would not be suitable for revealing the local neighbors in a complicated structure.

In addition to the incremental addition of data points, my method allows the user to delete past observations by utilizing the forgetting factor in Ross et al.'s model [193]. However, my method does not support updating feature values of past observations because their model is not designed for such a case. Exploring different ways to support this operation could be one direction for future work.

Another limitation of the work is related to the animated transitions. If we keep receiving new data points in a very short amount of time (e.g., less than a second), the staged animated transitions [19] may not have enough time to complete. In this situation, we could consider not employing the animation. Despite that possibility, the method can still be effective in maintaining the mental map as it can keep the node positions with the geometric transformation. As an additional option, we can store the new data points for a period of time, and then update the result when there is enough duration for the animation.

The position estimation method is designed mainly for cases where new data points have an incomplete number of dimensions and keep collecting the values until they reach the same number of dimensions as the existing data points. It is also possible to apply the method in other situations. For example, when new data points have more dimensions than the existing data points, we can plot the new points by applying the incremental PCA with only the dimensions that the existing data points have. Even though some dimensions may be discarded, the uncertainty measure can be used to inform the user how much uncertainty is introduced. Another example is when some dimensions of the data are no longer able to be used (e.g., at some point a work station is removed from the assembly line). By allowing the user to select which dimensions should be included in the PCA calculation, the method can also be applied for this case.

The scalability of the visualization is also worth discussing. The prototype system visualizes the PCA result as a 2D scatterplot. Therefore, the scalability issue mainly depends on the scatterplot itself. One way we can approach this issue is to delete or aggregate data points that are not necessary to be visualized. For example, we can filter out old data points in the visualization (e.g., data points that are a day old) or aggregate data points based on their similarities. This approach also solves the same scalability issue in the data points with the outer-ring colors. A similar issue can occur when paths are drawn to show the new data points' movements. If there are many new data points, it could create cluttered lines. One way to reduce this clutter is to filter the paths. For example, we can set a criterion based on the length of their movements or

on a chosen threshold for specific feature values.

4.7 Summary

The work in this chapter enhances the usability of a representative DR method for interactive analysis of streaming data. The enhanced method is able to address both the interactivity and interpretability of the visualization. The visual stability and the capability of handling varying data dimensions offered by the incremental method lead to effective visualizations for streaming data analysis. For example, the case studies demonstrate how time-varying data features, such as errors or clusters, could more easily be identified.

Chapter 5
Network Data Analysis: Contrastive Network Representation Learning

Networks are commonly used to model various types of relationships in real-world applications, such as social networks [50], cellular networks [41], and communication networks [29]. Comparative analysis of networks is an essential task in practice, where we want to identify differentiating factors between two networks or the uniqueness of one network compared to another [66, 220]. For instance, when a neuroscientist is studying the effect of Alzheimer's disease on a human brain [82], they want to compare the brain network of a patient with Alzheimer's disease to that of a healthy subject. Also, for collaboration networks of researchers in different fields [146], an analyst in a funding agency may want to discover any unique ways of collaborations in the fields for decision making.

Several approaches have been proposed for network comparison [220]. When two different networks have the same node-set and the pairwise correspondence between nodes is known, we can compute a similarity between two networks (e.g., a Euclidean distance between two adjacency matrices). When the node-correspondence is unknown or does not exist, a network-statistics based approach is commonly used (e.g., the clustering coefficient, network diameter, or node degree distribution). Another popular approach is using graphlets [220]—small, connected, and non-isomorphic subgraph patterns in a graph (e.g., the complete graph of three nodes). The similarities of two networks can be characterized by comparing the frequency of appearance of each graphlet in each network.

While the existing approaches can provide a (dis)similarity between different networks, they compare networks only based on one selected measure (e.g., node degree),

which is often insufficient. Also, these approaches only provide network-level similarities, and thus cannot compare networks in more detailed levels (e.g., a node-level). Without such a detailed-level comparison, it is difficult to find which part of a network relates to its uniqueness.

To address these challenges, this chapter introduces a new approach that integrates the concept of contrastive learning (CL) [3, 263] together with network representation learning (NRL), which I call *cNRL*. Within cNRL, the NRL enables the characterization of networks with comprehensive measures without overwhelming a user with information by embedding nodes into a low-dimensional space; the CL allows for discovering unique patterns in one dataset relative to another. By leveraging the benefits of both, we can reveal unique patterns in one network by contrasting with another, in a thorough (i.e., using multiple essential measures to capture the network characteristics) and detailed (i.e., analyzing a node or subnetwork level) manner.

With the above approach, I consider the generality and interpretability of cNRL, and contribute a method called *i-cNRL*. First, i-cNRL is designed not to require node-correspondences or network alignment [66], and thus is applicable to various networks. Also, unlike many other NRL methods (e.g., node2vec [93] and graph neural networks (GNNs) [259]), i-cNRL offers interpretability [6], providing information about the meaning of an identified pattern and the reason why that pattern can be seen in only that network.

In summary, this chapter's main contributions include:

- A new approach, called contrastive network representation learning (cNRL), which aims to reveal unique patterns in one network relative to another network.
- A method exemplifying cNRL, called i-cNRL, which (1) offers general applicability, including networks without node-correspondence or network alignment, (2) provides interpretability for helping understand revealed patterns, and (3) equips automatic hyperparameter selection for CL.
- Experiments with multiple network models and real-world datasets, which demonstrate the capability of comparative network analysis.

- Quantitative and qualitative comparisons with other potential designs of cNRL methods.

5.1 Problem Definition

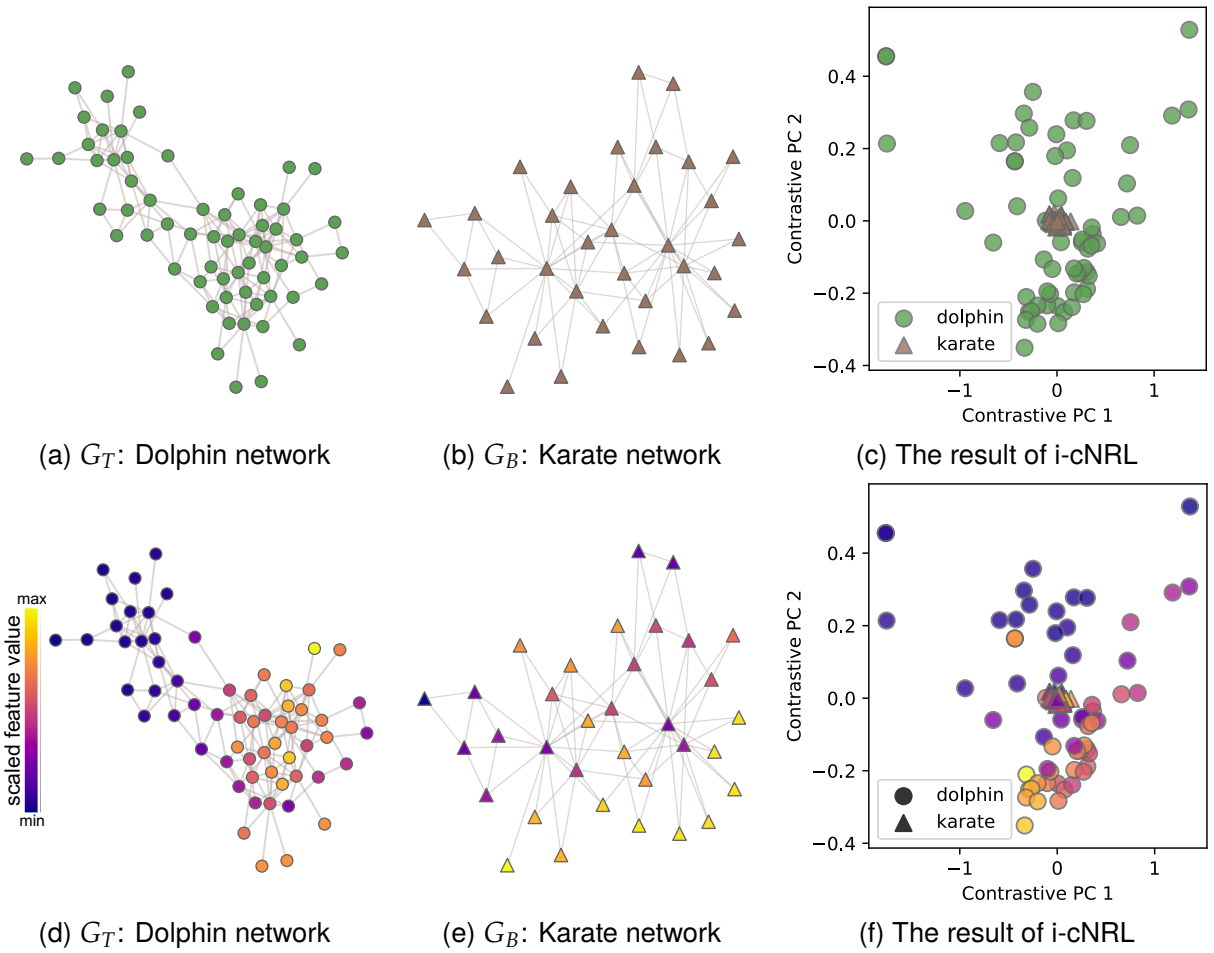
This section defines the problem to be addressed by contrastive network representation learning. Given two different networks, a target network G_T and a background network G_B, we want to seek unique patterns in G_T relative to G_B. Similar to contrastive learning [263], the unique patterns can be represented as relationships (e.g., the structural differences among network nodes) that appear in G_T but do not appear in G_B.

For example, when finding unique patterns in a scale-free network G_T (i.e., its node-degree distribution follows a power law) relative to a random network G_B (i.e., each node pair is connected with a fixed probability) [25], we should be able to capture the unique patterns related to node degrees since G_T has more variety in node degrees. For practical usage, the unique patterns could relate to more complicated centralities, measures, combinations of them, and many more. Note that, as with the existing work of contrastive learning, cNRL does not aim to discriminate G_T from G_B, but to identify unique patterns in G_T.

5.2 Analysis Example

To provide an illustrative example of analysis with cNRL, I begin by comparing two different social networks. I use the Dolphin social network [165] as G_T and the Zachary's karate club network [256] as G_B. Fig. 5.1-a, b depict the network structures of these networks. The statistics of these networks can be found in Tab. 5.1 (see `N1` and `N2`). By comparing these two networks, we want to reveal unique patterns in the Dolphin social network and identify which network characteristics relate to the patterns.

i-cNRL is applied to the two networks and then plot a 2D embedding result with contrastive PCA (cPCA) [3], as shown in Fig. 5.1-c. The x- and y-directions in Fig. 5.1-c represent the first and second contrastive principal components (cPCs), respectively. Details of i-cNRL and related techniques will be described in Sec. 5.4. Fig. 5.1-c shows that the nodes in G_T are more widely distributed, whereas the nodes in G_B are placed



(a) G_T: Dolphin network (b) G_B: Karate network (c) The result of i-cNRL

(d) G_T: Dolphin network (e) G_B: Karate network (f) The result of i-cNRL

Figure 5.1: (a) and (b) show the dolphin social network and the Zachary's karate club network, used as G_T and G_B for i-cNRL, respectively. (c) shows the i-cNRL results with 2D embedding. (d), (e), and (f) colorcode each node in (a), (b), and (c) based on the top-contributed feature (`F1-10`) of the first contrastive principal component (cPC1): $(\Phi_{\text{mean}})(\mathbf{x})$ with 'eigenvector' as the base feature $\mathbf{x}$ (see Tab. 5.2).

only around the center, which reveals some patterns specific to G_T when compared with G_B.

Moreover, since i-cNRL offers interpretability to the learned results, we can analyze why the above patterns appear. As shown in Tab. 5.2, the method provides cPC loadings (refer to Sec. 2.3.3), of which the absolute value indicates how large each learned feature contributes to each cPC direction. Each learned feature can be represented as a combination of the relational function f and the base feature $\mathbf{x}$ [194] (see Sec. 5.4 for details). Tab. 5.2 indicates that feature `F1-10` has the highest contribution to cPC1. From

Table 5.1: Statistics of network datasets.

ID	Name	# of nodes	# of links	Directed
N1	Dolphin [165]	62	159	False
N2	Karate [256]	34	78	False
N3	Random	100	471	True
N4	Price	100	294	True
N5	p2p-Gnutella08 [152, 192]	6,301	20,777	True
N6	Price 2	6,301	18,897	True
N7	Enhanced Price	6,301	18,281	True
N8	Combined-AP/MS [48, 255]	1,622	9,070	False
N9	LC-multiple [190, 255]	1,536	2,925	False
N10	School-Day1 [210]	236	5,899	False
N11	School-Day2 [210]	238	5,539	False

Table 5.2: Learned features and their cPC loadings for the dolphin vs. karate example.

ID	relational function f	base feature $\mathbf{x}$	cPC1	cPC2
F1-1	$(\mathbf{x})$	total-degree	0.00	-0.02
F1-2	$(\mathbf{x})$	betweenness	-0.00	-0.00
F1-3	$(\mathbf{x})$	closeness	0.00	0.00
F1-4	$(\mathbf{x})$	eigenvector	-0.04	0.00
F1-5	$(\mathbf{x})$	PageRank	0.04	0.04
F1-6	$(\mathbf{x})$	Katz	0.00	-0.02
F1-7	$(\Phi_{\text{mean}})(\mathbf{x})$	total-degree	-0.06	-0.08
F1-8	$(\Phi_{\text{mean}})(\mathbf{x})$	betweenness	0.05	-0.01
F1-9	$(\Phi_{\text{mean}})(\mathbf{x})$	closeness	-0.08	0.01
F1-10	$(\Phi_{\text{mean}})(\mathbf{x})$	eigenvector	**0.26**	0.02
F1-11	$(\Phi_{\text{mean}})(\mathbf{x})$	PageRank	-0.11	**0.15**
F1-12	$(\Phi_{\text{mean}})(\mathbf{x})$	Katz	-0.08	-0.09
F1-13	$(\Phi_{\text{max}})(\mathbf{x})$	PageRank	-0.01	0.00
F1-14	$(\Phi_{\text{mean}} \circ \Phi_{\text{mean}})(\mathbf{x})$	total-degree	-0.06	-0.00
F1-15	$(\Phi_{\text{mean}} \circ \Phi_{\text{max}})(\mathbf{x})$	PageRank	0.01	-0.00

the relational function $(\Phi_{\text{mean}})(\mathbf{x})$ and the base feature 'eigenvector' [179], this feature is interpreted as "the mean eigenvector centrality of the neighbors of a node."

To investigate the relationships between this feature and the i-cNRL result, the network nodes in Fig. 5.1-a, b, c are colorcoded based on the feature values, as shown in Fig. 5.1-d, e, f. We can see that, in Fig. 5.1-f, the nodes around the top-left corner tend

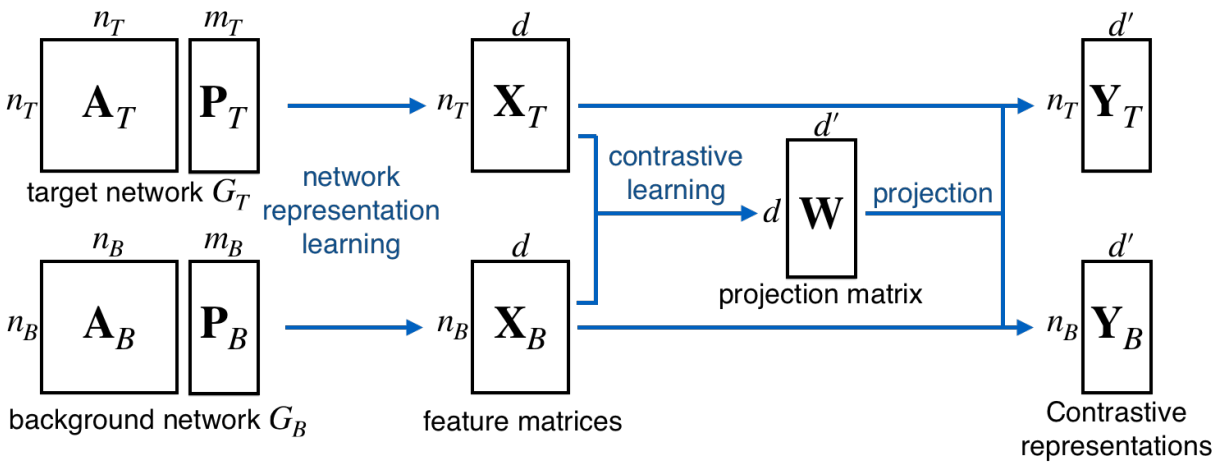


Figure 5.2: The general architecture for cNRL.

to have smaller feature values while the nodes around the bottom-right tend to have higher values. By comparing with Fig. 5.1-d, we notice that these two node groups correspond to the top-left and bottom-right communities in Fig. 5.1-d. Since the feature value shows the mean eigenvector centrality of the neighbors of a node, the nodes in the top-left community tend to have a low eigenvector centrality including their neighbors. On the other hand, the nodes in the right-bottom community have neighbors with a high eigenvector centrality. Fig. 5.1-e indicates that G_B does not have such clearly separated communities by the feature values, unlike G_T. Therefore, i-cNRL learns the patterns highly related to the eigenvector centralities of each node's neighbors, which can clearly separate the two communities in the Dolphin social network.

5.3 cNRL Architecture

Fig. 5.2 shows a general architecture for cNRL. Notations used for the following sections are listed in Tab. 5.3. The current CL methods [3, 4, 62, 203, 263] require target and background feature matrices ($\mathbf{X}_T$ and $\mathbf{X}_B$) sharing the same features as inputs. However, matrices that represent target and background networks (G_T and G_B) such as adjacency matrices ($\mathbf{A}_T$ and $\mathbf{A}_B$) might have a different number of nodes or no correspondence in nodes of $\mathbf{A}_T$ and $\mathbf{A}_B$. Thus, we cannot directly apply the CL methods to target and background networks (G_T and G_B). To address this issue, the core idea of cNRL consists of two main steps: (1) generating feature matrices $\mathbf{X}_T$ and $\mathbf{X}_B$ from networks G_T and G_B

Table 5.3: Summary of notations.

Notations for CNRL	
G_T, G_B	target and background networks
$\mathbf{A}_T$, $\mathbf{A}_B$	adjacency matrices of G_T and G_B
$\mathbf{P}_T$, $\mathbf{P}_B$	matrices of node attributes of G_T and G_B
n_T, n_B	numbers of nodes in G_T and G_B
m_T, m_B	numbers of attributes in G_T and G_B
l_T, l_B	numbers of edges in G_T and G_B
d, d'	numbers of features learned by NRL and CL
$\mathbf{X}_T$, $\mathbf{X}_B$	target and background feature matrices
$\mathbf{W}$	projection matrix learned by CL
$\mathbf{Y}_T$, $\mathbf{Y}_B$	contrastive representations of $\mathbf{X}_T$ and $\mathbf{X}_B$
Notations for DeepGL	
$\mathbf{x}$	base feature (e.g., in-degree)
f	relational function
Φ^-, Φ^+, Φ	relational feature operators for in-, out-, total neighbors
S	summary measure (e.g., mean, sum, and maximum)
$\mathcal{F}_i$	set of learned features with i relational feature operators
$\mathcal{F}$	set of learned features: $\mathcal{F} = \{\mathcal{F}_0, \cdots, \mathcal{F}_h\}$
h	maximum numbers of relational feature operators to use
Notations for cPCA	
$\mathbf{C}_T$, $\mathbf{C}_B$	covariance matrices
α	contrastive parameter

respectively by using NRL, and (2) applying CL on $\mathbf{X}_T$ and $\mathbf{X}_B$.

Below the details of each part of the cNRL architecture are described with requirements on inputs, NRL, and cNRL algorithms. The description focuses only on node feature learning to provide a simple and clear explanation. However, the architecture is generic enough to be used for link (or edge) feature learning.

Inputs. cNRL takes G_T and G_B as inputs. These networks can be any combination of being undirected or directed, unweighted or weighted, and non-attributed or attributed. The numbers of G_T and G_B nodes (i.e., n_T and n_B) do not have to be the same. Similarly, the numbers of attributes m_T and m_B may be different.

Network representation learning. The first step in Fig. 5.2 is applying an NRL method in order to transform the inputs G_T and G_B to feature matrices $\mathbf{X}_T$ and $\mathbf{X}_B$, respectively.

CL requires that $\mathbf{X}_T$ and $\mathbf{X}_B$ share the same features by nature of its learning purpose. Therefore, for this process, we need to use an NRL method that can produce the same features across networks.

Contrastive learning. Once we obtain $\mathbf{X}_T$ and $\mathbf{X}_B$, which have the same d learned features, we can apply any of the CL methods using $\mathbf{X}_T$ and $\mathbf{X}_B$ as target and background datasets, respectively. CL generates a parametric mapping (or a projection matrix $\mathbf{W}$) from d features learned by NRL to d' contrastive features ($d' \leq d$). With this projection matrix, $\mathbf{X}_T$ and $\mathbf{X}_B$ can be transformed to contrastive representations $\mathbf{Y}_T$ and $\mathbf{Y}_B$, respectively. As the existing CL works [3, 4, 62, 203, 263] only produced $\mathbf{Y}_T$ for their analysis, the generation of $\mathbf{Y}_B$ is optional. However, as demonstrated in Fig. 5.1-c, by visualizing both $\mathbf{Y}_T$ and $\mathbf{Y}_B$ in one plot, we can clearly see whether CL has found unique patterns in G_T relative to G_B.

5.4 Interpretable cNRL Method

As a specific method using the architecture above, this section introduces i-cNRL, which employs DeepGL [194] for NRL and cPCA [3] for CL, with the design rationale for the selection of these algorithms.

5.4.1 Network Representation Learning

As stated in Sec. 5.3, NRL needs to generate $\mathbf{X}_T$ and $\mathbf{X}_B$, which have the same features. To achieve this, we can employ any *inductive* NRL method [194] (e.g., GraphSAGE [101] and FastGCN [42]). However, the interpretability should be provided in the contrastive representations obtained by cNRL; thus, an NRL method needs to generate interpretable features as the learned result. As a result, DeepGL is specifically used in the first step of i-cNRL.

5.4.1.1 DeepGL

The method learns node and link features consisting of the *base feature* $\mathbf{x}$ and *relational function* f. For a concise explanation, I describe DeepGL for only node feature learning.

A base feature $\mathbf{x}$ is any simple feature or measure we can obtain for each node. For example, $\mathbf{x}$ can be (weighted) in-, out-, total-degree, degeneracy (or k-core numbers),

PageRank [179], or a node attribute (e.g., gender of a node in a social network).

A relational function f is a combination of *relational feature operators*, which is applied to a base feature. A relational feature operator summarizes base feature values of one-hop neighbors of a node. For example, the operator can be a computation of the mean, sum, maximum base feature values of one-hop neighbors' of a node. Also, the neighbors can be either in-, out-, total-neighbors. Together with the summary measure S (e.g., mean), the operators can be denoted Φ_S^-, Φ_S^+, and Φ_S, respectively. For example, $\Phi_{\text{mean}}^-(\mathbf{x})$ computes the mean $\mathbf{x}$ of the in-neighbors of a node. Moreover, the relational feature operator can be applied repeatedly. For example, $f = (\Phi_{\text{mean}}^+ \circ \Phi_{\text{max}}^-)(\mathbf{x})$ first computes the maximum $\mathbf{x}$ of in-neighbors for each out-neighbor of a node and then produces the mean of these maximum values. As described with the examples above, $\mathbf{x}$ and f are combinations of simple measures and operators; thus, both are interpretable.

In DeepGL, we can select as many different base features and relational feature operators as we want to consider. The learning process contains h number of iterations (indicated by the user), and in the end we obtain all the learned features $\mathcal{F} = \{\mathcal{F}_0, \mathcal{F}_1, \cdots, \mathcal{F}_h\}$, each of which is a relational functions over a base feature $f(\mathbf{x})$. During each iteration, DeepGL prunes redundant features based on the similarities of the obtained feature values. Tab. 5.2 shows an example of learned features from the Dolphin social network [165].

5.4.1.2 Use of Transfer Learning with DeepGL for cNRL

As described above, the learned features $\mathcal{F}$ by DeepGL are the combinations of the base features and relational functions. Once we obtain $\mathcal{F}$ from one network, we can naturally compute $\mathcal{F}$ for other networks. That is, DeepGL is inductive and can be used for transfer learning [194].

In cNRL, we need to decide which network(s), G_T and/or G_B, should be used for learning $\mathcal{F}$. One possible choice is applying DeepGL for both to learn the features $\mathcal{F}_T$ and $\mathcal{F}_B$. Then, we can use the union of these features (i.e., $\mathcal{F}_T \cup \mathcal{F}_B$) for producing feature matrices $\mathbf{X}_T$ and $\mathbf{X}_B$. Since cNRL aims to identify unique patterns in G_T relative to G_B, only a set of features capturing G_T's characteristics is required. Thus, DeepGL is

applied to G_T and the learned features for both G_T and G_B are used to generate $\mathbf{X}_T$ and $\mathbf{X}_B$. It can also avoid unnecessary computation for learning $\mathcal{F}_B$ from G_B.

5.4.2 Contrastive Learning

The above NRL step generates feature matrices $\mathbf{X}_T$ and $\mathbf{X}_B$. The remaining step is learning contrastive representations $\mathbf{Y}_T$ and $\mathbf{Y}_B$ through CL. While we can use any CL method, one of the goals is to provide interpretability. Since DeepGL generates interpretable features for $\mathbf{X}_T$ and $\mathbf{X}_B$, we can provide interpretable $\mathbf{Y}_T$ and $\mathbf{Y}_B$ by using a method that reveals interpretable relationships between d features learned by NLR and d' features learned by CL. Among current CL methods [3, 4, 62, 203, 263], only contrastive PCA (cPCA) [3] can provide such relationships by utilizing the linearity of its algorithm in a similar manner to the classical PCA [125]. Thus, cPCA is selected for the second step of i-cNRL, though, it can be replaced with any other interpretable CL methods developed in the future.

5.4.2.1 Contrastive PCA (cPCA)

Here, I briefly introduce cPCA described in Sec. 2.3.1 again. Similar to the classical PCA, cPCA first applies centering to each feature of $\mathbf{X}_T$ and $\mathbf{X}_B$ and then obtains their corresponding covariance matrices $\mathbf{C}_T$ and $\mathbf{C}_B$. Let $\mathbf{v}$ be any unit vector of d length. Then, with a given direction $\mathbf{v}$, the variances for $\mathbf{X}_T$ and $\mathbf{X}_B$ can be written as: $\sigma_T^2(\mathbf{v}) \stackrel{\text{def}}{=} \mathbf{v}^\mathsf{T}\mathbf{C}_T\mathbf{v}$, $\sigma_B^2(\mathbf{v}) \stackrel{\text{def}}{=} \mathbf{v}^\mathsf{T}\mathbf{C}_B\mathbf{v}$. The optimization that finds a direction $\mathbf{v}^*$ where $\mathbf{X}_T$ has high variance but $\mathbf{X}_B$ has low variance can thus be written as:

$$\mathbf{v}^* = \underset{\mathbf{v}}{\text{argmax}}\ \sigma_T^2(\mathbf{v}) - \alpha\sigma_B^2(\mathbf{v}) = \underset{\mathbf{v}}{\text{argmax}}\ \mathbf{v}^\mathsf{T}(\mathbf{C}_T - \alpha\mathbf{C}_B)\mathbf{v} \tag{5.1}$$

where α is a contrast parameter ($0 \leq \alpha \leq \infty$). Similar to the classical PCA, we can obtain top-d' cPCs as the learned features. With projection matrix $\mathbf{W}$ consisting of d' cPCs (i.e., $\mathbf{W}$ is a $d \times d'$ matrix), we can obtain the contrastive representation $\mathbf{Y}_T$ of $\mathbf{X}_T$.

The above contrast parameter α controls the trade-off between having high target variance and low background variance. When $\alpha = 0$, cPCs only maximize the variance of $\mathbf{X}_T$, the same as those in the classical PCA. As α increases, cPCs place greater emphasis on directions that reduce the variance of $\mathbf{X}_B$. Fig. 5.3 shows the results of cPCA with

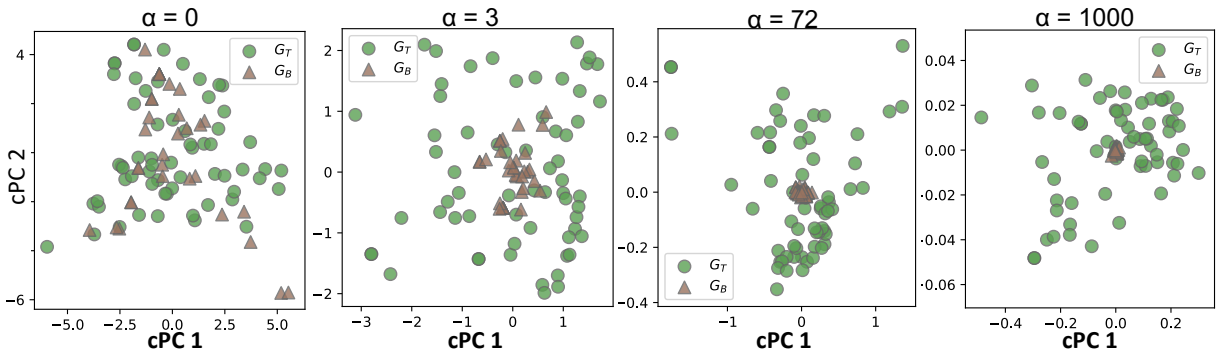


Figure 5.3: The cPCA results with different α values, applied on feature matrices $\mathbf{X}_T$ and $\mathbf{X}_B$ generated from G_T (the dolphin network) and G_B (the Karate network) in Fig. 5.1. When $\alpha = 0$, the result is the same with applying PCA on $\mathbf{X}_T$. A decrease of $\mathbf{X}_B$'s variances is observed as α increases. The result with $\alpha = 72$ corresponds to the results in Fig. 5.1.

different α values. Because α has a strong impact on the result, Abid and Zhang et al. [3] introduced the semi-automatic selection of α utilizing spectral clustering [180]. i-cNRL goes one step further to provide a fully automatic selection of α (see Sec. 5.4.2.3).

5.4.2.2 Representation Learning with cPCA in cNRL

By applying cPCA to $\mathbf{X}_T$ and $\mathbf{X}_B$, we can generate the projection matrix $\mathbf{W}$ and contrastive representations $\mathbf{Y}_T$ and $\mathbf{Y}_B$. Because each learned feature by DeepGL could have a different scale, as a default, i-cNRL applies the standardization to each of $\mathbf{X}_T$ and $\mathbf{X}_B$ for both learning and projection.

To provide interpretable relationships between NLR features d and CL features d', cPC loadings are computed as introduced in Sec. 2.3.3. These cPC loadings indicate how strongly each of the d input features contributes to the corresponding cPC. Tab. 5.2 shows an example of cPC loadings for the first and second cPCs. As demonstrated in Sec. 5.2, by referring to a list of the learned features via NRL and cPC loadings, we can interpret the obtained representations $\mathbf{Y}_T$ and $\mathbf{Y}_B$.

5.4.2.3 Automatic Contrastive Parameter Selection

I now show how to automatically select the parameter α in cPCA. Since we want to maximize the variation in the target feature matrix while simultaneously minimizing the variation in the background feature matrix, we can solve the following ratio problem:

$$\max_{\mathbf{W}^\top \mathbf{W} = I_{d'}} \frac{\mathrm{tr}(\mathbf{W}^\top \mathbf{C}_T \mathbf{W})}{\mathrm{tr}(\mathbf{W}^\top \mathbf{C}_B \mathbf{W})}. \tag{5.2}$$

While directly solving (5.2) may be difficult, there is a convenient iterative algorithm due to Dinkelbach [61]. The algorithm consists of two steps. Given $\mathbf{W}_t$, we perform

- $\alpha_t \leftarrow \dfrac{\mathrm{tr}(\mathbf{W}_t^\top \mathbf{C}_T \mathbf{W}_t)}{\mathrm{tr}(\mathbf{W}_t^\top \mathbf{C}_B \mathbf{W}_t)}$
- $\mathbf{W}_{t+1} \leftarrow \arg\max_{\mathbf{W}^\top \mathbf{W} = I_{d'}} \mathrm{tr}(\mathbf{W}^\top (\mathbf{C}_T - \alpha_t \mathbf{C}_B)\mathbf{W})$.

Clearly, α_t is just the objective value of our ratio problem (5.2) evaluated at the current solution $\mathbf{W}_t$. It is easy to show that α_t monotonically increases to the maximum value, and the convergence is usually very quick (e.g., less than 10 iterations). Conveniently, the second step for finding the next solution $\mathbf{W}_{t+1}$ is just the original cPCA problem, where we use α_t as the trade-off parameter. We can also regard cPCA as a (crude) one-shot algorithm for the ratio problem (5.2) where the user specifies α. One problem of the method above is that α_t reaches close to infinite when $\mathbf{C}_B$ is nearly singular. To avoid this, my method simply adds a small constant value ϵ, as a default $\epsilon = 10^{-3}$, to each diagonal element of $\mathbf{C}_B$. Note that the above algorithm of Dinkelbach [61] has been used in discriminant analysis [97,122], whose motivation is entirely different from contrastive learning.

5.4.3 Complexity Analysis

The time and space complexities of i-cNRL are comparable to those of DeepGL and cPCA. DeepGL's time and space complexities for learning from G_T are $O(d(l_T + dn_T))$ and $O(dn_T)$, respectively, where l_T is the number of links in G_T. Note that the time and space complexities for computing base features are assumed lower than these. When including the transfer learning step to obtain $\mathbf{X}_B$, the space complexity becomes $O(d(n_T + n_B))$. For a fixed α, cPCA has the similar time and space complexities with PCA, which are $O(d^2(n_T + n_B) + d^3))$ and $O(d^2)$. Even with the automatic selection of α in Sec. 5.4.2.3, we can assume that these complexities stay the same. This is because the automatic selection usually only needs a small number of iterations (e.g., less than 10) and does not require storing of additional information. Thus, in total, i-cNRL has the time complexity $O(d(l_T + d(n_T + n_B) + d^2))$ and the space complexity $O(d(n_T + n_B + d))$. However, in practice, d, the number of features learned by NRL, should be much smaller

than the numbers of nodes and links of G_T and G_B. Under this assumption, the time and space complexities are $O(d(l_T + d(n_T + n_B)))$ and $O(d(n_T + n_B))$, respectively. This indicates that the computational cost is largely due to DeepGL.

5.5 Related Work

To the best of my knowledge, my work is the first to introduce contrastive learning for networks and provide a general and interpretable method under this approach. There exists little work in the exact area. Thus, we here review typical NRL and CL techniques.

5.5.1 Network Representation Learning (NRL)

Various NRL methods have been developed for learning latent representations of network nodes and/or links. For a comprehensive description of NRL methods, refer to the recent survey papers [37, 259]. Here focuses on describing the closely related work using inductive and cross-network embedding methods.

5.5.1.1 Inductive NRL

GraphSAGE [101] is an inductive NRL method that shares many similar ideas with DeepGL [194]. Analogous to the relational functions f in DeepGL, GraphSAGE learns *aggregator functions*. However, GraphSAGE proposes more complex aggregators using LSTM and max-pooling concepts, compared to DeepGL's simple aggregators (e.g., mean). Moreover, GraphSAGE tunes parameters required by the aggregators and matrices that decide the weight for each learned feature, instead of the feature pruning in DeepGL. These differences might enable GraphSAGE to better capture complex characteristics of networks without manual parameter tuning; however, the learned features might be difficult to interpret. FastGCN [42] takes a similar approach to GraphSAGE except that FastGCN employs node sampling to save memory space. Also, HetGNN [258] enhances the aggregators to learn representations of heterogeneous networks. Thus, these methods, including other GNN variants [259] (e.g., GAT [236] and h/cGAO [85]), still suffer from lack of interpretability in the learned features. Although GNNExplainer [253] aims to provide interpretable explanations for predictions made by these

methods, it does not support explaining the learned features themselves.

5.5.1.2 Cross-Network Embedding

The inductive methods learn the features that can be generalized for unobserved nodes or other networks from one input network. On the contrary, the cross-network methods generate embeddings directly from multiple input networks. Most of the cross-network methods focus on finding similarities of nodes across networks, such as for node classification [204], network similarity calculation [166], and network alignment [110]. While CrossMVA [44] is developed mainly for network alignment, it can produce embeddings that contain both similarity and dissimilarity information. However, a major drawback of CrossMVA is that anchor nodes are necessary as inputs (i.e., at least we need to know a small portion of node-correspondence), which we cannot obtain in many cases (e.g., the example in Sec. 5.2). Also, CrossMVA's embeddings of the dissimilarity information only preserve discriminative structures across networks; as a result, it cannot find unique patterns in a specific network.

5.5.2 Contrastive Learning (CL)

Unlike discriminant analysis, such as linear discriminant analysis [122], which aims to discriminate data points based on their classes, CL [263] focuses on finding patterns that contrast one dataset with another [3]. Several extended CL machine learning methods have been developed. For example, there are contrastive versions of latent Dirichlet allocation, hidden Markov models, and regressions [87, 263]. More recently, including cPCA [3], CL methods for representation learning have been introduced [3, 4, 62, 203]. For example, Dirie et al. [62] proposed contrastive multivariate singular spectrum analysis (cMSSA) for decomposition of time-series data. Similar to cPCA, cMSSA could provide the interpretability by computing the PC loadings; however, cMSSA is not suitable for our case that handles non-time series data. On the other hand, contrastive variational autoencoder (cVAE) [4, 203] can be used as a CL method in cNRL. The strength of cVAE over cPCA is that it can find unique patterns in a target dataset even when its data points and latent features have nonlinear relationships. However, cVAE relies on multiple layers of neural networks (NNs), and thus the results of cVAE are

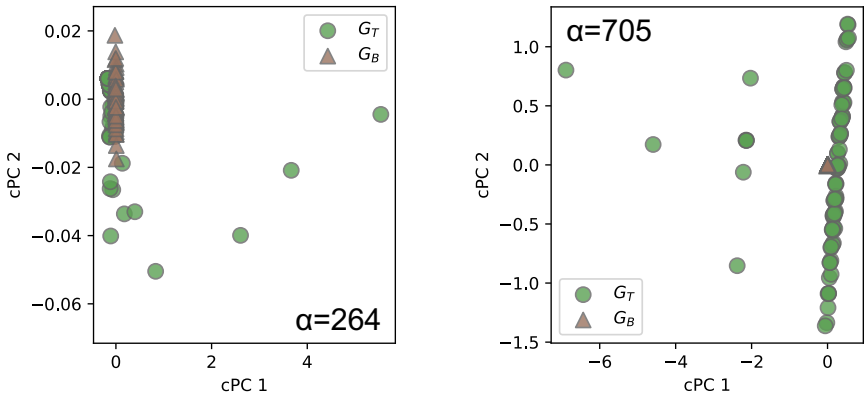
difficult to interpret as similar to other NN-based methods. Therefore, to use cVAE for interpretable cNRL, we need additional effort to help interpret the results.

5.6 Experimental Evaluation

The previous sections have introduced the concepts of cNRL and i-cNRL, as well as the related work. Sec. 5.2 has also demonstrated the effectiveness of i-cNRL in comparing social networks. To further evaluate the method, we first test i-cNRL with synthetic datasets that are generated with popular network models. Then, we see several analysis examples using i-cNRL with publicly available real-world datasets (see Tab. 5.1). Lastly, Sec. 5.6.3 provides quantitative and qualitative comparisons among i-cNRL and other potential cNRL implementations. Each subsection lists only the information closely related to the findings. Details of learning parameters and results are provided in Appendix B.3.

5.6.1 Evaluation with Network Models

i-cNRL is applied to compare two types of synthetic networks: *random* and *scale-free* networks (`N3` and N4 in Tab. 5.1). The random and scale-free networks are generated with the Gilbert's random graph [25] and the Price's preferential attachment models [179], respectively. Here we see two 2D embedding results, using one network as G_T and the other as G_B (Fig. 5.4-a, b). Each of the results shows unique patterns in G_T. The cPC loadings in Tab. 5.4 show that the Price network's unique patterns are related to the degree centralities (e.g., total-degree). This seems to be due to the fact that most nodes have the same number of links in a random network while a scale-free network contains hubs with a large number of links. In contrast, we can see that the random network's uniqueness is mostly related to k-core numbers. This is because the Price's model generates a network by adding a new node and then connecting it to another fixed number of nodes (e.g., 3 nodes) which are selected with a certain computed probability. As a result, all nodes in the network have the same k-core numbers (e.g., 3-core).

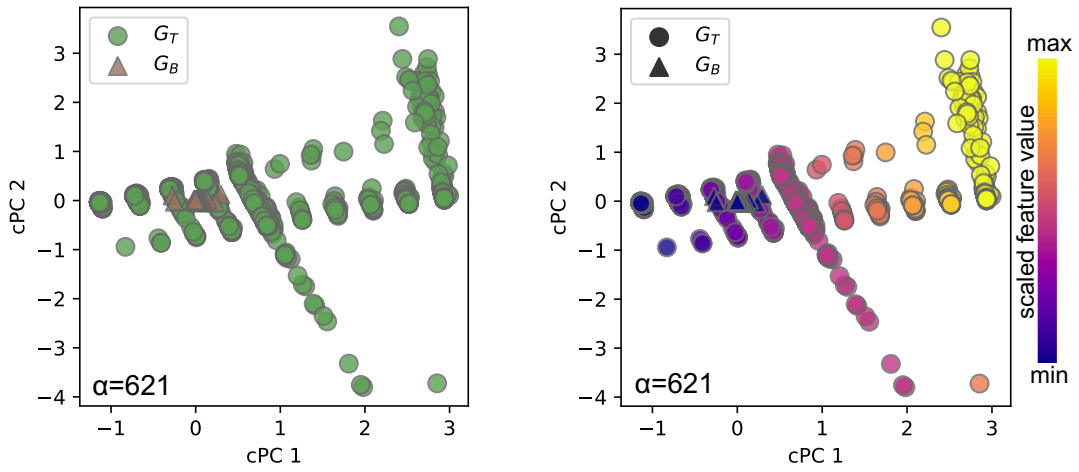


(a) G_T: Price, G_B: Random (b) G_T: Random, G_B: Price

Figure 5.4: Results for Sec. 5.6.1 with 2D embeddings by i-cNRL.

Table 5.4: The features with the top-3 absolute loadings for cPC 1 for different pairs of networks highlighted in gray.

ID	relational function f	base feature $\mathbf{x}$	cPC 1	cPC 2
G_T: Price, G_B: Random (Sec. 5.6.1)				
F2-1	$(\mathbf{x})$	total-degree	0.55	0.00
F2-1	$(\mathbf{x})$	out-degree	-0.40	0.00
F2-3	$(\mathbf{x})$	Katz	-0.19	0.06
G_T: Random, G_B: Price (Sec. 5.6.1)				
F3-1	$(\mathbf{x})$	k-core	1.00	-0.13
F3-2	$(\mathbf{x})$	total-degree	0.18	0.47
F3-3	$(\mathbf{x})$	in-degree	-0.10	-0.25
G_T: p2p-Gnutella08 , G_B: Price 2 (Sec. 5.6.2.1)				
F4-1	$(\mathbf{x})$	k-core	1.01	-0.10
F4-2	$(\mathbf{x})$	total-degree	0.22	0.30
F4-3	$(\mathbf{x})$	in-degree	-0.12	-0.17
G_T: p2p-Gnutella08 , G_B: Enhanced Price (Sec. 5.6.2.1)				
F5-1	$(\mathbf{x})$	total-degree	-0.23	0.00
F5-2	$(\mathbf{x})$	in-degree	0.12	0.05
F5-3	$(\mathbf{x})$	Katz	0.10	-0.05
G_T: LC-multiple , G_B: Combined-AP/MS (Sec. 5.6.2.2)				
F6-1	$(\Phi_{\text{mean}})(\mathbf{x})$	Katz	0.36	0.00
F6-2	$(\Phi_{\text{mean}})(\mathbf{x})$	eigenvector	-0.19	-0.01
F6-3	$(\Phi_{\text{mean}})(\mathbf{x})$	total-degree	-0.14	0.02
G_T: School-Day2 , G_B: School-Day1 (Sec. 5.6.2.3)				
F7-1	$(\Phi_{\text{mean}} \circ \Phi_{\text{max}})(\mathbf{x})$	PageRank	0.15	0.02
F7-2	$(\Phi_{\text{mean}} \circ \Phi_{\text{max}})(\mathbf{x})$	closeness	0.11	-0.04
F7-3	$(\Phi_{\text{mean}} \circ \Phi_{\text{max}})(\mathbf{x})$	betweenness	-0.09	-0.01



(a) G_T: p2p-Gnutella08, G_B: Price 2 (b) G_T: p2p-Gnutella08, G_B: Price 2

Figure 5.5: Results for Sec. 5.6.2.1. (a) presents the 2D embedding by i-cNRL. (b) shows the nodes in (a) colored by the k-core number (`F4-1` in Tab. 5.4).

5.6.2 Case Studies

5.6.2.1 Study 1: Network Model Refinement

Designing a network model that can simulate real-world networks is fundamental to understand network formation mechanisms, to perform hypothetical analyses (e.g., if there will be growth of the number of nodes, what will happen?), to generate more available datasets for machine learning, and so on [90]. This case study demonstrates the usage of i-cNRL to guide a refinement of network models.

Here, we use a peer-to-peer (P2P) network, specifically the Gnutella peer-to-peer file sharing network [192] available in SNAP[1] (`N5` in Tab. 5.1) as a modeling subject. Once we have a P2P network generation model, we can use it for analyzing network robustness, studying effective searching strategies on a P2P network, etc [157].

P2P networks are often scale-free [157], so we use the Price's model [179] to mimic a P2P network. To identify the characteristics that the Price's model does not simulate well, the P2P network (`N5`) and the Price network (`N6`) are set as G_T and G_B, respectively.

The result is shown in Fig. 5.5-a. From the cPC loadings in Tab. 5.4, we notice that the k-core number (`F4-1`) has a strong contribution to `cPC1`. Thus, we colorcode the result based on the k-core number, as shown in Fig. 5.5-b. We can clearly see that

[1]SNAP, `https://snap.stanford.edu/`, accessed: 2019-2-11

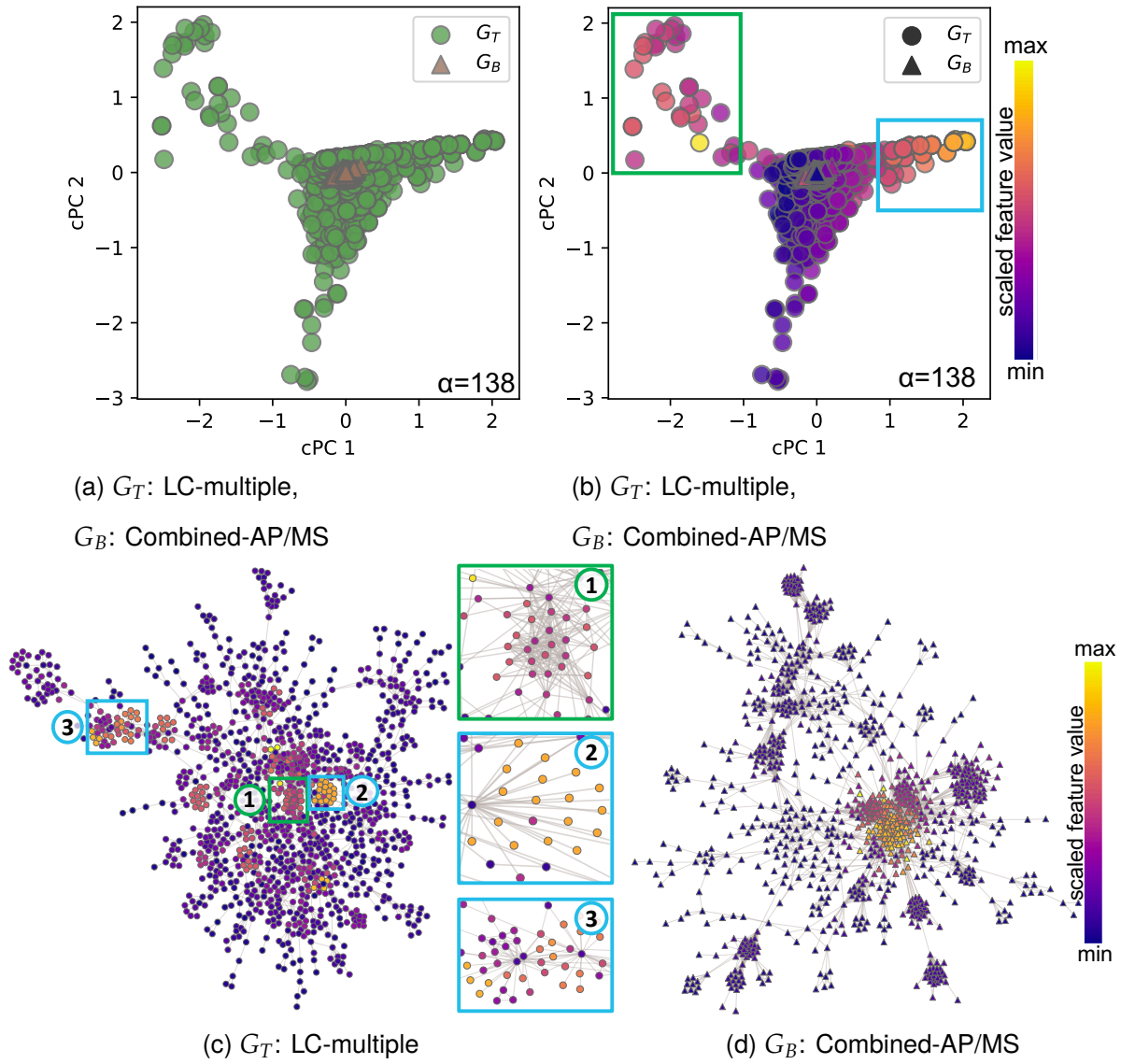
the P2P network has variations in the k-core number, but the Price network does not. Because the k-core number indicates that a node at least connects to other k nodes, the Price network makes a significant difference in the network robustness from the P2P network.

From the result above, we refine the Price model to generate various k-core numbers. As discussed in Sec. 5.6.1, the problem comes from the fact that the Price's model always adds a new node with a fixed number of links. Similar to the dual-Barabási-Albert model [174], we can avoid the problem by attaching a new node to a variable number of links according to a probability distribution. Specifically, we set the model to select the number of links from 1 to 10 with specified probabilities (for details, refer to Sec. B.3.3.2). Then, we generate a network with this model, which is referred to as the Enhanced Price (`N7`) network in Tab. 5.1. Next, we apply i-cNRL to the P2P (as G_T) and enhanced Price (as G_B) networks. The resultant cPC loadings are listed in Tab. 5.4. While G_T seems to still have the uniqueness in degree centralities, it does not in the k-core number. By iteratively performing refinement procedures such as the one above, we can build a better network model to simulate real-world networks.

5.6.2.2 Study 2: Comparison of Two Networks

This case study compares "interactome" networks—networks of physical DNA-, RNA-, and protein-protein interactions [255]. Specifically, we compare two interactome networks, Combined-AP/MS (`N8` in Tab. 5.1) and LC-multiple (`N9`), available in CCSB Interactome Database[2]. Both networks represent the interactome of the yeast *S. cerevisiae*; however, they are obtained through different analysis approaches. Combined-AP/MS is generated from two studies using a "high-throughput" approach, specifically, affinity purification/mass spectrometry (AP/MS) [48]. In contrast, LC-multiple is the literature-curated (LC) network from multiple "low-throughput" experiments [190]. Because each analysis approach has its own strength in identifying the yeast's interactions, the generated networks may vary. Comparing these networks is essential to understand the quality and characteristics of each approach [255].

[2]CCSB Interactome Database, `http://interactome.dfci.harvard.edu/`, accessed: 2019-1-28



(a) G_T: LC-multiple, G_B: Combined-AP/MS

(b) G_T: LC-multiple, G_B: Combined-AP/MS

(c) G_T: LC-multiple

(d) G_B: Combined-AP/MS

Figure 5.6: Results for Sec. 5.6.2.2. (a) presents the 2D embedding by i-cNRL. (b) shows the nodes in (a) colored by the feature—f: $(\Phi_{\text{mean}})(\mathbf{x})$, $\mathbf{x}$: the Katz centrality (F6-1 in Tab. 5.4). (c) and (d) show the network structures with the same colorcoding.

Here we analyze the uniqueness in LC-multiple by using LC-multiple and Combined-AP/MS as G_T and G_B, respectively. The 2D embedding result by i-cNRL is shown in Fig. 5.6-a. We first notice that, in G_T, there are two distinct regions: one spreading out towards the top-left and the other in the bottom-right quadrant. To understand why this pattern appears, we obtain the cPC loadings (Tab. 5.4) and color the nodes based on values of the feature that has the top cPC loading for cPC1 (i.e., F6-1, f: $(\Phi_{\text{mean}})(\mathbf{x})$ and $\mathbf{x}$: the Katz centrality). The result is shown in Fig. 5.6-b. We observe that either

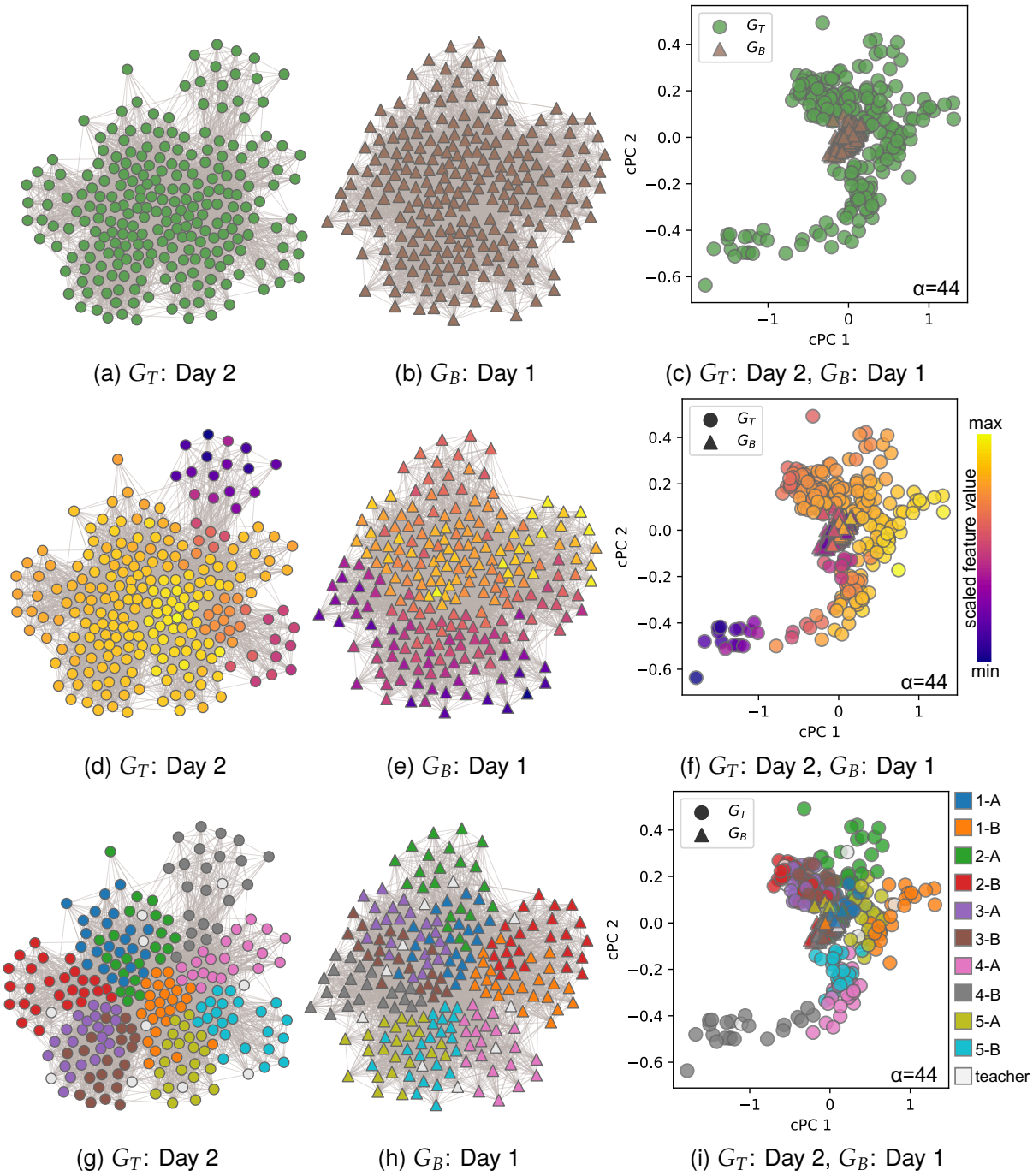
going to the left or right side along cPC1 tends to produce a high value of this feature, as annotated with the green and teal rectangles, respectively. While this feature has a strong positive loading for cPC1, another feature in Tab. 5.4—F6-2, f: $(\Phi_{\text{mean}})(\mathbf{x})$ and $\mathbf{x}$: the eigenvector centrality—has a strong negative loading. Therefore, if a node has a higher value for F6-2, it tends to be placed on the more left side in Fig. 5.6-b. This indicates that the green rectangle region in Fig. 5.6-b seems to have high values for both of these features while the teal region has low values for the latter feature (F6-2). This could happen because the eigenvector centrality tends to be low when a node is in a weakly connected region [179] while the Katz centrality is high whenever a node is linked by many others.

To visually observe the above patterns, the network structures of G_T and G_B are drawn with SFDP [115] and then colored based on the values of F6-1 (Fig. 5.6-c, d). Here only shows the largest component [179] of each network (i.e., the nodes connected with only several nodes are filtered out). Fig. 5.6-d shows that one strongly connected region around the center contains all nodes with high feature values. On the other hand, in Fig. 5.6-c, multiple regions contain nodes with high feature values. To further investigate this pattern, we select the nodes corresponding to the green and teal regions in Fig. 5.6-b and then highlight these nodes in Fig. 5.6-c. Afterward, we zoom into the related regions of the highlighted nodes. Fig. 5.6-c① shows a region related to the nodes in the green rectangle, while Fig. 5.6-c② and ③ are two example regions related to the teal rectangle region. We can see that the nodes in Fig. 5.6-c① are strongly connected, but not in Fig. 5.6-c② and ③. From these observations, i-cNRL reveals that only G_T has two different types of nodes linked to the high Katz centrality node(s) in either strongly or weakly connected region.

5.6.2.3 Study 3: Analysis of Network Changes

As an example of analyzing dynamic networks, this case study compares two different days of contact networks in a primary school[3] [210]. The networks represent face-to-face contact patterns between students and teachers, which are collected with RFID devices.

[3]Available in SocioPatterns, http://www.sociopatterns.org/, accessed: 2019-1-28



(a) G_T: Day 2 (b) G_B: Day 1 (c) G_T: Day 2, G_B: Day 1

(d) G_T: Day 2 (e) G_B: Day 1 (f) G_T: Day 2, G_B: Day 1

(g) G_T: Day 2 (h) G_B: Day 1 (i) G_T: Day 2, G_B: Day 1

Figure 5.7: Results for Sec. 5.6.2.3. (a) and (b) show the network structures of G_T and G_B. (c) presents the 2D embedding by i-cNRL. (d-f) show the colorcoded nodes in (a-c) based on the feature—f: $(\Phi_{\text{mean}} \circ \Phi_{\text{max}})(\mathbf{x})$, $\mathbf{x}$: the PageRank (F7-1 in Tab. 5.4). (g-i) show the nodes colored by the class name where the first number indicates the grade (e.g., '1-A' is the first grade class). The networks include 'teacher' nodes.

Information of the network at each day is listed in Tab. 5.1 (`N10` and `N11`). Fig. 5.7-a, b visualize the network structures drawn using SFDP. These networks also have multiple node attributes including genders, grades, and class names. In addition to multiple network centralities, we utilize the attribute information by including gender as the base feature, i.e., encoding 'male', 'female', and 'unknown' as -1, 1, and 0, respectively.

To analyze changes in contact patterns, we set the networks of the second day and the first day as G_T and G_B, respectively. Fig. 5.7-c shows the 2D embedding result. To interpret G_T's unique patterns, we review the cPC loadings listed in Tab. 5.4 and color the nodes in Fig. 5.7-a, b, c based on the learned feature `F7-1`—f: $(\Phi_{\text{mean}} \circ \Phi_{\text{max}})(\mathbf{x})$, $\mathbf{x}$: PageRank. The results are shown in Fig. 5.7-d, e, f. We can see that i-cNRL discovers that G_T has both strongly (colored with more yellow in Fig. 5.7-d, f) and weakly connected regions from others (colored with more purple), while all of G_B's nodes have relatively strong connections between each other, as seen in the laid-out result in Fig. 5.7-b.

According to the study by Stehlé et al. [210], the students tended to have more contact within the same class than between classes. To relate the class information and the found unique patterns, the nodes (i.e., students) are colorcoded based on their class, as shown in Fig. 5.7-g, h, i. From these results, we notice that i-cNRL well separates groups of students who have less (e.g., gray, pink, or teal nodes) and more (e.g., orange nodes) contact between classes in G_T.

5.6.3 Comparison with Other Potential Designs

i-cNRL utilizes DeepGL and cPCA for cNRL's two essential components, NRL and CL, to provide interpretable results. However, if the interpretability is not required, we can replace each of the learning methods with other alternatives. Here we compare three different designs for cNRL: (1) DeepGL & cPCA, (2) GraphSAGE [101] & cPCA, and (3) DeepGL & cVAE [4, 203].

5.6.3.1 Quantitative Results

Here we compare the quality of contrastive representations obtained with each design. A good contrastive representation should more widely distribute nodes in the target network than the background, and it should also show different patterns in the

target and background networks. For example, as shown in Fig. 5.3, cPCA ($\alpha = 72$) provides a better contrastive representation than PCA ($\alpha = 0$). To compare the aspects above, we use three different dissimilarity measures: dispersion ratio, Bhattacharyya distance [30], and Kullback-Leibler (KL) divergence [240] from a set of nodes in $\mathbf{Y}_B$ to that in $\mathbf{Y}_T$. The dispersion ratio represents how widely nodes in $\mathbf{Y}_T$ are scattered relative to $\mathbf{Y}_B$. The Bhattacharyya distance indicates closeness or overlaps of nodes in $\mathbf{Y}_T$ and $\mathbf{Y}_B$. The KL divergence of $\mathbf{Y}_T$ from $\mathbf{Y}_B$ shows the difference between their probability distributions of nodes. For all the above measures, the higher the value, the better the method design.

We calculate the dispersion ratio of $\mathbf{Y}_T$ to $\mathbf{Y}_B$ with: $\frac{\mathrm{tr}(\mathbf{Y}'^{\top}_T \mathbf{Y}'_T)/n_T}{\mathrm{tr}(\mathbf{Y}'^{\top}_B \mathbf{Y}'_B)/n_B}$, where $\mathbf{Y}'_T$ and $\mathbf{Y}'_B$ are the scaled matrices of $\mathbf{Y}_T$ and $\mathbf{Y}_B$ obtained by applying the standardization to a concatenated matrix of $\mathbf{Y}_T$ and $\mathbf{Y}_B$. We use $\mathbf{Y}'_T$ and $\mathbf{Y}'_B$, instead of $\mathbf{Y}_T$ and $\mathbf{Y}_B$, to avoid the scaling differences in the embedding's axes across the three designs. For the Bhattacharyya distance and KL divergence, since we do not have the exact probability distributions of $\mathbf{Y}_T$ and $\mathbf{Y}_B$, the estimation methods described in the works by Bi et al. [30] and Wang et al. [240] are employed.

For GraphSAGE, the GraphSAGE-maxpool model is selected because it produces better results [101]. We use the default parameter values indicated by the authors of GraphSage [101] and cVAE [4], except that we set 24 as the number of features leaned by GraphSAGE (see Sec. B.3.4). For the input features of GraphSAGE, we set the same base features used for DeepGL (see Tab. B.1 for details). We obtain 2D embeddings with the cPCs (with cPCA) or *salient latent variables* (with cVAE). Since cVAE relies on the probabilistic encoders, the results could be different for each trial, and thus we compute the mean value of each measure for 10 trials.

Tab. 5.5 shows a comparison of the three methods on different networks using the measures above. We can see that in general DeepGL & cPCA and GraphSAGE & cPCA have better scores than DeepGL & cVAE. Between DeepGL & cPCA and GraphSAGE & cPCA, DeepGL & cPCA tends to provide better results except for the dolphin and Karate networks, which have small numbers of nodes.

Table 5.5: Comparison of contrastive representation quality.

		dispersion ratio			Bhattacharyya			KL of $\mathbf{Y}_T$ from $\mathbf{Y}_B$		
G_T	G_B	DG& cPCA	GS& cPCA	DG& cVAE	DG& cPCA	GS& cPCA	DG& cVAE	DG& cPCA	GS& cPCA	DG& cVAE
Dolphin	Karate	174	**9,754**	1.78	1.40	**1.73**	0.93	6.82	**12.76**	0.83
P2P	Price 2	**21,744**	1,801	2.46	**7.52**	4.72	1.00	**45.73**	14.09	36.13
LC-multi.	C.-AP/MS	**376**	54	2.95	1.52	**1.76**	0.29	**18.49**	16.61	15.01
Sch.-Day2	Sch.-Day1	**57**	6	1.93	**1.81**	0.61	0.56	**5.82**	1.80	0.82

*DG=DeepGL, GS=GraphSAGE, P2P=p2p-Gnutella08, C.-AP/MS=Combined-AP/MS

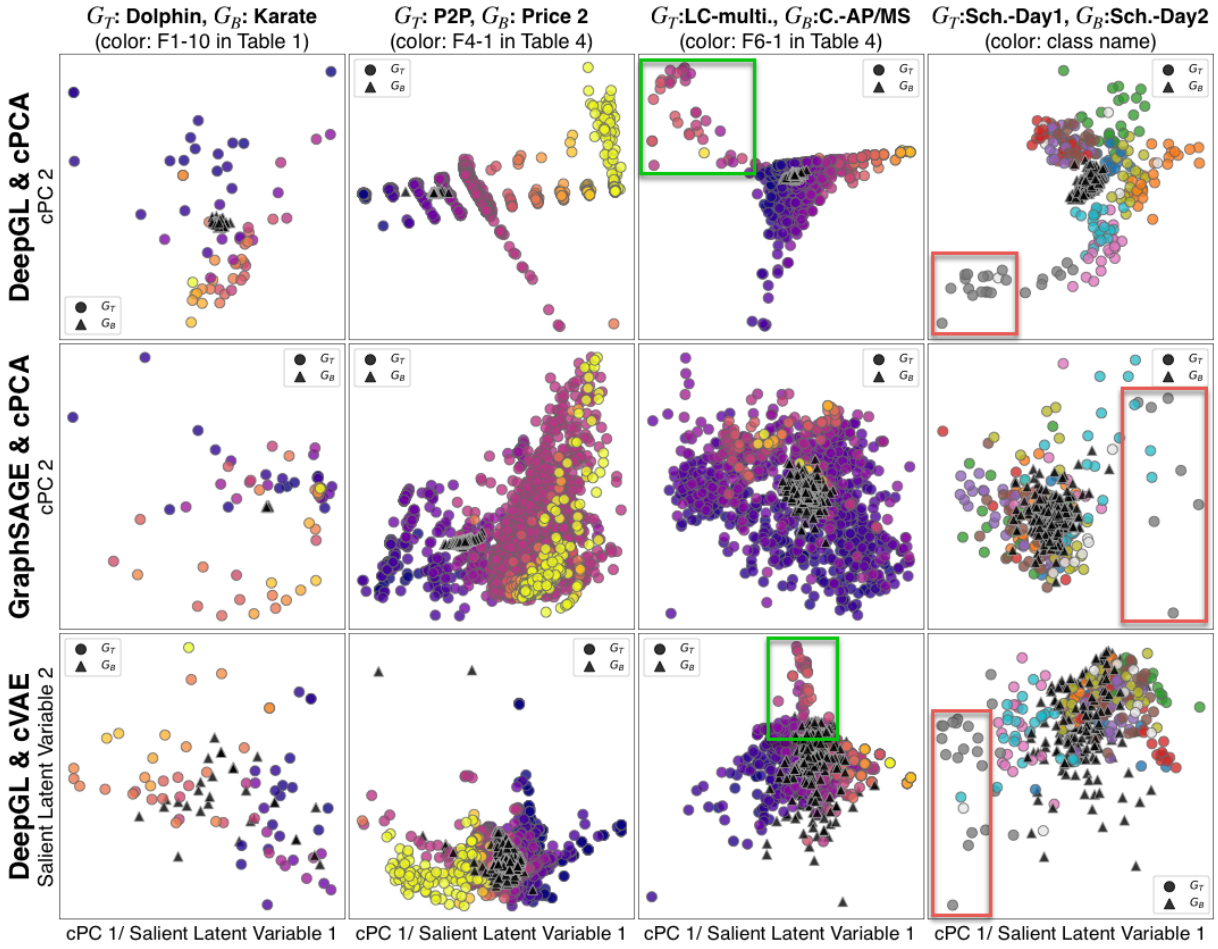


Figure 5.8: Visual comparison of the 2D embeddings.

5.6.3.2 Qualitative Results

We visually compare the embedding results to review more detailed differences, as shown in Fig. 5.8. For cVAE, we see the results that have the longest Bhattacharyya

distance from 10 trials. Because GraphSAGE and cVAE do not provide interpretable features, for the comparison, the nodes of the target network are colorcoded by the feature values from the DeepGL results. In specific, the left three columns in Fig. 5.8 are colored based on values of the feature that has the top absolute loadings for `cPC1` and the far right column is colored by their class name.

We can see that although the quality of the contrastive representation in Tab. 5.5 is different, these different designs seem to identify similar unique patterns. For instance, all the results of P2P and Price 2 show monotonic increase of the feature value (`F4-1`—k-core numbers). Also, for LC-mupltiple and Combined-AP/MS, both DeepGL & cPCA and DeepGL & cVAE depict clearly separated patterns, as indicated with the green rectangles while GraphSAGE & cPCA does not show the same pattern. Furthermore, in each result of the school networks, we can see a distinct group that consists of gray nodes, as annotated with the red rectangles.

From the above quantitative and qualitative comparisons, we can see that DeepGL & cPCA (i.e., i-cNRL) generates similar quality results when compared with the alternatives. However, the other two designs do not provide interpretable results.

5.7 Summary

This chapter introduces contrastive network representation learning (cNRL), which aims to reveal unique patterns in one network relative to another. Furthermore, I demonstrate a method of cNRL, i-cNRL, that is more generic and interpretable. With these contributions, the work in this chapter provides a new approach for network comparison.

Chapter 6
Network Data Analysis: A Visual Analytics Framework for Contrastive Network Analysis

As discussed in Chapter 5, in practice, comparative analysis of two networks is a vital task [66, 220], especially for the identification of unique structures of one network compared to another. In this chapter, I call this task *contrastive network analysis*.

Despite the demands for network comparison, there is little adequate visual analytics support. Most of the existing methods (e.g., [10, 129, 205]) presuppose the existence of node-correspondence (i.e., pairwise correspondence between nodes in two different networks) [220]. This is a critical limitation since we usually do not know such information in advance when the networks are collected from different resources. One potential solution is identifying the node-correspondence by using network alignment (or graph matching) [66, 220]. However, these algorithms notoriously have high computational costs, and thus are only suitable for treating small networks (e.g., 100 nodes). Also, there may not exist a clear correspondence between nodes.

Another approach for visual comparison of networks is based on statistical measures (e.g., network density) [73], centralities (e.g., degree centrality) [254], graphlets [144], or a combination of these [238]. For example, with graphlets [186] (small, connected, and non-isomorphic subgraph patterns in a network), the similarities of two networks can be measured by comparing the frequency of appearance of each graphlet in each network. While these approaches can provide a (dis)similarity between different networks, they compare networks only based on simple measures, which are often insufficient. Also, they only provide network-level similarities, and thus cannot compare networks at more

detailed levels (e.g., a node-level). Without a detailed-level comparison, it is difficult to find which part of a network relates to its uniqueness.

To address the above problems, this chapter introduces a novel visual analytics framework, *ContraNA*, for comparative network analysis, which integrates *contrastive network representation learning (cNRL)* described in Chapter 5 into interactive visualization. Empowered by cNRL, the framework allows for discovering unique characteristics of one network by contrasting with another in a comprehensive (i.e., using multiple advanced measures) and detailed (i.e., analyzing a node or subnetwork level) manner without node-correspondence information. Specifically, ContraNA employs an interpretable version of cNRL (i-cNRL) to provide human-understandable explanations of discovered characteristics that are further revealed by novel visual representations. I enhance i-cNRL by designing an interactive visual interface that allows analysts to integrate their domain knowledge into the automated analysis. Particularly, I introduce a method to visually identify the uniqueness in one network based on the i-cNRL result, a visual summary to intuitively inform network features that highly contribute to the result, and interactive linkings with the existing network visualizations to explain and refine the result.

In summary, the work in this chapter provides main contributions below:

- A cNRL-based visual analytics framework, ContraNA, which aims to support a new network analysis approach, named contrastive network analysis, to effectively reveal unique characteristics in one network relative to another.
- Enhancements of i-cNRL with a visual interface that provides four major abilities—**DIIF**: (1) **D**iscovery of uniqueness in networks, (2) **I**nterpretability of features generated by i-cNRL, (3) **I**ntuitive analysis with common visualizations, and (4) **F**lexibility of adjusting i-cNRL based on analysts' interests.
- A controlled user study with multiple real-world datasets, which assess the effectiveness and usefulness of ContraNA for contrastive network analysis.

6.1 Related Work

There exist three general approaches in visual comparison: juxtaposition, superposition, and explicit encoding [89]. Through a comprehensive survey, Gleicher [88] provided a framework of considerations for visual comparison, such as tasks, challenges, strategies, and designs. Here we review the relevant works in visual network comparison.

6.1.1 Static Network Comparison

Comparing two or more static networks has been a classic problem in visualization research. Alper et al. [10] presented several superposition designs for node-link and adjacency matrix visualizations to support weighted network comparison. TileMatrix [162] uses juxtaposition to place the triangular adjacency matrices of two networks onto upper and lower areas of a square matrix. On the other hand, John et al. [123] juxtaposed each pair of weighted links in a matrix cell. MatrixWave further extended this approach to support the comparison of multi-layer networks [261]

Researchers have focused on developing techniques for comparing brain networks due to their special characteristics (e.g., very dense) and importance. Shi et al. [205] opted to visualize links that are significantly different between two brain networks. Yang et al. [252] used a clustering algorithm with NodeTrix [111], a hybrid of node-link and adjacency matrix representations. Fujiwara et al. [74] enabled the comparison of a larger number of brain networks by providing an overview with dimensionality reduction. Some other domains have been addressed as well, such as genome interaction [129] and egocentric networks [155].

However, all the above methods require the information of exact node-correspondence, unlike ContraNA. While a few works [13, 137, 160] applied network alignment [66] to find node-correspondence before visualization, they do not scale well due to the computation cost.

6.1.2 Dynamic Network Comparison

Dynamic networks contain nodes and/or links changing over time. A comprehensive survey is provided by Beck et al. [26]. Here, we focus on the comparison of networks at different timestamps.

One approach is based on the juxtaposition of networks at different timestamps. Federico et al. [69] applied a 2D network layout that produces stable node positions across time and then juxtaposed networks at multiple time points in a 2.5D view. On the other hand, TimeArcs [56] lays out a network at each time point in 1D, and uses an arc diagram to display links. A wall-size display was used to juxtapose an array of networks [149]. Moreover, animated transitions have been employed, which can be viewed as juxtaposition in the temporal domain, e.g., GraphDiaries [19] and DiffAni [196].

Moreover, several works summarize a dynamic network based on the similarity of the network at each timestamp. For example, Small MultiPiles [18] groups similar weighted adjacency matrices across consecutive time points and then shows a representative matrix for each group. EgoLines [260] effectively visualizes a k-hop dynamic egocentric network with a "subway map" metaphor. van den Elzen et al. [230] utilized dimensionality reduction to overview the similarities of networks across time. A similar approach was used to visualize dynamic brain networks [20] and compare dominance variation in animal groups [46].

Lately, researchers have started to utilize time-series or topological analysis to summarize or identify important trends in a dynamic network. Examples include using graph wavelet transform to classify nodes [55] and persistent homology to capture topological changes [99]. Fujiwara et al. [77] applied change point detection [11] to segment a dynamic network and generate summaries. Several works extended this approach in other cases, such as visualizing streaming networks [130, 178].

Again, the above methods still require the information of node-correspondence. To overcome this limitation, ContraNA utilizes NRL to comprehensively capture the network's topological and semantic features.

6.1.3 Comparison without Node-Correspondence

Several systems were developed to support network comparison without the limitation of knowing node-correspondence. ManyNets [73] uses a tabular interface to list several basic network statistics (e.g., degree centrality) for each network. von Landesberger et al. [238] used graphlet frequencies and other network-statistics measures and to generate a self-organization map for arranging networks on a 2D grid. A similar approach was used by Harrigan et al. [102] to visualize egocentric networks, and by Kwon et al. [144] to show similar networks given an input network. In addition to node-level features, Gove [91] suggested network-level features (e.g., density) that are easier to interpret and faster to compute. Along this line, Graph Thumbnails [254] uses the k-core number in a nested circle packing representation of networks.

While the above methods can be used for comparing networks without node-correspondence, they lack the ability to compare networks from multiple levels. ContraNA addresses this by employing the state-of-the-art NRL method, allowing for comparison at both node and subgraph levels. Further, by leveraging CL, ContraNA focuses on revealing the uniqueness in one network relative to another, which is different from the purpose of the above works (i.e., identifying similarities of networks).

In sum, the existing methods have limited flexibility in use due to the requirement of node-correspondence or to insufficient analysis ability due to the absence of multiple-level comparison. ContraNA addresses these issues by utilizing cNRL, which is described in Chapter 5. Then, with interactive visualizations, ContraNA further supplements cNRL's limitations that are identified in Sec. 6.2.

6.2 Design Considerations

i-cNRL described in Chapter 5 can generate a contrastive representation which highlights the uniqueness of a target network. However, to thoroughly understand the uniqueness, I opt to empower the automated analysis with interactive visualization, which can tightly integrate the knowledge and adaptability of human experts with the statistical learning of machines [198, 217]. I comprehensively identify a set of limita-

tions to i-cNRL for contrastive network analysis in depth, which leads to the following design considerations for the visual analytics framework, ContraNA. In general, I aim to amplify the **D**iscovery, **I**nterpretability, **I**ntuitiveness, and **F**lexibility (**DIIF**) in visual contrastive network analysis.

DC1: ***Support the discovery*** *of whether a target network is unique compared to a background network, and which part of the network relates to the uniqueness.* The uniqueness of a target dataset relative to the base is embedded in the contrastive representation $\mathbf{Y}_T$ generated by CL-based representation learning methods, including cNRL. Many previous works attempted to display this data to reveal the uniqueness [3,4,62]. However, because $\mathbf{Y}_T$ only contains the information of the target network G_T, reviewing only $\mathbf{Y}_T$ is not sufficient to understand how well the CL method finds uniqueness. Also, it is difficult to identify which data points (i.e., network nodes in our case) highly relate to the found uniqueness. The visual analytics framework should support discovering the uniqueness and the associated nodes by presenting the information in both the target and background networks.

DC2: ***Enhance the interpretability*** *of the features learned by NRL and the cPCs generated by CL.* Investigating the relationships among the network features, cPCs, and the representation $\mathbf{Y}_T$ is important to interpret the uniqueness of G_T. While i-cNRL is designed to provide interpretable network features and cPCs, understanding them from i-cNRL's direct outputs is not straightforward. For example, DeepGL could generate a sophisticated relational function such as $(\Phi^{+}_{\text{sum}} \circ \Phi_{\text{max}} \circ \Phi^{-}_{\text{mean}})(\mathbf{x})$. Moreover, examining cPC loadings for each feature would be time-consuming when DeepGL produces many network features. The framework should provide visualizations to facilitate easy understanding of the above information.

DC3: ***Offer intuitiveness*** *in understanding a target network's uniqueness by relating it to common network visualizations.* The contrastive representation $\mathbf{Y}_T$ generated could contain complicated patterns that are difficult to understand. Thus, it is not intuitive enough to just view these patterns directly based on the i-cNRL results in the embedding space. To help analyze such patterns, the framework should provide links between the

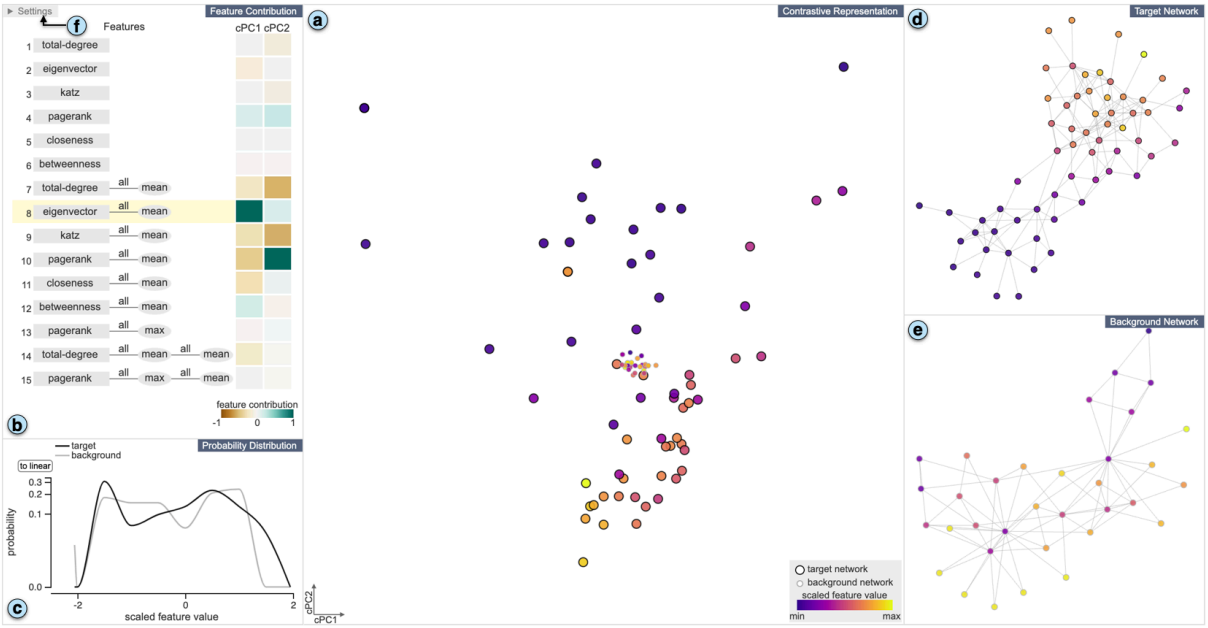


Figure 6.1: The analyst is using ContraNA to conduct a contrastive analysis of the Dolphin social network [165] (the target network) and the Zachary's karate club network [256] (the background network). (a) A contrastive representation (CR) view shows contrastive representations of target and background networks. (b) A feature contribution (FC) view visualizes network features generated by DeepGL and their contributions to each cPC (i.e., scaled cPC loadings). (c) A probability distribution (PD) view depicts target and background networks' probability distributions of the selected network feature in (b). (d)(e) A network layout (NL) view draws laid-out target and background networks, respectively. (f) The analyst can change several settings of the algorithm and visualizations from the drop-down menu.

results of i-cNRL and commonly used visualizations for network analysis, such as laid-out networks and probability distributions of network centralities.

DC4: ***Provide the flexibility*** *to interactively adjust the i-cNRL parameters to generate results based on the analysts' interest.* The results of i-cNRL heavily depend on the parameters used for each embedding step. For example, changing a value of the contrast parameter α might reveal different unique characteristics in G_T. For analysts with advanced knowledge of NRL and CL, the framework should provide abilities for interactively tuning the i-cNRL results based on their needs.

6.3 Framework Overview

Grounded by the **DIIF** design considerations, ContraNA is developed by augmenting the back-end i-cNRL algorithm with interactive visualization (Fig. 6.1) to support visual

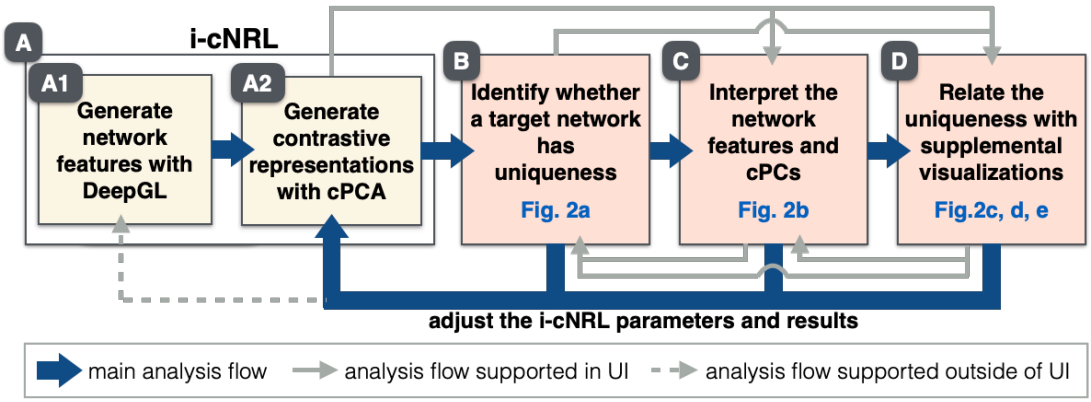


Figure 6.2: Contrastive network analysis workflow with ContraNA.

contrastive network analysis.

Fig. 6.2 shows a workflow of conducting contrastive network analysis with ContraNA. The workflow starts from (A) generation of the i-cNRL results that includes NRL with DeepGL and CL with cPCA (Fig. 5.2). Afterward, the analyst can first (B) identify whether or not there are any unique characteristics that can be only found in a target network from the contrastive representations visualized by ContraNA (Fig. 6.1-a). If such characteristics exist, to understand the uniqueness, the analyst can (C) interpret the network features and cPCs generated by i-cNRL with visualizations in Fig. 6.1-b. They can also (D) analyze the contrastive representations, network features, and cPCs by relating them with probability distributions (Fig. 6.1-c) and laid-out networks (Fig. 6.1-d, e). Based on findings during the exploration, the analyst might want to adjust the parameters of i-cNRL.

The above procedure is my expected main analysis workflow as indicated by the thick blue arrows in Fig. 6.2. However, the ContraNA UI provides the flexibility in the analysis activities, shown by the solid gray arrows in Fig. 6.2. For example, the analyst might want to start to (D) see laid-out networks in order to grasp the topological differences between target and background networks at a glance, and then (B) examine the differences with the contrastive representations. Also, such an interactive analysis often requires to go back and forth between different views in order to validate findings obtained in one view.

Due to the high computational cost of NRL with DeepGL (e.g., 20 seconds for a

network of 6,000 nodes and 20,000 links), the interactive parameter adjustment only for cPCA is supported. After the analyst updates DeepGL's parameters and generates the network features, they can analyze the results with the ContraNA UI.

ContraNA is developed as a web application. For the back-end algorithms, Python is used to integrate the i-cNRL implementation (Chapter 5). The front-end UI is implemented with a combination of HTML5, JavaScript, D3 [33], and WebGL. D3 is used for the feature contribution and probability distribution views (Fig. 6.1-b, c). For the other views (Fig. 6.1-a, d, e), WebGL is utilized to support efficient rendering and interaction as networks often consist of many nodes and links (e.g., several thousand nodes). WebSocket is used to communicate between the front- and back-end modules.

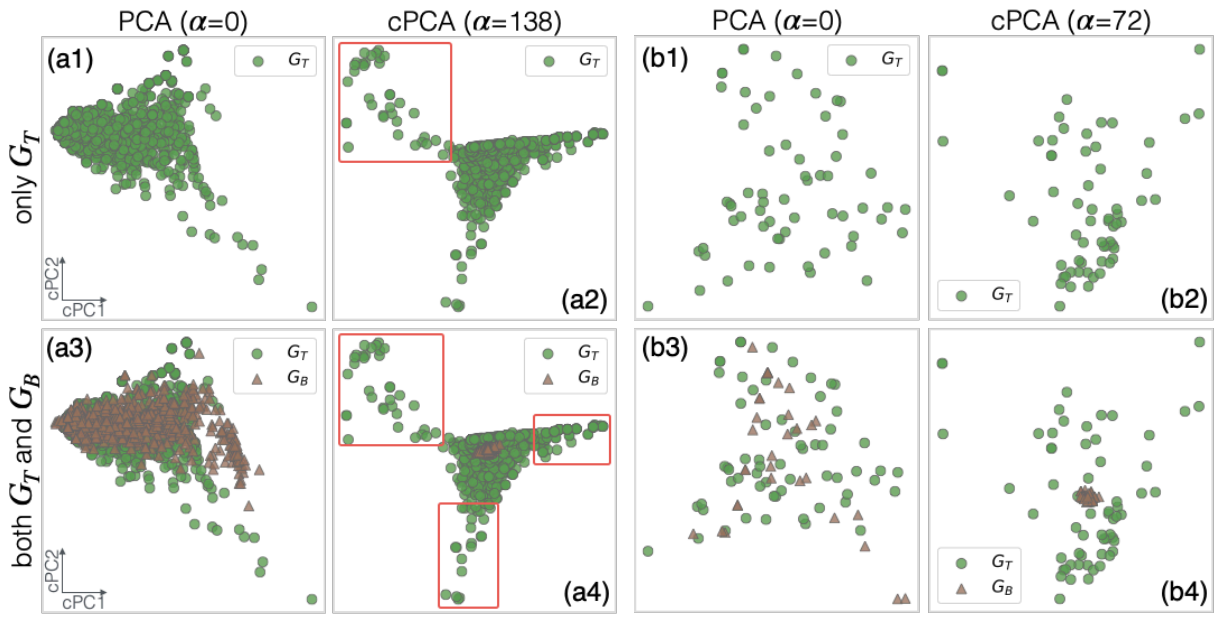
6.4 ContraNA Visual Interface

As shown in Fig. 6.1, the ContraNA UI consists of four interactively coordinated views, including a contrastive representation (CR) view, a feature contribution (FC) view, a probability distribution (PD) view, and a network layout (NL) view, designed with the considerations in Sec. 6.2. This section describes the views provided by the UI through comparison of two social networks, the Dolphin social network [165] as G_T and Zachary's karate club network [256] as G_B, which are also analyzed in Sec. 5.2. A demonstration video of the interface is available at the online site listed in Appendix A.

6.4.1 Visualization of Contrastive Representations

With the results generated by i-cNRL, the first step of the analysis workflow (Fig. 6.2-A), ContraNA's CR view (Fig. 6.1-a) visualizes the results to reveal whether or not there is uniqueness in the target network compared to the background network, serving as the following step (Fig. 6.2-B, **DC1-Discovery**).

Visual Identification of Target Network's Uniqueness. Similar to existing works [3, 4, 62, 203], a potential solution is comparing the results of ordinary PCA and cPCA. For example, given the two protein interaction networks, LC-multiple [190, 255] and Combined-AP/MS [48,255], Fig. 6.3-a1, a2 show contrastive representations $\mathbf{Y}_T$ generated with i-cNRL using the contrastive parameter $\alpha = 0$ (PCA) and $\alpha = 138$ (cPCA),



(a) G_T: LC-multiple [190, 255], G_B: Combined-AP/MS [48, 255]

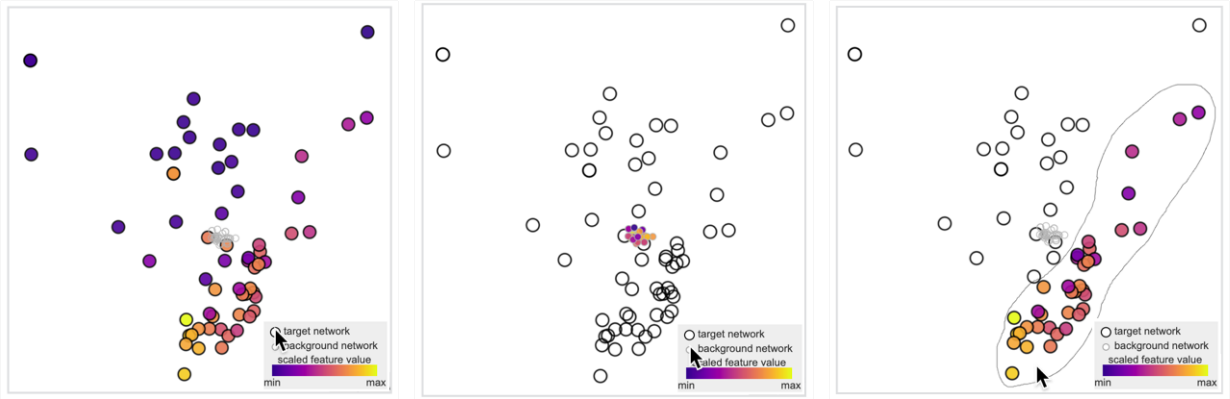
(b) G_T: Dolphin [165], G_B: Karate [256]

Figure 6.3: Two-dimensional projections based on (contrastive) representations obtained with PCA (when $\alpha = 0$) and cPCA.

respectively. In Fig. 6.3-a2, comparing with Fig. 6.3-a1, we can see the emergence of a new cluster, as annotated with the red rectangle. It indicates that cPCA successfully finds directions (i.e., cPCs) where G_T has a higher variance than G_B (i.e., the uniqueness). However, in many cases, it is difficult to see clear pattern differences between the results of PCA and cPCA, as shown in Fig. 6.3-b1, b2 with the networks of dolphins [165] as G_T and Karate club members [256] as G_B.

The problem is mainly because we do not know how nodes in a background dataset distribute in the embedding space generated by CL. Thus, I introduce a method that plots the contrastive representations of target and background datasets, $\mathbf{Y}_T$ and $\mathbf{Y}_B$, together. As shown in Fig. 6.3-a3, a4, b3, b4, $\mathbf{Y}_T$ and $\mathbf{Y}_B$ are visualized as green circles and brown triangles, respectively.

When a network has high variance in the embedded space, its nodes are widely distributed along cPCs. Thus, the uniqueness of a target network G_T can be identified by comparing the scatteredness of nodes in $\mathbf{Y}_T$ and $\mathbf{Y}_B$. As shown in Fig. 6.3-a4, b4, cPCA

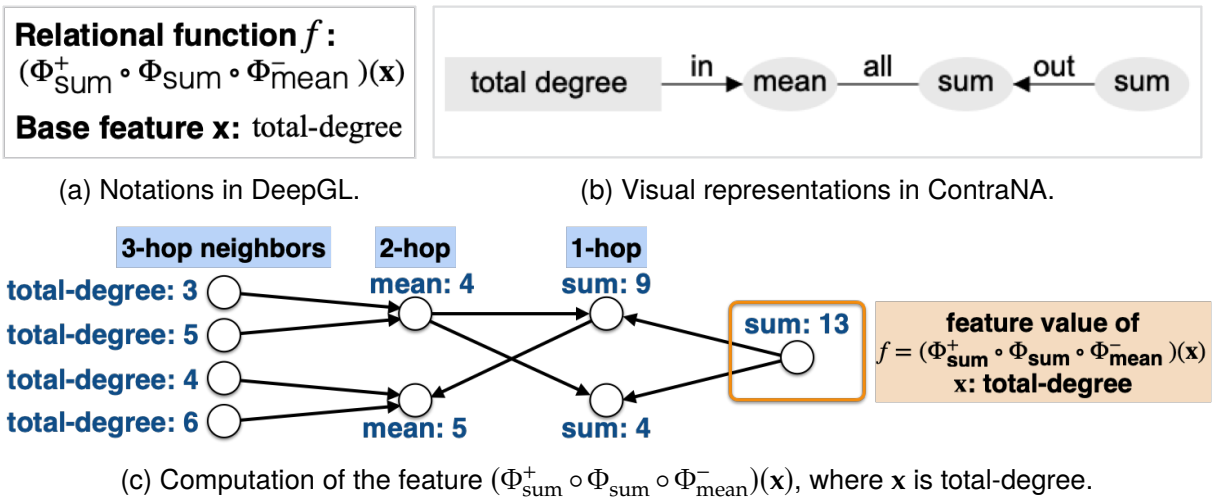


(a) Highlighting of G_T (b) Highlighting of G_B (c) Selection with lasso

Figure 6.4: Node highlighting and selection supported in the CR view.

reveals that $\mathbf{Y}_T$ has much higher scatteredness than $\mathbf{Y}_B$. Moreover, we can easily grasp which parts of a target network have strong uniqueness. Similar to other representation learning methods (e.g., PCA and MDS [222]), a distance in the embedding space of cPCA represents a dissimilarity between nodes. Thus, when the target network nodes are highly unique, they are placed far away from the nodes in the background network (e.g., the nodes in the red rectangles of Fig. 6.3-a4).

Integration into ContraNA. The above visualization is employed as the CR view of ContraNA (Fig. 6.1-a), where the values of a network feature selected in the FC view (see Fig. 6.1-b and Sec. 6.4.2) are colorcoded with a purple-yellow scheme [208]. To encode nodes in target and background networks, I first explored different shapes, including circles, triangles, and squares; however, circles and squares are hard to distinguish and triangles require much higher rendering cost with WebGL than circles and squares. I then used circles with different sizes and borders, with larger and black-border circles for the target network and smaller and gray-border circles for the background network. Moreover, the analyst can highlight the target or the background network by hovering over the corresponding legend as shown in Fig. 6.4-a, b. The CR view also provides fundamental interactions, such as zooming, panning, and lasso selection (Fig. 6.4-c). From the different scatteredness of G_T and G_B nodes in Fig. 6.1-a, we can decide that there exists uniqueness in the Dolphin network.



(a) Notations in DeepGL. (b) Visual representations in ContraNA.

(c) Computation of the feature $(\Phi^{+}_{\text{sum}} \circ \Phi_{\text{sum}} \circ \Phi^{-}_{\text{mean}})(\mathbf{x})$, where $\mathbf{x}$ is total-degree.

Figure 6.5: Representations of network features in DeepGL and ContraNA. Here, as an example, we use a complex feature (consisting of three relational feature operators) that DeepGL may produce. (a) and (b) represent the same feature: *the sum of out-neighbors of the sum of all-neighbors of the mean of in-neighbors of total-degrees*. (c) shows an example of the computational flow of this feature. In (c), the circles and arrows represent nodes and directed links of a network.

6.4.2 Interpretation of Network Features and cPCs

With the above observation from the CR view described above, we move on to interpret the network features and cPCs (Fig. 6.2-C, **DC2-Interpretability**) with the feature contribution (FC) view (Fig. 6.1-b).

Visual Representation of Network Features. The left part of the FC view lists all the network features generated by DeepGL. They usually consist of a few relational feature operators (RFOs), which are represented with mathematical notations (Fig. 6.5-a). However, it is difficult for analysts to interpret features with such notations. An intuitive visual representation of the features (Fig. 6.5-b) is designed.

As described in Sec. 5.4, a network feature learned by DeepGL consists of the base feature (e.g., total-degree), summary measures (e.g., mean), and neighbor types (e.g., in-neighbors). The FC view uses a gray rectangle and an ellipse with text labels to denote a base feature and a summary measure, respectively. Then, they are connected with a line and, to indicate the neighbor type, annotate with a text label (*in*, *out*, or *all*). Additionally, for in- and out-neighbors, an arrowhead is used to indicate the direction. Lastly, they are ordered from left to right based on the computational flow to obtain

the feature value. The resultant representation in Fig. 6.5-b visually summarizes the neighborhood relationships and the computational flow, which is further explained in Fig. 6.5-c.

Visualization of cPC Loadings. The right part of the FC view visualizes cPC loadings described in Sec. 5.4.2.1 as a heatmap. Each row and column correspond to a network feature and cPC, respectively. Scaled cPC loadings (or *feature contributions*) between $[-1, 1]$ are generated by dividing each cPC's loadings by their maximum absolute value. Then, the scaled cPC loadings are encoded with a brown-to-blue-green diverging colormap [58, 103]. The magnitude of the loading represents how strongly a feature contributes to the corresponding cPC. For example, the feature at the eighth row in Fig. 6.1-b (F8: the mean of all-neighbors' eigenvector centralities [179]), has the most influence on cPC1. Also, the sign of the loading indicates the contributed direction along the cPC (+: positive; −: negative). For example, in Fig. 6.1-a where each node is colored by F8, we can see that the feature values of G_T generally vary from low to high along the positive x-direction.

By default, ContraNA automatically selects the feature that most strongly contributes to cPC1 (e.g., F8 in Fig. 6.1-b) and highlights the corresponding row with a yellow background. The analyst can select a different feature, and all other views are updated based on the selected feature (e.g., node colors in the CR view).

By using the CR and FC views together, we discover that the uniqueness of the Dolphin network G_T highly relates to F8. From the nodes colored by the feature values (Fig. 6.1-a), we can see that the nodes around the top-left have low values while the nodes around the bottom-right tend to have higher values.

6.4.3 Relating to Common Network Visualizations

With above results, we further analyze the uniqueness by relating F8 to common network visualizations (Fig. 6.2-D). ContraNA provides two perspectives for network analysis (**DC3-Intuitiveness**): probability distributions and laid-out networks. Probability distributions are often used to compare the distributions of target and background networks' centralities (e.g., whether the degree distribution follows the power law [25]),

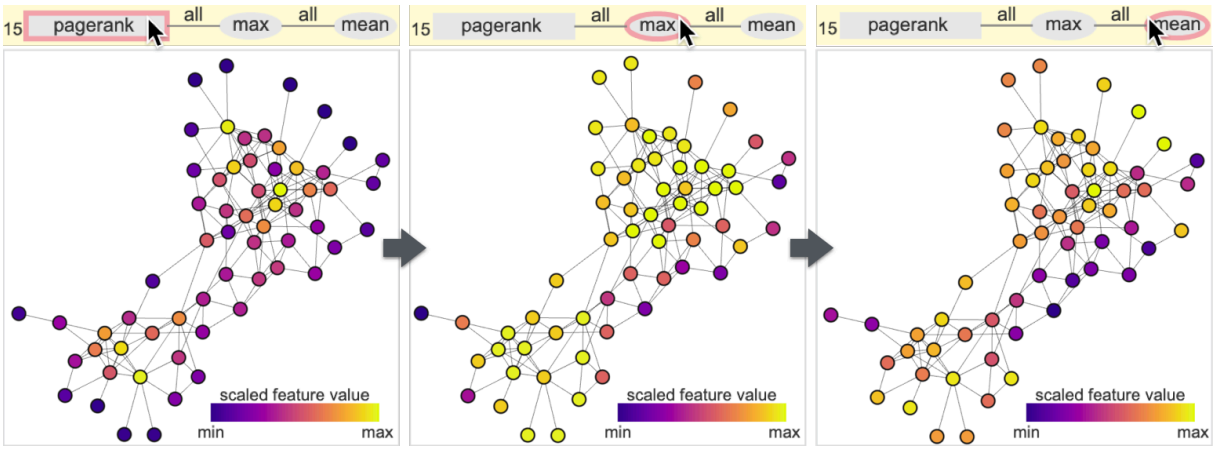


Figure 6.6: Visualization of intermediate computational results of feature F15.

and laid-out networks are helpful for viewing the topological differences (e.g., whether multiple communities exist).

Linking with Probability Distributions. The probability distribution (PD) view (Fig. 6.1-c) shows the distributions of the selected feature values in the FC view (i.e., F8 in Fig. 6.1-b), for target and background networks. Its x- and y-coordinates represent a (scaled) feature value and its probability (or relative frequency), respectively. Both logarithmic and linear scales for the y-coordinate are supported. The probability distribution lines are colorcoded with the same colors used for the node borders in the CR view (i.e., black: target network, gray: background network).

Linking with Network Layouts. The network layout (NL) view in Fig. 6.1-d, e visualizes laid-out target and background networks, with the scalable force-directed placement [115]. Same as the CR view, each node is colored based on the selected feature in the FC view (e.g., F8 in Fig. 6.1-b) and outlined in black (target network) or gray (background network). The NL view also supports several basic interactions such as zooming, panning, and a lasso selection, and is fully linked with other views. For example, by reviewing Fig. 6.1-a, b, d together, we notice that the two node groups found previously (i.e., nodes with small and high F8 values, placed around the top-left and bottom-right in Fig. 6.1-a) seem to correspond to distinct communities at the bottom-left and top-right in Fig. 6.1-d. This can be confirmed by performing lasso selection on the nodes in Fig. 6.1-a, as demonstrated in Fig. 6.4-c.

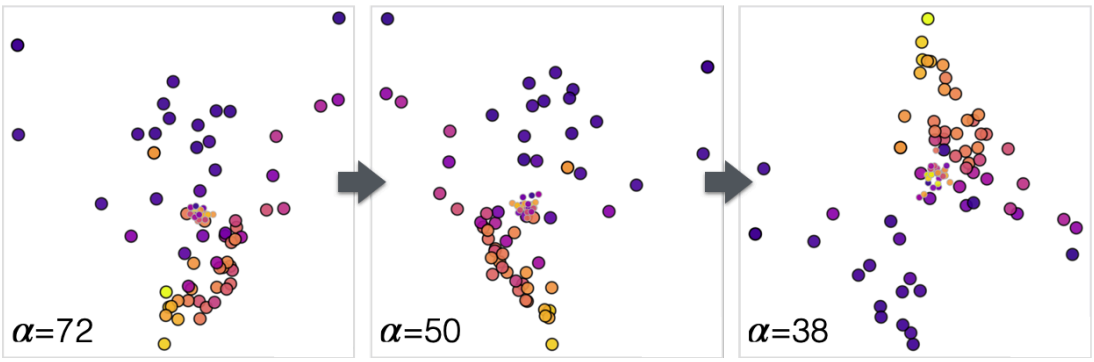
Understanding Complicated Network Features. The linkings above can be utilized to further help understand the network feature that consists of multiple RFOs. As shown in Fig. 6.6, by hovering over either the base feature or summary measure in the FC view, the NL view and the CR view show the intermediate computational results of the feature values. For instance, Fig. 6.6 (from left to right) visualizes PageRank values of G_T's nodes, the maximum of all-neighbors of PageRank values, and the mean of all-neighbors of them. Thus, the analyst can visually understand how the base feature values spread across the neighbors and how the final network feature values are derived.

Through the analysis from Fig. 6.2-A to D, we can conclude that the Dolphin network G_T has unique characteristics relative to the Karate network G_B. The uniqueness highly relates to F8: eigenvector centralities of each node's neighbors, and it clearly reveals the separation of the two communities in G_T, which cannot be seen in G_B.

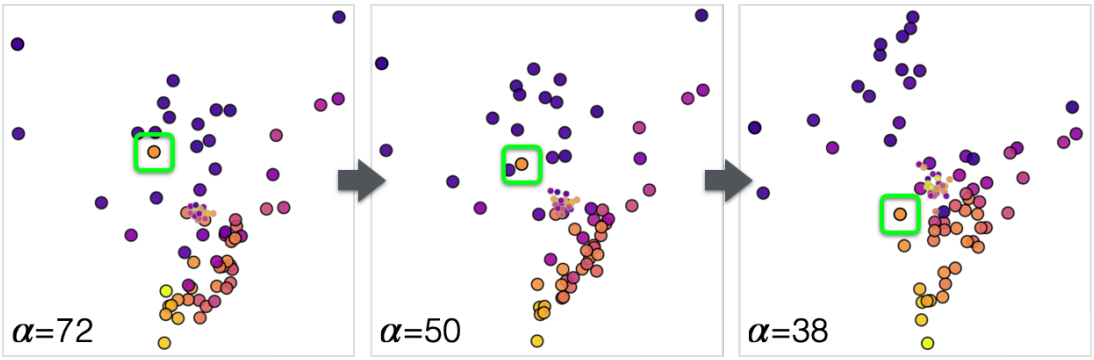
6.4.4 Refinement of Contrastive Representations

The cPCA used in i-cNRL automatically selects the contrastive parameter α and computes cPCs to generate the optimized contrastive representations, i.e., maximizing the variation in $\mathbf{X}_T$ while simultaneously minimizing the variation in $\mathbf{X}_B$ (Fig. 5.2). However, the analyst may want to loosen or strengthen the reduction of the variation of $\mathbf{X}_B$ in order to elucidate the found patterns or discover different patterns. For example, around the top-left in Fig. 6.1-a, an orange node, with a high value of F8, is mixed up with the nodes with lower values (as annotated in the green box in Fig. 6.7-b). Also, the resultant cPCs might not apt to interpret visually found patterns. For example, in Fig. 6.1-a, the value of F8 tends to increase along the diagonal line, but not along cPC1 (the x-axis). To handle such cases, ContraNA supports interactive adjustments of α and cPCs (**DC4-Flexibility**).

Adjustment of Contrastive Parameter. ContraNA allows the analyst to interactively change the contrastive parameter α with a range slider (Fig. 6.1-f), based on the efficiency of cPCA (e.g., the completion time is less than 3ms for 10,000 nodes with 10 features [76]). However, the update of α in cPCA causes an arbitrary sign flipping for each cPC, similar



(a) Without sign adjustment.



(b) With sign adjustment.

Figure 6.7: Changes of node projection by updating contrastive parameter α (a) without and (b) with sign adjustment. Corresponding animations are available online (Appendix A).

to PCA [75, 226]. Fig. 6.7-a shows an example of the flipping along both horizontal and vertical directions when α is changed, making it difficult to follow.

To address this issue, a similar solution used for PCA [226] is employed. For each of cPC1 and cPC2, the cosine similarity between the coordinates of all nodes before and after the update is computed; then if the similarity is negative, the sign generated by cPCA is flipped. Fig. 6.7-b shows the result with the sign adjustment. As α decreases to 38, the orange node annotated with the green rectangle moves toward the right-bottom and the separation of nodes with low (purple) and higher values (pink, orange, and yellow) becomes more salient.

Adjustment of Contrastive Principal Components. I introduce an interactive method for customizing cPCs, which can be used for both PCA and cPCA. First, in the CR view, a preferable axis direction for cPC1 can be drawn as a straight line (Fig. 6.8-

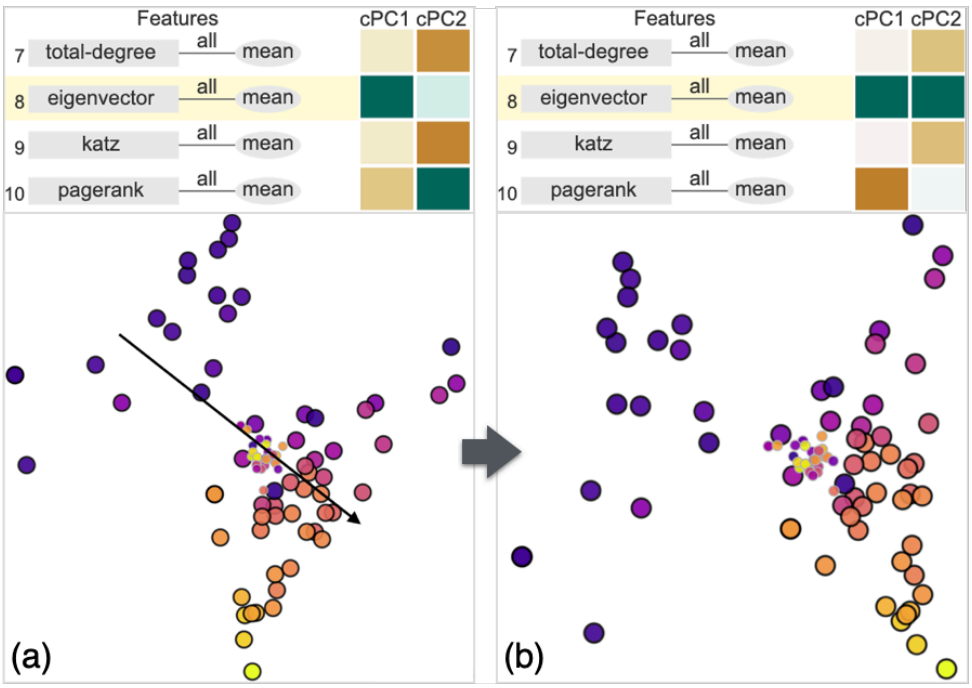


Figure 6.8: Adjustment of the cPCs. (a) and (b) show FCs of Features 7–10 and the CR view before and after the adjustment, respectively.

a). The projection is rotated based on the angle between the drawn line and cPC1 (Fig. 6.8-b). As a result, the cPC loadings shown in Fig. 6.1-b also need to be updated. Similar to the rotation in ordinary PCA [191], the cPC loadings can be obtained by simply multiplying a rotation matrix with the above user-defined angle. For example, Fig. 6.8-a, b show a subset of the cPC loadings corresponding to F7-10 before and after the rotation. We can see that F8 has a strong contribution to both cPC1 and cPC2 and F10 has a stronger contribution to cPC1 than before.

Note that we can also use a method developed by Kwon et al. [143] for general scatterplots, including cPCA projection results. It generates new axes based on the user-drawn freeform line over the plot and nonlinear transformation. However, the above linear transformation is used so that we can update cPC loadings, which are important to interpret the result of cPCA.

6.5 Controlled User Study

With the visual interface described in Sec. 6.4, we can interactively perform the same analyses demonstrated in Sec. 5.6.2. Here, as an additional evaluation of the visual interface, a controlled user study was conducted to assess the usefulness of ContraNA for contrastive network analysis. The user study aimed to answer these research questions: **(Q1)** Can analysts effectively identify unique characteristics in a target network (compared to a background network), and **(Q2)** Can analysts properly interpret and explain the found uniqueness? I expected that **Q1** would be primarily addressed by the contrastive representation view, and that all the other coordinated views would help answer **Q2** I provide the materials used for the study online (Appendix A), including the datasets listed in Tab. 6.1, their visualized results with ContraNA, and questionnaires.

6.5.1 Study Design

As far as I know, ContraNA is the first framework designed for contrastive network analysis, and thus I was not able to find a baseline system to compare against. Therefore, the following study was designed to evaluate the usability of ContraNA in terms of discovering a target network's uniqueness and interpreting it.

Datasets. As shown in Tab. 6.1, I generated random networks (Random 1, 2) with Gilbert's random graph [25] and scale-free networks (Price 1, 2) with the Price's preferential attachment models [179], as well as used several public datasets. The analysis tasks were categorized into three by carefully selecting target and background networks: (a) no uniqueness is in G_T (*# of RFOs* is `N/A`), (b) the uniqueness in G_T can be identified and interpreted with a network feature containing only the base feature (*# of RFOs* = 0), and (c) containing RFOs (*# of RFOs* $\geq$ 1). As the number of RFOs increases, a network feature becomes more complicated and the task becomes harder.

Participants. 12 participants were recruited (4 females and 8 males; aged 18–44) at a local university, with 10 from computer science and 2 from political science. There were 1 postdoc-fellow, 10 PhDs, and 1 Master's. Participants were pre-screened to ensure that they have fundamental knowledge of network science. Their self-reported

familiarity with network analysis had the median of 5 (▁▁▃▅▇), on a scale of 1 (not familiar) to 7 (use regularly). Out of 7 network centralities/measures used in the study (i.e., degree, closeness, betweenness, eigenvector, Katz centralities, PageRank, and k-core number [179]), participants' knowledge of these had the median of 3 (▁▃▁▁▃).

Apparatus. The study was conducted on an iMac (4 GHz Intel Core i7, 16GB 1,600 MHz DDR3) with a 27-inch display ($5,120 \times 2,880$ pixels), connected with an Apple Magic Mouse 2. The UI was presented with Google Chrome in full-screen mode. Because the refinement of contrastive representations (Sec. 6.4.4) was not relevant to the study tasks, the related functionalities in ContraNA were disabled.

Tasks and Design. Based on **Q1** and **Q2**, given target and background networks, participants were asked to perform comparative analysis using ContraNA and complete two subtasks **(ST1)** and **(ST2)**: **(ST1)** identifies whether or not the target network has any uniqueness compared to the background network, and **(ST2)** explains the found uniqueness (if any) or the reason of concluding there is no uniqueness. ST1 required a selection from options of *Yes*, *No*, and *I'm not sure*; for ST2, participants were asked to write down their explanation. A within-subjects design was employed for this study. Each participant completed three comparative network analysis tasks in the main study, using three different pairs of networks (Tasks A, B, and C in Tab. 6.1). The order of tasks was counterbalanced across participants.

Procedure. At the beginning, participants provided their demographics and backgrounds on a survey. A brief tutorial was then presented including explanations of the definition of the uniqueness, the above 7 network centralities/measures, the usage of ContraNA, and 3 concrete analysis examples. Afterward, participants completed a training session, allowing them to get familiar with ContraNA and the task, followed by the real study consisting of three tasks. The datasets used in the tutorials, training, and study tasks are shown in Tab. 6.1. Think aloud protocol was used during the training and task sessions. They were allowed to ask questions about the ContraNA UI and the network centralities and measures. No time limit was set for the tasks. Lastly, participants provided their feedback with the NASA TLX [105], a questionnaire about

Table 6.1: Networks used for the controlled user study and case studies, where n and l represent the numbers of nodes and links, respectively.

	Target Network G_T	*Background Network* G_B	*# of RFOs**
Tutorial 1	Price1 ($n = 100, l = 294$)	Random1 ($n = 100, l = 471$)	0
Tutorial 2	Random1 ($n = 100, l = 471$)	Random2 ($n = 100, l = 525$)	N/A
Tutorial 3,	LC-multiple [255] ($n = 1{,}536, l = 2{,}925$)	Combined-AP/MS [255] ($n = 1{,}622, l = 9{,}070$)	1
Training	School-Day2 [210] ($n = 238, l = 5{,}539$)	School-Day1 [210] ($n = 236, l = 5{,}899$)	2
Task A	Brain-Low [131] ($n = 233, l = 2{,}627$)	Brain-High [131] ($n = 246, l = 3{,}355$)	N/A
Task B,	p2p-Gnutella08 [153] ($n = 6{,}301, l = 20{,}777$)	Price2 ($n = 6{,}301, l = 18{,}897$)	0
Task C	Dolphin [165] ($n = 62, l = 159$)	Karate [256] ($n = 34, l = 78$)	1

* # of RFOs in a network feature that highly contributes to the uniqueness in G_T.

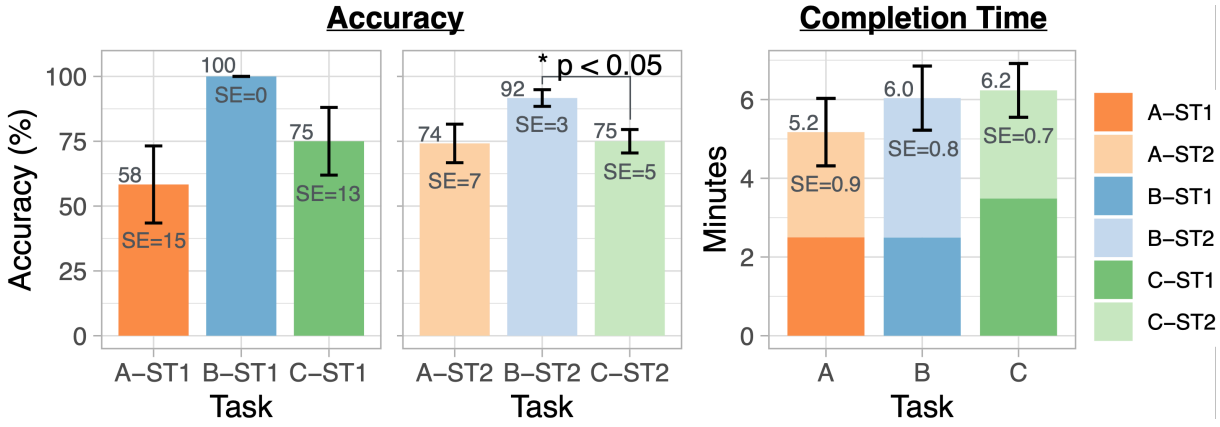


Figure 6.9: Accuracy (left) and completion time (right) for each subtask.

ContraNA's visual interface, and a semi-structured interview. The whole study lasted around 1 hour per participant.

6.5.2 Results

This section reports the controlled study results including task accuracy, completion time, and participants' subjective feedback.

Accuracy. The accuracy for each subtask is shown in Fig. 6.9-left. Two network

science experts independently rated participants' explanations in ST2 with a scale of 1 (the worst) to 5 (the best) based on correctness and comprehensiveness. Weighted Cohen's kappa coefficient indicates high reliability of the ratings ($\kappa = 0.83$, in the range of 0.81–1.00: *almost perfect agreement*) [47].

In general, Task B has the highest mean accuracy for both ST1 (100%) and ST2 (92%), which might be because the uniqueness of the target network can be understood easily with the base feature. However, for ST1, a Cochran's Q test [169] does not show any significant differences across tasks. For ST2, a Friedman test [169] reveals significant differences ($\chi^2 = 7.55$, $p < 0.05$). A post-hoc analysis using Wilcoxon signed-rank exact test with Bonferroni correction [28] indicates that Task B has significantly higher accuracy than Task C ($p < 0.05$) that has the most difficulty. Additionally, participants' scores of ST2 show a weak positive correlation (Pearson's correlation coefficient $\rho = 0.31$) with the numbers of network centralities/measures they knew, which generally represent their level of expertise in network science. Thus, higher expertise seems to help provide better explanations.

Completion Time. Fig. 6.9-right shows the completion time for each task. However, a Friedman test does not show any significant difference across tasks. There is a weak negative correlation ($\rho = -0.33$) between the completion times and the numbers of known network centralities/measures (i.e., the expertise helped finish tasks faster). For Tasks A and B, ST2 (2.7 minutes and 3.5 minutes, respectively) took longer than ST1 (both 2.5 minutes). But for Task C, it is the opposite (ST1: 3.5 minutes, ST2: 2.7 minutes). From my observation, the reason might be that participants tried to find the explanation (ST2) before deciding their answer to ST1.

Subjective Feedback. Fig. 6.10 lists participants' ratings with the NASA TLX. Generally, ST2 has higher mean values than ST1 in each task; however, a Wilcoxon signed-rank exact test does not show any significant difference in each pair of subtasks. Participants expressed relatively high mental demand and effort for performing the tasks, which is plausible because the network analysis needs high concentration. Fig. 6.11 shows the questionnaire results on the impression of ContraNA. Overall, participants felt that

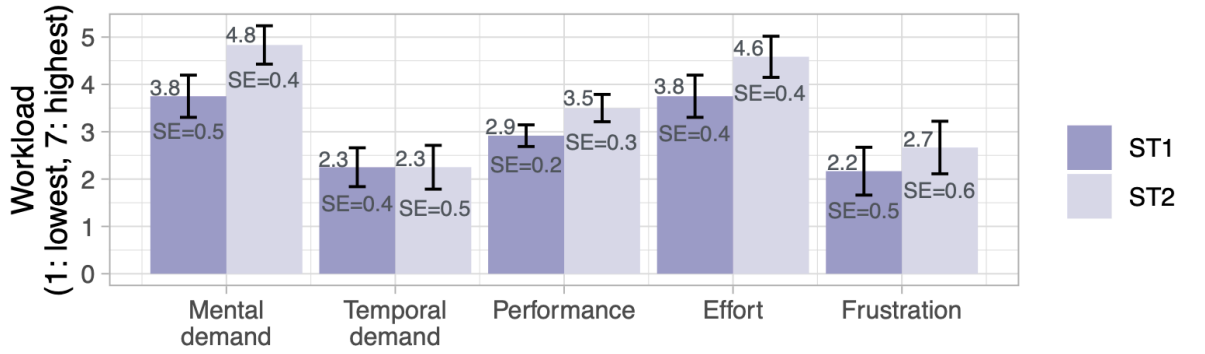


Figure 6.10: NASA TLX results (the lower the better).

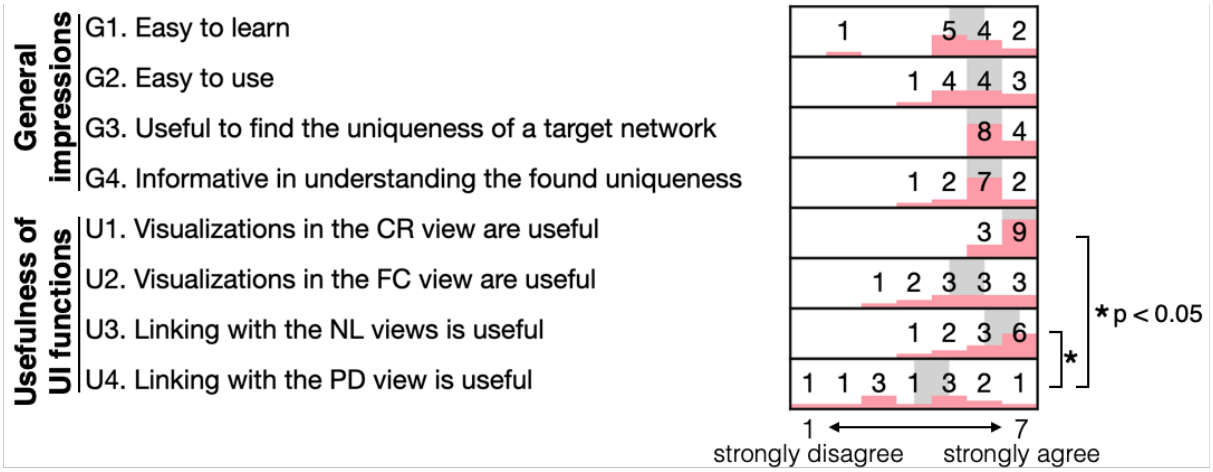


Figure 6.11: Histograms of participants' ratings on the overall impression and usefulness of each UI function (the higher the better). Numbers over the bins represent the frequency. Median ratings are indicated in gray.

ContraNA is easy to learn, easy to use, and useful to perform ST1 and ST2. For the usefulness of each UI function, the CR, FC, and NL views receive high ratings, especially the CR view, whereas the PD view has relatively low scores. Also, a Friedman test ($\chi^2 = 18.0$, $p < 0.001$) and a post-hoc analysis using the Wilcoxon signed-rank exact test with Bonferroni correction [28] show significant differences of the PD view from the CR ($p < 0.05$) and NL ($p < 0.05$) views on participants' ratings. One reason I obtained from the interviews is that the uniqueness can be identified and explained with other views, while the PD view is not necessary, although it is helpful to confirm uniqueness.

During the interview, the participants' preference was collected for the feature representations in DeepGL (Fig. 6.5-a) and ContraNA (Fig. 6.5-b). Ten out of 12 participants preferred ContraNA's representation because it is *"visually more clear"* (*p4, 6*) and *"more*

intuitive to understand" (other 8 participants). The rest of the participants preferred DeepGL's notation because using mathematical symbols has less ambiguity. Eleven participants applauded the usefulness of ContraNA's visualization of intermediate computational results (Fig. 6.6), which was used to understand complicated network features: *"Those are particularly useful because you can see the levels of how these [i.e., features] are getting computed like that"* (*p3*). Two participants with expert knowledge mentioned that they wanted to use the opposite order from the current representation (i.e., from left to right, placing RFOs first and then a base feature) because they mentally converted each network feature in this order. However, others stated that they were used to understand each feature from a base feature.

6.6 Discussion

ContraNA has been validated with a controlled user study. This section provides a thematic discussion on additional aspects of ContraNA as well as the studies.

Limitations in Visual Scalability. While the studies indicate the usefulness and effectiveness of ContraNA, it is not without limitations. ContraNA employs scatterplots, node-link diagrams, and heatmaps in its interface, but these techniques suffer from scalability issues. We can enhance these techniques with filtering, aggregation, and focus+context methods to mitigate the issues [51]. A specific scalability issue in ContraNA is the visual representation of network features, where ContraNA uses rectangles in the FC view, ellipses, lines with text labels. This may limit the number of RFOs to display in a network feature. However, this is not a major issue, because NRL methods (including DeepGL) that generate features based on relationships of node neighbors are generally utilized with only a few hops (typically 2 or 3) of neighbors [135, 194]. Another issue could be caused when NRL produces a large number of features (e.g., 100 features). This issue can be addressed by only displaying features that highly contribute to cPCs as such features are most important to interpret the cNRL results.

Ambiguity of Uniqueness. In spite of high mental demand and effort, participants achieved high accuracy in identifying (**Q1**) and explaining (**Q2**) unique structures

of a target network when it actually exists (Task B and C). However, when a target network did not have clear uniqueness (Task A), the accuracy for ST1 was relatively low, though there was no significant difference (Sec. 6.5.2). Potential reasons might be associated with the ambiguity of uniqueness and participants' expectation, as noted by *p1*: *"It wasn't too difficult [to learn and use the CR view] but I had a question of how much separation is enough to define uniqueness."* While the CR view in Task A showed the similar scatteredness between target and background networks, participants were able to find some small regions that seemed to relate to the uniqueness if they had an expectation for uniqueness. I found that all 5 participants who answered *Yes* for Task A-ST1 did not provide a convincing explanation, with mean accuracy for Task A-ST2 52% (Fig. 6.9). How to better define and inform a threshold of containing the uniqueness should be addressed in the further work.

Importance of Interpretability and cNRL. One notable result is that participants spent similar time in completing ST1 and ST2. This result differs from the original expectation: ST1 would be finished much faster because they only needed to review the CR view and select an answer, while ST2 required the use of multiple views and writing an explanation. From my observation, I noticed that although they quickly recognized the uniqueness from the CR view, before selecting the answer, they tried to understand the reasons behind to convince themselves. This points out the importance of providing the interpretability in algorithms, including NRL and CL methods. This fact also influenced the mean accuracy of Task C-ST1. Three participants chose *I'm not sure* because they were not able to completely understand why the target network was unique while the potential uniqueness was found, which may be due to their lower expertise in network science.

All the views except for the PD view seemed to be useful according to participants. From my interviews, several participants mentioned that for easier tasks (e.g., Task B) it was not necessary to use the PD view; for more difficult tasks (e.g., Task C), the PD view was not helpful to reveal the uniqueness. This indicates the limitation of network comparison based on probability distributions, which is a popular analysis

approach, and the necessity of more advanced embedding based approaches, such as cNRL. Further, when asked about how to perform similar tasks without ContraNA, participants provided approaches of either comparing probability distributions of basic network centralities or comparing laid-out networks. Also, they mentioned that they might be able to find the uniqueness with their stated approach but it would *"be awful"* (*p9*) and *"take longer"* (*p1*), and *"I might miss some uniqueness"* (*p5*). In contrast, using ContraNA is *"much easier because it supports a lot of stuff you need to deal with... Comparing the target and background in CR view is really helpful. If you see spreading patterns [of a target], it might be unique."* (*p11*).

Usage with Other Algorithms. ContraNA employs i-cNRL because of its interpretability; however, most of ContraNA functionalities are generic enough to be well adapted with other NRL and CL methods in the architecture. For example, if the interpretability is not required, DeepGL can be replaced with GraphSAGE [101], where only network features are changed. Thus, ContraNA is still applicable by updating the visual representations of features in the FC view. Similarly, we can switch cPCA with the other CL methods, such as cVAE [4,203] which can find uniqueness in a target dataset even when its data points and latent features have nonlinear relationships. As we cannot obtain the features' contributions, in this case, we can simply remove the heatmap from the FC view. Also, once other interpretable CL methods are developed, we do not need any changes to integrate them into ContraNA. Another potential extension is cooperating with link feature learning, which is also supported by DeepGL. In this case, we just need to add visual encodings of links to the views of ContraNA.

Adaption for Application Domains. As presented, the linking with laid-out networks is important to intuitively understand uniqueness. Networks are often visualized in a specific manner according to the application domain. For example, when analyzing brain networks, neuroscientists often use adjacency-matrix based visualizations or 2D/3D node-link diagrams [71]. This is because the former is useful to find correlated brain regions with matrix-reordering algorithms [27] and the latter can help relate analysis results to the actual locations in a brain. By customizing the NL views, ContraNA

can support such analysis tasks in this specific domain. Also, as shown in Sec. 6.5.2, the way to understand network features generated by DeepGL is different by the analyst. Therefore, we should consider adding settings to customize the representation of network features based on the analyst's preference.

Future Extension to Dynamic Networks. As discussed in Sec. 6.1.2, dynamic network comparison is another common analysis target in the visualization field. One case study in the previous chapter (Sec. 5.6.2.3) has demonstrated the comparison of networks obtained at two different time points. However, when we want to compare networks at more than two time points (e.g., networks at 100 different time points) or two independent dynamic networks, the comparison should consider not only the structural/topological properties but also the temporal properties. Capturing the temporal properties with network representation learning is an emerging research direction [207]. For example, researchers have started to extend GNNs for dynamic networks (often called dynamic GNNs) by integrating the concept of recurrent neural networks (RNNs) [197] and temporal self-attention mechanisms [199]. As with DeepGL for a static network, once interpretable dynamic GNN models are developed, we can use such models in the cNRL architecture for dynamic network comparison.

6.7 Summary

This chapter presents ContraNA, a visual analytics framework for network comparison, which utilizes two machine learning schemes—network representation learning and contrastive learning—together with an intuitive visual interface. ContraNA provides the capability for effectively identifying and understanding unique characteristics of one network relative to another, supporting four key capabilities as outlined by **DIIF**. The controlled user study also reflects the usefulness and effectiveness of ContraNA with carefully-designed analysis tasks.

Chapter 7
Conclusion

This dissertation introduces several new dimensionality reduction (DR) methods incorporating interactive visualization with the aim to enhance the interpretability, usability, and flexibility of DR. Specifically, ccPCA (Chapter 2) and the visual analytics system using ccPCA support analyzing a DR result by identifying the essential features characterizing clusters appeared in the DR result and highlighting information of the essential features with scalable visualizations. Notably, ccPCA and the system are designed to be applicable to any DR method. MulTiDR (Chapter 3) not only produces an effective visual overview of multivariate time-series data through a two-step DR but also provides auxiliary information for interpreting the overview by utilizing ccPCA. Incremental PCA enhanced for visualizing streaming multidimensional data (Chapter 4) adds two new capabilities to an existing incremental PCA: preservation of the user's mental map and the estimation of principal components of incoming data points that have an incomplete number of features. These capabilities are further supported with the automatic tracking of incoming data points and an uncertainty visualization of the estimated principal components. i-cNRL (Chapter 5) and ContraNA (Chapter 6) provide a new analysis approach that can identify unique characteristics in one network with respect to another network by utilizing network representation learning in conjunction with contrastive learning. i-cNRL and ContraNA are also carefully designed to provide the interpretability in the analysis results.

As the complexity of datasets increases in terms of the number of data points or the number of features, DR becomes an essential approach to visual analytics, providing an effective overview of the data and a guidance for further analysis. This dissertation has identified key challenges of DR when applied to visual analytics and has demonstrated

solutions for several tangible topics. The new DR methods and the associated visualizations can be utilized to provide better functionalities in various visual analytics systems using DR hereafter. This dissertation motivates and paves the way for future research towards designing practical DR methods for visual analytics.

Appendix A
Online Supplementary Materials

The links below include the demonstration videos of visualizations, systems, datasets, and implementations used for each corresponding chapter.

- Chapter 2: `https://takanori-fujiwara.github.io/s/dr-cl/`
- Chapter 3: `https://takanori-fujiwara.github.io/s/multidr/`
- Chapter 4: `https://takanori-fujiwara.github.io/s/inc-dr/`
- Chapter 5: `https://takanori-fujiwara.github.io/s/cnrl/`
- Chapter 6: `https://takanori-fujiwara.github.io/s/contrana/`

Appendix B
Appendix for Chapter 5

B.1 Datasets

For the evaluation, we use the datasets in various data repositories including SNAP, CCSB Interactome Database, and SocioPatterns as well as the synthetic datasets. To allow the reproducibility of this work, I provide links to the original network datasets, processed datasets, and feature matrices learned by DeepGL and GraphSAGE in `https://takanori-fujiwara.github.io/s/cnrl/`.

B.2 Implementation Details

The cNRL architecture is implemented with Python 3. The implemented cNRL architecture allows the user to apply any NRL and CL methods that provide "fit" and "transform" methods (as similar to machine learning methods supported in scikit-learn[1]). For the implementation of i-cNRL, DeepGL and cPCA are integrated into the cNRL architecture. Because there is no implementation of DeepGL available from Python[2], DeepGL is implemented with graph-tool[3]. For cPCA, the implementation available online[4] is modified to add the automatic contrastive automatic selection described in Sec. 5.4.2.3.

B.3 Experiment Details

The source code for generating the experimental results is available in `https://takanori-fujiwara.github.io/s/cnrl/`.

[1] scikit-learn, `https://scikit-learn.org/`, accessed 2020-2-10.

[2] Implementation using Java with Neo4j database is available from `https://github.com/neo4j-graph-analytics/ml-models`, accessed 2020-2-10.

[3] graph-tool, `https://graph-tool.skewed.de/`, accessed 2020-2-10.

[4] ccPCA, `https://github.com/takanori-fujiwara/ccpca`, accessed 2020-2-10.

B.3.1 Learning Parameters of i-cNRL

B.3.1.1 DeepGL Settings

Because DeepGL is introduced as a comprehensive inductive NRL framework, there are multiple settings we can adjust. The terminologies used here are the same as [194]. Refer to [194] for those not explained in this dissertation (indicated with *italic* fonts below). For all the cNRL performed in the studies, DeepGL is used with $h = 3$ and the logarithmic binning to *transform* feature values with 0.5 as the *transformation parameter*, but without the *feature diffusion*. For the other settings, generally, as many different relational feature operators and base features are used as possible for each network dataset. As for the relational feature operators, for directed networks, all the combinations of $\{\Phi_S^-, \Phi_S^+, \Phi_S\}$ are used with $S = \{\text{mean}, \text{sum}, \text{max}, \text{L}^2\text{norm}\}$ (i.e., 12 operators in total). For undirected networks, Φ_S where $S = \{\text{mean}, \text{sum}, \text{max}, \text{L}^2\text{norm}\}$ is used. As for the base feature $\mathbf{x}$, all centralities and measures available in graph-tool are used. However, for each network, some of these features have produced 'NaN' values (e.g., closeness). In that case, such features are excluded from the base features. Tab. B.1 shows the base features used for each analysis. Additionally, for *scoring* and *pruning* of the learned $\mathcal{F}_i$, the same method used in [194] is applied with the *tolerance/feature similarity threshold*, λ. As λ becomes larger, the number of features learned by NRL (i.e., d) increases. A different λ value is set for each analysis, as listed in Tab. B.1. In general, for the undirected networks, relatively higher values ($\lambda = 0.7$) are used because the number of base features used is smaller when compared with the directed networks.

B.3.1.2 cPCA Settings

For all results, cPCA is used with the automatic contrastive parameter selection and default settings. That is, the standardization is applied to each of $\mathbf{X}_T$ and $\mathbf{X}_B$ for both learning and projection and the automatic contrastive parameter selection with $\epsilon = 10^{-3}$.

B.3.2 Full Sets of cPC Loadings

The full sets of cPC loadings obtained with i-cNRL for each analysis in Sec. 5.6.1 and Sec. 5.6.2 are listed in Tab. B.2-B.5.

Table B.1: The detailed DeepGL settings for each analysis.

G_T	G_B	$\mathbf{x}$	λ
Dolphin	Karate	{total-degree, betweenness, closeness, eigenvector, PageRank, Katz}	0.7
Price	Random	{in-degree, out-degree, total-degree, PageRank, betweenness, Katz, k-core}	0.3
Random	Price	{in-degree, out-degree, total-degree, PageRank, betweenness, Katz, k-core}	0.3
p2p-Gnutella08	Price 2	{in-degree, out-degree, total-degree, PageRank, betweenness, Katz, k-core}	0.5
p2p-Gnutella08	Enhanced Price	{in-degree, out-degree, total-degree, PageRank, betweenness, Katz, k-core}	0.5
LC-multiple	Combined-AP/MS	{total-degree, betweenness, eigenvector, PageRank, Katz}	0.7
School-Day2	School-Day1	{gender, total-degree, closeness, betweenness, eigenvector, PageRank, Katz}	0.7

B.3.3 Network Generation Models and Parameters

The Gilbert's and Price's network models are used to generate Random (`N3`), Price (`N4`), and Price 2 (`N6`) in Tab. 5.1. Also, in Sec. 5.6.2.1, I have introduced the enhanced Price's network model as the solution to generate a network of which nodes have different k-core numbers—Enhanced Price (`N7`) in Tab. 5.1. The following description explains the details of the parameters used for the network generation and the enhanced Price's model.

B.3.3.1 Parameters for the Gilbert's and Price's Models

The Gilbert's model generating a random network requires the fixed probability of a connection of each pair of nodes. The probability is set to 0.05 for generating Random (`ID 4`). The Price's model requires the fixed number of out-degree of newly added nodes as its parameter. This parameter is set to 3 for both Price (`N4`) and Price 2(`N6`).

B.3.3.2 Enhanced Price's Model

For the enhanced Price's model, I modify the Price's model to be able to generate nodes with various k-core numbers. To achieve this, the enhanced Price's model allows the

Table B.2: All cPC loadings for Sec. 5.6.1.

relational function f	base feature $\mathbf{x}$	cPC 1	cPC 2
G_T: Price, G_B: Random			
$(\mathbf{x})$	in-degree	-0.19	-0.06
$(\mathbf{x})$	out-degree	-0.40	-0.00
$(\mathbf{x})$	total-degree	0.55	0.00
$(\mathbf{x})$	PageRank	0.00	-0.00
$(\mathbf{x})$	betweenness	-0.00	0.00
$(\mathbf{x})$	Katz	-0.19	0.06
$(\mathbf{x})$	k-core	-0.00	-0.00
$(\Phi^{-}_{\text{mean}})(\mathbf{x})$	in-degree	0.01	-0.00
$(\Phi^{-}_{\text{mean}} \circ \Phi^{-}_{\text{mean}})(\mathbf{x})$	in-degree	0.00	0.00
G_T: Random, G_B: Price			
$(\mathbf{x})$	in-degree	-0.10	-0.25
$(\mathbf{x})$	out-degree	0.01	-0.02
$(\mathbf{x})$	total-degree	0.18	0.47
$(\mathbf{x})$	PageRank	0.02	0.01
$(\mathbf{x})$	betweenness	-0.01	-0.00
$(\mathbf{x})$	Katz	-0.09	-0.24
$(\mathbf{x})$	k-core	1.00	-0.13
$(\Phi^{-}_{\text{mean}})(\mathbf{x})$	in-degree	0.00	0.00
$(\Phi^{-}_{\text{mean}} \circ \Phi^{-}_{\text{mean}})(\mathbf{x})$	in-degree	-0.00	0.00

user to set multiple positive integer numbers of out-degree of newly added nodes. Here we denote this input as $\kappa = \{\kappa_1, \cdots, \kappa_u\}$ where u is the length of the input. To select one number from κ when a new node is added, we need to set the probability of selecting each number. We denote the probabilities as $p = \{p_1, \cdots, p_u\}$ where $\sum p = 1$.

To generate Enhanced Price (`ID 8`), these parameters are set to κ ={1, 2, 3, 4, 5, 6, 7, 8, 9, 10} and p ={0.3, 0.25, 0.15, 0.1, 0.075, 0.05, 0.025, 0.025, 0.0125, 0.0125}.

B.3.4 Settings of GraphSAGE and cVAE

Here describes the detailed settings and parameters of GraphSAGE and cVAE used in Sec. 5.6.3. The source code provided by the authors of GraphSAGE[5] and cVAE[6] is used. For GraphSAGE, the unsupervised model `graphsage_maxpool` is used with 24 as the number of features learned (i.e., `dim_1`=12 and `dim_2`=12) while following the default

[5]GraphSAGE: `https://github.com/williamleif/GraphSAGE`, accessed 2020-2-10.
[6]Contrastive VAE: `https://github.com/abidlabs/contrastive_vae`, accessed 2020-2-10.

Table B.3: All cPC loadings for Sec. 5.6.2.1.

relational function f	base feature $\mathbf{x}$	cPC 1	cPC 2
G_T: p2p-Gnutella08 , G_B: Price 2			
$(\mathbf{x})$	in-degree	-0.12	-0.17
$(\mathbf{x})$	out-degree	0.04	-0.00
$(\mathbf{x})$	total-degree	0.22	0.30
$(\mathbf{x})$	PageRank	0.04	0.00
$(\mathbf{x})$	betweenness	-0.00	-0.00
$(\mathbf{x})$	Katz	-0.11	-0.13
$(\mathbf{x})$	k-core	1.01	-0.10
$(\Phi^{-}_{\text{mean}})(\mathbf{x})$	in-degree	-0.00	0.00
$(\Phi^{-}_{\text{mean}})(\mathbf{x})$	out-degree	0.00	0.00
$(\Phi^{-}_{\text{mean}})(\mathbf{x})$	betweenness	0.00	-0.00
$(\Phi_{\text{mean}})(\mathbf{x})$	out-degree	-0.00	-0.00
$(\Phi^{-}_{\text{mean}} \circ \Phi^{-}_{\text{mean}})(\mathbf{x})$	in-degree	-0.00	-0.00
$(\Phi^{-}_{\text{mean}} \circ \Phi^{-}_{\text{mean}})(\mathbf{x})$	out-degree	0.00	-0.00
$(\Phi_{\text{mean}} \circ \Phi^{-}_{\text{mean}})(\mathbf{x})$	out-degree	0.00	0.00
G_T: p2p-Gnutella08 , G_B: Enhanced Price			
$(\mathbf{x})$	in-degree	0.12	0.05
$(\mathbf{x})$	out-degree	0.05	-0.00
$(\mathbf{x})$	total-degree	-0.23	0.00
$(\mathbf{x})$	PageRank	-0.00	0.00
$(\mathbf{x})$	betweenness	0.00	-0.00
$(\mathbf{x})$	Katz	0.10	-0.05
$(\mathbf{x})$	k-core	0.00	-0.00
$(\Phi^{-}_{\text{mean}})(\mathbf{x})$	in-degree	-0.00	0.00
$(\Phi^{-}_{\text{mean}})(\mathbf{x})$	out-degree	0.00	-0.00
$(\Phi^{-}_{\text{mean}})(\mathbf{x})$	betweenness	-0.00	0.00
$(\Phi_{\text{mean}})(\mathbf{x})$	out-degree	0.00	0.00
$(\Phi^{-}_{\text{mean}} \circ \Phi^{-}_{\text{mean}})(\mathbf{x})$	in-degree	-0.00	-0.00
$(\Phi^{-}_{\text{mean}} \circ \Phi^{-}_{\text{mean}})(\mathbf{x})$	out-degree	0.00	-0.00
$(\Phi_{\text{mean}} \circ \Phi^{-}_{\text{mean}})(\mathbf{x})$	out-degree	0.00	0.00

values for other parameters (e.g., `learning_rate`=0.00001 and `model_size`='small'). cVAE is used with the default parameters (i.e., `intermediate_dim` = 12, `latent_dim` = 2, `batch_size` = 64, and `epochs` = 500).

B.3.5 Automatic Contrastive Parameter Selection

Fig. B.1 shows transitions of α value during the automatic selection in i-cNRL. For all the experiments, we can see that α reaches the convergence before 10 iterations.

Table B.4: All cPC loadings for Sec. 5.6.2.2.

relational function f	base feature $\mathbf{x}$	cPC 1	cPC 2
$(\mathbf{x})$	total-degree	-0.02	0.13
$(\mathbf{x})$	betweenness	-0.00	-0.00
$(\mathbf{x})$	eigenvector	0.01	0.12
$(\mathbf{x})$	PageRank	0.01	0.00
$(\mathbf{x})$	Katz	-0.00	-0.24
$(\Phi_{\text{mean}})(\mathbf{x})$	total-degree	-0.14	0.02
$(\Phi_{\text{mean}})(\mathbf{x})$	betweenness	0.00	0.00
$(\Phi_{\text{mean}})(\mathbf{x})$	eigenvector	-0.19	-0.01
$(\Phi_{\text{mean}})(\mathbf{x})$	PageRank	-0.00	-0.00
$(\Phi_{\text{mean}})(\mathbf{x})$	Katz	0.36	0.00
$(\Phi_{\text{max}})(\mathbf{x})$	betweenness	0.00	0.00
$(\Phi_{\text{max}})(\mathbf{x})$	PageRank	-0.01	0.00
$(\Phi_{\text{mean}} \circ \Phi_{\text{mean}})(\mathbf{x})$	total-degree	-0.03	-0.01
$(\Phi_{\text{mean}} \circ \Phi_{\text{mean}})(\mathbf{x})$	betweenness	0.00	-0.00
$(\Phi_{\text{mean}} \circ \Phi_{\text{mean}})(\mathbf{x})$	PageRank	-0.01	-0.01
$(\Phi_{\text{mean}} \circ \Phi_{\text{max}})(\mathbf{x})$	betweenness	0.00	0.00
$(\Phi_{\text{mean}} \circ \Phi_{\text{max}})(\mathbf{x})$	PageRank	0.01	0.01
$(\Phi_{\text{max}} \circ \Phi_{\text{max}})(\mathbf{x})$	betweenness	-0.00	-0.00
$(\Phi_{\text{max}} \circ \Phi_{\text{max}})(\mathbf{x})$	PageRank	0.00	-0.00

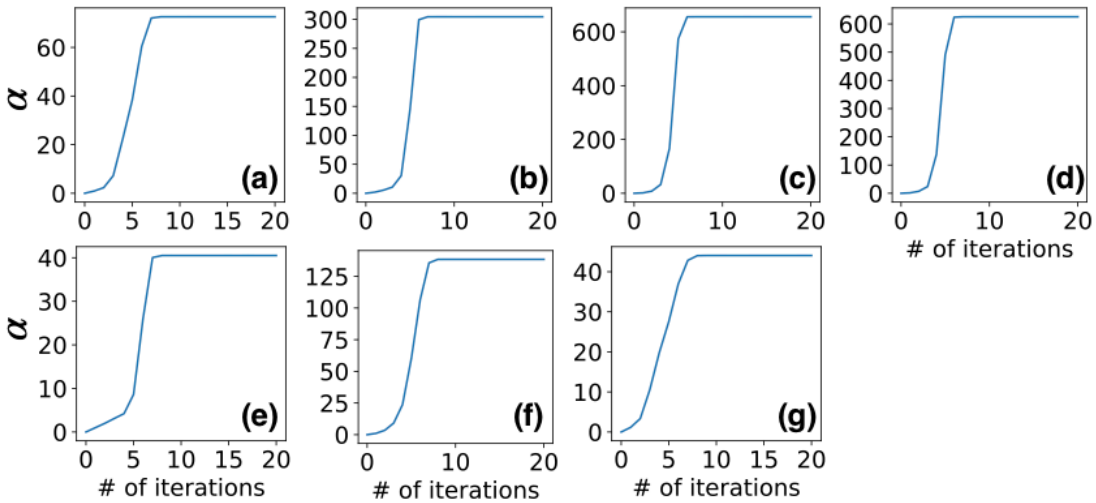


Figure B.1: Transitions of α with the automatic selection: (a) G_T: Dolphin, G_B: Karate, (b) G_T: Price, G_B: Random, (c) G_T: Random, G_B: Price, (d) G_T: p2p-Gnutella08, G_B: Price 2, (e) G_T: p2p-Gnutella08, G_B: Enhanced Price, (f) G_T: LC-multiple, G_B: Combined-AP/MS, and (g) G_T: School-Day2, G_B: School-Day2.

Table B.5: All cPC loadings for Sec. 5.6.2.3.

relational function f	base feature $\mathbf{x}$	cPC 1	cPC 2
$(\mathbf{x})$	total-degree	0.05	0.03
$(\mathbf{x})$	closeness	-0.01	0.00
$(\mathbf{x})$	betweenness	-0.01	0.00
$(\mathbf{x})$	eigenvector	-0.06	0.02
$(\mathbf{x})$	PageRank	0.00	-0.04
$(\mathbf{x})$	Katz	0.02	-0.02
$(\mathbf{x})$	gender	-0.01	-0.00
$(\Phi_{\text{mean}})(\mathbf{x})$	total-degree	0.07	0.01
$(\Phi_{\text{mean}})(\mathbf{x})$	betweenness	-0.02	-0.01
$(\Phi_{\text{mean}})(\mathbf{x})$	gender	-0.01	-0.00
$(\Phi_{\text{sum}})(\mathbf{x})$	gender	0.01	0.00
$(\Phi_{\text{max}})(\mathbf{x})$	total-degree	-0.01	0.03
$(\Phi_{\text{max}})(\mathbf{x})$	closeness	0.03	-0.00
$(\Phi_{\text{max}})(\mathbf{x})$	betweenness	-0.02	-0.00
$(\Phi_{\text{max}})(\mathbf{x})$	eigenvector	-0.03	0.01
$(\Phi_{\text{max}})(\mathbf{x})$	PageRank	0.03	-0.01
$(\Phi_{\text{max}})(\mathbf{x})$	Katz	-0.01	-0.02
$(\Phi_{\text{max}})(\mathbf{x})$	gender	-0.00	0.00
$(\Phi_{\text{L}^2\text{norm}})(\mathbf{x})$	gender	0.01	0.00
$(\Phi_{\text{mean}} \circ \Phi_{\text{mean}})(\mathbf{x})$	total-degree	-0.06	-0.01
$(\Phi_{\text{mean}} \circ \Phi_{\text{mean}})(\mathbf{x})$	betweenness	0.04	0.00
$(\Phi_{\text{mean}} \circ \Phi_{\text{mean}})(\mathbf{x})$	gender	0.02	0.01
$(\Phi_{\text{mean}} \circ \Phi_{\text{max}})(\mathbf{x})$	total-degree	-0.05	0.04
$(\Phi_{\text{mean}} \circ \Phi_{\text{max}})(\mathbf{x})$	closeness	0.11	-0.04
$(\Phi_{\text{mean}} \circ \Phi_{\text{max}})(\mathbf{x})$	betweenness	-0.09	-0.01
$(\Phi_{\text{mean}} \circ \Phi_{\text{max}})(\mathbf{x})$	eigenvector	-0.08	0.03
$(\Phi_{\text{mean}} \circ \Phi_{\text{max}})(\mathbf{x})$	PageRank	0.15	0.02
$(\Phi_{\text{mean}} \circ \Phi_{\text{max}})(\mathbf{x})$	Katz	-0.06	-0.05
$(\Phi_{\text{mean}} \circ \Phi_{\text{max}})(\mathbf{x})$	gender	0.00	-0.00
$(\Phi_{\text{max}} \circ \Phi_{\text{mean}})(\mathbf{x})$	betweenness	-0.00	0.00
$(\Phi_{\text{max}} \circ \Phi_{\text{mean}})(\mathbf{x})$	gender	0.00	0.00
$(\Phi_{\text{max}} \circ \Phi_{\text{sum}})(\mathbf{x})$	gender	0.00	0.00
$(\Phi_{\text{max}} \circ \Phi_{\text{L}^2\text{norm}})(\mathbf{x})$	gender	0.00	0.00

References

[1] H. Abdi and D. Valentin. Multiple correspondence analysis. *Encyclopedia of Measurement and Statistics*, pp. 651–657, 2007.

[2] A. Abid, M. J. Zhang, V. K. Bagaria, and J. Zou. Contrastive principal component analysis. *arXiv:1709.06716*, 2017.

[3] A. Abid, M. J. Zhang, V. K. Bagaria, and J. Zou. Exploring patterns enriched in a dataset with contrastive principal component analysis. *Nature Communications*, 9(1):2134, 2018.

[4] A. Abid and J. Zou. Contrastive variational autoencoder enhances salient features. *arXiv:1902.04601*, 2019.

[5] E. Acuna and C. Rodriguez. The treatment of missing values and its effect on classifier accuracy. In *Classification, Clustering, and Data Mining Applications*, pp. 639–647. Springer, 2004.

[6] A. Adadi and M. Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence. *IEEE Access*, 6:52138–52160, 2018.

[7] E. Ahmed, I. Yaqoob, I. A. T. Hashem, I. Khan, A. I. A. Ahmed, M. Imran, and A. V. Vasilakos. The role of big data analytics in internet of things. *Computer Networks*, 129:459–471, 2017.

[8] D. Akca. Generalized procrustes analysis and its applications in photogrammetry. Technical report, ETH Zurich, 2003.

[9] A. B. Alencar, K. Börner, F. V. Paulovich, and M. C. F. de Oliveira. Time-aware visualization of document collections. In *Proc. SAC*, pp. 997–1004, 2012.

[10] B. Alper, B. Bach, N. Henry Riche, T. Isenberg, and J.-D. Fekete. Weighted graph comparison techniques for brain connectivity analysis. In *Proc. CHI*, pp. 483–492, 2013.

[11] S. Aminikhanghahi and D. J. Cook. A survey of methods for time series change point detection. *Knowledge and Information Systems*, 51(2):339–367, 2017.

[12] E. Anderson. The species problem in iris. *Annals of the Missouri Botanical Garden*, 23(3):457–509, 1936.

[13] K. Andrews, M. Wohlfahrt, and G. Wurzinger. Visual graph comparison. In *Proc. IV*, pp. 62–67, 2009.

[14] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. OPTICS: Ordering points to identify the clustering structure. In *Proc. SIGMOD*, pp. 49–60, 1999.

[15] D. Arendt, D. Best, R. Burtner, and C. L. Paul. CyberPetri at CDX 2016: Real-time network situation awareness. In *Proc. VizSec*, pp. 1–4, 2016.

[16] B. Bach, P. Dragicevic, D. Archambault, C. Hurter, and S. Carpendale. A review of temporal data visualizations based on space-time cube operations. In *Proc. EuroVis*, 2014.

[17] B. Bach, P. Dragicevic, D. Archambault, C. Hurter, and S. Carpendale. A descriptive framework for temporal data visualizations based on generalized space-time cubes. *Computer Graphics Forum*, 36(6):36–61, 2017.

[18] B. Bach, N. Henry-Riche, T. Dwyer, T. Madhyastha, J.-D. Fekete, and T. Grabowski. Small MultiPiles: Piling time to explore temporal patterns in dynamic networks. *Computer Graphics Forum*, 34(3):31–40, 2015.

[19] B. Bach, E. Pietriga, and J.-D. Fekete. GraphDiaries: Animated transitions and temporal navigation for dynamic networks. *IEEE Transactions on Visualization and Computer Graphics*, 20(5):740–754, 2014.

[20] B. Bach, C. Shi, N. Heulot, T. Madhyastha, T. Grabowski, and P. Dragicevic. Time Curves: Folding time to visualize patterns of temporal evolution in data. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):559–568, 2016.

[21] R. Ballester-Ripoll, P. Lindstrom, and R. Pajarola. TTHRESH: Tensor compression for multidimensional visual data. *IEEE Transactions on Visualization and Computer Graphics*, 2019 (Early Access).

[22] O. Banos, R. García, J. Holgado-Terriza, M. Damas, H. Pomares, I. Rojas, A. Saez, and C. Villalonga. mHealthDroid: A novel framework for agile development of mobile health applications. In *Proc. IWAAL*, vol. 8868, pp. 91–98, 2014.

[23] O. Banos, C. Villalonga, R. García, A. Saez, M. Damas, J. Holgado-Terriza, S. Lee, H. Pomares, and I. Rojas. Design, implementation and validation of a novel open framework for agile development of mobile health applications. *BioMedical Engineering OnLine*, 14:S6, 08 2015.

[24] Z. Bar-Joseph, D. K. Gifford, and T. S. Jaakkola. Fast optimal leaf ordering for hierarchical clustering. *Bioinformatics*, 17(1):S22–S29, 2001.

[25] A.-L. Barabási et al. *Network science*. Cambridge university press, 2016.

[26] F. Beck, M. Burch, S. Diehl, and D. Weiskopf. A taxonomy and survey of dynamic graph visualization. *Computer Graphics Forum*, 36(1):133–159, 2017.

[27] M. Behrisch, B. Bach, N. Henry Riche, T. Schreck, and J.-D. Fekete. Matrix reordering methods for table and network visualization. *Computer Graphics Forum*, 35(3):693–716, 2016.

[28] A. Benavoli, G. Corani, and F. Mangili. Should we really use post-hoc tests based on mean-ranks? *Journal of Machine Learning Research*, 17(1):152–161, 2016.

[29] G. Bhanot, A. Gara, P. Heidelberger, E. Lawless, et al. Optimizing task layout on the Blue Gene/L supercomputer. *IBM Journal of Research and Development*, 49(2.3):489–500, 2005.

[30] S. Bi, S. Prabhu, S. Cogan, and S. Atamturktur. Uncertainty quantification metrics with varying statistical information in model calibration and validation. *AIAA Journal*, pp. 3570–3583, 2017.

[31] I. Borg and P. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer Series in Statistics, 2005.

[32] H. Bosch, D. Thom, F. Heimerl, E. Püttmann, S. Koch, R. Krüger, M. Wörner, and T. Ertl. ScatterBlogs2: Real-time monitoring of microblog messages through user-guided filtering. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2022–2031, 2013.

[33] M. Bostock, V. Ogievetsky, and J. Heer. D^3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, 2011.

[34] M. Brehmer, M. Sedlmair, S. Ingram, and T. Munzner. Visualizing dimensionally-reduced data: Interviews with analysts and a characterization of task sequences. In *Proc. BELIV*, pp. 1–8, 2014.

[35] R. Bro, E. Acar, and T. G. Kolda. Resolving the sign ambiguity in the singular value decomposition. *Journal of Chemometrics*, 22(2):135–140, 2008.

[36] B. Broeksema, A. C. Telea, and T. Baudel. Visual analysis of multi-dimensional categorical data sets. *Computer Graphics Forum*, 32(8):158–169, 2013.

[37] H. Cai, V. W. Zheng, and K. C.-C. Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616–1637, 2018.

[38] R. J. G. B. Campello, D. Moulavi, and J. Sander. Density-based clustering based on hierarchical density estimates. In *Proc. PAKDD*, pp. 160–172, 2013.

[39] N. Cao, C. Lin, Q. Zhu, Y.-R. Lin, X. Teng, and X. Wen. Voila: Visual anomaly detection and monitoring with streaming spatiotemporal data. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):23–33, 2018.

[40] J. D. Carroll and J.-J. Chang. Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition. *Psychometrika*, 35(3):283–319, 1970.

[41] H. Chen and B. M. Sharp. Content-rich biological network constructed by mining pubmed abstracts. *BMC Bioinformatics*, 5(1):147, 2004.

[42] J. Chen, T. Ma, and C. Xiao. FastGCN: Fast learning with graph convolutional networks via importance sampling. In *Proc. ICLR*, 2018.

[43] S. Cheng, K. Mueller, and W. Xu. A framework to visualize temporal behavioral relationships in streaming multivariate data. In *Proc. NYSDS*, pp. 1–10, 2016.

[44] X. Chu, X. Fan, D. Yao, Z. Zhu, et al. Cross-network embedding for multi-network alignment. In *Proc. WWW*, pp. 273–284, 2019.

[45] W. S. Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of the American statistical association*, 74(368):829–836, 1979.

[46] D. Coelho, I. Chase, and K. Mueller. PeckVis: A visual analytics tool to analyze dominance hierarchies in small groups. *IEEE Transactions on Visualization and Computer Graphics*, 26(4):1650–1660, 2020.

[47] J. Cohen. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological Bulletin*, 70(4):213, 1968.

[48] S. R. Collins, P. Kemmeren, X.-C. Zhao, J. F. Greenblatt, et al. Toward a comprehensive atlas of the physical interactome of *Saccharomyces cerevisiae*. *Molecular & Cellular Proteomics*, 6(3):439–450, 2007.

[49] T. Crnovrsanin, J. Chu, and K.-L. Ma. An incremental layout method for visualizing online dynamic graphs. *Journal of Graph Algorithms and Applications*, 21(1):55–80, 2017.

[50] T. Crnovrsanin, C. Muelder, R. Faris, D. Felmlee, and K.-L. Ma. Visualization techniques for categorical analysis of social networks with multiple edge sets. *Social Networks*, 37:56–64, 2014.

[51] W. Cui. Visual analytics: A comprehensive overview. *IEEE Access*, 7:81555–81573, 2019.

[52] J. P. Cunningham and Z. Ghahramani. Linear dimensionality reduction: Survey, insights, and generalizations. *Journal of Machine Learning Research*, 16(1):2859–2900, 2015.

[53] A. Cuzzocrea and D. Zall. Parallel coordinates technique in visual data mining: Advantages, disadvantages and combinations. In *Proc. InfoVis*, pp. 278–284, 2013.

[54] H. Dai, Y. Tao, and H. Lin. Visual analytics of urban transportation from a bike-sharing and taxi perspective. In *Proc. VINCI*, pp. 1–8, 2019.

[55] A. Dal Col, P. Valdivia, F. Petronetto, F. Dias, C. T. Silva, and L. G. Nonato. Wavelet-based visual analysis of dynamic networks. *IEEE Transactions on Visualization and Computer Graphics*, 24(8):2456–2469, 2017.

[56] T. N. Dang, N. Pendar, and A. G. Forbes. TimeArcs: Visualizing fluctuations in dynamic networks. *Computer Graphics Forum*, 35(3):61–69, 2016.

[57] A. Dasgupta, D. L. Arendt, L. R. Franklin, P. C. Wong, and K. A. Cook. Human factors in streaming data analysis: Challenges and opportunities for information visualization. *Computer Graphics Forum*, 37(1):254–272, 2018.

[58] M. Davis. Palettable. `https://jiffyclub.github.io/palettable/`. Accessed: 2020-3-31.

[59] Ç. Demiralp. Clustrophile: A tool for visual clustering analysis. *arXiv:1710.02173*, 2017.

[60] D. V. Dimitrov. Medical internet of things and big data in healthcare. *Healthcare Informatics Research*, 22(3):156–163, 2016.

[61] W. Dinkelbach. On nonlinear fractional programming. *Management Science*, 13(7):492–498, 1967.

[62] A.-H. Dirie, A. Abid, and J. Zou. Contrastive multivariate singular spectrum analysis. In *Proc. Allerton Conference*, pp. 1122–1127, 2019.

[63] M. Dowling, J. Wenskovitch, J. Fry, L. House, and C. North. SIRIUS: Dual, symmetric, interactive dimension reductions. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):172–182, 2018.

[64] S. Dray and J. Josse. Principal component analysis with missing values: A comparative survey of methods. *Plant Ecology*, 216(5):657–667, 2015.

[65] D. Dua and C. Graff. UCI machine learning repository. `http://archive.ics.uci.edu/ml`, 2019.

[66] F. Emmert-Streib, M. Dehmer, and Y. Shi. Fifty years of graph matching, network alignment and network comparison. *Information Sciences*, 346:180–197, 2016.

[67] A. Endert, C. Han, D. Maiti, L. House, and C. North. Observation-level interaction with statistical models for visual analytics. In *Proc. VAST*, pp. 121–130, 2011.

[68] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. KDD*, pp. 226–231, 1996.

[69] P. Federico, W. Aigner, S. Miksch, F. Windhager, and L. Zenk. A visual analytics approach to dynamic social networks. In *Proc. i-KNOW*, pp. 1–8, 2011.

[70] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Human Genetics*, 7(2):179–188, 1936.

[71] A. Fornito, A. Zalesky, and E. Bullmore. *Fundamentals of Brain Network Analysis*. Academic Press, 2016.

[72] J. Fournet and A. Barrat. Contact patterns among high school students. *PLOS One*, 9(9), 2014.

[73] M. Freire, C. Plaisant, B. Shneiderman, and J. Golbeck. ManyNets: An interface for multiple network analysis and visualization. In *Proc. CHI*, pp. 213–222, 2010.

[74] T. Fujiwara, J.-K. Chou, A. M. McCullough, C. Ranganath, and K.-L. Ma. A visual analytics system for brain functional connectivity comparison across individuals, groups, and time points. In *Proc. PacificVis*, pp. 250–259, 2017.

[75] T. Fujiwara, J.-K. Chou, S. Shilpika, P. Xu, L. Ren, and K.-L. Ma. An incremental dimensionality reduction method for visualizing streaming multidimensional data. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):418–428, 2020.

[76] T. Fujiwara, O.-H. Kwon, and K.-L. Ma. Supporting analysis of dimensionality reduction results with contrastive learning. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):45–55, 2020.

[77] T. Fujiwara, J. K. Li, M. Mubarak, C. Ross, C. D. Carothers, R. B. Ross, and K.-L. Ma. A visual analytics system for optimizing the performance of large-scale networks in supercomputing systems. *Visual Informatics*, 2(1):98–110, 2018.

[78] T. Fujiwara, Shilpika, N. Sakamoto, J. Nonaka, K. Yamamoto, and K.-L. Ma. A visual analytics framework for reviewing multivariate time-series data with dimensionality reduction. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1601–1611, 2021.

[79] T. Fujiwara, X. Wei, J. Zhao, and K.-L. Ma. Interactive dimensionality reduction for comparative analysis. *IEEE Transactions on Visualization and Computer Graphics*, 2021 (early access).

[80] T. Fujiwara, J. Zhao, F. Chen, and K.-L. Ma. A visual analytics framework for contrastive network analysis. In *Proc. VAST*, pp. 48–59, 2020.

[81] T. Fujiwara, J. Zhao, F. Chen, Y. Yu, and K.-L. Ma. Interpretable contrastive learning for networks. *arXiv:2005.12419*, 2020.

[82] C. Gaiteri, S. Mostafavi, C. J. Honey, P. L. De Jager, and D. A. Bennett. Genetic variants in Alzheimer disease-molecular and brain network approaches. *Nature Reviews Neurology*, 12(7):413, 2016.

[83] E. R. Gansner, Y. Hu, and S. G. Kobourov. GMap: Drawing graphs as maps. In *Proc. GD*, pp. 405–407, 2009.

[84] E. R. Gansner, Y. Hu, and S. C. North. Interactive visualization of streaming text data with dynamic maps. *Journal of Graph Algorithms and Applications*, 17(4):515–540, 2013.

[85] H. Gao and S. Ji. Graph representation learning via hard and channel-wise attention networks. In *Proc. KDD*, pp. 741–749, 2019.

[86] S. Garte. The role of ethnicity in cancer susceptibility gene polymorphisms: The example of CYP1A1. *Carcinogenesis*, 19(8):1329–1332, 1998.

[87] R. Ge and J. Zou. Rich component analysis. In *Proc. ICML*, pp. 1502–1510, 2016.

[88] M. Gleicher. Considerations for visualizing comparison. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):413–423, 2018.

[89] M. Gleicher, D. Albers, R. Walker, I. Jusufi, C. D. Hansen, and J. C. Roberts. Visual comparison for information visualization. *Information Visualization*, 10(4):289–309, 2011.

[90] A. Goldenberg, A. X. Zheng, S. E. Fienberg, and E. M. Airoldi. A survey of statistical network models. *Foundations and Trends® in Machine Learning*, 2(2):129–233, 2010.

[91] R. Gove. Gragnostics: Fast, interpretable features for comparing graphs. In *Proc. IV*, pp. 201–209, 2019.

[92] J. C. Gower, G. B. Dijksterhuis, et al. *Procrustes Problems*, vol. 30. Oxford University Press on Demand, 2004.

[93] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proc. KDD*, pp. 855–864, 2016.

[94] G. Guennebaud, B. Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010. Accessed: 2019-3-6.

[95] H. Guo, S. Di, R. Gupta, T. Peterka, and F. Cappello. La VALSE: Scalable log visualization for fault characterization in supercomputers. In *Proc. EGPGV*, pp. 91–100, 2018.

[96] R. Guo, T. Fujiwara, Y. Li, K. M. Lima, S. Sen, N. K. Tran, and K.-L. Ma. Comparative visual analytics for assessing medical records with sequence embedding. *Visual Informatics*, 4(2):72–85, 2020.

[97] Y.-F. Guo, S.-J. Li, J.-Y. Yang, T.-T. Shu, and L.-D. Wu. A generalized Foley-Sammon transform based on generalized Fisher discriminant criterion and its application to face recognition. *Pattern Recognition Letters*, 24(1):147–158, 2003.

[98] S. Hachul and M. Jünger. Drawing large graphs with a potential-field-based multilevel algorithm. In *Proc. GD*, pp. 285–295, 2004.

[99] M. Hajij, B. Wang, C. Scheidegger, and P. Rosen. Visual detection of structural changes in time-varying graphs using persistent homology. In *Proc. PacificVis*, pp. 125–134, 2018.

[100] P. Hall, D. Marshall, and R. Martin. Incremental eigenanalysis for classification. In *Proc. BMVC*, pp. 286–295, 1998.

[101] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Proc. NIPS*, pp. 1024–1034, 2017.

[102] M. Harrigan, D. Archambault, P. Cunningham, and N. Hurley. EgoNav: Exploring networks through egocentric spatializations. In *Proc. AVI*, pp. 563–570, 2012.

[103] M. Harrower and C. A. Brewer. Colorbrewer.org: An online tool for selecting colour schemes for maps. *Cartographic Journal*, 40(1):27–37, 2003.

[104] R. A. Harshman. Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-mode factor analysis. *UCLA Working Papers in Phonetics*, 16:1–84, 1970.

[105] S. G. Hart and L. E. Staveland. Development of NASA-TLX (task load index): Results of empirical and theoretical research. In P. A. Hancock and N. Meshkati, eds., *Human Mental Workload*, vol. 52 of *Advances in Psychology*, pp. 139–183. North-Holland, 1988.

[106] J. A. Hartigan. Printer graphics for clustering. *Journal of Statistical Computation and Simulation*, 4(3):187–213, 1975.

[107] J. A. Hartigan and M. A. Wong. A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[108] H. Hassani and R. Mahmoudvand. Multivariate singular spectrum analysis. In *Singular Spectrum Analysis*, pp. 49–86. Springer, 2018.

[109] T. Hastie and R. Tibshirani. Discriminant analysis by Gaussian mixtures. *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 155–176, 1996.

[110] M. Heimann, H. Shen, T. Safavi, and D. Koutra. REGAL: Representation learning-based graph alignment. In *Proc. CIKM*, pp. 117–126, 2018.

[111] N. Henry, J.-D. Fekete, and M. J. McGuffin. NodeTrix: A hybrid visualization of social networks. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1302–1309, 2007.

[112] C. Higuera, K. J. Gardiner, and K. J. Cios. Self-organizing feature maps identify proteins critical to learning in a mouse model of down syndrome. *PLOS one*, 10(6):e0129126, 2015.

[113] T. Höllt, N. Pezzotti, V. van Unen, F. Koning, B. P. Lelieveldt, and A. Vilanova. CyteGuide: Visual guidance for hierarchical single-cell analysis. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):739–748, 2018.

[114] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417, 1933.

[115] Y. Hu. Efficient, high-quality force-directed graph drawing. *Mathematica Journal*, 10(1):37–71, 2005.

[116] L. Huang, A. L. Shea, H. Qian, A. Masurkar, H. Deng, and D. Liu. Patient clustering improves efficiency of federated machine learning to predict mortality and hospital stay time using distributed electronic medical records. *Journal of Biomedical Informatics*, 99:103291, 2019.

[117] A. Inselberg and B. Dimsdale. Parallel coordinates for visualizing multi-dimensional geometry. In *Proc. Visualization*, pp. 361–378, 1990.

[118] A. J. Izenman. Linear discriminant analysis. In *Modern Multivariate Statistical Techniques*, pp. 237–280. Springer, 2013.

[119] D. Jäckle, F. Fischer, T. Schreck, and D. A. Keim. Temporal MDS plots for analysis of multivariate data. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):141–150, 2016.

[120] D. H. Jeong, C. Ziemkiewicz, B. Fisher, W. Ribarsky, and R. Chang. iPCA: An interactive system for PCA-based visual analytics. *Computer Graphics Forum*, 28(3):767–774, 2009.

[121] D. H. Jeong, C. Ziemkiewicz, W. Ribarsky, and R. Chang. Understanding principal component analysis using a visual analytics tool. Technical Report No. CVC-UNCC-09-16, UNC Charlotte, 2009.

[122] Y. Jia, F. Nie, and C. Zhang. Trace ratio problem revisited. *IEEE Transactions on Neural Networks*, 20(4):729–735, 2009.

[123] M. John and M. Baumann. A visual approach for the comparative analysis of character networks in narrative texts. In *Proc. PacificVis*, pp. 247–256, 2019.

[124] P. Joia, F. Petronetto, and L. G. Nonato. Uncovering representative groups in multidimensional projections. *Computer Graphics Forum*, 34(3):281–290, 2015.

[125] I. T. Jolliffe. Principal component analysis and factor analysis. In *Principal Component Analysis*, pp. 115–128. Springer, 1986.

[126] F. Jourdan and G. Melangon. Multiscale hybrid MDS. In *Proc. IV*, pp. 388–393, 2004.

[127] E. Kandogan. Star coordinates: A multi-dimensional visualization technique with uniform treatment of dimensions. In *Proc. InfoVis*, pp. 9–12, 2000.

[128] D. Keim, G. Andrienko, J.-D. Fekete, C. Görg, J. Kohlhammer, and G. Melançon. Visual analytics: Definition, process, and challenges. In A. Kerren, J. T. Stasko, J.-D. Fekete, and C. North, eds., *Information Visualization: Human-Centered Issues and Perspectives*, pp. 154–175. Springer Berlin Heidelberg, 2008.

[129] P. Kerpedjiev, N. Abdennur, F. Lekschas, C. McCallum, K. Dinkla, H. Strobelt, J. M. Luber, S. B. Ouellette, A. Azhir, N. Kumar, et al. HiGlass: Web-based visual exploration and analysis of genome interaction maps. *Genome biology*, 19(1):1–12, 2018.

[130] S. P. Kesavan, T. Fujiwara, J. K. Li, C. Ross, M. Mubarak, C. D. Carothers, R. B. Ross, and K.-L. Ma. A visual analytics framework for reviewing streaming performance data. In *Proc. PacificVis*, pp. 206–215, 2020.

[131] A. N. Khambhati, J. D. Medaglia, E. A. Karuza, S. L. Thompson-Schill, and D. S. Bassett. Subgraphs of functional brain networks identify dynamical constraints of cognitive control. *PLOS Computational Biology*, 14(7):e1006234, 2018.

[132] H. A. Kiers. Hierarchical relations among three-way methods. *Psychometrika*, 56(3):449–470, 1991.

[133] H. A. Kiers. Towards a standardized notation and terminology in multiway analysis. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 14(3):105–122, 2000.

[134] H. Kim, J. Choo, C. Lee, H. Lee, C. K. Reddy, and H. Park. PIVE: Per-iteration visualization environment for real-time interactions with dimension reduction and clustering. In *Proc. AAAI*, pp. 1001–1009, 2017.

[135] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *Proc. ICLR*, 2016.

[136] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

[137] D. Koop, J. Freire, and C. T. Silva. Visual summaries for graph collections. In *Proc. PacificVis*, pp. 57–64, 2013.

[138] O. Kouropteva, O. Okun, and M. Pietikäinen. Incremental locally linear embedding. *Elsevier Pattern Recognition*, 38(10):1764–1767, 2005.

[139] H.-P. Kriegel, P. Kröger, J. Sander, and A. Zimek. Density-based clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(3):231–240, 2011.

[140] M. Krstajic and D. A. Keim. Visualization of streaming data: Observing change and context in information visualization techniques. In *Proc. BigData*, pp. 41–47, 2013.

[141] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[142] B. C. Kwon, B. Eysenbach, J. Verma, K. Ng, C. De Filippi, W. F. Stewart, and A. Perer. Clustervision: Visual supervision of unsupervised clustering. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):142–151, 2018.

[143] B. C. Kwon, H. Kim, E. Wall, J. Choo, H. Park, and A. Endert. AxiSketcher: Interactive nonlinear axis mapping of visualizations through user drawings. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):221–230, 2017.

[144] O.-H. Kwon, T. Crnovrsanin, and K.-L. Ma. What would a graph look like in this layout? A machine learning approach to large graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):478–488, 2018.

[145] C. Lai, Y. Zhao, and X. Yuan. Exploring high-dimensional data through locally enhanced projections. *Journal of Visual Languages and Computing*, 48:144–156, 2018.

[146] V. Larivière, Y. Gingras, and É. Archambault. Canadian collaboration networks: A comparative analysis of the natural sciences, social sciences and the humanities. *Scientometrics*, 68(3):519–533, 2006.

[147] M. H. Law and A. K. Jain. Incremental nonlinear dimensionality reduction by manifold learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(3):377–391, 2006.

[148] Y. LeCun, C. Cortes, and C. J.C. Burges. The MNIST database of handwritten digits. `http://yann.lecun.com/exdb/mnist/`, 1999. Accessed: 2019-3-7.

[149] A. Lee, D. Archambault, and M. Nacenta. Dynamic Network Plaid: A tool for the analysis of dynamic networks. In *Proc. CHI*, pp. 1–14, 2019.

[150] C. Lee, Z. Luo, K. Y. Ngiam, M. Zhang, K. Zheng, G. Chen, B. C. Ooi, and W. L. J. Yip. Big healthcare data analytics: Challenges and applications. In *Handbook of Large-Scale Distributed Computing in Smart Healthcare*, pp. 11–41. Springer, 2017.

[151] J. A. Lee and M. Verleysen. Quality assessment of dimensionality reduction: Rank-based criteria. *Neurocomputing*, 72(7-9):1431–1443, 2009.

[152] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data*, 1(1):2–es, 2007.

[153] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, 2014. Accessed: 2020-4-10.

[154] A. Levey and M. Lindenbaum. Sequential Karhunen-Loeve basis extraction and its application to images. *IEEE Transactions on Image Processing*, 9(8):1371–1374, 2000.

[155] D. Liu, F. Guo, B. Deng, H. Qu, and Y. Wu. egoComp: A node-link-based technique for visual comparison of ego-networks. *Information Visualization*, 16(3):179–189, 2017.

[156] D. Liu, P. Xu, and L. Ren. TPFlow: Progressive partition and multidimensional pattern extraction for large-scale spatio-temporal data analysis. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):1–11, 2019.

[157] L. Liu, J. Xu, D. Russell, P. Townend, and D. Webster. Efficient and scalable search on scale-free P2P networks. *Peer-to-Peer Networking and Applications*, 2(2):98–108, 2009.

[158] S. Liu, W. Cui, Y. Wu, and M. Liu. A survey on information visualization: Recent advances and challenges. *Visual Computer*, 30(12):1373–1393, 2014.

[159] S. Liu, D. Maljovec, B. Wang, P.-T. Bremer, and V. Pascucci. Visualizing high-dimensional data: Advances in the past decade. *IEEE Transactions on Visualization and Computer Graphics*, 23(3):1249–1268, 2017.

[160] S. Liu, X. Wang, J. Chen, J. Zhu, and B. Guo. TopicPanorama: A full picture of relevant topics. In *Proc. VAST*, pp. 183–192, 2014.

[161] S. Liu, J. Yin, X. Wang, W. Cui, K. Cao, and J. Pei. Online visual analytics of text streams. *IEEE Transactions on Visualization and Computer Graphics*, 22(11):2451–2466, 2016.

[162] X. Liu and H.-W. Shen. The effects of representation and juxtaposition on graphical perception of matrix visualization. In *Proc. CHI*, pp. 269–278, 2015.

[163] S. Loisel and Y. Takane. Comparisons among several methods for handling missing data in principal component analysis (PCA). *Advances in Data Analysis and Classification*, pp. 1–24, 2018.

[164] H. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos. A survey of multilinear subspace learning for tensor data. *Pattern Recognition*, 44(7):1540–1551, 2011.

[165] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, et al. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54(4):396–405, 2003.

[166] G. Ma, N. Ahmed, T. Willke, D. Sengupta, et al. Deep graph similarity learning for brain data analysis. In *Proc. CIKM*, pp. 2743–2751, 2019.

[167] W. E. Marcílio, D. M. Eler, and R. E. Garcia. An approach to perform local analysis on multidimensional projection. In *Proc. SIBGRAPI*, pp. 351–358, 2017.

[168] M. Marusteri and V. Bacarea. Comparing groups for statistical differences: How to choose the right statistical test? *Biochemia Medica*, 20(1):15–32, 2010.

[169] E. McCrum-Gardner. Which is the correct statistical test to use? *British Journal of Oral and Maxillofacial Surgery*, 46(1):38–41, 2008.

[170] L. McInnes, J. Healy, and J. Melville. UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv:1802.03426*, 2018.

[171] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K.-R. Mullers. Fisher discriminant analysis with kernels. In *Neural Networks for Signal Processing IX: Proc. IEEE Signal Processing Society Workshop*, pp. 41–48, 1999.

[172] H. Miyazaki, Y. Kusano, N. Shinjou, F. Shoji, M. Yokokawa, and T. Watanabe. Overview of the K computer system. *Fujitsu Scientific & Technical Journal*, 48(3):302–309, 2012.

[173] A. Morrison, G. Ross, and M. Chalmers. Fast multidimensional scaling through sampling, springs and interpolation. *Information Visualization*, 2(1):68–77, 2003.

[174] N. Moshiri. The dual-Barabási-Albert model. *arXiv:1810.10538*, 2018.

[175] C. Muelder, B. Zhu, W. Chen, H. Zhang, and K.-L. Ma. Visual analysis of cloud computing performance using behavioral lines. *IEEE Transactions on Visualization and Computer Graphics*, 22(6):1694–1704, 2016.

[176] T. Mühlbacher, H. Piringer, S. Gratzl, M. Sedlmair, and M. Streit. Opening the black box: Strategies for increased user involvement in existing algorithm implementations. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1643–1652, 2014.

[177] D. Müllner. Modern hierarchical, agglomerative clustering algorithms. *arXiv:1109.2378*, 2011.

[178] S. Murugesan, K. Bouchard, J. Brown, M. Kiran, D. Lurie, B. Hamann, and G. H. Weber. State-based network similarity visualization. *Information Visualization*, 19(2):96–113, 2020.

[179] M. Newman. *Networks*. Oxford university press, 2018.

[180] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Proc. NIPS*, pp. 849–856, 2002.

[181] L. G. Nonato and M. Aupetit. Multidimensional projection for visual analytics: Linking techniques with distortions, tasks, and layout enrichment. *IEEE Transactions on Visualization and Computer Graphics*, 25(8):2650–2673, 2019.

[182] E. Oja and J. Karhunen. On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *Journal of Mathematical Analysis and Applications*, 106(1):69–84, 1985.

[183] L. Pagliosa, P. Pagliosa, and L. G. Nonato. Understanding attribute variability in multidimensional projections. In *Proc. SIBGRAPI*, pp. 297–304, 2016.

[184] M. Parimala, D. Lopez, and N. Senthilkumar. A survey on density based clustering algorithms for mining large spatial databases. *International Journal of Advanced Science and Technology*, 31(1):59–66, 2011.

[185] N. Pezzotti, B. P. Lelieveldt, L. van der Maaten, T. Höllt, E. Eisemann, and A. Vilanova. Approximated and user steerable tSNE for progressive visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 23(7):1739–1752, 2017.

[186] N. Pržulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23(2):e177–e183, 2007.

[187] P. E. Rauber, S. G. Fadel, A. X. Falcao, and A. C. Telea. Visualizing the hidden activity of artificial neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):101–110, 2017.

[188] P. E. Rauber, A. X. Falcão, and A. C. Telea. Visualizing time-dependent data using dynamic t-SNE. In *Proc. EuroVis*, pp. 73–77, 2016.

[189] M. Redmond and A. Baveja. A data-driven software tool for enabling cooperative information sharing among police departments. *European Journal of Operational Research*, 141(3):660–678, 2002.

[190] T. Reguly, A. Breitkreutz, L. Boucher, B.-J. Breitkreutz, et al. Comprehensive curation and analysis of global interaction networks in *Saccharomyces cerevisiae*. *Journal of Biology*, 5(4):11, 2006.

[191] M. B. Richman. Rotation of principal components. *Journal of Climatology*, 6(3):293–335, 1986.

[192] M. Ripeanu, A. Iamnitchi, and I. Foster. Mapping the Gnutella network. *IEEE Internet Computing*, 6(1):50, 2002.

[193] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1-3):125–141, 2008.

[194] R. A. Rossi, R. Zhou, and N. Ahmed. Deep inductive graph representation learning. *IEEE Transactions on Knowledge and Data Engineering*, 32(3):438–452, 2020.

[195] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv:1609.04747*, 2016.

[196] S. Rufiange and M. J. McGuffin. DiffAni: Visualizing dynamic graphs with a hybrid of difference maps and animation. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2556–2565, 2013.

[197] L. Ruiz, F. Gama, and A. Ribeiro. Gated graph recurrent neural networks. *IEEE Transactions on Signal Processing*, 68:6303–6318, 2020.

[198] D. Sacha, L. Zhang, M. Sedlmair, J. A. Lee, J. Peltonen, D. Weiskopf, S. C. North, and D. A. Keim. Visual interaction with dimensionality reduction: A structured literature analysis. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):241–250, 2017.

[199] A. Sankar, Y. Wu, L. Gou, W. Zhang, and H. Yang. DySAT: Deep neural representation learning on dynamic graphs via self-attention networks. In *Proc. WSDM*, pp. 519–527, 2020.

[200] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[201] P. H. Schönemann and R. M. Carroll. Fitting one matrix to another under choice of a central dilation and a rigid motion. *Psychometrika*, 35(2):245–255, 1970.

[202] D. W. Scott. On optimal and data-based histograms. *Biometrika*, 66(3):605–610, 1979.

[203] K. A. Severson, S. Ghosh, and K. Ng. Unsupervised learning with contrastive latent variable models. In *Proc. AAAI*, vol. 33, pp. 4862–4869, 2019.

[204] X. Shen, Q. Dai, S. Mao, F.-L. Chung, and K.-S. Choi. Network together: Node classification via cross-network deep network embedding. *IEEE Transactions on Neural Networks and Learning Systems*, 32(5):1935–1948, 2021.

[205] L. Shi, H. Tong, and X. Mu. BrainQuest: Perception-guided brain network comparison. In *Proc. ICDM*, pp. 379–388, 2015.

[206] F. Shilpika, B. Lusch, M. Emani, V. Vishwanath, M. E. Papka, and K.-L. Ma. MELA: A visual analytics tool for studying multifidelity HPC system logs. In *Proc. DAAC*, pp. 13–18, 2019.

[207] J. Skardinga, B. Gabrys, and K. Musial. Foundations and modelling of dynamic networks using dynamic graph neural networks: A survey. *IEEE Access*, 9:79143–79168, 2021.

[208] N. Smith and S. van der Walt. mpl colormaps. `https://bids.github.io/colormap/`. Accessed: 2020-3-31.

[209] J. Stahnke, M. Dörk, B. Müller, and A. Thom. Probing projections: Interaction techniques for interpreting arrangements and errors of dimensionality reductions. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):629–638, 2016.

[210] J. Stehlé, N. Voirin, A. Barrat, C. Cattuto, et al. High-resolution measurements of face-to-face contact patterns in a primary school. *PLOS one*, 6(8), 2011.

[211] M. Steiger, J. Bernard, S. Mittelstädt, H. Lücke-Tieke, D. Keim, T. May, and J. Kohlhammer. Visual analysis of time-series similarities for anomaly detection in sensor networks. *Computer Graphics Forum*, 33(3):401–410, 2014.

[212] C. D. Stolper, A. Perer, and D. Gotz. Progressive visual analytics: User-driven visual exploration of in-progress analytics. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1653–1662, 2014.

[213] H. Strange and R. Zwiggelaar. *Open Problems in Spectral Dimensionality Reduction*. Springer, 2014.

[214] G.-D. Sun, Y.-C. Wu, R.-H. Liang, and S.-X. Liu. A survey of visual analytics techniques and applications: State-of-the-art research and future challenges. *Journal of Computer Science and Technology*, 28(5):852–867, 2013.

[215] Y. Sun, H. Song, A. J. Jara, and R. Bie. Internet of things and big data analytics for smart and connected communities. *IEEE Access*, 4:766–773, 2016.

[216] M. J. Swain and D. H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[217] G. K. Tam, V. Kothari, and M. Chen. An analysis of machine-and human-analytics in classification. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):71–80, 2017.

[218] Y. Tanahashi, C.-H. Hsueh, and K.-L. Ma. An efficient framework for generating storyline visualizations from streaming data. *IEEE Transactions on Visualization and Computer Graphics*, 21(6):730–742, 2015.

[219] J. Tang, J. Liu, M. Zhang, and Q. Mei. Visualizing large-scale and high-dimensional data. In *Proc. WWW*, pp. 287–297, 2016.

[220] M. Tantardini, F. Ieva, L. Tajoli, and C. Piccardi. Comparing methods for comparing networks. *Scientific Reports*, 9(1):17557, 2019.

[221] The USDA Nutrient Data Laboratory, the Food and Nutrition Information Center, and Information Systems Division of the National Agricultural Library. USDA food composition databases. `https://ndb.nal.usda.gov/ndb/`, 2018. Accessed: 2019-3-7.

[222] W. S. Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17(4):401–419, 1952.

[223] TransitFeeds. Nashville MTA GTFS. `https://transitfeeds.com/p/nashville-mta/220`. Accessed: 2018-8-22.

[224] F. S. Tsai. Dimensionality reduction techniques for blog visualization. *Expert Systems with Applications*, 38(3):2766–2773, 2011.

[225] L. R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[226] C. Turkay, E. Kaya, S. Balcisoy, and H. Hauser. Designing progressive and interactive analytics processes for high-dimensional data analysis. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):131–140, 2017.

[227] C. Turkay, A. Lundervold, A. J. Lundervold, and H. Hauser. Representative factor generation for the interactive visual analysis of high-dimensional data. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2621–2630, 2012.

[228] C. Turkay, N. Pezzotti, C. Binnig, H. Strobelt, B. Hammer, D. A. Keim, J.-D. Fekete, T. Palpanas, Y. Wang, and F. Rusu. Progressive data science: Potential and challenges. *arXiv:1812.08032*, 2018.

[229] US Environmental Protection Agency. Air quality system data mart. `https://www.epa.gov/airdata`, 2019. Accessed: 2020-4-21.

[230] S. van den Elzen, D. Holten, J. Blaas, and J. J. van Wijk. Reducing snapshots to points: A visual analytics approach to dynamic network exploration. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):1–10, 2016.

[231] L. van der Maaten. Learning a parametric embedding by preserving local structure. In *Proc. AISTATS*, pp. 384–391, 2009.

[232] L. van der Maaten. Accelerating t-SNE using tree-based algorithms. *Journal of Machine Learning Research*, 15(1):3221–3245, 2014.

[233] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[234] L. van der Maaten, E. Postma, and J. van den Herik. Dimensionality reduction: A comparative review. *Journal of Machine Learning Research*, 10:66–71, 2009.

[235] J. J. van Wijk and W. A. Nuij. Smooth and efficient zooming and panning. In *Proc. InfoVis*, pp. 15–23, 2003.

[236] P. Veličković, G. Cucurull, A. Casanova, A. Romero, et al. Graph attention networks. In *Proc. ICLR*, 2018.

[237] T. von Landesberger, F. Brodkorb, P. Roskosch, N. Andrienko, G. Andrienko, and A. Kerren. MobilityGraphs: Visual analysis of mass mobility dynamics via spatio-temporal graphs and clustering. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):11–20, 2016.

[238] T. von Landesberger, M. Gorner, and T. Schreck. Visual analysis of graphs with multiple connected components. In *Proc. VAST*, pp. 155–162, 2009.

[239] J.-L. Wang, J.-M. Chiou, and H.-G. Müller. Functional data analysis. *Annual Review of Statistics and Its Application*, 3:257–295, 2016.

[240] Q. Wang, S. R. Kulkarni, and S. Verdú. Divergence estimation for multidimensional densities via k-nearest-neighbor distances. *IEEE Transactions on Information Theory*, 55(5):2392–2405, 2009.

[241] Y. Wang, J. Li, F. Nie, H. Theisel, M. Gong, and D. J. Lehmann. Linear discriminative star coordinates for exploring class and cluster separation of high dimensional data. *Computer Graphics Forum*, 36(3):401–410, 2017.

[242] Y. Wang and K.-L. Ma. Revealing the fog-of-war: A visualization-directed, uncertainty-aware approach for exploring high-dimensional data. In *Proc. BigData*, pp. 629–638, 2015.

[243] K. Webga and A. Lu. Discovery of rating fraud with real-time streaming visual analytics. In *Proc. VizSec*, pp. 1–8, 2015.

[244] J. Weng, Y. Zhang, and W.-S. Hwang. Candid covariance-free incremental principal component analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):1034–1040, 2003.

[245] J. Wenskovitch, I. Crandell, N. Ramakrishnan, L. House, and C. North. Towards a systematic combination of dimension reduction and clustering in visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):131–141, 2018.

[246] M. Williams and T. Munzner. Steerable, progressive multidimensional scaling. In *Proc. InfoVis*, pp. 57–64, 2004.

[247] W. Wu, Y. Zheng, K. Chen, X. Wang, and N. Cao. A visual analytics approach for equipment condition monitoring in smart factories of process industry. In *Proc. PacificVis*, pp. 140–149, 2018.

[248] J. Xia, F. Wu, F. Guo, C. Xie, Z. Liu, and W. Chen. An online visualization system for streaming log data of computing clusters. *Tsinghua Science and Technology*, 18(2):196–205, 2013.

[249] P. Xu, H. Mei, L. Ren, and W. Chen. ViDX: Visual diagnostics of assembly line performance in smart factories. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):291–300, 2017.

[250] R. Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645—678, 2005.

[251] T. Yang, J. Liu, L. McMillan, and W. Wang. A fast approximation to multidimensional scaling. In *Proc. IEEE Workshop on Computation Intensive Methods for Computer Vision*, 2006.

[252] X. Yang, L. Shi, M. Daianu, H. Tong, Q. Liu, and P. Thompson. Blockwise human brain network visual comparison using NodeTrix representation. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):181–190, 2017.

[253] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec. GNNExplainer: Generating explanations for graph neural networks. In *Proc. NIPS*, pp. 9240–9251, 2019.

[254] V. Yoghourdjian, T. Dwyer, K. Klein, K. Marriott, and M. Wybrow. Graph Thumbnails: Identifying and comparing multiple graphs at a glance. *IEEE Transactions on Visualization and Computer Graphics*, 24(12):3081–3095, 2018.

[255] H. Yu, P. Braun, M. A. Yıldırım, I. Lemmens, et al. High-quality binary protein interaction map of the yeast interactome network. *Science*, 322(5898):104–110, 2008.

[256] W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):452–473, 1977.

[257] M. D. Zeiler. ADADELTA: An adaptive learning rate method. *arXiv:1212.5701*, 2012.

[258] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla. Heterogeneous graph neural network. In *Proc. KDD*, pp. 793–803, 2019.

[259] Z. Zhang, P. Cui, and W. Zhu. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2020 (early access).

[260] J. Zhao, M. Glueck, F. Chevalier, Y. Wu, and A. Khan. Egocentric analysis of dynamic networks with EgoLines. In *Proc. CHI*, pp. 5003–5014, 2016.

[261] J. Zhao, Z. Liu, M. Dontcheva, A. Hertzmann, and A. Wilson. MatrixWave: Visual comparison of event sequence data. In *Proc. CHI*, p. 259–268, 2015.

[262] F. Zhou, X. Lin, C. Liu, Y. Zhao, P. Xu, L. Ren, T. Xue, and L. Ren. A survey of visualization for smart manufacturing. *Journal of Visualization*, 22(2):419–435, 2019.

[263] J. Y. Zou, D. J. Hsu, D. C. Parkes, and R. P. Adams. Contrastive learning using spectral methods. In *Proc. NIPS*, pp. 2238–2246, 2013.