

caption

**Data Summaries for Scalable Visual Analysis**

**DISSERTATION**

**Submitted in Partial Fulfillment of  
the Requirements for  
the Degree of**

**DOCTOR OF PHILOSOPHY (Computer Science)**

**at the**

**NEW YORK UNIVERSITY  
TANDON SCHOOL OF ENGINEERING**

**by**

**Gromit Yeuk-Yin Chan**

**September 2021**

# **Data Summaries for Scalable Visual Analysis**

## **DISSERTATION**

**Submitted in Partial Fulfillment of**

**the Requirements for**

**the Degree of**

**DOCTOR OF PHILOSOPHY (Computer Science)**

**at the**

**NEW YORK UNIVERSITY  
TANDON SCHOOL OF ENGINEERING**

**by**

**Gromit Yeuk-Yin Chan**

**September 2021**

**Approved:**

---

**Department Chair Signature**

---

**Date**

University ID: N14847514

Net ID: yyc337

Approved by the Guidance Committee:

Major: Computer Science

---

**Cláudio T. Silva**  
Professor of Computer Science  
New York University

---

Date

---

**Juliana Freire**  
Professor of Computer Science  
New York University

---

Date

---

**Luis Gustavo Nonato**  
Professor of Computer Science  
Universidade de São Paulo

---

Date

---

**Enrico Bertini**  
Associate Professor of Computer Science  
New York University

---

Date

---

**Arvind Satyanarayan**  
Assistant Professor of Computer Science  
Massachusetts Institute of Technology

---

Date

Microfilm or other copies of this dissertation are obtainable from

UMI Dissertation Publishing

ProQuest CSA

789 E. Eisenhower Parkway

P.O. Box 1346

Ann Arbor, MI 48106-1346

## Vita

Gromit Yeuk-Yin Chan was born in Hong Kong in February 1993. He completed a B.Eng in Computer Engineering and a BBA in General Business Management from the Hong Kong University of Science and Technology. While completing his undergraduate studies, he worked as a Research Assistant in VisLab directed by prof. Huamin Qu. He started his Ph.D. in August 2016, working on a variety of areas with prof. Claudio Silva and prof. Juliana Freire, including large-scale visual data summarization, scalable algorithms for Machine Learning, and visual analytics applications. He has received an honorable mention for SIGMOD 2017 Best Demonstration and has published first-authored full papers at IEEE VIS, WWW, and SIGKDD. He has also interned at Bosch Research, diNo group from University of Paris, and Adobe Research during his doctoral studies.

## Acknowledgements

First and foremost, I would like to thank my Ph.D. committee: prof. Claudio Silva, prof. Juliana Freire, prof. Gustavo Nonata, prof. Enrico Bertini, and prof. Arvind Satyanarayan for their valuable feedback on my dissertation.

I would like to thank my advisors, prof. Claudio Silva and prof. Juliana Freire, for the unconditional support and care throughout my Ph.D. studies. I had acquired numerous visualization and database knowledge from their coaching, funding, and world-class research facilities. Apart from that, Claudio taught me how to understand the research world, identify goals to achieve, and treat people as individuals. Juliana taught me how to stay focused on details and show me an obvious but highly underrated (in my opinion) recipe of research success, which is to spend as much time as you can on research.

I would like to thank prof. Gustavo Nonato and prof. Enrico Bertini for the many discussions on different research topics and ideas. Gustavo and I always came up with lots of exciting ideas and worked together on unlimited possibilities. Enrico and I always had lots of inspirational discussions to brainstorm ideas with sharp angles. It was my pleasure to work with both of them during my studies.

I would like to thank my mentor of my first research internship, Dr. Panpan Xu, for the hands-on coaching to publish top papers. Panpan single-handedly taught me how to formulate, code, and write top-quality research work in my first year of Ph.D. I realized it was the moment I had a much better low-level understanding of how to do research. I would also like to thank her for the opportunity to do an internship at Bosch, given I was a first-year Ph.D. student and had no major publication experience.

I would like to thank my mentors at Adobe Research, especially Dr. Fan Du, Dr. Tung Mai, and Dr. Eunyee Koh. It was fun to work with Fan for a year of HCI research and then with Tung for another year of ML research. Afterward, I was grateful to be invited by Eunyee for a job talk and then a full-time research position.

I would like to thank prof. Themis Palpanas for the three-month visit to his lab in Paris. Themis opened a new world for me of time series research. I hope that I can work more on this discipline in the foreseeable future.

I would like to thank my collaborators on the projects in the lab, especially Dr. Alice Chu, Dr. Harish Doraiswamy, Dr. Brian Barr, and Kyle Overton. Their constant help and feedback ensured my productivity in the lab.

I would like to thank all members of the VIDA lab for the research environment and fun over the years. The supportive atmosphere provides me the serenity to focus on research.

I would like to thank prof. Huamin Qu for introducing me to the research world in my last year of undergraduate. Huamin discovered my potential in research and encouraged me to pursue a Ph.D. overseas. I am grateful to his persuasion to give up my job offer from a top consulting firm and commit to a research career, which totally changed my life.

I would like to thank Bowen, Jeffrey, Lawrence, Martin, and Yao, for being very special friends.

I would like to thank my dad and mum for their support throughout my life. I would also like to thank my uncle, prof. Weisi Lin, for the help throughout my studies. I would like to express my thoughts to my grandpa, who passed away few months before I could finish my Ph.D.

Finally, I would like to thank Annie for supporting me throughout my graduate studies, even though we have a 12-hour time zone difference.

To dad, mum and Annie.

## ABSTRACT

Data Summaries for Scalable Visual Analysis

by

Gromit Yeuk-Yin Chan

Advisors: Prof. Cláudio T. Silva, Ph.D. and Prof. Juliana Freire, Ph.D.

Submitted in Partial Fulfillment of the Requirements for  
the Degree of Doctor of Philosophy (Computer Science)

September 2021

# Contents

Vita . . . . .	iv
Acknowledgements . . . . .	v
Abstract . . . . .	viii
List of Figures . . . . .	xiv
List of Tables . . . . .	xv
<b>1 Introduction</b>	<b>1</b>
1.1 Contributions . . . . .	3
1.2 Outline . . . . .	5
<b>2 Overview of Large Scale Visual Analytics</b>	<b>6</b>
2.1 Sampling . . . . .	6
2.2 Feature Extraction . . . . .	7
2.3 Interactions . . . . .	7
2.4 Data Summarization . . . . .	8
2.5 Summary . . . . .	9
<b>3 Bipartite Graph Summary</b>	<b>10</b>
3.1 Related Work . . . . .	12
3.1.1 Bipartite relation visualization . . . . .	12
3.1.2 Bipartite relation clustering and visualization . . . . .	13
3.2 Minimum Description Length (MDL) for Bipartite Graph Summa- rization . . . . .	14
3.2.1 Two-part representation of a bipartite graph . . . . .	14
3.2.2 The MDL principle . . . . .	16
3.3 Computing MDL Representation . . . . .	18

	x	
3.3.1	The <i>BM-MDL</i> algorithm . . . . .	20
3.3.2	Speeding up With LSH . . . . .	23
3.4	Design Requirements . . . . .	23
3.5	The ViBR System . . . . .	26
3.5.1	Visual adjacency list . . . . .	26
3.5.2	User Interaction . . . . .	29
3.6	Evaluation . . . . .	32
3.6.1	Evaluation of robustness and speed . . . . .	32
3.6.2	Example Usage Scenarios . . . . .	33
3.6.3	Congress voting analysis . . . . .	34
3.6.4	Vehicle fault data analysis . . . . .	37
3.6.5	Expert Interview . . . . .	38
3.7	Conclusion . . . . .	39
<b>4</b>	<b>Time Series Summary</b>	<b>40</b>
4.1	Related Work . . . . .	42
4.1.1	Time Series Visualization . . . . .	42
4.1.2	Time Series Mining and Visualization . . . . .	44
4.2	Extracting Similar Time Series Subsequences . . . . .	45
4.2.1	Background on Time Series Subsequence Mining . . . . .	46
4.2.2	Time Series Visual Symbolic Representation . . . . .	47
4.2.3	Optimizing Groupings from Sequential Patterns . . . . .	49
4.2.4	Illustration of Visual Summarization Result . . . . .	51
4.3	The TiVY Algorithm . . . . .	51
4.3.1	Computing Symbolic Representations of Time Series . . . . .	52
4.3.2	Computing Support of Sequential Patterns . . . . .	54
4.3.3	Extracting the Optimal Groupings . . . . .	55
4.4	Visualizing Time Series At Scale . . . . .	56
4.4.1	Analytical Tasks . . . . .	56
4.4.2	Visualizing Time Series Summary . . . . .	58
4.5	Evaluation . . . . .	62
4.5.1	Datasets and Apparatus . . . . .	62
4.5.2	Quantitative Evaluation . . . . .	63
4.5.3	Use Cases . . . . .	64

	xi
4.6 Conclusion . . . . .	67
<b>5 Machine Learning Model Summary</b>	<b>68</b>
5.1 Related Work . . . . .	70
5.1.1 Visualization of Model Internals . . . . .	70
5.1.2 Visualization of Logical Models . . . . .	71
5.1.3 Feature Vector Visualization . . . . .	72
5.2 Pipeline to Summarize ML Models . . . . .	73
5.2.1 Goals and Target Users . . . . .	73
5.2.2 Constructing an Explanation Matrix . . . . .	74
5.2.3 Patterns in the Explanation Matrix . . . . .	75
5.2.4 Feature Engineering . . . . .	77
5.2.5 Information Theoretic Approach to Co-cluster the Explanation Matrix . . . . .	79
5.2.6 The MELODY Algorithm . . . . .	81
5.3 Visual Analytics System:MELODY MATRIX . . . . .	85
5.3.1 Visualization Design . . . . .	85
5.3.2 Interactions . . . . .	89
5.4 Case Study . . . . .	91
5.4.1 Datasets and Models . . . . .	91
5.5 User Study . . . . .	95
5.5.1 Participants . . . . .	96
5.5.2 Tasks . . . . .	96
5.5.3 Results and Design Lessons . . . . .	97
<b>6 Topological Summary of Heterogeneous Model Explanations</b>	<b>100</b>
6.1 Related Work . . . . .	104
6.2 Topology Background . . . . .	104
6.3 Topological Representation and Assessment of Local Explanation .	107
6.4 Design Process For Visualization . . . . .	109
6.4.1 Step 1: Tasks . . . . .	110
6.4.2 Step 2: Data Types . . . . .	110
6.4.3 Step 3: Data Abstraction . . . . .	112
6.4.4 Step 4: Visual Encodings . . . . .	113

6.5	Evaluation . . . . .	116
6.5.1	Experiment 1: Choosing Parameters in an Explanation Method	116
6.5.2	Experiment 2: Comparing Different Explanation Methods .	119
6.6	Conclusion . . . . .	122
<b>7</b>	<b>Conclusions and Future Work</b>	<b>123</b>
<b>Appendices</b>		<b>125</b>
<b>A</b>	<b>Graph Summary</b>	<b>126</b>
A.1	Subroutines in BM-MDL Algorithm . . . . .	126
<b>B</b>	<b>Time Summary</b>	<b>128</b>
B.1	Illustration of Synthetic Dataset Creation . . . . .	128
B.2	Parameters Setting In Evaluation . . . . .	129
<b>C</b>	<b>Model Summary</b>	<b>130</b>
C.1	Feature Engineering Implementation on Explanation Matrix . . . . .	130
<b>D</b>	<b>Topological Explanation Summary</b>	<b>132</b>
D.1	Tuning Parameters in the Mapper Algorithm . . . . .	132

# List of Figures

1.1	Approaches Addressing Visual Scalability . . . . .	2
1.2	Examples of Visual Data Summarization . . . . .	3
3.1	Two Part Representation of Bipartite Graph . . . . .	15
3.2	Illustration of Bipartite Graph Summarization . . . . .	17
3.3	Visual Outcomes for MDL Summarization with Regularization . . . . .	19
3.4	Interface of Bipartite Graph Summarization System . . . . .	25
3.5	Design Considerations of Bipartite Graph Summary Visualization . . . . .	28
3.6	Filtering Interactions on Bipartite Graph Summary . . . . .	30
3.7	Faceting Bipartite Graph Summary . . . . .	31
3.8	Run Time and Quality Experiment on Bipartite Graph Summarization Algorithm . . . . .	34
3.9	Case Study of Bipartite Graph Summarization on Congress Votes . . . . .	35
4.1	Motivation of Time Series Summarization . . . . .	41
4.2	Pipeline of Time Series Summarization . . . . .	45
4.3	Illustration of Symbolic Aggregated Approximation . . . . .	46
4.4	Examples of Time Series Similarity Measures . . . . .	47
4.5	Illustration of Time Series Summary . . . . .	51
4.6	Illustration of DTW Clustering with LSH . . . . .	55
4.7	Interface of Time Series Summary Visualization . . . . .	59
4.8	Run Time Experiment on Time Series Summarization Algorithm . . . . .	63
4.9	Rendering Performance on Time Series Summary Visualization . . . . .	64
4.10	Use Case of Time Serie Summarization on Stock Market Data . . . . .	65
5.1	Visual Interpretation of Machine Learning Model Summary . . . . .	69

5.2	Examples of Explanation Matrices from Real World Data . . . . .	75
5.3	Feature Engineering Strategies for Explanation Matrix . . . . .	76
5.4	Illustration of Interactive ML Model Summarization . . . . .	85
5.5	System Overview for ML Model Summary Visualization . . . . .	86
5.6	Design Considerations for ML Model Summary Visualization . . . . .	88
5.7	Use Case of ML Model Summary on CNN Image Classifier . . . . .	91
5.8	Use Case of ML Model Summarization on Tabular Data . . . . .	94
5.9	Explanation Summary of a Text Classifier . . . . .	95
5.10	Usability Score for ML Model Summary Visualization . . . . .	97
5.11	Observations from User Study on ML Model Summary . . . . .	98
6.1	Illustration of Local Explanation's Stability . . . . .	101
6.2	Illustration of Local Explanation's Heterogeneity . . . . .	102
6.3	Pipeline of Topology Based Explanation Summarization . . . . .	103
6.4	Approximate Reeb graph of a Point Cloud . . . . .	105
6.5	Synthetic Data for Determining the Data Inputs for Visualization .	111
6.6	Synthetic Data Illustration on the Mapper Algorithm . . . . .	112
6.7	Examples of Comparisons between Topological Representations and Cluster Visualization . . . . .	113
6.8	Design Alternatives for Topological Summary Visualization . . . . .	114
6.9	Experiment of Assessing the Topological Representations with Varying Numbers of Features . . . . .	116
6.10	Experiment of Assessing the Stability of Explanations with Topological Representations . . . . .	118
6.11	Experiment of Assessing the Baselines of Local Explanations with Synthetic Dataset . . . . .	119
6.12	Experiment of Assessing the Baselines of Local Explanations with Bank Dataset . . . . .	121
B.1	Illustration of Synthetic Data Generation . . . . .	128
D.1	Parameter Tuning for the Experiment in Figure 6.9 . . . . .	133
D.2	Parameter Tuning for the Experiment in Figure 6.10 . . . . .	133
D.3	Parameter Tuning for the Experiment in Figure 6.11 . . . . .	134
D.4	Parameter Tuning for the Experiment in Figure 6.12 . . . . .	134

# List of Tables

3.1	Task Analysis of Bipartite Graph Summarization . . . . .	24
4.1	Examples of Frequent Patterns from Time Series Symbolic Representations . . . . .	49
5.1	Run Time of ML Model Summarization Algorithm . . . . .	92
5.2	Task Completion Rate for User Study on ML Model Summarization	96

# Chapter 1

## Introduction

“A picture is worth a thousand words.” Visualization has always been one of the most important techniques to enhance user’s ability to reason and understand statistical data [1]. Statistical charts, such as line charts and bar charts, are the de facto means to present information among different data-intensive systems such as Excel or business intelligence (BI) dashboards. The reason lies in the human ability to recognize graphical representations much better than textual information [2]. By connecting the customers and their purchases with a node-link diagram, we can see the popular products. By joining the temporal data with a line, we can see the trends. By drawing the IF-ELSE statements as a flow chart, we can see the logic. Therefore, visualization delivers the insights for people to understand numbers and take action.

However, in the era of big data, people usually need to make decisions based on millions of data. Simply plotting all the relevant information on a single chart without preprocessing is far from ideal when users need to interact with millions of transactions and trends. This problem is called *Visual Scalability* – the goal of visual analysis is to acquire *insights*, and is affected by the *size of the database* and *number of distinct items in the visualization* [3]. In other words, visualization becomes ineffective when *overplotting* occurs. To address visual scalability, works apart from data summarization mainly fall into three separate approaches: sampling, feature extraction, and interaction. Sampling reduces the amount of data visualized (e.g., nodes in large graph [4]) without losing much context so that the picture becomes clearer. Feature extraction attempts to encode or highlight several statistical

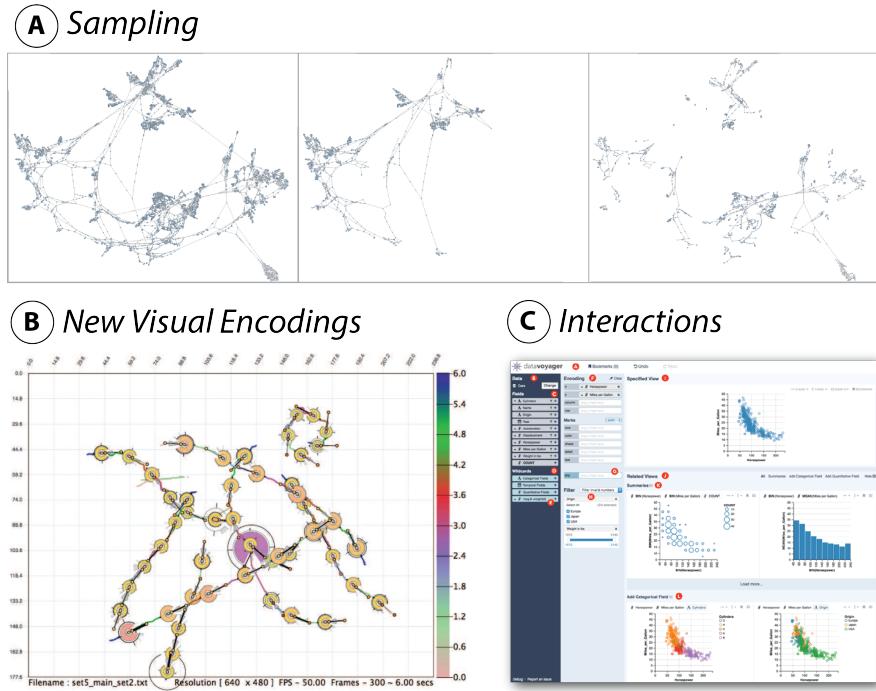


Figure 1.1: Examples of approaches reducing visual scalability. **(a)** graph sampling [4]. **(b)** feature extraction for cell movement [5]. **(c)** interactions for visual recommendation [9].

information (e.g., average or counts) to address specific tasks [5, 6, 7]. Interaction, which usually follows the visual analytic mantra: overview first, zoom and filter, then details-on-demand [8], empowers users to select and explore a subset of information at a time [9].

For data summarization, we are interested in grouping the data before visualizing them. Generally, data summarization aims at finding a compact description of a dataset. For example, similar items are grouped together to patterns, and these patterns are prioritized for display so that the visualization will only show a few but important insights without much redundancy. In other words, if we can compress the visual complexity of the data, we can visualize it with the least amount of encodings, thus making the visualization scalable.

To provide concrete examples of visual data summarization, we can take a look at Figure 1.2. From the left-hand side of each chart, it is hard to identify what kinds of patterns existed in the visualization. However, by arranging the same dataset

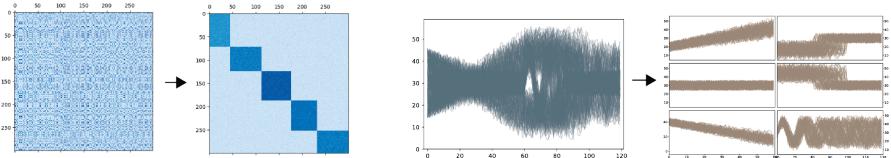


Figure 1.2: Examples of visual data summarization.

into appropriate groups and orders, the patterns such as blocks and trends become visible. Thus, we define the process of visual data summarization as the grouping of homogeneous data, which results in clear and compact visualization. Overall, there are two main challenges in visual data summarization. First, it is crucial to formulate the summarization goals into data modeling frameworks for different data primitives. The formulation should be robust to more complex combinations of patterns in the dataset. Second, the algorithms to summarize the data should scale to large data, given the needs of interactive data explorations. Users should be able to compute the results on the fly using few computation resources like personal laptops. The summaries should also come with the ability to accept user inputs for refinements throughout the interactive data analysis and support top-down and detail-on-demand analysis.

## 1.1 Contributions

In this thesis, we propose methods to efficiently summarize three types of data: bipartite graph, time series, and classification models. Summarizing bipartite graphs provides an example to group discrete attributes like the nodes in the data. Then, we extend the idea to time series data, which are real-valued sequences. Afterward, we explore the ways to summarize logic in a classification model that has no structured data at the beginning. In short, we summarize the contribution of each chapter of this thesis.

**Summarizing Bipartite Graph.** Bipartite graphs model the key relations in many large-scale real-world data: customers purchasing items, legislators voting for bills, people's affiliation with different social groups, faults occurring in vehicles, etc. However, it is challenging to visualize large-scale bipartite graphs with tens of thousands or even more nodes or edges.

We propose a novel visual summarization technique for bipartite graphs based

on the minimum description length (MDL) principle. The method simultaneously groups the two different sets of nodes and constructs aggregated bipartite relations with balanced granularity and precision. It addresses the key trade-off that often occurs for visualizing large-scale and noisy data: acquiring a clear and uncluttered overview while maximizing the information in it. We formulate the visual summarization task as a co-clustering problem and propose an efficient algorithm based on Locality Sensitive Hashing (LSH) that can easily scale to large graphs under reasonable interactive time constraints.

**Summarizing Time Series.** Time series capture the behavior of many large-scale real-world processes, from stock market trends to urban activities and health informatics. However, it is challenging to visualize many time series that cover a long time span.

We propose a new algorithm that summarizes time series using sequential patterns. It transforms the series into a set of symbolic sequences based on subsequence visual similarity using Dynamic Time Warping (DTW). Then, it constructs optimum disjoint groupings of similar subsequences based on the frequent sequential patterns. The grouping result, time-series visual summary, addresses a key trade-off in visualizing large and noisy time series data: an uncluttered superposition with a minimum number of small multiples. Unlike common clustering techniques, our algorithm extracts similar subsequences (of varying lengths) that are aligned in time.

**Summarizing Classification Models.** Feature importance explanation methods on machine learning models have been popular to provide accurate explanations to complex models and different data primitives such as tabular, image, or text. However, unlike other global explanation methods such as decision trees or rules that help users with little expertise in machine learning to understand predictive models, feature importance explanations have been almost exclusively used to explain the models locally for a single instance.

We propose an interactive visual summarization technique to understand and explore feature importance explanations at scale, leveraging their versatility and generality to a scalable understanding of predictive models. By grouping the instances and important features, we can present the model’s rationale in a matrix-based visualization to help general users develop and verify the classification model’s

logical expressions without in-depth knowledge of machine learning.

**Summarizing Heterogeneous Model Interpretations.** Different feature importance explanation methods generate predictive model interpretations with different intrinsic meanings and values. It thus becomes hard to compare the behavior and quality of these methods.

We propose a topology-based framework to extract a simplified low-dimensional representation of the set of local explanations for binary classifications. This is accomplished by first modeling the relationship between the explanation space and the class predictions as a scalar function and computing the topological skeleton of this function. The persistence diagram of the features of this skeleton acts as a signature for such functions, which are then used to analyze and compare different explanation methods.

## 1.2 Outline

Chapter 2 presents an in-depth overview for approaches addressing the challenges of visual scalability. Chapter 3 presents the Bipartite Graph Summary for the interactive summarization of large bipartite graphs. Chapter 4 describes the Time Series Summary for summarizing the subsequence clusters in the large time-series data. Chapter 5 presents the Machine Learning Model Summary for the scalable understanding of feature importance-based explanations that convey a classifier’s rationale to a dataset. Chapter 6 presents our work to summarize and compare heterogeneous black box explanation models through topology based data summarization. Finally, Chapter 7 concludes the dissertation, highlighting potential future work.

# Chapter 2

## Overview of Large Scale Visual Analytics

As mentioned in Chapter 1, visual scalability is the main challenge in the goal of visualizing large-scale data to users. While visual scalability can refer to several aspects like high dimensional data or multimodal distributions [10], this thesis addresses the most common scalability problem – *overplotting*. Such a problem easily occurs with the increasing cardinality (i.e., number of rows) of the dataset. In an overly plotted visualization, the patterns of the dataset are mitigated and ambiguous. For example, too much overlapping in scatterplots or node-link diagrams will only display giant “dark masses” in the charts [11]. In this chapter, we describe four main areas, including data summarization, that address these visualization challenges.

### 2.1 Sampling

A popular approach to address the visual scalability problem is to sample the visualized data so that the subset of data can present the visual structure of the whole dataset with clarity. Intuitively, the challenge is to scale down the data size without omitting rare but important data items representing the patterns. This can be done automatically or interactively. For automatic sampling processes, we can compare the patterns between the original data and sampled data with statistical tests. For example, in graph sampling, we can define the graph properties as a set

of distributions such as the counts of nodes with different in-degrees or out-degrees, then we can compare the distributions between the sampled data and full data with hypothesis testing [12]. On the other hand, we can visualize different subsamples that plot the dataset with varying sampling rates and let users explore the data with various resolutions [13]. By interacting with different subsamples, users can acquire various sampling outcomes to make more accurate judgements.

Sampling has a clear advantage of efficient computations which enables ad-hoc explorations of large data. However, it does not access to full information of the dataset. Such a data loss can induce uncertainty which can adversely affect user trust levels or lead to misconceptions when the users are overly trusting [14].

## 2.2 Feature Extraction

Another way to deal with visual scalability is to extract and visualize the important features from the data that directly address the needs of data exploration. For example, if users are interested in distributions of points grouped by different labels in a scatter plot, we can group the dense regions for points in each label with smooth shapes [15]. Also, in network visualization, the node-link diagrams can be represented as Graph Thumbnails that emphasize the coarse structural properties for comparisons among different graphs using small multiples [16]. These visualization techniques usually change the existing visual representations to overcome perceptual scalability issues. However, the new representations might need additional cognitive training that might take a long time to be adapted to the general public [17]. Also, since the features extracted often aim to solve a specific problem, the insights obtained from such visualization might not apply to more general tasks.

## 2.3 Interactions

Visual analytics often address the scalability problem by leveraging user interactions. One rule of thumb is the “Visual Analytics Mantra” – “Overview first, zoom and filter, then details-on-demand” [8]. Interactive visualization often uses dynamic queries [18] that updates the subsets of data to be visualized continuously when the users adjust sliders or select items in the interfaces. The main challenge for this

area is the response time for providing the results. Slow responses (500ms or more) per interaction will cause users to reduce their activities and cover fewer data in the exploration, which ultimately results in fewer insights [19]. To address this issue, we can use pre-computed data cubes [20] or prefetching mechanisms [21, 22] to compute the anticipated results before the users start the interactions.

The limitation of user interactions to explore large-scale data is the positive linear relationships between data size and the number of interactions. Users often use interactions to slice and dice the whole dataset into moderate-sized subsets for browsing. Thus, when the size of data increases, analyzing the data becomes more time-consuming and labor-intensive.

## 2.4 Data Summarization

Our thesis falls into the category of visual data summarization. In general, data summarization can be seen as the approach to group the visualized elements so that in each group, the visual elements are homogenous (i.e., look similarly). The assumption for its effectiveness is that there are many repetitions and redundancy of shapes (e.g., lines with the same trend) in the data, and users' goal is to discover these structures. Some notable examples include the visual summarization of event sequences [23] and spatio-temporal data [24] to discover the respective patterns for various actionable insights. Since the visual items within each group can be easily compressed as they have similar shapes, the visualization can be drastically simplified while representing all information in the dataset. In other words, visual scalability is addressed by *rearranging* the visual elements.

To apply data summarization techniques to large-scale visual analytics, we need to address two challenges. First, we need to define an objective function to evaluate if a grouping is good or not. This allows us to group the data without supervised metrics and determine the most visually friendly visual outcome. One common way is to use information theory, which essentially calculates the number of bits needed to recover the original data from a compression result. In visualization, we can treat it as the residuals between an aggregation of a group and visual elements inside the group. Also, contrary to clustering, data summarization does summarize both the inputs and the attributes among the inputs. For example, we

should split a time series into multiple groups of segments if it contains multiple trends throughout the duration. Then, second, given an objective function, how can we group the visual elements efficiently? As the grouping is concerned about the structures beyond a single dimension (i.e., clustering rows) in the dataset, the number of possible partitions can skyrocket quickly. Therefore, efficient methods that compress the computation time to a linear scale are needed, especially in interactive visual analysis scenarios.

Our thesis uses four types of data to demonstrate the use of data summarization to address visual scalability, with an increasing difficulty. First, we use bipartite graph to demonstrate how to summarize discrete nodes into a compact group of similar nodes. Then, we demonstrate how to achieve similar outcome for continuous variables with subsequence summarization of real-valued time series. Afterwards, when users search for patterns in complex Machine Learning models which there are not even a concrete data representation, we demonstrate how to reduce the problem into a data summarization problem. Lastly, when heterogeneous data needs to be compared, like different black box local explanations for predictive models, we show how to simplify them with topological data summaries.

## 2.5 Summary

Data summarization is a powerful data mining approach to group data into a smaller set of homogenous visual attributes. With data summaries, we can plot the visual attributes in separate charts to achieve visual clarity in statistical charts with large-scale data. Apart from data summarization, sampling is another approach to reduce the data needed to be displayed, feature extraction can consolidate a subset of properties for visualization, and interactions are useful to explore the whole data one part at a time.

# Chapter 3

## Bipartite Graph Summary

Understanding bipartite relations is the key to gain insight from data in a variety of application domains. Such activity can often be seen in user preferences identification on movie recommender systems [25], market basket analysis on sales records [26] , political leanings analysis on roll-call vote records [27] and relationship discovery in urban open data [28, 29]. As "a picture is worth a thousand words," visualization plays an important role in landing a good hypothesis or direction for domain experts to analyze bipartite relations at scale.

Recently many visualization techniques have been proposed to support bipartite relation analysis [30, 31, 32, 33, 34]. Nonetheless, the increasing volume and complexity of the data bring new challenges. First, revealing all information at once will exceed human's cognitive ability to conduct effective analysis. In our use cases (Section 3.6.2) data either contains more than 170, 000 bipartite connections or contains 5,000 - 43,000 nodes depending on the subset selected for analysis. Plainly showing the data will be considered as infeasible. A better way to help analysts start the data exploration is to construct a broad overview of the data instead of showcasing each individual entity and bipartite connection. Second, noises are prevalent in real-world datasets, thus the insights are common to contain artifacts as well. Thus, what analyst needs is a robust visualization technique that reveals the most general and salient patterns in the entire dataset.

To address these challenges, we describe a novel visual summarization technique for bipartite relational data using an *information-theoretic* approach. We apply the Minimum Description Length (MDL) principle [35] which provides a criteria

to optimize the aggregation of bipartite relations to create a high-level overview, balancing visual complexity and information loss in the display. We visually represent the original data with the aggregated bipartite connections, and in the meanwhile model the information loss with the corrections needed to recover the original data from the aggregated graph.

Using the MDL principle in visual data summarization has been seen in hierarchical data [36] and event sequence data [23], but applying it to large scale bipartite relation data imposes new challenges and opportunities. Apart from formulating the principle for bipartite relations, we also describe how to speed it up with locality sensitive hashing (LSH) [37], which effectively improves the running time without significantly degrading the results. We further introduce a tailored and space efficient visualization design inspired by visual adjacency lists [38] to display the aggregated bipartite relations. The compact visual design fits nicely in a small multiples display for visual comparison of bipartite relations across different subsets of data, which can be created by faceting on a selected node attribute. A comprehensive visual analytic system is developed as well to support data exploration at varying levels-of-detail, correlating domain-specific node attributes with the relation patterns, and filtering and selecting subsets for focused analysis to cope with the challenges in usabilities [39]. In short, our contributions are as follows:

1. We apply the MDL principle to pack large scale and noisy bipartite relation data into a highly compressed representation which is suitable for a coarse-level overview of the data.
2. We propose an efficient algorithm based on LSH to optimize the MDL optimization process to facilitate interactive analysis.
3. We introduce novel visual analytics techniques and interaction designs for exploring large scale multivariate bipartite graph data.
4. We present two example usage scenarios with real-world datasets and domain specific analytic tasks that demonstrate the usability, effectiveness and general applicability of our technique.

## 3.1 Related Work

Bipartite graph exists in many application domains and a variety of visualization techniques have been developed in the past. Here we categorize the related work into plain bipartite graph visualization and bipartite graph aggregation for a high-level overview of the data.

### 3.1.1 Bipartite relation visualization

The most common approach to visualize bipartite relations is to position two sets of nodes on separated regions on the display and draw edges as (curved) lines between the nodes. Related techniques can be seen in various visualizations including semantic substrates [40], PivotPath [41], Jigsaw [42], and parallel node-link bands [43]. All of them applied such principle to display bipartite relation and encode additional attributes with node sizes and colors, edge widths, opacities and coloring, etc. Another layout style for bipartite graph is unimodal, which treats the bipartite graph as a whole without spatial separation. Such visualization techniques use color, shape or other visual channels to distinguish the sets, which can be seen in FacetAtlas [44], OntoVis [45] and Anchored Maps [46]. Besides bipartite graphs can also be represented by adjacency matrices. The visualization techniques usually build upon the simplest form of adjacency matrices and enhance it with additional visual cues or interactive functionalities [47, 48]. Row and column seriation techniques further facilitate pattern recognition in matrix displays [49, 48, 50]. Besides node-link diagrams and adjacency matrices, bipartite graphs can also be visually represented by a hybrid of the two, highlighting the bi-cluster structure detected by automatic algorithms as in Bixplorer [30], or visualizing additional topological structures on top of the bipartite relations [51].

Bipartite graphs can also be generally represented as sets covered in a recent survey about the state-of-the-art of set visualization techniques [52]. In general, most of the techniques visualize a moderately sized dataset with at most hundreds of sets/items. Our work focuses on providing a concise overview of large scale bipartite graphs with tens of thousands or even more nodes and edges.

The techniques discussed above typically display a bipartite graph in its original form with individual nodes and bipartite connections. To handle large scale data

aggregation is usually necessary, which we discuss in the next section.

### 3.1.2 Bipartite relation clustering and visualization

To support scalable analysis and pattern detection on bipartite graph data, work has been focusing on graph summarization through edge or node aggregation. We group the graph summarization algorithms and the corresponding visualization techniques into two major categories:

Algorithms including CHARM [53] and LCM [54] extracts bi-cliques in coordinated bipartite relations. Bixplorer [30, 55, 56], BiSet [33], and BiDots [34] utilize them to identify and visualize bi-cliques. The visualization techniques display such structures using overlays on top of matrices/node-link diagrams, bundles edges within a bi-clique in node-link diagrams [57, 58, 33], or combine both to form a hybrid representation [30, 31]. However, real-world data is usually noisy and missing links may create a lot of fragmented bi-cliques that overwhelm the users. Although interactive exploratory visualization techniques can partly alleviate this issue [33, 34], a high-level overview still cannot be obtained easily.

Another category of bipartite graph clustering algorithms simultaneously group the nodes in the two partitions, relaxing the requirements on bi-cliques. Spectral co-clustering [59] and spectral bi-clustering [60] were classical methods. Recently, some bipartite visualization techniques utilize those algorithms to perform data aggregation and reduce visual clutter in the display, including Xu *et al.* [32] and Ming *et al.* [61].

ViBR falls into this research domain, although we formulate the co-clustering problem with the information theoretic minimum description length principle (MDL) [35], which allows the method to directly quantify and minimize the information loss in the visual display. Recently, Veras and Collins [36] and Chen *et al.* [23] apply it to construct high-level visual summary of hierarchical and temporal event sequence data respectively to balance the conciseness and information content in visualization. The MDL principle has also been applied to graph summarization in the database and data mining research community. Navlakha *et al.* [62] is the first work proposing summarization of general graphs with bounded error based on a two-part representation of summary graph and corrections.

A similar approach (SCMiner) has been proposed by Feng *et al.* [63] for weighted

bipartite graphs. However, our algorithm is accelerated with LSH which experimentally reveals that our algorithm has over 300 times speed gain as compared to SCMiner and three times speed gain compared to Cross-association [64] (a matrix clustering algorithm), making it more suitable for interactive exploration of data.

We further propose a novel design to visualize the aggregation inspired by the visual adjacency list design for dynamic graph visualization [38]. The visual design provides a compact overview of the bipartite relations and facilitates visual comparison across different subsets in the data. The analysts can facet on a selected node attribute and compare the bipartite connections.

## 3.2 Minimum Description Length (MDL) for Bipartite Graph Summarization

In the following discussion we use  $U$  and  $V$  to denote the two sets of nodes and the bipartite graph is  $R \subseteq U \times V$ . In this section we first introduce a two-part representation of a bipartite graph, inspired by the two-part representation of general graphs in [62], which consists of a summary graph and a set of residual edges. Combining it with the MDL principle we formulate an optimization goal to identify a simultaneous grouping (i.e., co-clustering) of  $U$  and  $V$  such that the corresponding summary graph can describe the original data balancing complexity and information loss.

### 3.2.1 Two-part representation of a bipartite graph

The two-part representation is illustrated in Figure 3.1. Given a simultaneous grouping of the nodes in  $U$  and  $V$  (Figure 3.1(a)), it consists of:

**A summary graph**  $\mathbb{S}$  with the meta-nodes and their interconnections as illustrated in Figure 3.1(b). An edge is created between two meta-nodes if the connection is dense in the original graph. For example, it is almost a bi-clique between  $\{1, 2, 3\}$  and  $\{a, b, c\}$ , therefore in the summary graph an edge is created between these two meta-nodes. On the other hand only one edge exists in the original graph between  $\{1, 2, 3\}$  and  $\{d\}$ , so the summary graph between these two has no edges.

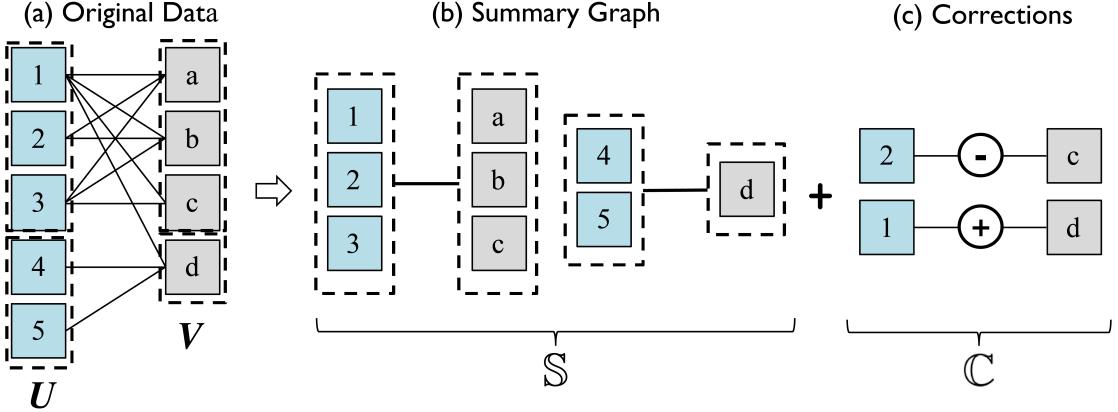


Figure 3.1: A bipartite graph can be represented as a summary graph  $\mathbb{S}$  with corrections  $\mathbb{C}$ . The original set requires 11 units of spaces (11 edges) while  $\mathbb{S}$  and  $\mathbb{C}$  together only require 4 (2 in  $\mathbb{S}$  and 2 in  $\mathbb{C}$ ).

**A list of corrections**  $\mathbb{C}$  that can recreate the original data from the summary graph. The summary graph is an approximate representation of the original data. With the meta-edges we can infer that the interconnection between two clusters is dense. However it is not enough to recover the exact bipartite connections. We need to include additional corrections to remove the non-existing edges. For example, between  $\{1, 2, 3\}$  and  $\{a, b, c\}$ , every edge exists except for  $(2, c)$ , therefore we add an additional correction to remove  $(2, c)$  which does not exist in the original graph (Figure 3.1(c)). On the other hand, even when meta-edges do not exist in the summary graph, it is still possible for some edges to appear in the original data, therefore another type of correction add edges back. For example, between  $\{1, 2, 3\}$  and  $\{d\}$  no edges exist except for  $(1, d)$ , so we add back  $(1, d)$  (Figure 3.1(c)).

Combining the summary graph and the corrections we can fully recover the original graph. The two-part representation is therefore a *lossless* representation of the original data. The summary graph  $\mathbb{S}$  can provide a coarse-level overview of the data and the corrections  $\mathbb{C}$  model the information loss in the display. The visual complexity dramatically decreases in the overview and user can immediately grasp the dominant connectivity patterns in the bipartite graph. Visual abstraction of the data is even more critical for understanding bipartite relations with thousands or even millions of nodes and edges when it becomes almost impossible to fit all the raw data on a single screen.

The remaining problem is how to identify a summary graph which can best

represent the underlying data balancing the visual complexity and information loss. This problem eventually boils down to identifying an optimal grouping of the nodes in  $U$  and  $V$  based on which we can bundle the edges to form the summary graph.

### 3.2.2 The MDL principle

We propose an algorithm to obtain an optimal grouping of the nodes in  $U$  and  $V$  simultaneously following the minimum description length (MDL) principle. The MDL principle states that the best model (or hypothesis) of a dataset should minimize its total description length  $L$ , which consists of the model description length and the description length of the original data with the help of the model:

$$L = L(M) + L(D|M)$$

For a bipartite graph, the model is the summary graph  $\mathbb{S}$  and given a summary graph we can use the corresponding corrections part  $\mathbb{C}$  to recover the original data. Our goal is to obtain an optimal grouping of the nodes such that it can minimize the total description length of the summary graph and the corrections. To state it more formally, we denote a bipartite graph as  $R \subseteq U \times V$ . Our goal is to identify a partition of  $U$  and  $V$  such that it can minimize the total description length:

$$L_R(P, Q) = L(\mathbb{S}) + L(\mathbb{C})$$

where  $P$  is a partition of  $U$ ,  $Q$  is a partition of  $V$ ,  $\mathbb{S} \subseteq P \times Q$  is the summary graph and  $\mathbb{C}$  is the set of corrections. The definition of  $\mathbb{C}$  is:

$$\mathbb{C} = (\cup_{(p,q) \in \mathbb{S}} p \times q) \oplus R$$

where  $\oplus$  denotes the disjunctive union between sets. Since we only need to store the meta-edge information for the summary graph, the description length is  $L(\mathbb{S}) \propto \|\mathbb{S}\|$  and the description length of the corrections is  $L(\mathbb{C}) \propto \|\mathbb{C}\|$ . We further introduce the parameters  $\alpha$ ,  $\beta_P$  and  $\beta_Q$  to control the penalty of the corrections and the number of clusters, similar to Veras and Collins *et al.* [36] and Chen *et al.* [23]. To

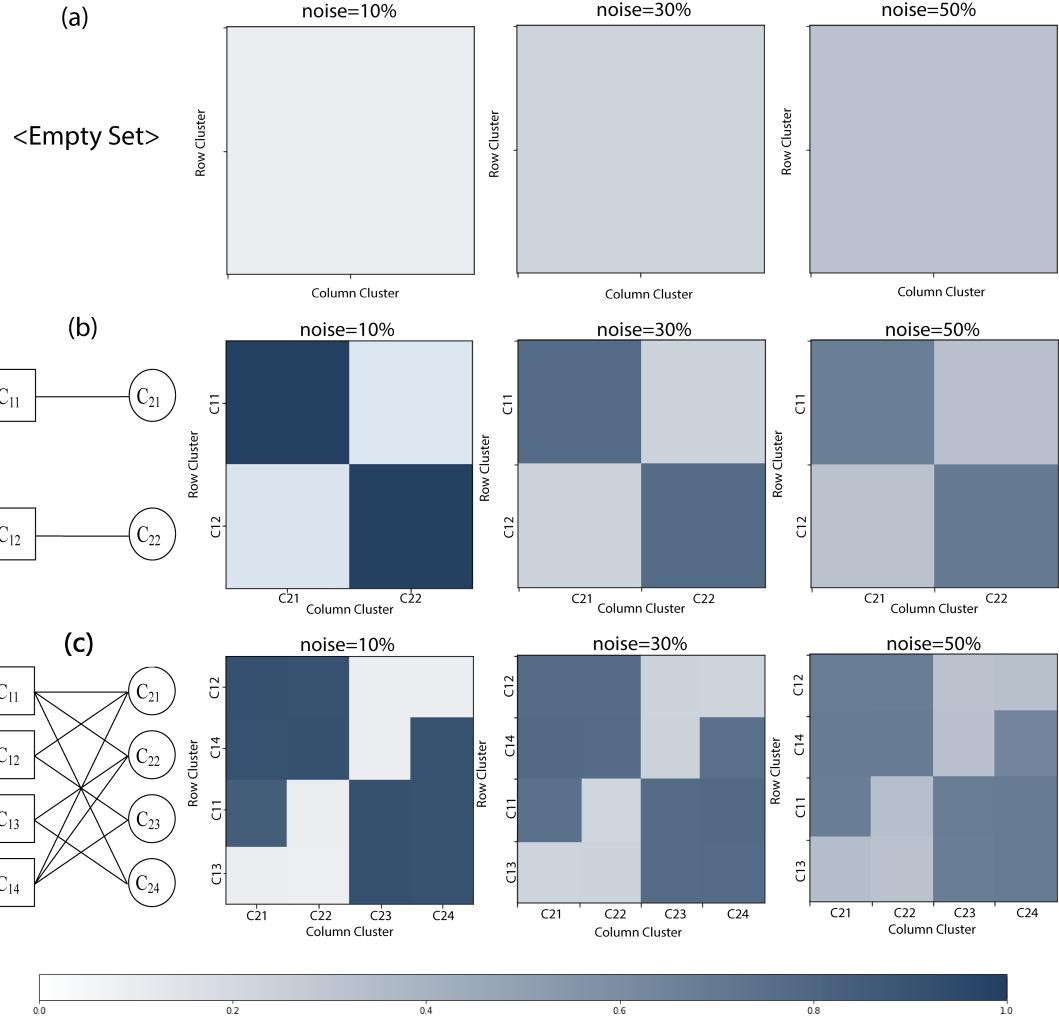


Figure 3.2: Illustration of summarization results of the MDL formulation using three synthetic datasets with different ground truth co-cluster structures as shown on the left of the figures. Each partition is computed three times with noise equal to 10%, 30%, and 50%. The result shows that our approach captures the structures in the synthetic data and is robust to noises.

sum up, our goal is to find  $P$  and  $Q$  that can minimize the loss function:

$$L_R(P, Q) = \|\mathbb{S}\| + \alpha \|(\cup_{(p,q) \in \mathbb{S}} p \times q) \oplus R\| + \beta_P \|P\| + \beta_Q \|Q\| \quad (3.1)$$

where  $\beta_P \|P\| + \beta_Q \|Q\|$  can be considered as two regularization terms which penalize large number of node clusters. Larger  $\beta_P$  and  $\beta_Q$  results in smaller numbers of clusters. An example of the effect can be found in Figure 3.3.

We illustrate our approaches' outcome by visually inspecting the partitioning results for different bipartite graphs with ground truth co-cluster structure with the help of our algorithm (discussed in the next section). The results are shown in Figure 3.2. We generate three synthetic datasets in the same way as in SCMiner [63]: an empty set with no co-cluster structures Figure 3.2(a), one with two one-to-one relations (Figure 3.2(b)), and one relatively complex graph with more node clusters (Figure 3.2(c)). We increase the noise gradually to 10%, 30%, and 50% by creating additional or missing edges in the bipartite graph for each synthetic dataset. It can be observed that overall our approach can capture the structure in the bipartite graph without any supervised number of groups and has good robustness over noises. Our method creates less debris, as shown in an empty set Figure 3.2(a) that when noises are increasing, the partition structure is less likely to break into more concrete pieces, and the overall structure is maintained.

### 3.3 Computing MDL Representation

In this section, we first introduce a basic algorithm to find partition  $P$  and  $Q$  and the corresponding summary graph  $\mathbb{S}$  that can minimize the description cost described in Equation 3.1. Then we describe a speed up strategy that applies LSH [37], an efficient nearest neighbor search algorithm. We report the results of a series of empirical experiments to verify the robustness of the algorithm and compare it with other co-clustering algorithms.

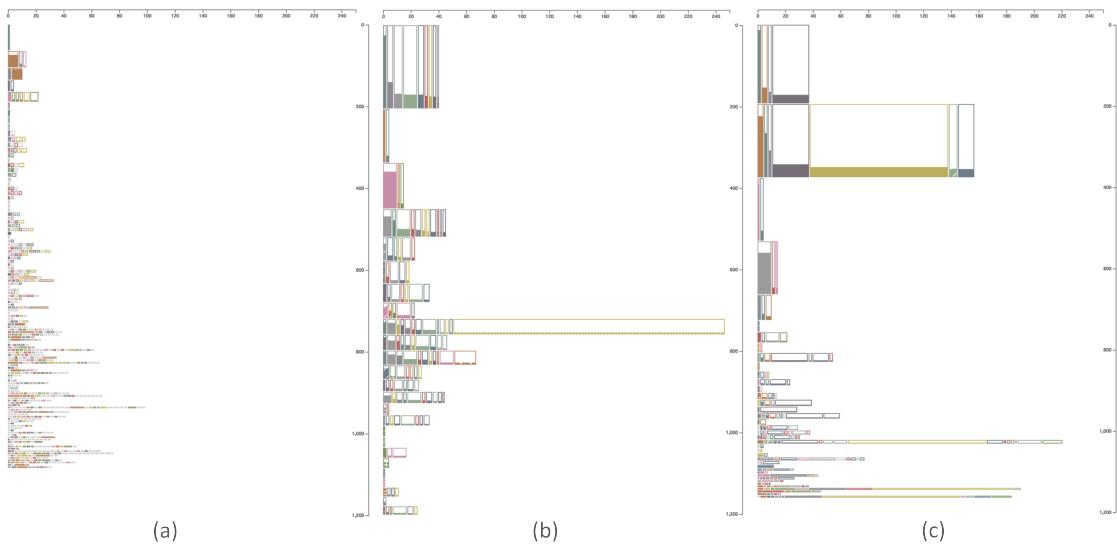


Figure 3.3: Summarization results of our vehicle dataset consisting of 2967 vehicle logs with around 250 fault codes using different values of  $\beta$ . (a)  $\beta = 1$ ; (b)  $\beta = 10$ ; (c)  $\beta = 100$ ; Only co-clusters with more than 10% density are included.

---

**ALGORITHM 1:** BM-MDL

---

**Input:** Two sets of nodes  $U$  and  $V$  and the bipartite relation  $R \subseteq U \times V$

**Output:** Partition of  $U$ , denoted as  $P$  and partition of  $V$ , denoted as  $Q$ , summary graph  $\mathbb{S} \subseteq P \times Q$

```

graph  $\mathbb{S} \subseteq P \times Q$ 
    /* Initialization step
    1  $P = \{\{u\} | u \in U\}; Q = \{\{v\} | v \in V\}$ 
    2  $\mathbb{S} = \{(\{u\}, \{v\}) | (u, v) \in R\}$ 
    3  $\Delta L_{max} = 1$ 
        /* Iterative merging step, until no cost reduction is possible */
    4 while  $\Delta L_{max} > 0$  do
        /* Merge clusters in  $P$ 
        5  $p_0 = random\_select(P)$ 
        6  $\Delta L_{max} = 0, p_{max} = undefined$ 
        7 for  $p \in two\_hop\_neighbors(p_0, \mathbb{S})$  do
             $\Delta L = 0$ 
            for  $q \in neighbors(p, \mathbb{S}) \cup neighbors(p_0, \mathbb{S})$  do
                 $\Delta L += cost\_reduction\_for\_bundling((p, q), (p_0, q), R, \mathbb{S})$ 
            end
             $\Delta L += \beta_P$ 
            if  $\Delta L > \Delta L_{max}$  then
                 $\Delta L_{max} = \Delta L, p_{max} = p$ 
            end
        end
        /* Merge two clusters if cost reduction is possible */
        17 if  $\Delta L_{max} > 0$  then
            merge( $p_0, p_{max}, R, \mathbb{S}$ )
        end
        /* Same procedure as for  $Q$ ...
    20 end
    21 return  $P, Q$ , and  $\mathbb{S}$ 

```

---

### 3.3.1 The *BM-MDL* algorithm

We first propose a basic version of the algorithm named *BM-MDL* (bipartite graph mining with MDL) based on the approach proposed by Navlakha *et al.* [62]. The algorithm follows a bottom-up and greedy approach. Initially each node is treated as an individual cluster. In each iteration, we identify a pair of clusters to merge that will result in the maximum reduction in description length. The process stops when the total description length no longer decreases. As a simple

speed up strategy, we use a randomized approach which picks a cluster randomly and merge it with the best node in its hop-2 neighborhood, similar to Navlakha *et al.* [62]. For example, in Figure 3.1, if node 1 is first chosen, the algorithm will try to merge it with its 2-hop neighbors including node 2, 3, 4 and 5. Merging node 1 and 2 creates two meta-edges  $(\{1,2\}, \{a\})$  and  $(\{1,2\}, \{b\})$  in  $\mathbb{S}$  and two additional correction edges  $(1,c)$  and  $(1,d)$  in  $\mathbb{C}$ . Assuming  $\alpha = 1$ ,  $\Delta L = 2 + \beta_P$  since there are two edges less in total and the number of node clusters in  $P$  reduces by one. Similarly, the algorithm calculates  $\Delta L$  by merging node 1 with node 3, 4 or 5, this result in  $\Delta L = 3, 1, 1$  respectively with an additional constant  $\beta_P$ . The algorithm therefore will choose node 3 and merge it with node 1. The procedure is described in detail in Algorithm 1. The subroutine *cost\_reduction\_for\_bundling* in line 10 calculates the change in description length by merging two meta-edges in the summary graph, which is a necessary step for merging two meta-nodes. The subroutine *merge* in line 18 updates the partitions  $P$  or  $Q$  and the summary graph  $\mathbb{S}$  by merging two meta-nodes. In Appendix (Section A) we provide more detail on the two subroutines *cost\_reduction\_for\_bundling* and *merge*. In each iteration, the algorithm computes the description length reduction for all the hop-2 neighbors of a node. Assuming that the nodes have average degree of  $d$ ,  $O(d^2)$  hop-2 candidate pairs have to be checked for each iteration [62].

---

**ALGORITHM 2:** BM-MDL-LSH

---

**Input:** Two sets of nodes  $U$  and  $V$  and the bipartite relation  $R \subseteq U \times V$

**Output:** Partition of  $U$ , denoted as  $P$  and partition of  $V$ , denoted as  $Q$

```

/* Initialization step */  

1   $P = \{\{u\} | u \in U\}; Q = \{\{v\} | v \in V\}$   

2   $\mathbb{S} = \{(\{u\}, \{v\}) | (u, v) \in R\}$   

3   $\Delta L_{max} = 1$   

4   $\theta = 0.99, \theta_{cutoff} = 0.1, \lambda_{decay} = 0.9$   

/* Iterative merging step */  

5  while  $\theta > \theta_{cutoff}$  do  

6       $T_P = build\_lsh\_table(P, \mathbb{S}, \theta)$   

7       $T_Q = build\_lsh\_table(Q, \mathbb{S}, \theta)$   

8      while  $\Delta L_{max} > 0$  do  

9          /* Merge meta-nodes in  $P$  */  

10          $p_0 = random\_select(P)$   

11          $\Delta L_{max} = 0, p_{max} = undefined$   

12         for  $p \in query\_lsh\_table(p_0, T_P)$  do  

13              $\Delta L = 0$   

14             for  $q \in neighbors(p, \mathbb{S}) \cup neighbors(p_0, \mathbb{S})$  do  

15                  $\Delta L += cost\_reduction\_for\_bundling((p, q), (p_0, q), R, \mathbb{S})$   

16             end  

17              $\Delta L += \beta_P$   

18             if  $\Delta L > \Delta L_{max}$  then  

19                  $\Delta L_{max} = \Delta L, p_{max} = p$   

20             end  

21         end  

22         /* Merge two clusters if cost reduction is possible */  

23         if  $\Delta L_{max} > 0$  then  

24              $merge(p_0, p_{max}, R, \mathbb{S})$   

25         end  

26         /* Same procedure as for  $Q$ ... */  

27     end  

28      $\theta *= \lambda_{decay}$   

29 end  

30 return  $P, Q$ , and  $\mathbb{S}$ 

```

---

### 3.3.2 Speeding up With LSH

The basic version of the algorithm is extremely time consuming due to the need to compute and compare the potential cost reductions for merging each pair of clusters with 2-hops in the bipartite graph. To speed up the algorithm, we employ locality sensitive hashing (LSH) [37], which is an efficient method for nearest neighbor search. We use LSH to efficiently identify the clusters with the most similar bipartite connections measured by Jaccard similarity. The procedure is described in Algorithm 2. Since LSH allows very efficient search for nodes with similar bipartite connections, the number of candidate pairs to check is much less than  $O(d^2)$  [37].

Notice that the inner while loop in Algorithm 2 is similar to Algorithm 1 except that now instead of identifying the best clusters to merge in the hop-2 neighbors, we only search among those pairs of clusters with Jaccard similarity coefficient above a certain threshold  $\theta$  (line 12) which can be efficiently done approximately with LSH [37]. Using the same example in Figure 3.1 and above, with a proper setting of  $\theta$ , if node 1 is first chosen, the algorithm will compare with node 3 only since they have the greatest similarity of connecting edges, eventually 1 and 3 will be merged as the description length will decrease. The outer loop sets  $\theta$  at a relatively high value (close to 1.0 as in line 4) initially and gradually decrease it by a fixed decay rate  $\lambda$ . This allows the algorithm to prioritize the most similar clusters in early iterations but still being exhaustive in the search at later stages.

## 3.4 Design Requirements

While creating the visual representations and analytics system we faced many design decisions. To formulate our desiderata we interviewed a group of data scientists in the automotive industry, whose main responsibility is to analyze large amount of vehicle log data capturing the occurrences of different fault signals in cars. One typical question they are trying to answer with such log data is: are there any groups of cars that exhibit the same set of symptoms over the course of their lifetimes? Insights like this enable large scale troubleshooting and exposes hidden market segments since cars with similar faults will likely need common parts for replacement or similar services in the repair shops. We list the analytic tasks

Use case 1: vehicle fault analysis ( $R \subseteq \text{vehicles} \times \text{faults}$ )	General case: bipartite graph analysis ( $R \subseteq U \times V$ )	Use case 2: roll-call vote analysis ( $R \subseteq \text{legislators} \times \text{bills}$ )
Identify vehicles with similar faults	<b>T.a1</b> Identify nodes in $U$ with similar bipartite connections	Identify legislators that vote similarly
Identify faults that co-occur in cars	<b>T.a2</b> Similar as T.a1 for $V$	Identify bills voted by similar legislators
Compare faults that occur in different vehicle clusters	<b>T.a3</b> Compare linkages between node clusters in $U$ and $V$	Compare bills voted by different clusters of legislators
Assess the deviations of fault occurrence patterns for vehicles in the same cluster	<b>T.a4</b> Assess the amount of corrections needed to recover $R$ from $\mathbb{S}$	Assess the deviations of voting patterns for legislators in the same cluster
Compare fault occurrences for cars with different shared properties e.g. engine types	<b>T.b</b> Compare bipartite connections for nodes with different attribute values	Compare voting records of different parties

Table 3.1: Task analysis. The two use cases correspond to the example usage scenarios are described in Section 3.6.2.

gathered from the interview in the first column in Table 3.1.

We further investigate the possibility of generalizing the tasks in vehicle log analysis to other application domains. In Table 3.1 (column 2) we first generalize these tasks for an abstract bipartite graph with node attributes. Then we instantiate the analytic tasks for another real-world application: roll-call vote analysis. The instantiated tasks are also valid and critical in the corresponding application domain (political science) as verified by a domain expert<sup>1</sup>. Eventually we aim at designing a system that can address a core set of analytic tasks that appears in a wide range of applications applying bipartite graph to model the key relations in the data.

We categorize the tasks in Table 3.1 into two groups (**T.a1-4** and **T.b**). **T.a1-4** focus on the topological structure of the bipartite relation. Applying the graph summarization algorithm (BM-MDL-LSH) and visualizing the aggregated result help analysts quickly gain an overview of the data to support **T.a1-3**. However, the summary graph  $\mathbb{S}$  alone is an inaccurate representation of the original data. To help analysts better assess the significance and reliability of the aggregated bipartite relations (**T.a4**), we also need to visually encode the amount of corrections  $\mathbb{C}$  needed to recover the original graph. **T.b** focuses on the attribute values of nodes and how they are associated with the bipartite connections. The domain-specific attributes (e.g. engine type in vehicle data, gender or occupation in movie preference data,

---

<sup>1</sup>We interviewed a research scientist working in the area of political science.

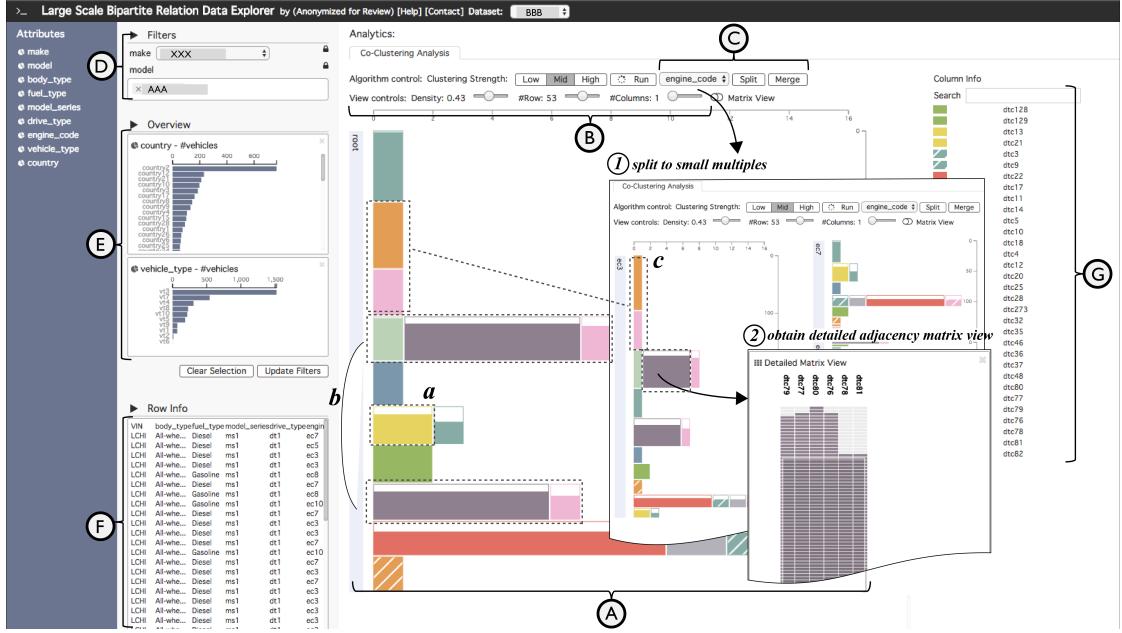


Figure 3.4: ViBR interface. First analyst selects the data using the filters (D) and computes a summarization (A) filtered by the density and the sizes of the clusters (B). From the adjacency list (A) she observes different fault patterns (**a** and **b**). Splitting the summary view into small multiples (1), several unique groups of faults (e.g. **c**) only occur in vehicles with a particular engine code. Next level of detail is shown by bringing up a matrix view for a particular block (2). Labels are always provided in the legend bar with text search (G). The node attribute value distributions and detail node information are displayed in (E) and table (F) respectively.

and party in roll-call votes data) add context to further understand and interpret the topological structure.

Besides supporting the analytic tasks described above, the system should also enable detail-on-demand data exploration (**R1**) since a static, high-level visual summary of data is seldom sufficient and analysts need to interactively drill down to the details for verification or identify more fine-grained structures in the data. Furthermore, the co-clustering algorithm creates node clusters and meta-edges with varying strengths. To help the analysts identify salient patterns in the data we should provide filtering mechanisms (**R2**) accordingly.

## 3.5 The ViBr System

We have designed ViBR to address the analytic tasks and design requirements discussed in Section 3.4, based on the summary graph generated by our technique described in Section 3.2. ViBR allows users to gain an overview of large scale bipartite relations with the summary graph (**T.a1-4**), adjust the granularity of the visualization to drill down into details (**R1**), filter the data to focus on the significant and salient cluster structures (**R2**) and apply the information encoded in domain specific node attributes for comparison, explanation and verification (**T.b**).

### 3.5.1 Visual adjacency list

To support scalable visual exploration, we design an adjacency list style visualization which is illustrated in Figure 3.5 (b). The visualization represents the clusters on one side of the graph (i.e. clusters in  $U$ ) as different rows and their outgoing connections to clusters on the other side (i.e. clusters in  $V$ ) as colored blocks stacked from left to right. Different colors represent different node clusters in  $V$ . The height and width of the blocks are proportional to the number of nodes contained in the two clusters. Some blocks are not entirely filled to indicate that there are missing edges in the original graph. The filled height of the blocks is determined by the density of the edges. The density is the number of edges between two clusters  $p \in P$  and  $q \in Q$  divided by the maximum number of possible edges.

$$\text{density}(p, q) = \frac{\|p \times q \cap R\|}{\|p\| \cdot \|q\|} \quad (3.2)$$

$\text{density}(p, q) = 1$  if the edges between  $p$  and  $q$  form a bi-clique. The blocks are sorted from left to right based on the density of the edges connecting the two clusters of nodes.

Compared with other visualization techniques such as node-link diagram with two parallel lists of nodes (Figure 3.5(a)), flow map (Figure 3.5(c)) and adjacency matrix (Figure 3.5(d)), the visual design results in an aligned and compact representation. It benefits the searching and understanding of bipartite relations and is adaptive to visualize graphs with different degrees of density. The key connections in the graph are prioritized and they can be easily identified by scanning vertically.

In flow map (Figure 3.5 (c)) the aggregated nodes in  $P$  and  $Q$  are arranged in two parallel vertical lists and the aggregated links are drawn as curved edges connecting the corresponding nodes. The widths of the edges are modulated based on the density value computed with Equation 3.2. One advantage of the design is that it allows the labels to be placed horizontally which can greatly improve the readability and interpretability of the visualization. This advantage, however, diminishes when the graph becomes much larger with thousands or even more nodes. Our major concern about the flow map is the visual clutter caused by the edges crossing each other, which makes it a challenging task to gain an overview of the bipartite connections and compare subsets of data, even for graphs at a moderate scale. Recent works identify bi-cliques in the graph and bundle the edges correspondingly to reduce visual clutter [58, 33]. However these methods may fall short for large scale bipartite graphs which could contain many small bi-cliques. Adjacency matrix (Figure 3.5 (d)) is another possible design. In adjacency matrices we use the filled proportion (similar to visual adjacency list) to encode the density of the aggregated connections. Adjacency matrices are suitable for visualizing dense interconnections [65, 66]. However it lacks space efficiency: for graphs with relatively sparse interconnections the empty blocks still have to occupy the screen space and the 'data-ink' ratio is not high. Besides that, it is still a challenging task to display readable row and column labels.

One important property of the adjacency list is that by filtering the blocks with low density and small node cluster size in  $U$  and/or  $V$  we can obtain an extremely compact overview of the bipartite relations. In Figure 3.6 we illustrate how the visual adjacency list is gradually simplified by increasing the threshold on density and the size of the node clusters. This property is especially useful for creating small multiples of adjacency lists to compare the bipartite relations for different subsets of data as illustrated in Figure 3.7. We use animated transition in the system to further facilitate understanding.

For the default display of the aggregated graph, we choose the adjacency list style design as it is a compact design and it distinguishes the node clusters with vertical positions and color. Aligning rows in this way allows the separation of node clusters at the first glance. In most of the cases it looks much clearer. Color has scalability issues in assigning distinct values, but choosing it over the long

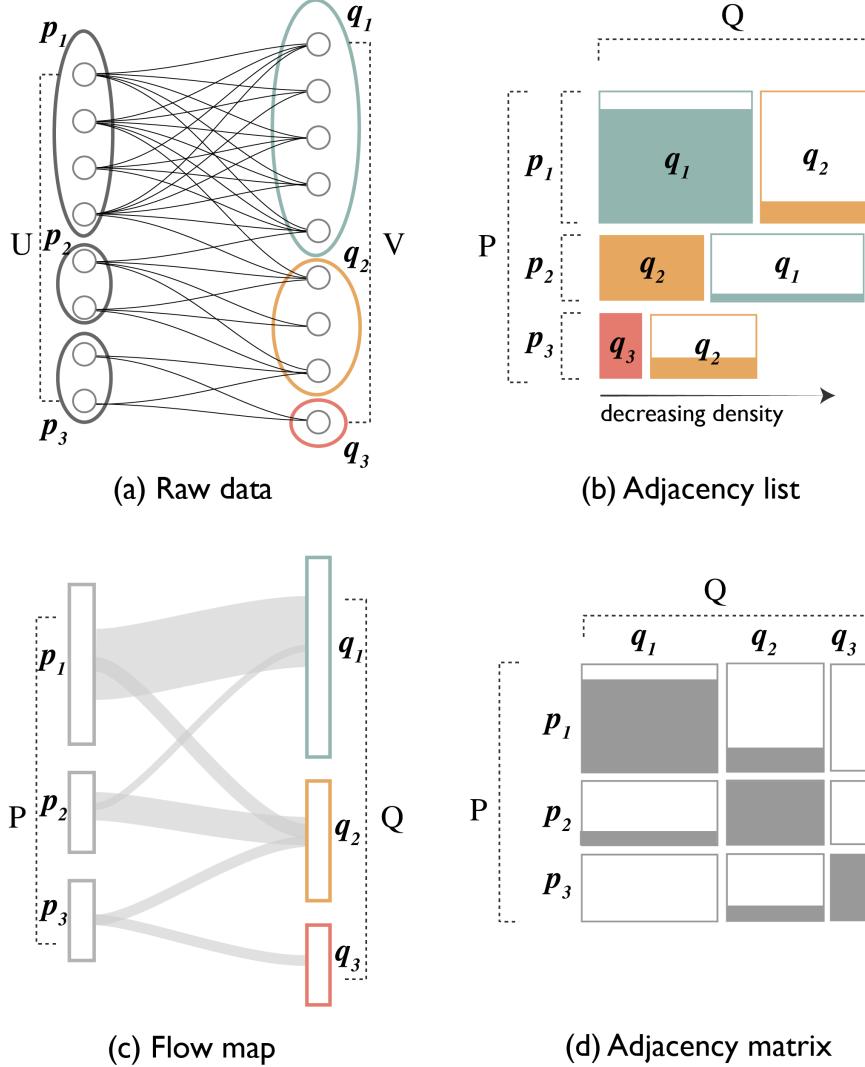


Figure 3.5: Design alternatives: (a) original data with two parallel list of nodes in  $U$  and  $V$ . The node clusters identified by the algorithm are highlighted; (b) the adjacency list style design. Color encode the different node clusters in  $V$ . Each rectangle block corresponds to an aggregated edge between two clusters in  $U$  and  $V$ . The height and the width of the block are proportional to the number of the nodes in the corresponding clusters. Filled proportion in each block encodes the density of the aggregated edge. Blocked sorted from left to right by decreasing density; (c) flow map with aggregated nodes and edges; (d) adjacency matrix with aggregated rows and columns. Filled proportion in each block encodes same value as in adjacency list.

horizontal axis as in adjacency matrices creates a greater utilization of space. We further extend the palette from ColorBrewer [67] with textures [68] such as . In the example usage scenarios with real-world data (Section 3.6.2) we realize that the distribution of bipartite connections are usually quite skewed and the color plus texture encoding can create enough variations to differentiate the major node clusters. We acknowledge that further user study is needed for a comprehensive understanding of the perceptual scalability and crafting a set of optimal designs for the texture patterns. We provide a legend (Figure 3.4 (G)) indicating the belonging nodes to each colored cluster. The system also supports text search of the node names in the legend.

One drawback of the visual adjacency list is that the two sets of nodes ( $U$  and  $V$ ) are not treated symmetrically in the visual representation. For  $U$ , the stacked heights make it easier to compare and sum the sizes of the node clusters. For  $V$ , the color encoding makes it easier to label the individual nodes in different clusters with additional legend. In practice, we also provide adjacency matrix as an alternative in the user interface and the analyst can switch between these two representations.

### 3.5.2 User Interaction

For effective exploratory data analysis, interaction is equally important as visual representation. ViBR supports the following user interactions:

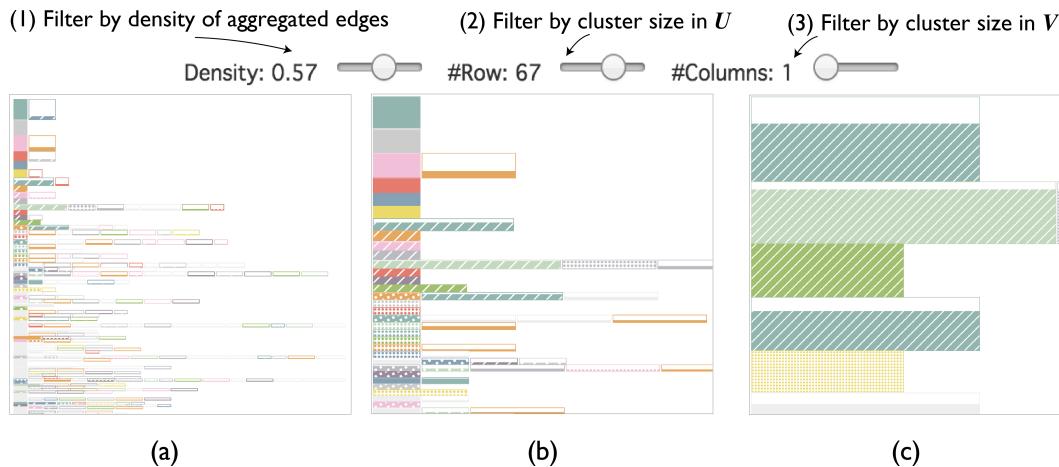


Figure 3.6: Filter visual adjacency list with different threshold settings on: (1)the density of the links, (2)the size of the clusters in  $U$  and (3)the size of the clusters in  $V$ . From (a) to (c) the visualization is gradually simplified and the significant patterns are highlighted.

**Filtering:** The system supports several filtering mechanisms such that analyst can focus on a particular subset or the most significant clusters in the data:

1. *Filter nodes by attribute values:* ViBR supports selecting a subset of nodes based on their attribute values (Figure 3.4 (D)) such that the analysts can focus on a relevant segment of the data.
  2. *Filter blocks in the adjacency list:* Although the number of node partitions can be controlled, noises in data can still produce small pieces that reduce the available spaces and affect the clarity of color encoding. Therefore, three filters are available (Figure 3.6, Figure 3.4 (B)) to remove noises based on different criteria: the density of the blocks and the corresponding size of the node clusters in  $U$  and  $V$ . Users can choose to keep only blocks that are significant in density or representative in size. The purpose is to emphasize important relations for comparison and facilitate understanding. Figure 3.6(a-c) illustrates the effect of different filtering threshold settings.

**Compare bipartite connections by node attributes:** For comparative analysis of bipartite relations the system supports creating small multiples by slicing on a selected node attribute (Figure 3.7). In the example illustrated in Figure 3.7 we show the overview of the bipartite relation (Figure 3.7 (a)) and the result of facetting

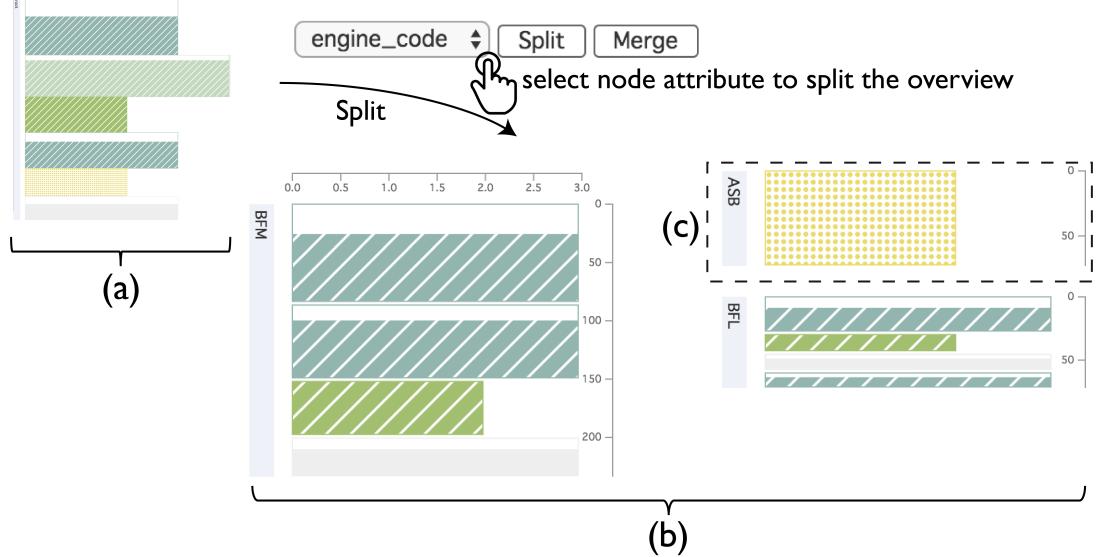


Figure 3.7: The system supports creating small multiples (b) from the original visual summary (a) by slicing on a selected node attribute. This supports comparative analysis across different subsets of data. The result shows a unique group of nodes with very distinctive bipartite connections (c).

it on a particular node attribute (Figure 3.7 (b)). The result shows a unique group of nodes with very distinctive bipartite connections (Figure 3.7 (c)) as the blocks have complete different color compared to the other two small multiples.

**Detail-on-demand:** Users can select a block to perform drill down inspection. The visual summary provides a high-level picture of the bipartite connections. However user may request more details for either verifying the results of the clustering algorithm or understanding the characteristics of a particular cluster. VIBR supports several different mechanisms for drill down inspection:

1. When a mouse hovers over a block, a tooltip is displayed to show the number of rows and columns and the density of the block.
2. When user clicks on a block, the detailed information of the corresponding nodes in  $U$  and  $V$  will be displayed and updated in two different tables (Figure 3.4 (F) and (G)).
3. When user double clicks on a block representing the high level summary, a new window will be created to reveal low level details of the bipartite relation within it using an adjacency matrix style visualization (Figure 3.4(2)). The

co-clustering algorithm is invoked again to reorder the rows and columns in the matrix to highlight the existence of any internal structures. User can click on the matrix view then brush through the entries, so that the two data tables will be further refined to highlight the information of the selected rows and columns. Under certain circumstances improper parameter settings in BM-MDL-LSH algorithm may result in an overly aggressive compression of the data. The visualization may henceforth display nodes with drastically different bipartite connections as one single cluster. Details provided by the matrix view can help users verify the resemblance of items within the same cluster to answer their hypothesis.

**Brushing on coordinated views:** The system visualizes node attribute value distributions with univariate charts: bar charts for categorical variables, histograms for (binned) numerical variables. These univariate charts are linked to the bipartite graph visualization: upon selection of a cluster the filtered data are highlighted in bar charts or histograms.

## 3.6 Evaluation

### 3.6.1 Evaluation of robustness and speed

In this section we present the experiments to evaluate our technique in terms of robustness and speed. The evaluation takes references from other similar bipartite relation summarization technique including the most recent SCMiner [63]. We also include a binary matrix reordering algorithm named Cross-association (CA) for reference [64]. We use the Python implementation for all the four algorithms. SCMiner, CA and BM-MDL-LSH use Numpy <sup>2</sup> which wraps C for numerical computation. We run the experiments on a Mac (OS Version High Sierra) with 2.3GHz Intel i7 CPU with 16GB RAM.

To evaluate the speed and the scalability the algorithm, we compare the running speed across SCMiner, CA, BM-MDL and BM-MDL-LSH. SCMiner and BM-MDL are similar [63, 62] since they all use 2-hop search to find candidate pairs of nodes to merge. In the experiment, we use the 1M MovieLens dataset [69]. The dataset

---

<sup>2</sup><http://www.numpy.org/>

contains 1M movie ratings from  $\sim$ 6000 users on  $\sim$ 4000 movies (the density is  $\sim$ 4%). We compute the running speed by gradually increasing the sampling rate of the movies and the users. The results in Figure 3.8 show that replacing the 2-hop search with LSH drastically reduces the running time of BM-MDL. Based on the calculation it improves the speed by more than 10 times for the full dataset. In the meanwhile, SCMiner cannot finish under reasonable time constraints — the running time exceeds 1 hour already for 50% data. Compared with the closest candidate CA, BM-MDL-LSH achieves around three times speed gain, which makes it a more practical choice for interactive bipartite relation exploration where the analyst can iteratively select different subsets in the data and run the algorithm to discover the underlying co-clusters.

Besides that we also further verify that BM-MDL-LSH does not introduce significant degradation in the results when compared to BM-MDL, in terms of description length reduction. We compare the percentage of reduction in description length in Figure 3.8 for the two algorithms running on the sampled 1M MovieLens data, with the same parameter settings for  $\alpha$ ,  $\beta_P$  and  $\beta_Q$ . The result shows that BM-MDL-LSH in fact improve the results, which can be explained by the higher threshold (close to 1) we set for LSH at the initial runs that prevents dissimilar pairs of nodes from being merged. In the meanwhile, with an aggressive setting of  $\beta$ s (to enforce less cluster numbers), BM-MDL groups even dissimilar nodes in the initial iterations, which results in far from ideal reduction in the description length.

### 3.6.2 Example Usage Scenarios

We introduce two example usage scenarios to demonstrate the effectiveness of our techniques. First we work with a researcher in political science to apply ViBR to analyze the roll-call voting records on 668 subjects (e.g., bills, amendments, resolutions, nominations) in the 115th United States House of Representatives in 2017. The data is collected from GovTrack.us<sup>3</sup>. The graph is constructed based on 435 individual legislators' votes on each subject. Overall we count 170,237 favorable votes. For each favorable vote we create a bipartite connection between the corresponding legislator and the subject.

---

<sup>3</sup><https://www.govtrack.us/>

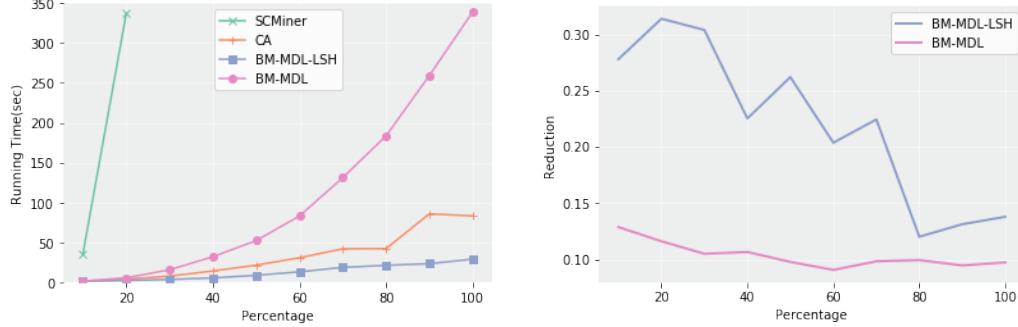


Figure 3.8: Left: comparison of running times of SCMiner, CA, BM-MDL and BM-MDL-LSH on the 1M MoiveLens dataset [69] with different sampling percentage. BM-MDL-LSH consistently outperforms others in terms of running speed. We discard all the results that require more than 1200s to complete in the plot. Right: comparison of the description length reduction of BM-MDL and BM-MDL-LSH. It shows that BM-MDL-LSH does not degrade the basic methods and even outperforms it in most of the cases.

The second work consists of a group of data scientists from the automotive industry focusing on a dataset about around 7 million vehicles' after-market repair information. The dataset records the diagnostic trouble codes (DTCs) for each vehicle. We create the bipartite relations based on the occurrences of the DTCs in the individual vehicle. Each computation and visualization is bounded within an individual car model. Therefore, the analysis of each car model consists of vehicles amount ranged from 2,000 to 40,000. The exact vehicle identification number (VIN) and the DTCs are anonymized for privacy concerns.

### 3.6.3 Congress voting analysis

A background of the 115th Congress is as follows: the Republicans occupied more than half of the seats in both the Senate and the House of Representatives, plus they had won the presidential election in 2016. Therefore, it is clear that they act as the ruling party while the Democrats act as the opposition party. To understand the dynamics in the congress we conducted the case study with a researcher in political science and she helped document the findings.

**Overview the bipartite structure in votes.** The expert first runs the co-clustering algorithm and creates a small multiple display by faceting on the party affiliation (Figure 3.9(b)). The clusters of subjects being voted (e.g. bills,

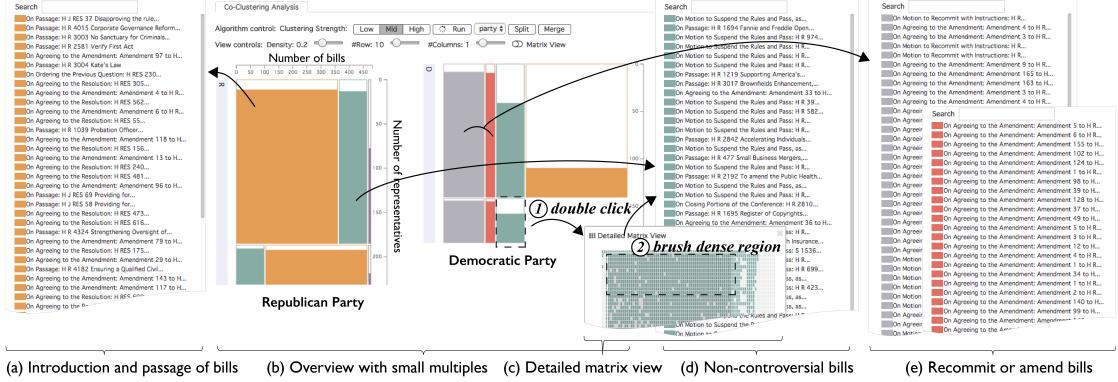


Figure 3.9: Using ViBR to analyze the bipartite structure of roll-call votes in the US House of Representatives (115th Congress). (a) The Republicans mainly vote for bills that favor the legislation (orange block); (b) Overview of the bipartite relations. The two small multiples summarize the voting patterns of the Republicans and the Democrats respectively; (c) Detailed matrix view of one block; (d) Both parties vote for bills that are non-controversial. (e) The Democrats mostly vote for amendments (gray and red block). Details can be found in Section 3.6.3.

amendments) are assigned with different colors. The relative areas of the colored blocks in each subgraph shows that the two parties indeed have very different sets of favored bills (or amendments and etc.) and the representatives usually vote according to the consensus of their parties. Besides that, based on the total filled area of the blocks, the Republicans vote “yes” more than the Democrats, showing the proposed bills and rules are favorable to the ruling party.

**Understanding roles of different parties in the legislation process.** In both subgraphs we observe a teal colored block, indicating that there are quite a few number of proposals supported by both the parties. The expert clicked one teal colored block in the Democrat’s partition (Figure 3.9 ①) and opened the detailed matrix view (Figure 3.9(c)) for further analysis. The co-clustering algorithm runs again to reorder the rows and columns in the matrix view to reveal the internal structures in the block. The expert brushed on the matrix view to review more details (Figure 3.9 ②) and observed that the Democrats mostly voted “yes” on the subjects about *“On Motion to Suspend the Rules and Pass”* (Figure 3.9(d)), which refer to the act of quickly passing the non-controversial bills for more efficiency in the legislation process.

The orange colored subjects are rarely voted in favor by the Democrats while

being supported by the majority of the Republicans as shown by its high density in the partition (Figure 3.9(b)). These subjects are mainly related to these categories (Figure 3.9(a)): 1. *Agreeing to the resolution for consideration*, which means agreeing to initiate the introduction of new bills; 2. *Ordering the previous question*, which means the motion to end debate on a pending proposal and bring it to an immediate vote; and 3. *Passage*, which means passing the proposal. It shows that as the Republican party is now the majority party, more bills and legislations of their interests and policy preferences will be proposed by them. Therefore they will mostly agree on these actions which are conducive to the establishment of new laws to fulfill their party interests. On the other hand, Democrats are more likely to oppose such initiatives and prevent the bills from passing, since those would rarely be compatible with their interests.

On the other hand, the subjects with votes mostly from the Democrats (gray and red blocks) fall into these categories (Figure 3.9(e)): 1. *Motion to recommit with instructions*, which means to send back the proposal for amendment; and 2. *Agreeing to the amendment*, which refers to the amendment of several details in the bill. Our expert then comes up with an interesting question: “*Why do Democrats like to vote for amendments?*” She searches for the number of amendments voted from the Democrats, and discovers that they support amendments more than the Republicans. She then concludes that the Democrat’s support on amendments is a clear reflection of the bargaining process in the legislature, as well as the general balance of power between the two parties. Amendment is a subtle issue, which may bring benefits to both the Democrats and the Republicans. As the minority party in the House, the Democrats may find the bill drafted by the Republicans far from their ideal policy position; however, this does not mean the Democrats have no bargaining power in the Congress. Instead, through negotiations and even fierce debates in the House, the Democrats may propose changes to specific articles in the drafted bill through amendments, such that their interests and preferences are represented. This helps explain why while the Democrats are reluctant to pass the bill, they are more willing to seize the opportunity for amendments.

### 3.6.4 Vehicle fault data analysis

In the second example usage scenario, we turn our attention to a completely different application domain and analyze the occurrences of faults in vehicles. The vehicle faults are recorded as DTC codes, which are warning messages generated by different electric control units (ECUs) in vehicles. They indicate abnormality in various sensor measurements or other types of malfunction in the hardware and software system. A large portion of the DTC codes are standardized across different makes and models while some of them remain unique to individual brands. Given that the vehicles are increasingly complex, the repair shops rely heavily on the historical record of DTCs to track the health statuses of the vehicles. When a vehicle visits a repair shop the mechanics will collect the on-board DTC logs and use them to perform more precise diagnosis and identify the suitable repair procedures.

Large scale analysis of DTC occurrences in vehicles help understand the demographics of vehicle faults, which would have great impact on the automotive industry. It enables large scale troubleshooting such that early warnings can be generated on the emerging fault patterns for the auto manufacturers before the problem strikes a larger vehicle population. Besides that, it also enables experience based repair by analyzing the most effective repair procedures for a particular set of co-occurring DTCs. Regardless of the size of vehicle data being analyzed, our analyst follows similar pipeline (Figure 3.4) supported by ViBR to acquire insights that help understand the demographics of vehicle faults:

**Search for co-clusters with high confidence.** Our analyst's first action would be filtering out co-clusters by densities and sizes. In general the bipartite relations between the vehicles and the DTCs are quite sparse. Among the  $\sim 3000$  different types of DTCs most of them seldom occur. To seek for meaningful and significant clusters which can reflect the systematic fault patterns within a vehicle population, it is necessary to focus on clusters with a larger number of vehicles and relatively higher density. Therefore, the analyst adjusts the filters (Figure 3.4 (B)) available in the system such that the dense and significant blocks become more visible. (Figure 3.4 (A)) shows the filtered result. There are around 10 large groups of vehicles with different combinations of co-occurring DTCs. It reveals that vehicle repair and maintenance requires highly customized services and inspecting the data

visually is an effective way to uncover such multi-class situation.

**Compare the co-occurring faults for different clusters of vehicles.** The vehicle clusters differ by either having a unique set of DTCs, or they could have a common set of DTCs but differ by some additional ones. For example, the temperature sensor problems (yellow block, Figure 3.4 *a*) only appear in one of the vehicle clusters, independent of all the other DTCs. While the other two groups of vehicles (Figure 3.4 *b*) share a similar set of DTCs (purple+ pink blocks) but differ by additional seat belt problems (light green block). Based on such observations the analysts can isolate different sets of vehicles for focused analysis.

#### Analyze the correlation of fault patterns with vehicle properties.

Given the overview the analyst wants to understand whether the fault patterns are associated with a particular subpopulation of vehicles that share the same properties such as *engine type* or *country*. This can be done by partitioning the vehicles on a selected property and separating the overview into small multiples for comparison. For example, in Figure 3.4 (1), each small multiple displays a summarized bipartite relation for a subset of vehicles with the same engine code. The analyst finds out that the sensor faults (orange) and pressure actuator faults (pink) only occur in vehicles with a particular type of engine (Figure 3.4 *c*).

**Inspect details in the matrix view.** To obtain detailed information the analysts open the matrix view. In Figure 3.4 (2) we give an example where a dense purple colored block is double clicked and the matrix view is displayed for the block showing the occurrences of DTCs in individual vehicles. The analyst further brushed on the corresponding entries to obtain fine-grained information of the vehicles and the DTCs such as the *VIN number* and the *body type* of the vehicles (Figure 3.4 (F)) and the description of the DTCs (Figure 3.4 (G)).

### 3.6.5 Expert Interview

After recording the exploratory analysis process and findings from the experts in the automotive industry and political science respectively, we gathered feedback on the effectiveness and usability of the visualizations and their thoughts on potential extensions/other applications with a few guiding questions, following the suggestions proposed in [70].

**Visual design.** Overall, the domain experts appreciate the clarity and novelty

of the summarization as well as the visualization output. Our political science expert looks forward to not only examining votes for legislative bills, but also the exact content of it. Given the summarization now available, she can analyze the importance of certain policy issues through text analysis of the terms and phrases, which contributes to agenda setting. She believes text visualization techniques like word clouds can seamlessly be incorporated into the current system to help produce more insights regarding political issues.

**Outlook to further impact.** The domain experts from the automotive industry wants to add more critical labeling information in the data such as parts that are replaced in a vehicle. These can be associated with the DTC clusters identified by the algorithm, allowing repair shops to perform faster and more accurate data-driven diagnostics and even alert drivers for potential failure. Furthermore, our political science expert would like to apply the current techniques to help her research in the area of *international political economy* to understand the contemporary globalization pattern. She would like to summarize the trade between countries distributed in different geopolitical regions and compare the patterns over a period of time. She believes that it can revolutionize the traditional way of showing data within her research community.

### 3.7 Conclusion

In this chapter, we visit the problem of summarizing large scale bipartite relations and introduce a novel interactive visual analytics approach to address the challenges. First, we propose an information-theoretic co-clustering algorithm based on the MDL principle. The algorithm runs efficiently on large scale bipartite graphs which makes it suitable to support interactive visual exploration. Then, we present a comprehensive visual analytics system with a novel visual adjacency list style design and multiple levels-of-detail to facilitate interactive exploration of large scale bipartite graphs with multivariate node attributes. We present usage scenarios with two real-world dataset and feedbacks from the domain experts to demonstrate its effectiveness.

# Chapter 4

## Time Series Summary

Time-series data analysis is prevalent in many applications. Data scientists and domain experts engage in a wide range of time series exploration tasks such as discovering patterns in urban activities (e.g., noises, traffic, and weather [71, 72, 73]), identifying trends in highly volatile financial markets [74], and grouping human mobility patterns from wireless telecommunication traffic data series [75, 76, 77, 78]. Analysts often rely on visual exploration as the first step to produce hypotheses and acquire insights from many time series.

Many visualization primitives have been proposed [79, 80, 81, 82] to support temporal data analysis. However, time-series visualization is challenging for complex, large data [83]. The difficulties of visualizing time series without properly handling the data are illustrated in Figure 4.1. First, different composite visualizations face a tradeoff between scalability and visual clarity. While plotting time series in each small multiple (i.e. (a) juxtaposition) is not feasible for large scale data, plotting every time series in one chart (i.e. (b) superposition) deteriorates the visual clarity quickly. The main reason for cluttering comes from the overlapping among various shapes, which furthermore inhibits the usage of clutter reduction techniques such as the aggregation by bands or density (Figure 4.1 (c) and (d)). Therefore, the intuitive solution of visualizing time series without overplotting nor under-utilizing small multiples is to cluster the whole time series with similar trends then visualize each trend with one chart per cluster (Figure 4.1(i)). Yet, the problem is not fully solved – time series in real life exhibit trends within *subsequences* in arbitrary durations and time intervals. Only considering the whole time series for

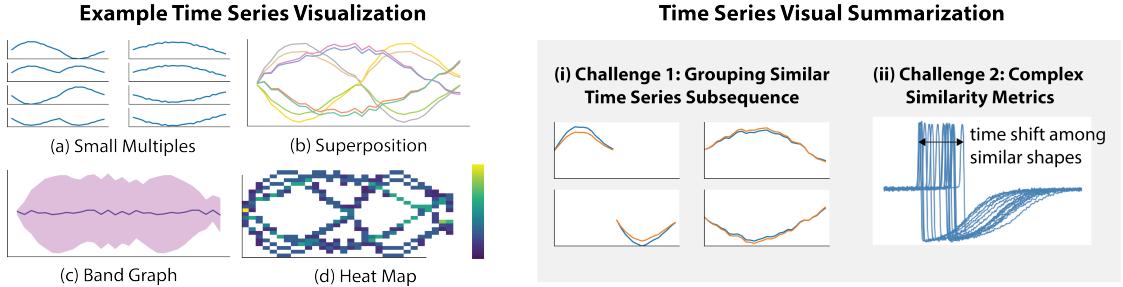


Figure 4.1: Illustration of the challenges of visualizing multiple time series. Without an effective grouping of time series, the basic visualizations (a) – (d) suffer from either visual clarity or scalability. (1) Clustering time series reduces the visual clutter, and (2) grouping the similar subsequences further optimizes the number of small multiples needed, resulting in a time series visual summary proposed in our work.

clustering reduces the chances for each series to be grouped, consequently hindering the reduction of small multiples and the emphasis of useful patterns. Thus, we argue that the most effective visualization technique for large scale time series is to extract their similar subsequences (Figure 4.1(i)). This opens a brand new research challenge: how to present the time series subsequences that balance visual clarity and the number of small multiples with computations that apply visualization oriented metrics and return results in interactive time. Without loss of generality, we define the time series subsequence clusters as a set of disjoint partitions of real-value data series, which differs our problem setting to overlapping subsequence clusters [84, 85, 86, 87, 88] (i.e. same data points visualized in multiple charts) and clustering discrete event sequences [89, 90].

Besides the challenges of clustering time series subsequence with arbitrary durations and time intervals, it is also computationally expensive to measure the similarity between time series in terms of their shapes. As shown in Figure 4.1, the second challenge for summarizing time series is the possibility of shifting among similar trends of time series (Figure 4.1(ii)). It means that alignment is needed among time series before measuring their similarity with metrics like Euclidean distances, which, we will show in Section 4.3, imposes a dramatic computation complexity for interactive data analysis.

As a result, we propose a novel visual summarization technique for multiple time series that uses a sequential pattern mining approach and reveals the time

series patterns that can appear in any time interval or set. We first propose a new symbolic representation of time series which encodes the visual structure of each time series subsequence in a discrete format, and how to compute them efficiently. We furthermore introduce efficient algorithms that use the symbolic sequential patterns as input and output optimal groupings of similar subsequences. The result is a visual representation that balances the tradeoff between optimizing the number of charts allocated for the time series and clutter.

Such a technique enables a scalable visual encoding to display time-series patterns with low visual complexity. To leverage the technique for scalable time-series visual analytics, we propose a carefully considered interactive system to facilitate visual pattern discovery and understanding. We discuss the rendering strategies to facilitate visual exploration and interaction of massive time-series data. In short, our contributions are as follows:

1. We propose a novel time-series summarization technique that consists of a new symbolic representation of time series and an optimization strategy to compactly group the similar time series subsequences using the representation.
2. We propose efficient algorithms to construct the representation and extract the time series patterns from the sequential patterns within seconds instead of hours in straightforward DTW and pattern mining computations.
3. We present an accessible and scalable visualization interface to explore large volumes of time series data with our time series summary.
4. We present two example usage scenarios with real-world datasets to demonstrate the usefulness of our technique.

## 4.1 Related Work

### 4.1.1 Time Series Visualization

Aigner *et al.* [91] summarize time series visualization techniques based on the data type (i.e. visualizing single or multiple time series) and dimension (i.e. whether it is linear, cyclic, or branching and whether it is a point or an interval). Bach *et al.* [92] provide a perspective of time series visualization as operations on a space-time cube. The operations include extraction, flattening, filling, and geometric

and content transformation. Historically, time series visualization started with the introduction of line charts in the 18th century [79]. It can be encoded in small multiples and sparklines [80] that used position as the only channel to encode temporal data. Later on, visualization techniques also included channels such as area or color to provide better scalability and density in the presentation. Examples include horizon chart [82], which splits and superimposes a line chart vertically with a few bands of ranges distinguished by color and color fields [93, 94, 95, 96], which use hues to encode the values. Recently, these techniques have been shown to result in different perceptions of similarities [97]. In particular, scalability is a major challenge when visualizing multiple time series. Charts could lose discrimination and become cluttered even with a fairly small number of time series (i.e eight in Javad *et al.* [83]). Moritz *et al.* thus proposed a heat map approach to treat the values in time series as independent pixels and plot the heat map of lines in one chart [81]. This method reveals the density of lines but neglects the common patterns from the contiguous movement in many time series.

Since these visualization techniques contain tradeoffs on different data characteristics, interaction plays an important role in facilitating multivariate time series analysis. TimeSearcher [98, 99, 100] uses brushing to select the time interval and a value range of interest to filter or query similar time series. BinX [101] provides options to aggregate time series in bins and visualizes the aggregate functions with additional box plots and ranges. LiveRAC [80] integrates time series data into a spreadsheet layout so that users can select and reorder the rows and columns to perform inspection and manipulation in batches. ChronoLenses [102] and Care-Cruiser [103] both use filtering and highlighting to provide focus and context information of such data. Continuum [104], EventRiver [105], FacetZoom [106], Midgaard [107], TimeNote [108], and KAVAGait [109] apply semantic zooming on a time axis to visualize the data with different granularity and visual metaphors.

These techniques typically display time series in their original form. To handle large-scale data, combining data abstraction techniques and visualization is usually necessary.

### 4.1.2 Time Series Mining and Visualization

Since time series visualization is vulnerable to occlusion, especially when the lines are interleaving, many visualization techniques apply data mining approaches for time series data analysis. See [110] for an in-depth survey of data abstraction on visualizing time series.

When abstracting data to visualize large-scale time series, approximation techniques are used to reduce complexity and dimension [111]. Symbolic Aggregate Approximation (SAX) [112] is widely used to discretize a long time series. It transforms long series into a short piecewise aggregate approximation (PAA) and then converts the data into a string of symbols. The approximation works well for visualizing periodic time series. VizTree [113] and SAX Navigator [114] uses a hierarchical tree to visualize the results of SAX; and Hao *et al.* visualize the motifs of these symbols with colored rectangles [115]. Muthumanickam *et al.* apply a piecewise linear approximation that minimizes the loss of gradient to visualize long time series [116]. GrammerViz [117] applies SAX to visualize variable length motifs. SOMFlow[118] applies similar hierarchical clustering approaches for interactive time series clustering

Projection and clustering are used to gather similar time series together to reveal underlying patterns. Steiger *et al.* visualize the pairwise distance matrix of multiple time series into 2D projections and a Voronoi diagram [119]. Ward *et al.* use N-grams to segment time series and project the data with PCA [120]. Van Goethem *et al.* apply a method that detects trends of time series at the starting time point and visualize the trends and subtrends with a river metaphor [121]. StreamStory [122] transforms time series into a set of states to visualize the relationships among the time series in a directed graph.

Instead of understanding time series as a whole, we can explore the statistical properties such as variances and correlations instead. Pinus [123] is a triangle matrix metaphor that visualizes the variances of time segments in all combinations of time intervals. Kothur *et al.* [124] visualize the time series as color fields that encode the correlations among other time series. TimeSeer [125] visualizes scagnostics with scatter plots of data attributes at each time index to identify anomalies.

We propose a novel time series summarization technique that transforms time series into compact symbolic sequences, considers the visual features of time series, and produces results aimed at reducing visual complexity. We also propose efficient

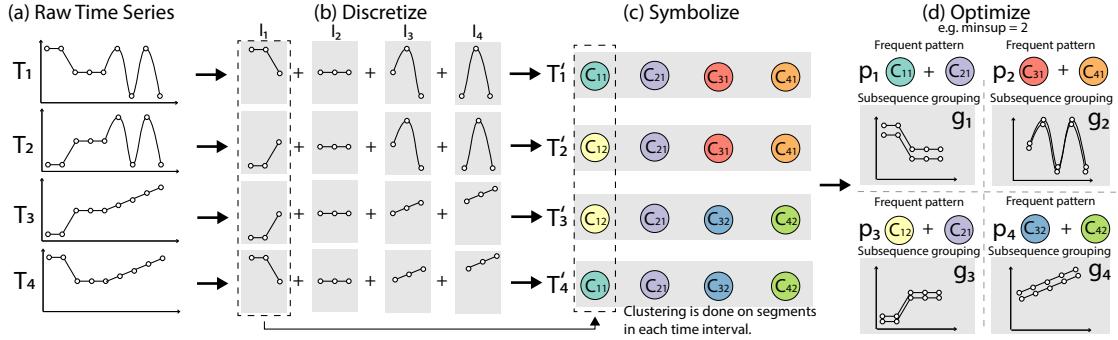
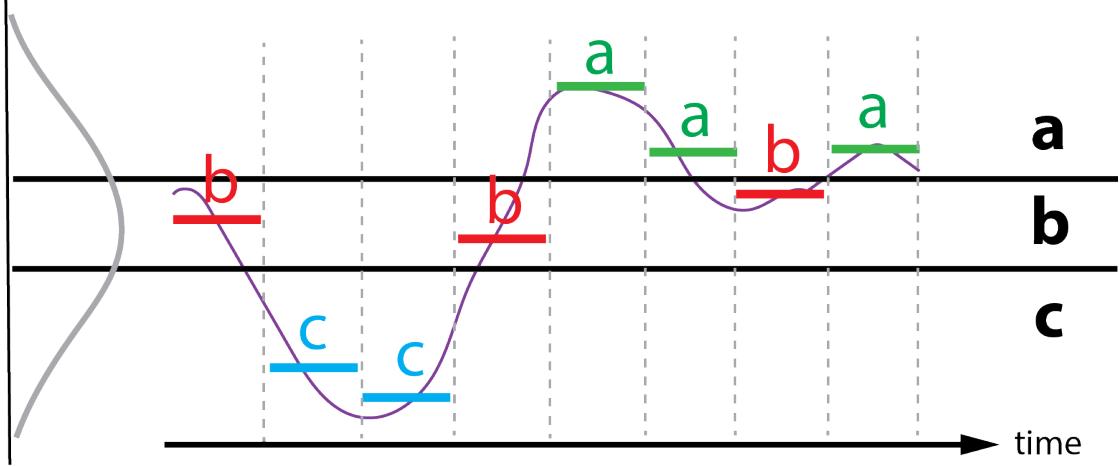


Figure 4.2: Pipeline of extracting the time series patterns ( $g_1, g_2, g_3, g_4$ ) from the time series in (a). (b) The series are discretized into sets of time segments with equal length. (c) The segments in each time interval are labeled with the cluster to which they belong. (d) The groupings of time segments are optimized based on the frequent patterns from the sequences in (c). In the above example, if  $\text{minsup} = 2$  (i.e. each output group has to contain at least 2 time series), the optimal groups will extract subsequences from  $(C_{11} + C_{21})$ ,  $(C_{12} + C_{21})$ ,  $(C_{31} + C_{41})$  and  $(C_{32} + C_{42})$ .

algorithms that achieve 62 times speed up in constructing sequences compared to a straightforward DTW clustering and attain interactive speed to extract the time series patterns, making it suitable for interactive exploration of data.

## 4.2 Extracting Similar Time Series Subsequences

In this section, we first present the theoretical foundations to extract similar time series subsequences as a visual summary. To cluster the time series subsequences for clear and compact visualization, we focus on (1) transforming the time series into **discrete sequences** and then (2) mining the **frequent patterns** from the sequences. Breaking down time series into discrete sequences is the popular approach to query similar subsequences [126, 112]. It treats the time series mining problem as a sequence pattern mining problem. The challenges for visualization are constructing the discrete subsequence that preserves visual similarity and extract sets of frequent patterns with visual compactness as the objective.



**Symbolic Representation = bccbaaba**

Figure 4.3: Illustration of Symbolic Aggregated Approximation (SAX).

#### 4.2.1 Background on Time Series Subsequence Mining

Before proposing our framework to achieve time series, we first provide some background information on how time series subsequence clustering is achieved in general, inspiring much intuition on how we approach the problem. To retrieve common time series subsequences, the key is to transform the real-valued time series into *discrete sequences*. Figure 4.3 illustrates a popular technique called Symbolic Aggregate approXimation (SAX) to achieve the discretization in many data mining tasks [112]. First, the time series is split into  $w$  segments with equal durations. Then, for each segment, an alphabet representing a range with equal probable distribution is assigned based on the average of the values within the segment (i.e., piecewise aggregate approximation (PAA)). Since the time series becomes a discrete sequence (or a string for an easier understanding), pattern mining techniques can be applied to obtain the repetitive substrings among all sequences in the dataset. These substrings thus imply the subsequence clusters in the time series.

SAX inspires our method in terms of discretizing the real-valued sequences into a symbolic representation. Our main difference is how we can compute the symbols that represent visual shapes instead of aggregated values. In Figure 4.3, we notice that time segments with different shapes can result in the same alphabets. Thus we need another way to define symbols for visualization.

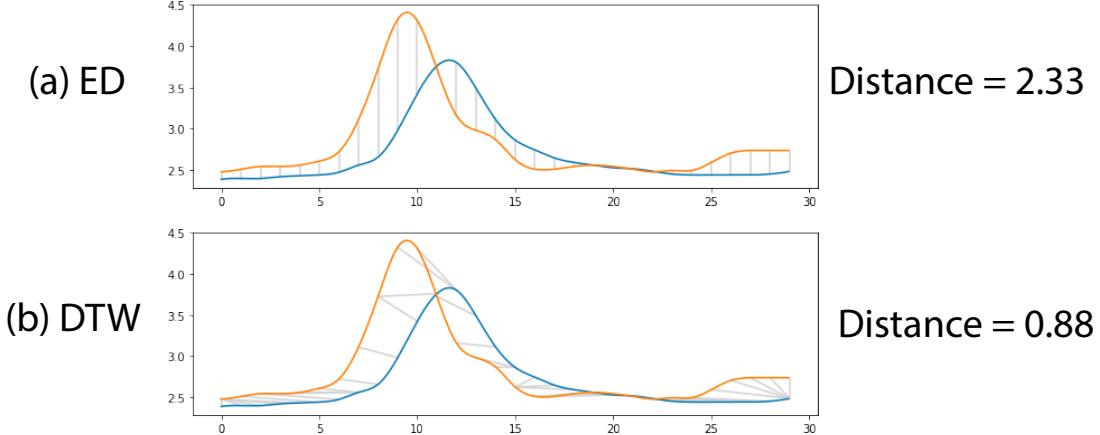


Figure 4.4: (a) Euclidean Distance (ED) sums up the  $L_2$  distance between the points of two time series at the same temporal positions. (b) Dynamic Time Warping (DTW) matches the points (i.e. the grey lines) first even though they are not aligned on the time axis.

#### 4.2.2 Time Series Visual Symbolic Representation

Let  $\mathcal{T}$  be a set of  $m$  real valued time series, where each series  $T \in \mathcal{T}$

$$T = t_1, t_2, \dots, t_n \quad \forall t_i \in \mathbb{R}, i \in I = \{1, \dots, n\} \quad (4.1)$$

has length  $n$ .  $I$  represents the time intervals where each  $T$  is defined. The proposed approach begins by decomposing the time series  $T$  into a sequence of contiguous time series segments (Figure 4.2(b))

$$T = s_1 \oplus s_2 \oplus \dots \oplus s_{n'}, \quad (4.2)$$

where  $\oplus$  is the concatenation operator and each  $s_i$  has size  $l$ . The value of  $l$  is defined by users based on the shortest possible intervals of final patterns they want to extract.

Given a subinterval of size  $l$ , we can split the time interval  $I$  where the times series are defined into  $n' = \frac{n}{l}$  subintervals  $I_i$  such that  $I = I_1 \cup I_2 \cup \dots \cup I_{n'}.$  Our goal is to identify groups of similar time series segments within each subinterval  $I_i$ , so that each group of similar segments is represented by a symbol. This procedure makes it possible to represent a time series as a sequence of symbols, one per

segment  $I_i$ , leading to a more compact and easily interpretable representation of the time series (Figure 4.2(c)). Specifically, let  $\mathcal{T}_i$  be the set of time series segments from  $\mathcal{T}$  in subinterval  $I_i$ . We can cluster the segments in  $\mathcal{T}_i$  according to their similarity, assigning a symbol to each cluster. If we denote the clusters in  $I_i$  by  $c_{i1}, c_{i2}, \dots, c_{ik_i}$ , where  $k_i$  is the number of clusters in  $I_i$ , we can represent time series  $T$  as  $T' = c_{1j_1}, c_{2j_2}, \dots, c_{n'j_{n'}}$  (Figure 4.2(c)), where  $c_{ij}$  is the symbol assigned to the  $j$ th cluster of interval  $I_i$  (here, we represent the cluster and the symbol assigned to it with the same notation). The symbol  $c_{ij}$  is chosen such that the segment  $s_i \in T$  defined on  $I_i$  is contained in  $c_{ij}$  (Figure 4.2(c)).

**Dynamic Time Warping as a visual similarity metric.** Choosing the distance metric that reflects visual similarity is essential to a good clustering result. Despite the abundance of similarity measures available in the literature [127, 128], several studies (including crowdsourcing, user study, and data mining benchmarks) indicate that Dynamic Time Warping (DTW) [129] performs better on average than other measures both in terms of perception [130, 97, 131] and classification accuracy [127]. Unlike Euclidean distance (ED) that only considers the corresponding points in two time-series, DTW allows for small shifts on the time axis to minimize the overall sum of distances. For example, in Figure 4.4, although the two time-series have similar patterns (i.e., one peak), ED computes a much larger distance than DTW because it does not perform a shape matching alignment of points before computing the distances. Such an alignment invokes the Gestalt rule of similarity: humans perceive lines with similar slopes as the same group [68]. For these reasons, in our work, we use DTW in [132] to calculate the pairwise similarity of each time segment.

**Clustering technique for symbolic representation.** We use agglomerative hierarchical clustering with gap statistics [133] to automatically choose the number of clusters. To facilitate effective visual analytics, we use a parameter  $\alpha$  as clustering strength with Gap statistics to adjust the number of clusters  $\hat{k}$ :

$$\hat{k} = \text{smallest } k \text{ such that } \text{Gap}(k) \geq \frac{1}{\alpha} \text{Gap}(k+1)$$

Pattern	Support	Pattern	Support	Pattern	Support
$C_{11}$	2	$C_{32}$	2	$C_{12} + C_{21}$	2
$C_{21}$	4	$C_{22}$	2	$C_{21} + C_{32}$	2
$C_{31}$	2	$C_{11} + C_{21}$	2	$C_{32} + C_{42}$	2
$C_{41}$	2	$C_{21} + C_{31}$	2		
$C_{12}$	2	$C_{31} + C_{41}$	2		

Table 4.1: Patterns in Figure 4.2(c) having  $\text{minsup} \geq 2$ .

### 4.2.3 Optimizing Groupings from Sequential Patterns

Once the symbolic sequences are established, we can apply sequential pattern mining techniques to retrieve common time series subsequences from the dataset. Before explaining the optimization process and its motivation, we first introduce some nomenclature, mainly the concept of patterns and minimum support ( $\text{minsup}$ ) and their implications on the optimization outcome.

A *pattern* (i.e. sequential pattern) is a sequence of contiguous symbols and also a set of intersected time series subsequences represented by the symbolic sequences in our case. The length of a pattern is the number of such symbols in the sequence. For instance, in Figure 4.2,  $C_{31} + C_{41}$  corresponds to a pattern of length 2. The *support* of a pattern  $p$ , denoted  $\text{sup}(p)$ , is defined as the number of time segments that are members of  $p$ . For example, support of 3 means that there are three segments in the pattern. The minimum number of time series segments that a pattern must contain to be *frequent* is called  $\text{minsup}$ . Intuitively,  $\text{minsup}$  plays a user-defined role in the meaning of *trends* in the time series data. It defines how many similar series are needed to form a trend. Also, the lower the  $\text{minsup}$  value, the greater the chance of obtaining longer frequent patterns, whereas larger  $\text{minsup}$  values produce shorter patterns with more similar segments.

If one directly visualizes the time series segments inside the frequent patterns, it can be certain that the time segments in each pattern are similar to each other. However, it presents two main problems. Using the example in Figure 4.2(c), the frequent patterns with  $\text{minsup} = 2$  are illustrated in 4.1. First, the number of frequent patterns can grow drastically when the  $\text{minsup}$  is set as a small number because of the combinatorial explosion [134]. Second, these patterns might overlap with each other, resulting in many repeated time segments displayed. Thus, our goal is to find the minimum number of subsequence groups where each group

contains time segments from the same pattern and the whole set of groups does not contain repeated subsequences (Figure 4.2(d)).

Formally, given  $minsup$ , the optimization process is accomplished as follows. For each set of subsequence groups  $\{g_{i_1}, g_{i_2}, \dots, g_{i_s}\} \in G_i$ , where  $G_i \in \mathcal{G}$  is one of the possible sets,

$$\begin{aligned} g_{i_j} &\subset p \text{ where } p \text{ is frequent} \\ g_{i_j} \cap g_{i_r} &= \emptyset \text{ and } sup(g_{i_j}) \geq minsup \end{aligned} \tag{4.3}$$

Among all possible sets of subsequence groups satisfying Equation 4.3, we look for the one with the minimum number of subsequence groups:

$$\hat{G} = \arg \min_{G \in \mathcal{G}} |G| \tag{4.4}$$

where  $|\cdot|$  accounts for the number of elements in the set.

Intuitively, we use frequent patterns to identify significant and similar time series subsequences in arbitrary lengths and discard those without enough supports. We further minimize the number of groups needed to partition the similar contiguous subsequences. Thus, we provide a minimal overview of unique temporal behaviors along the time period, dramatically reducing the visual complexity and allowing users to quickly grasp the dominant trends in a high volume of time series.

#### 4.2.3.1 Parameter Tuning

Like other clustering algorithms, the whole subsequence clustering framework contains three main parameters: window size  $I$ , clustering strength  $\alpha$ , and minimum support  $minsup$ . Typically, tuning the clustering results involves domain expertise, such as the suitable durations to be defined as the sizes of time windows (e.g., the number of days for a stock market period and the number of series eligible a trend). We can input these parameters for parameter tuning and have the cluster quality metric (like the Silhouette index) as the objective score for an exhaustive grid search. Due to the focus of interactive visualization computation, we do not dive deep into this query optimization problem. In Section 4.5.2 we will show that we can arrive at good results with minimal changes to these parameters.

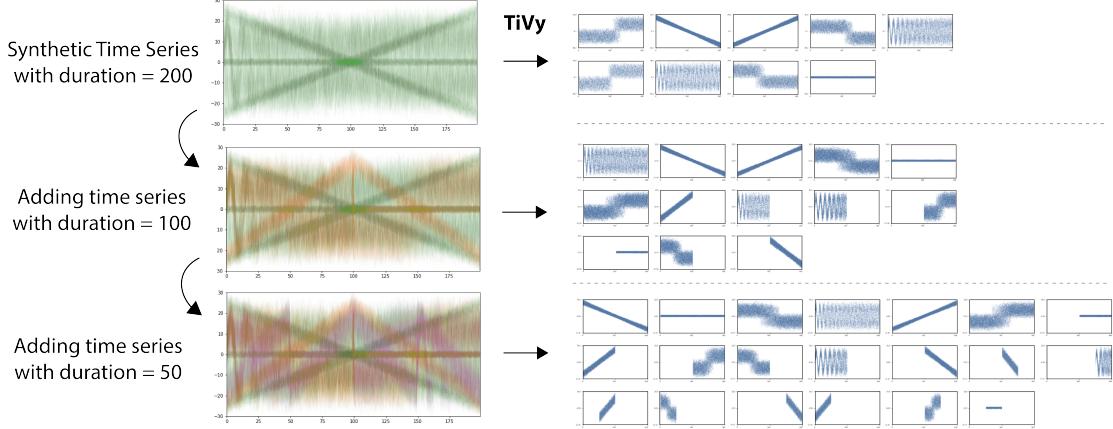


Figure 4.5: Illustration of TiVY’s visual summarization using synthetic dataset composing of six main trends with different durations. By adding more complexities of patterns hidden in different parts of the dataset (more details in Figure B.1), our approach successfully extracts them into accurate subsequence clusters.

#### 4.2.4 Illustration of Visual Summarization Result

To illustrate how our approach achieve the summarization of subsequence clusters in the dataset, we design a synthetic dataset with different shapes in multiple time intervals. The whole dataset contains six classes of shapes: cyclic ; normal ; increasing ; decrease ; upward shift ; and downward shift . In each class, the time series have some deviations in temporal alignment and amplitudes but plotting all of them within one plot would not cause too much perception differences. We also generate these data with different durations and combine them together to form the datasets shown in Figure 4.5. In Figure 4.5, we can see that by successfully separating the time series based on the similarity of subsequences, we can visualize the cluttered time series with a moderate number of charts while enabling each chart to be visually clear. In the next section, we will introduce an efficient algorithm to produce the results in Figure 4.5.

### 4.3 The TiVY Algorithm

We now present the TiVY (Time Series Visual Summary) algorithm to construct the symbolic representations and obtain an optimal grouping of time series subsequences (Algorithm 3). The algorithm mainly consists of three components:

---

**ALGORITHM 3: TiVY (Time Series Visual Summary)**


---

**Input :**  $T$  – a set of time series as a  $m \times n$  matrix  
 $window\_size$  – window size  
 $\alpha$  – clustering strength  
 $minsup$  – minimum support

**Output :**  $\hat{G}$  – A list of subsequence groups

```

1  $P \leftarrow \text{construct\_sequences}(T, window\_size, \alpha)$            /* transform time series to
   discrete sequences */
2  $D \leftarrow \{\}$                                          /* dictionary to store pattern support */
3  $\text{prefix\_scan}(P, minsup, null, D)$                  /* compute pattern support */
4  $\hat{G} \leftarrow \text{extract\_groups}(D, minsup)$           /* extract optimal groupings */

```

---

(1) constructing symbolic representations from the time series, (2) retrieving supports of the sequential patterns, and (3) construct an optimal grouping from the sequential patterns.

### 4.3.1 Computing Symbolic Representations of Time Series

As mentioned in Section 4.2.2 and Figure 4.2(c), transforming a real-valued time series to a discrete symbolic representation involves the clustering using DTW metric. However, a straightforward clustering with DTW is infeasible even with moderate-sized data. The distance computation has a time and space complexity of  $O(m^2)$ , where  $m$  is the length of the time series. Also, although hierarchical clustering can automatically choose the most optimal number of clusters, it requires a pairwise distance matrix resulting in  $O(n^2)$  time complexity, where  $n$  is the total number of time-series. Thus, the total time complexity to construct symbolic sequences from time series has a time complexity of  $O(m^2n^2)$ . Even though it results in an effective perceptual result, the lack of scalability hinders visual analytics usage. Thus, we propose a novel clustering method to speed up the computations. It mainly exploits the properties that the *upper bound* of DTW distance between two time-series is the Euclidean distance. Therefore, we can group the time series with Locality-sensitive hashing (LSH) [135]. Basically, we hash each time series  $t$  into an integer bucket using the following hash function :

$$h(t) = \left\lfloor \frac{t \cdot x + b}{w} \right\rfloor \quad (4.5)$$

---

**ALGORITHM 4:** construct\_sequences

---

**Input :**  $T$  – a set of time series  
      $w$  – window size  
      $\alpha$  – clustering strength

**Output :**  $P$  – A set of symbolic sequences

```

1  m ← T.shape[0]                                /* number of time series */
2  n ← T.shape[1]                                /* length of each time series */
3  S ← array_split( $T, w$ )                         /* split T into segments */
4  P ← empty_matrix(m, n/w)
5  for  $i=0; i < n / w; i++$  do
6    | s ← S[i]
7    | cluster_labels ← []
8    | for  $j=0; j < m; j++$  do
9    |   | cluster_labels[j] ← LSH(s[j])
10   | end
11   | sample ← []
12   | for label in unique(cluster_labels) do
13   |   | sample.push(random_select(s[cluster_labels == label,:]))
14   | end
15   | dist_sample ← pairwise_distance( sample , 'DTW')
16   | linkage ← create_linkage(dist_sample, 'ward')
17   | sample_labels ← gap_rule(linkage,  $\alpha$ )
18   | for label, sample_label in unique(cluster_labels), sample_labels do
19   |   | cluster_labels[cluster_labels == label] ← sample_label
20   | end
21   | P[:,i] ← cluster_labels
22 end

```

---

where  $x$  is a random vector with each element sampled from Normal distribution  $x_i \sim N(0, 1)$ ,  $w$  is a width representing the quantization bucket, and  $b$  is a random variable sampled from the Uniform Distribution  $b \sim \text{unif}[0, w]$ . The hashing function ensures that if two time series are similar in terms of Euclidean distance, the probability of going into the same bucket (i.e.  $P(h(t_i) = h(t_j))$ ) will be high. Then, instead of running the DTW clustering on the whole dataset, we can run it with one sample from each hash bucket instead, which prunes most of the DTW calculations. The outline of the algorithm `construct_sequences` (Algorithm 4 and Figure 4.6) is as follows:

1. The clustering starts on segments in each time interval (line 5-6).
2. We are first interested in grouping the segments that are very similar to each other (Figure 4.6(b)). Thus, our *coarse* clustering objective is to ensure similar

---

**ALGORITHM 5:** prefix\_scan

---

**Input** :  $\bar{P}$  – an  $\bar{m} \times \bar{n}$  submatrix containing  $\bar{m}$  symbolic sequences with length  $\bar{n}$   
 $minsup$  – minimum support  
 $prefix$  – prefix sequence of the submatrix  
 $D$  – A reference of dictionary  $P \rightarrow \{ Time\ Segments \}$

```

1 if  $\bar{P}$  is empty then
2   | return
3 end
4  $first\_col \leftarrow \bar{P}[1 : m, 1]$                                 /* symbols on the current level */
5 for symbol, idx in unique( $first\_col$ ) do
6   | if  $Size(idx) < minsup$  then
7     |   | continue
8   | end
9   |  $D[prefix + symbol] \leftarrow D[prefix][idx, 2 : \bar{n}]$ 
10  | prefix_scan( $\bar{P}[idx, 2 :]$ ,  $minsup$ ,  $prefix + symbol$ ,  $D$ )
11 end

```

---

series are grouped into the same cluster. As mentioned above, we can use LSH (line 8-10) to assign the coarse cluster labels to the time series.

3. To combine these clusters for the final result, we can conduct the clustering with DTW distances and Gap Statistics (line 16-17) with only *one* random sample from each coarse group (line 12-14). The pairwise DTW calculations needed thus become much fewer.
4. Finally, we propagate the cluster labels obtained from each sample to the rest of its members in the same coarse group (line 18-20), which completes the whole process (Figure 4.6(c)).

### 4.3.2 Computing Support of Sequential Patterns

After constructing the sequences, we need to calculate the supports of each patterns that appear in all sequences. Retrieving the supports of frequent patterns is a computationally expensive sequence mining problem. Most existing algorithms [136, 137, 138] retrieve sequential patterns without knowing all supports. However, it is easier to retrieve them in our case. Our symbolic sequence  $T'$  contains ordered symbols that never repeat within the same sequence. Thus, all patterns can be presented by a tree-based data structure based on the prefix of their symbolic sequences. (e.g. the time series represented in + + are subsets from

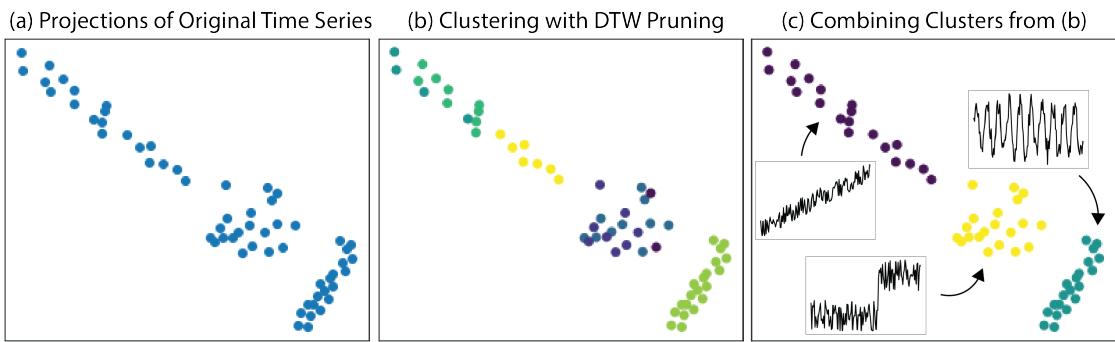


Figure 4.6: Clustering time series with DTW distance directly (from (a) to (c)) is an expensive computation. To speed up, we first (b) coarsely partition the data with LSH, then we can sample from each partition to compute DTW clustering with a smaller set of data.

the ones represented in  $C_{12} + C_{21}$ ) Thus, we extract patterns by grouping the time series symbol-by-symbol from the beginning of the symbolic sequences to the end (Algorithm 5). The algorithm handles one time interval in each recursion (line 4). The time series indexes (as rows in the input) are grouped together to the next recursion if they have the same symbol at the current interval (as column), or be split into different recursions if not (line 5). The time series will be stored in the dictionary if their patterns' supports are not smaller than  $minsup$ ; else, the recursion will stop (line 6-9).

### 4.3.3 Extracting the Optimal Groupings

Since the number of frequent patterns is exponential to the number of symbols [139], and the number of possible sets of groupings  $G$  is the binomial sums (i.e.,  $O(2^n)$ ), finding the set of optimal groupings is extremely hard. Thus, the proposed algorithm follows a greedy approach that evaluates the frequent patterns one by one, with a priority given to long frequent patterns. The intuition is that if the subsequence groups from long patterns are chosen, it is likely to reduce the groups from shorter patterns, which requires more groupings to cover the same number of time intervals. The outline of the algorithm `extract_groups` (Algorithm 6) is as follows:

1. As each group of subsequences contains time segments, which are subsets of a pattern (Equation 4.3), the algorithm extracts and removes the groups of subsequences from the dictionary of patterns' support  $D$  iteratively until it

becomes empty (line 2-3).

2. In each iteration, the algorithm only considers the longest patterns as the candidates for extraction (line 4-5).
3. From the candidate list, the algorithm tries to extract as many groups from these patterns as possible. Since the shorter patterns of these candidates must have the same or greater supports, maximizing the extraction of groups from the longer patterns prevents their subsets from being extracted. Such a condition of whether the extraction is maximized is considered during each time a candidate is evaluated (line 6).
4. In each evaluation, the supports of other candidates are minus by the number of intersections of time segments with the evaluated candidates (line 9-13). If the reduction of supports leads to a decrease of candidates that satisfy the *minsup* condition (line 14), the evaluated candidate will be removed in the dictionary without being extracted to the result (line 15-16). Otherwise, the candidate is extracted to the result (line 19), and the dictionary is updated (line 20-30). The update involves the removal of the same time segments shared between the selected candidate and both other candidates (line 20-25) and remaining patterns inside the dictionary (line 26-29).

## 4.4 Visualizing Time Series At Scale

In this section, we propose an efficient system to visualize time series at scale with the time series summaries. By efficiency, we do not only consider the visual styles to encode the time series effectively, but we also consider an accessible and computationally efficient way to render thousands of temporal information with a decent frame rate within a laptop. We first describe the analytical tasks supported by time series summaries. Then, we present an efficient implementation that leverages OpenGL and acts as a reusable widget in the Jupyter notebook.

### 4.4.1 Analytical Tasks

A visual summary of time-series effectively addresses the removal of visual clutter and optimization of layout compactness. To address a visual analytics

---

**ALGORITHM 6:** extract\_groups

---

**Input :**  $D$ : A dictionary of  $P \rightarrow \{Time\ Segments\}$   
 $minsup$ : Minimum support

**Output :**  $\hat{G}$  – A list of subsequence groups

```

1    $\hat{G} \leftarrow []$ 
2   while  $D \neq \emptyset$  do
3       delete  $D[p]$   $\forall p$  in  $D$  if  $Support(D[p]) < minsup$ 
4        $max\_length \leftarrow$  maximum length of patterns in  $D$ 
5        $Candidate \leftarrow \{p \mid p \text{ in } D\}$ , where  $length(p) = max\_length$ 
6       while  $Candidates \neq \emptyset$  do
7            $p_{candidate} \leftarrow \text{random\_select}(Candidates)$ 
8            $overlap\_patterns \leftarrow []$ 
9           for  $p$  in  $overlapped\_patterns(Candidates, p_{candidate})$  do
10              if  $Support(D[p] - D[p_{candidate}]) < minsup$  then
11                   $overlap\_patterns.push(p)$ 
12              end
13          end
14          if  $Size(overlap\_patterns) > 1$  then
15              delete  $D[p_{candidate}]$ 
16               $Candidates.remove(p_{candidate})$ 
17              continue
18          end
19           $\hat{G}.push(p_{candidate})$ 
20          for each pattern in  $overlap\_patterns$  do
21               $D[pattern] \leftarrow D[pattern] - D[p_{candidate}]$ 
22              if  $D[pattern] < minsup$  then
23                   $Candidates.remove(pattern)$ 
24              end
25          end
26           $combination \leftarrow \{p \mid p \text{ in } D\}$  where  $D[p_{candidate}] \cap D[p] \neq \emptyset$ 
27          for each pattern in  $combination$  do
28               $D[pattern] \leftarrow D[pattern] - D[p_{candidate}]$ 
29          end
30           $Candidates.remove(p_{candidate})$ 
31      end
32  end
```

---

approach that leverages such a summary, we reference the book from Aigner *et al.* [91], which describes a set of tasks involved in the time series data analysis. The overall tasks include classification, clustering, search and retrieval, pattern discovery, and prediction. Our visual summary falls into the category of clustering. The book references the classical paper regarding time series clustering by Van Wijk and Van Selow [140]. Also, a recent paper related to visualizing trends in time series [121] summarizes a set of clustering tasks from the book. Therefore, we base our design considerations on the clustering tasks from these three references. Furthermore, we include the system design tasks to maximize the real-world impact of our software. The tasks are summarized below. **T.1** and **T.2** focus on the tasks gathered from the surveyed time series clustering analysis and **T.3** and **T.4** focus on the interactions involved when visualizing clusters with additional attributes. Last, **T.5** focuses on the scalability and accessibility of the system.

**T.1** Identify common subsequences in large time series.

- a. Understand the support of each cluster.
- b. Understand the time interval and range of each cluster.

**T.2** Understand the distribution of each subsequences cluster.

- a. Assess the quality of each cluster.
- b. Understand the shape differences among clusters.

**T.3** Filter and zoom for specific clusters.

- a. Highlight clusters by their temporal values.
- b. Inspect individual time series within a cluster.

**T.4** Compare clusters with different attributes.

**T.5** Integrate to the real working environment.

- a. Implement as an interactive widget inside the computation notebook with low hardware requirement.
- b. Render visualization with low latency and scale to real-world datasets.

#### 4.4.2 Visualizing Time Series Summary

Designs to visualize multiple time series have been thoroughly studied [83]. However, when it comes to high volumes of temporal information, the main challenge is the technical scalability to render millions of data points representing the lines while performing various interactions seamlessly. We need to consider different

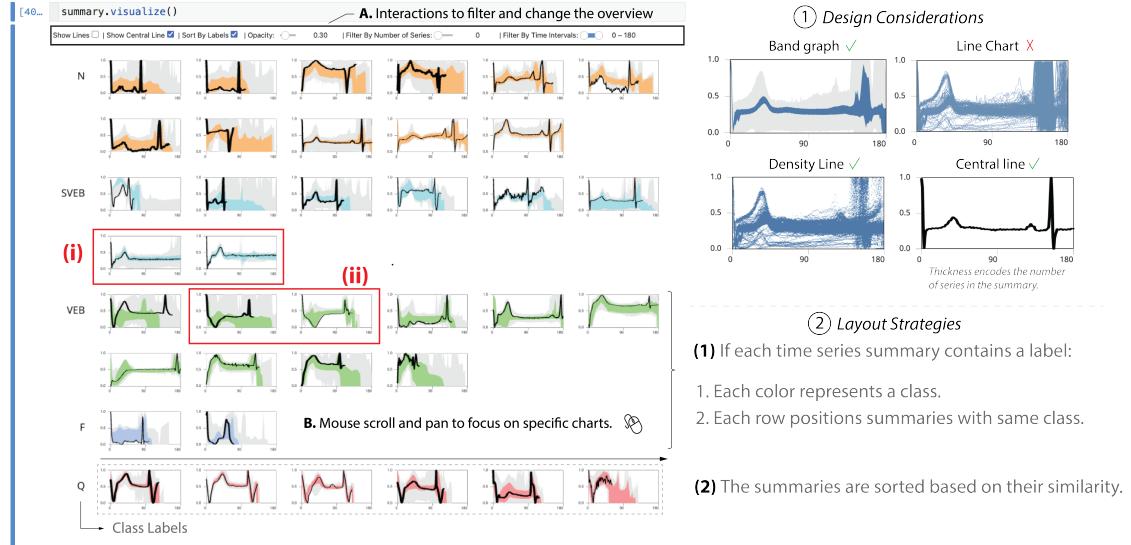


Figure 4.7: Time series summary for 100,000 ECG signals in a WebGL based widget in Jupyter notebook. Summaries of different shapes are visualized in different small multiples for categorizing different waveforms to help proper diagnosis and treatment (i, ii). (1) Visual Design: Each chart is shown with a bold line encoding the representation and size of the summary. The time series inside can be encoded as a band graph or density line chart. (2) Layout: Colors and positions encode the labels of the summary, and the charts are sorted based on their pairwise similarity.

tradeoffs to balance the software efficiency and accessibility. Thus, we now propose our time series visualization system (Figure 4.7), and discuss the visual encodings that consider both visual clarity and rendering efficiency.

#### 4.4.2.1 Visual Encodings

Since TIVY allows a holistic high-level summary of all subsequence clusters in a time series dataset, each group could be visualized with a precise shape (**T.2**) and displayed with the main statistics (**T.1**) such that users can quickly acquire an overview of multiple clusters plotted on the same screen. Moreover, it becomes feasible to use aggregated visual encodings and superposition visualization in Figure 4.1(b) and (c) to present a more significant number of time-series inside the cluster. Overall, our design considerations include the usage of band graph, line chart, density line chart, and a central line to encode the time series inside each summary (Figure 4.7 (1)).

**Band graph.** We provide two bands (i.e., range + 90% quantile) to visualize both

the extrema and tighter bounds of the time series inside the cluster to balance between the visual clarity and accuracy. The advantage is that since the time series within each chart is homogenous, the band can accurately depict the trends and hinder the noises (**T.2**). Also, in terms of rendering, we can use two polygons to render two bands, which are much simpler than rendering millions of points (**T.5**). We acknowledge the fact that further user study is needed to evaluate the tradeoffs between clarity and uncertainty when choosing the number of quantiles in uncertainty visualization.

**Density line chart.** To reveal original data points in the chart, we can use line charts to visualize the time series. However, direct input of data points to the rendering pipeline will easily hinder the system’s interactive performance. When users pan and zoom the whole graph, each data point in the time series has to undergo several linear transformations to define its new location in the screen. For example, 100,000 time series with a cardinality of 180 will require 18 million transformation operations. Worse still, to enable GPU acceleration such as OpenGL, lines are commonly represented as 2D meshes (i.e., triangulated polygons), which result in more vertices as input to the rendering pipeline. For a common laptop and latency in milliseconds allowed only, users will easily experience lagging during the interactions (**T.5**).

To address the increasing data points on rendering the line charts, we need to avoid the rendering complexity to be linearly proportional to the number of data points. One way to do so is to aggregate the input time series to 2-D density maps, which are bounded to a fixed number of bins. By having a slight overhead to compute the density map, we can instead input much smaller meshes for real-time interactions. The color opacity encodes the density such that we can inspect the distribution of temporal values inside the cluster (**T.2**).

**Center line.** To improve the reflection of the main statistics in the aggregated plot (i.e., band graph), we use the medoid [141] in the cluster to present the main shape. A medoid is the best representative line inside the cluster that has the minimum sum of pairwise distances with other lines. Choosing a line instead of using time-invariant averaging methods such as soft-DTW [142] or DTW Barycenter Averaging [143] to compute an “average” line avoids computationally expensive algorithms ( $O(N \cdot T^3 + N^2 \cdot T^2)$ ) for useful interactive analysis. We also encode the

line width with the number of time series inside the summary to better estimate cluster size (**T.1**).

#### 4.4.2.2 Layout Strategies

To allocate the time series in a compact layout, we position each time series chart as small multiples that fills the entire canvas from left to right and top to down (Figure 4.7 (2)). Besides, when users supply an attribute to each summary (e.g., class labels), we can encode the summaries with colors and group the same ones by vertical positions (Figure 4.7(i)) to present clearer comparisons (**T.4**). Also, summaries might share similar shapes among each other as well. Thus, we can sort the summaries by their similarity. It can be done by first sampling one series from each summary, then build a hierarchical cluster (i.e., linkage) with the samples and obtain the order from the leaves in the hierarchy (Figure 4.7(ii)).

#### 4.4.2.3 Interactions

Our widget provides the interactions to facilitate the exploration of time series summaries (Figure 4.7A).

**Pan and zoom.** Since our rendering pipeline provides great computation efficiency on the linear transformation on the graphics, users can easily pan and zoom the whole canvas to focus on a subset of time summaries in real-time (**T.2**).

**Filtering.** Since each summary contains statistics such as the number of time series and time intervals, our system provides two sliders that filter the summaries based on that. These allow the users to explore the important trends effectively within a specific time interval (**T.1**).

**Toggles for different visual encodings and layouts.** Our system provides different toggles to reveal different visual encodings on demand. Users can either display the summaries as band graphs or density lines and select whether they want the central line or not. Furthermore, users can select whether they want the summaries to be separated by the attributes or not (**T.4**).

#### 4.4.2.4 Implementation

Our algorithm and system are implemented with Python and can be used in Jupyter notebook and Jupyter Lab. WebGL is required to render the time series visualization. We use NumPy for most of our computations in TiVY and VisPy for rendering the meshes representing different visuals in the time series views.

To reduce the latencies during visualization interactions, such as filtering or switching the charts, we first compute the vertices of all possible visuals (i.e., polygons of band graphs, centerlines, and density lines) and store them in a buffer object. Then, when each visualization is selected and shown, we can directly import the locations to the rendering pipeline without creating the vertexes on the fly. Our visualization interface is available in <https://github.com/GromitC/Jupyter-Time-Series-Visualization>.

## 4.5 Evaluation

In this section, we conduct both quantitative and qualitative evaluations on the performances of the visual summary extraction pipeline, as well as use cases of the summary results. Our goal is to demonstrate that:

1. Our pipeline computes and renders results in interactive time for large scale data.
2. Using real-world datasets, our algorithm produces accurate and robust representations to explore large scale medical and financial data.

### 4.5.1 Datasets and Apparatus

All of our experiments are conducted in a MacBook Pro with 2.4 GHz 8-Core Intel Core i9 CPUs and 32GB RAM. For a fair comparison among different algorithms, we do not perform our algorithms under a multi-core or distributed environment. We use the following datasets for our experiments:

**Stock Time Series.** The dataset contains 4,470 company daily adjusted stock prices in NASDAQ between the year of 2015-16. The industry sector for each stock price are also provided.

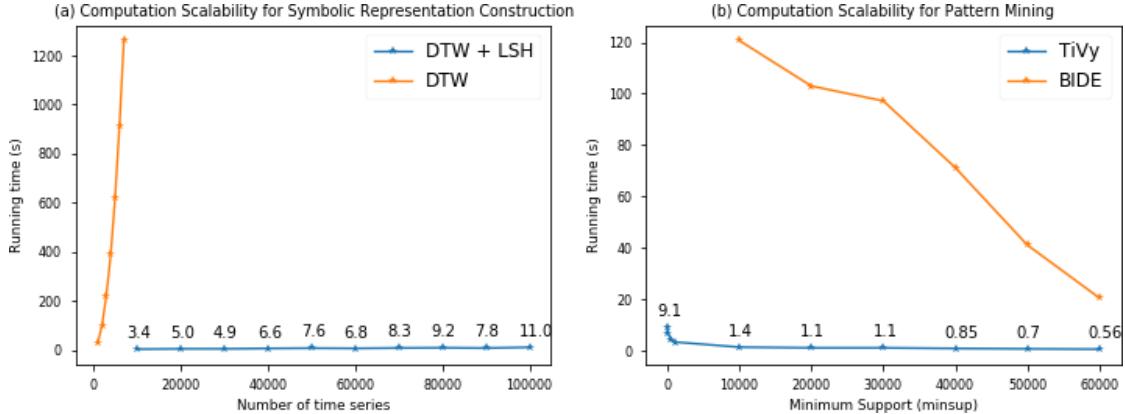


Figure 4.8: Run time improvement with LSH and indexing.

**ECG dataset.** We use the MIT-BIH Arrhythmia ECG dataset<sup>1</sup> that records 100,000 patients’ heart beat signals. Each signal has a cardinality of around 200 and we trim the trailing zeros for each signal. The heartbeats are annotated by five groups of heart conditions.

#### 4.5.2 Quantitative Evaluation

In this section, we report the effects of visual outcomes by different design choices in the algorithm, and the performances with different scalability and alternatives.

**Computation Scalability.** We report the run time using the 100,000 ECG data series as the example in Figure 4.8. To highlight the effect of LSH in reducing the quadratic time complexity of DTW distance metric and hierarchical clustering in the symbol construction step (Section 4.2.3), we sample the ECG data and compare the run time between the algorithms with and without hashing optimization. We fix our parameters on time windows and clustering strengths so that the final results are the same. In Figure 4.8(a), it shows that our LSH optimization is able to speed up the process from hours to within a couple of seconds. For the discrete pattern mining process (Section 4.3.2, 4.3.3), we compare our grouping algorithms with general purpose pattern mining approach [138]. In Figure 4.8(a), it shows that given our unique structure of symbolic patterns, we can speed up the computations from long time to seconds easily.

For the rendering pipeline, we report the preparation time of the buffers input to

---

<sup>1</sup><https://www.physionet.org/content/mitdb/1.0.0/>

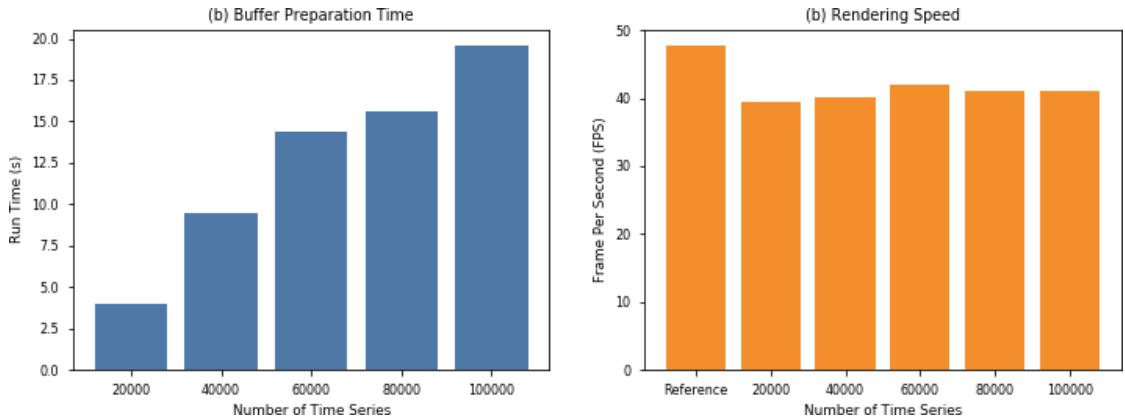


Figure 4.9: Rendering preparation time and FPS.

the system and the interaction latencies during the filter, pan, and zoom operations (Figure 4.9). We can see that by addressing the linear overhead of preparing the visual buffers to the system which is an one-off computation only (Figure 4.9(a)), we can achieve a seamless exploration of time series data in the interactive interface with good FPS (Figure 4.9(b)).

### 4.5.3 Use Cases

In this section, we present two usage scenarios to demonstrate the effectiveness of TiVY. First, we study whether our approach can summarize meaningful patterns for correct categorization of the ECG data. Second, we apply our technique to help financial analyst understand important trends in the financial stock market. For both cases, we fix the time window size to one-tenth of the total durations, clustering strength to 1, and minimum support to 50. The summarization outcomes for both of them capture more than 95% data points in the datasets.

#### 4.5.3.1 ECG Signal Classification

In this use case scenario, we demonstrate whether TiVY are able to visualize large amount of time series effectively. We use the MIT-BIH Arrhythmia ECG dataset <sup>2</sup>. ECG is widely used in medical practices to monitor cardiac health. Understanding the waveforms and attributing them to the correct categorization is important for proper diagnosis and treatment. In this dataset, each signal consists

<sup>2</sup><https://www.physionet.org/content/mitdb/1.0.0/>

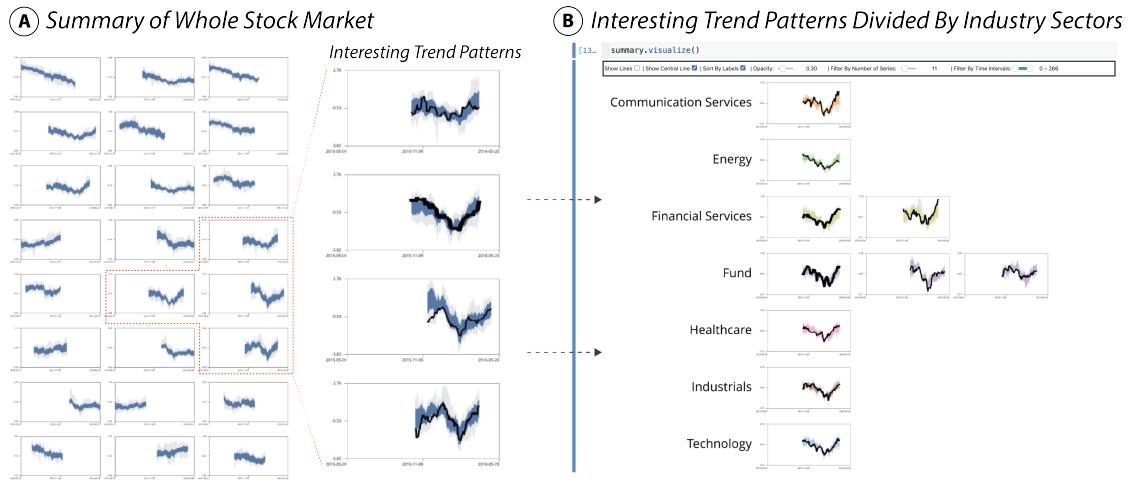


Figure 4.10: A time series visual summary is a set of time series subsequence clusters of varying lengths that groups visually similar time segments together. In a stock market use case, a financial analyst uses our algorithm, TIVY, to explore visual summaries of 4,470 stock market time series in 2015-16 for portfolio construction. (A) He first runs the algorithm to create subsequence clusters that cover main trends in the dataset. (B) After identifying different trends in the market, he identifies a common “v” shape pattern among stocks in the first three months of 2016 and splits the subsequence clusters by sectors to observe which sectors contain this pattern.

of heartbeats annotated by at least two cardiologists. The annotations are mapped to five groups suggested by AAMI: Normal (N), Supraventricular Ectopic Beat (SVEB), Ventricular Ectopic Beat (VEB), Fusion Beat (F) and Unknown Beat (Q).

**Exploring patterns among 100,000 time series.** The overall summarization result is shown in Figure 4.7. We presented the results to the cardiologists and they quickly identified some interesting patterns. For example, the patterns in Figure 4.7(i) are long flat lines of dropped beats, which are clear characteristics of junctional escape beat belonging to the SVEB group. Also, Figure 4.7(ii) shows strong contraction beats with a long pauses afterwards, which are symptoms related to premature ventricular contractions. These illustrate that TIVY successfully extracts the visually meaningful patterns among 100,000 ECG signals, which corroborates the verifications from two independent cardiologists.

#### 4.5.3.2 Financial Time Series Analysis

The second use case involved a financial analyst working with stock market data to construct a portfolio. The dataset contains 4,470 company daily stock price series between the year of 2015-16. The goal is to study the trends occurred in the financial market to construct a portfolio that balances the risks in different situations. The time series are normalized with zero mean and unit variance to calibrate the trends with different numerical prices. In this use case scenario, a portfolio manager uses TiVY to explore and select stocks for creating a portfolio for his client through the understanding of different visual behavior of the stock time series.

**Observing overall trends.** To begin with, the portfolio manager runs the algorithm on the whole 4,470 stock time series. The visual summary shows 23 trends that, in total, cover most of the time series data (Figure 4.10 (A)). By inspecting the shapes of the trends, he understands that most companies' prices were decreasing throughout the period. However, he also discovers some increasing trends as well as the ones with zig-zag shapes. He understands that the zig-zagging stocks are the ones that are being actively bought and sold (i.e., a price war). As his client is risk-averse (i.e., afraid of risks), he labels these stocks as the one not recommended for purchases.

**Focusing on selected trends.** Next, the portfolio manager wants to know if there are any more risky trends, so he filters the time series subsequence clusters with these “v” shapes (Figure 4.10(b)). The visualization reveals that this pattern happens in early 2016. Since the trend exists in many such companies, there must be a global economy issue that affects the whole market. The portfolio manager is aware that actively trading the stocks impose a higher risk in this situation.

**Exploring industry sectors.** The portfolio manager then tries to know if there are any relationships between the trend and the sectors, so he further splits the clusters by the sector. The result shows that the trends mainly happen in four sectors represented by four thick lines: Industrial, Technology, Healthcare, and mutual funds (non-sector) (Figure 4.10(c)). As a result, he decides to research passive trading strategies for these four categories of companies. Overall, the algorithm provides a visually compelling result for him to explore the stock trends freely and leverage the system to perform multi-model data analysis.

## 4.6 Conclusion

In this chapter, we propose a novel summarization technique to visualize large scale time series and a comprehensive visual analytics approach to address the challenges. First, we propose a sequential pattern mining approach that involves transforming time series to discrete sequences and optimize the time series subsequence patterns based on minimum support. Then, we present a sophisticated visual analytics system with a efficient visualization interface to display large scale time series data. We present experiments and two use cases with different real world dataset to showcase its effectiveness.

# Chapter 5

## Machine Learning Model Summary

Using feature importance explanations to explain predictive models' decisions has been prevalent in Artificial Intelligence research and real-life usage scenarios. For machine learning models, ranging from coefficients in simple linear functions to gradients of complex deep learning models, these internal attributes guide the models to make decisions by assigning the *weights* to different parts of the input in affecting the prediction outcomes. Therefore, different kinds of feature importance explanation methods have been developed to capture these weights to explain the model behavior. For example, saliency maps are used to highlight the important regions in the original images or texts to explain the relationships between the input and the predictions of deep neural networks [144, 145, 146]. Besides, there are local explanation models using linear weighted functions that fit the models in a close region around an input to explain the models locally [147, 148, 149]. Furthermore, instead of weights on the original inputs' dimensions, one can use weights on a set of analogies to explain an input's similarity to ground-truth labels [150, 151, 152] in the neural network.

As a result, together with decision trees and rules, feature importance explanation is regarded as one of the most three popular interpretation methods to understand machine learning logics [153, 154, 155]. To clarify, we define the interpretability of a machine learning model as the capability of explaining reasons for its prediction in ways humans can understand. This is of paramount importance

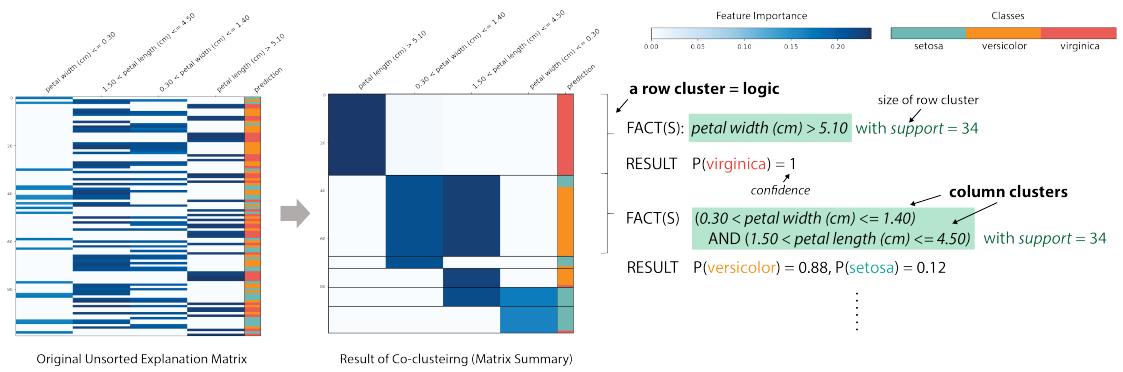


Figure 5.1: An illustration of explanation summary by applying LIME to a Random Forest classifier of Iris data. The summary can be perceived as a set of unordered conjunctive (AND) logics. Each row cluster can be interpreted as a logic with column clusters intersecting with the row cluster as conditions (FACTS). The predictions of the rows inside the row cluster convey the consequent (RESULT) of the logic as probability distributions of each possible class. The sizes of row clusters provide the support of each logic.

since the end users are mostly domain experts who need to be responsible for the deployment of models in real-life scenarios. In high stake decision making situations such as health care [156], criminal justices [157], and financial risk assessment [158], the models are required to provide these interpretations to satisfy the ethical regulations of data [159].

Among these three explanation methods (i.e., decision trees, rules, and feature importance), we observe much difference between the feature importance explanation and the other two. Although feature importance explanations are more prevalent in explaining accurate and complex models like deep neural networks, they can only be used to explain the model locally for a single instance. This can be described as a trade-off between accuracy and efficiency: while the explanation is more accurate and faithful to the original model, it loses the ability to generalize the whole dataset's understanding. Seeing this limitation, in this paper, we propose the idea of summarizing every feature importance explanation in the dataset and provide a scalable visual analysis interface to understand these explanations at scale. An illustration of the summarization outcome can be seen in Figure 5.1. The ingredients for an explanation summary are an input dataset (e.g., iris dataset), a classifier (e.g., a random forest classifier), and an explanation method (e.g., LIME). Grouping the explanation feature vectors from the input dataset provide us an

overview of the model as patterns of logic. We will explain each visual component in detail in Section 5.2.3, but overall our goal is to enable feature importance explanation to explain the classifier globally.

To generate the explanation summary, it consists of three steps. First, we standardize the explanations among different major feature importance explanation methods and across the whole dataset to a structured format. Then, we extract the patterns among all explanations to discover logics in the model with optimal granularity. Finally, we develop a visual interface to deliver interpretability by the interactive exploration of the details in the model’s decision making logic. In summary, the main contributions are as follows:

1. We propose a visual summarization technique for domain experts to understand classification models using feature importance explanations *on the whole dataset*.
2. To enable scalable data analysis, we also introduce efficient algorithms and data structures to summarize the explanation interactively.
3. We present three use cases using tabular, image, and text data and a user study to demonstrate the proposed method’s effectiveness.

## 5.1 Related Work

To facilitate human understanding towards complex models through visualization, research mainly focus on visualization on three aspects: model internals, logics induced from the models, and instance level feature vectors that describe the behavior of the model.

### 5.1.1 Visualization of Model Internals

Visualization has been applied readily to understand and interact with deep learning neural networks. In fact, a survey about deep learning visual analytics by Hohman *et al.* [160] has listed more than 40 representative works in this area in the last 5 years. We encourage the readers to read the survey paper for a deeper investigation to the subject.

The simplest form of a neural network can be represented by a node-link diagram in which each node represents a neuron and link presents a connection weight between two neurons [161]. As nowadays the ways neurons are connected become more sophisticated and opaque, various visual analytics approaches have been developed to understand different properties of the networks. RNNVis [61] and LSTMVis [162] address the understanding of recurrent neural network (RNN) by visualizing the bipartite relationship between hidden memories and input data and hidden memory dynamics with parallel coordinates respectively. Autoencoder is addressed by Seq2SeqVis which proposes a bipartite graph visualization to visualize the attention relationships between input and its possible translations to enable model debugging [163]. Another type of popular models for image classification is Convolutional Neural Networks (CNN). CNNVis [164], Blocks [165], AEVis [166] and Summit [167] are graph visualizations that aggregate similar neurons, connections, and similar activated image patches to convey learned visual representations from the model.

Besides visualizing the structures of a neural network, there are visual analytics systems that assist the model development processes in the industry. ActiVis [168] is a visual analytics system used by Facebook to explore industrial deep learning models. Google has developed TensorFlow Graph [169] and what-if tool [170] to help developers understand and test the behavior of different ML models.

The work in these criteria mainly addresses the visual analysis for model developers who have sufficient knowledge of the methodologies of their models. However, a more general AI tool requires the assessment and involvement of end-users, decision-makers, and domain experts. Addressing the needs of border XAI user experience, our work focuses on providing general explanations of ML models to users without requiring them to know the architectures.

### 5.1.2 Visualization of Logical Models

Logical models like decision trees [171] or rules [172, 173] can address the interpretation of complex models by using them to infer an approximated model from any ML models. Given a set of test data, the original model gives the predictions and the logical models use them as labels to train another classifier. The resulting classifier can be used to mimic the behavior of the original model

while providing good interpretability to the users. Through visualizing logical models, users gain knowledge of the model’s capability.

Rule Matrix [174] is proposed to build and visualize the surrogate rule list to understand the model’s behavior by interacting with the rules. Gamut [175] uses generalized additive models to construct a set of linear functions for each feature in the dataset to understand models through line charts. TreePOD [176] and Baobabview [177] visualizes the decision trees with different metrics incorporated for model understanding. iForest [178] visualizes random forests with data flow diagrams to interact with multiple decision paths.

For complex model, using logical models to explain the complex model is the consideration of *fidelity* – the accuracy of the explanation on the original model. It creates an additional layer of performance concerns. Therefore, feature importance explanation methods are proposed to explore the possibility to provide accurate explanations or even be embedded in the original model training process. Yet, they only return results for an instance and do not consider a general explanation to the whole dataset, our work addresses the challenges of visually constructing a global view for local explanations.

### 5.1.3 Feature Vector Visualization

Local explanation models give feature scores for each instance. The features can be the features from original data [147, 148, 145] or a set of external information like concepts [179] or training data [150, 151, 152]. Visual analysis can be directly applied to interact with the features [180] or the feature vectors can be visualized as a matrix where rows represent the instances and columns represent the features [181].

Besides, the data comes out from a deep neural network can appear as embeddings such that the linear distances between vectors represent their similarities as the model’s rationale. The main visualization technique to understand these feature vectors is projection [182, 183, 184, 185, 186, 187]. Treating the embedding as high dimensional data projection techniques such as tSNE, MDS, or PCA are applied to discover semantic groups inside the dataset from the resulting scatterplot. Users can assess the ML model and refine the original data from the brushing the filtering interactions in the projection.

Our technique identifies the scalability and usability challenges in the existing

visualization. The projection technique mainly suffers from cluttering and the lack of feature information in the visualization which is crucial for a comprehensive explanation. MELODY aims at providing compact representations of both data and features so that visual information is more precise. Also, we address the needs of explanation exploration with different granularity by the visual encoding illustrated by MELODY MATRIX, providing new ways to extend the powerful feature importance explanations to scalable visual analytics.

## 5.2 Pipeline to Summarize ML Models

In this section, we introduce our pipeline for creating a feature importance based visual interface to help domain experts understand, validate and inspect the behavior of a machine learning model.

### 5.2.1 Goals and Target Users

As mentioned in the beginning, our goal is to allow feature importance explanations to interpret the logics behind a machine learning model on the whole dataset. The interpretation should target on the *end users*, unlike model developers, to fully understand the model behavior so that they can apply it to problems from users' domains. To investigate what questions needed to be answered from domain experts, we can study the previous tasks addressed by interactive systems that provide analysis of interpretable models to these users. To be specific, we summarize the questions addressed by systems visualizing decision trees [177], rule list [174], and generalized additive model (GAM) [175] for end users.

**Q1 What kinds of logics has the model apply on the dataset?** When receiving inputs from a dataset, the trained machine learning model can be seen as an extraction of logics from the dataset to make predictions. For feature importance explanations, we present the logics as a set of facts (clauses) that leads to an outcome. In this way, users can focus on understanding the facts without the needs to study other representations.

**Q2 What is the outcome of each piece of logic?** For each piece of logic, we can acquire three important information: (1) the *outcome* (probability

distribution of predicted classes inside a logic); (2) the *accuracy* (how well the predicted distribution is consistent with true class distribution), and (3) the *support* (the amount of data in the logic). Evenly distributed outcome means the logic is general to many possible outcomes, low accuracy reveals the error region of the machine learning model, and low support means a rare logic in the dataset. These statistics provide a guidance for domain experts to investigate on each logic.

### **Q3 What are the predictions / logics involved in a logic / prediction?**

The question is concerned about customized starting points of the analysis. For predictions, the question is like “what are the reasons for a particular prediction?” And for logics, the question is like “how does the model use  $j$  a logic $_j$  for prediction?” These considerations are concerned with the verifications of the models using experts’ expertise. There might exist some strict regulations or expertise after years of development, thus it is important to integrate these knowledge in the analysis.

### **Q4 When and where does the model fail?** Checking the limitations of the machine learning model is one of the most important activities in machine learning tasks. There are many reasons for a model to fail (e.g. data quality, architecture design, unseen observations), and it is crucial to provide guidance to discover that.

#### **5.2.2 Constructing an Explanation Matrix**

To begin with, what is an *explanation* of a machine learning model? In current literature, we often see examples like highlights of texts in a paragraph (i.e. saliency maps [188]) or bar charts showing the important features that the model uses to predict an input [148, 147]. These are the gradients of input [144], which is one of the most popular methods to explain an input’s prediction based on a set of salient features. Recently, there is an increasing trend of producing explanations by analogy. For example, prototype learning [150] and TCAV [179] explain an image’s classifications by similar instances (“this image (*input*) looks like that image (*ground truth data*)”). However, these methods can only be used to explain a model locally for a single instance.

Interestingly, although these methods can only be applied to a single input, when

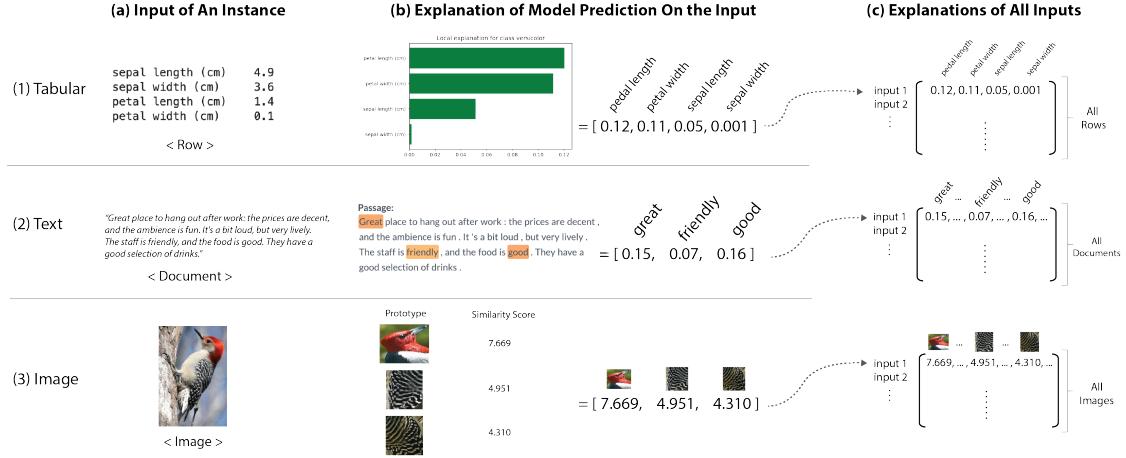


Figure 5.2: Examples of transforming the feature importance based explanations to real-valued vectors from different (a) data types and (b) explanation methods. The result of stacking the feature vectors from all inputs is a matrix containing the rationale of a model’s predictions to a dataset.

we apply them to every instance in the dataset, it could result in a structured matrix format that provides opportunities for scalable visual analysis. To illustrate, we present outputs from LIME, AllenNLP, and ProtoPNet, which are libraries having more than hundreds of stars in Github to explain predictions on tabular, text, and image data respectively (Figure 5.2). The similarity among these explanation methods, regardless of data types, is the mathematical output as a real-valued vector (Figure 5.2(b)). These vectors represent the important salient features that are used to explain an input’s prediction. If we treat each instance’s explanation as a sparse vector and vertically stack all instances’ results from the dataset, we will create an *explanation matrix* where rows represent instances and columns represent all the features used in the explanation methods (Figure 5.2(c)). As a result, the patterns inside the matrix will convey how the machine learning model works on the whole dataset.

### 5.2.3 Patterns in the Explanation Matrix

As mentioned above, an explanation matrix  $E$  consists of a set of rows  $R$  as instances in the dataset and columns  $C$  as explanatory features used in the explanation methods. The patterns obtained from mining the matrix are defined as a set of rows and columns partitions (co-clusters) which can be interpreted as textual logical expressions (Figure 5.1). The clusters can be arranged as a set of

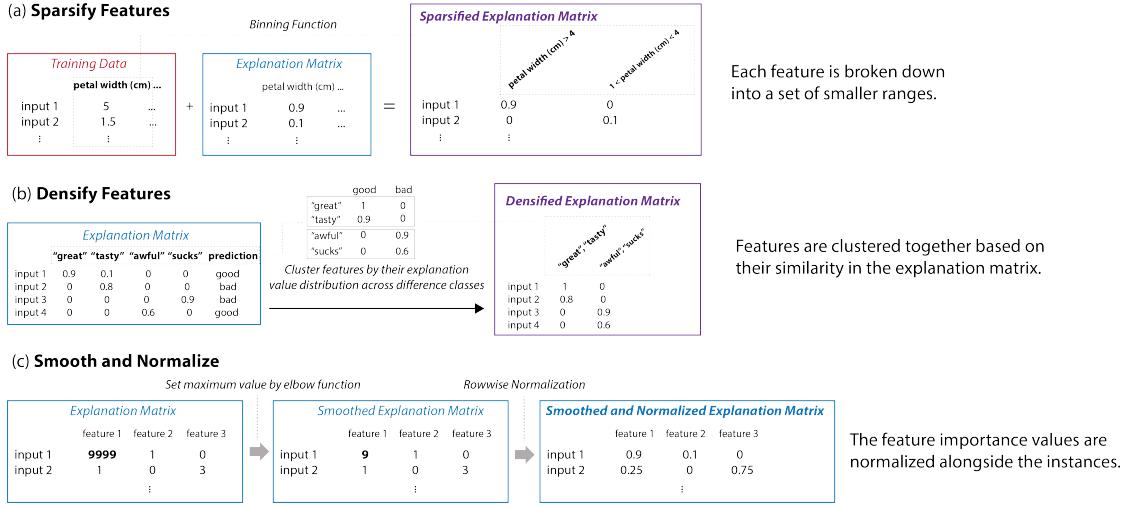


Figure 5.3: Different strategies improving the interpretability of the explanation matrix under different scenarios. (a) Replacing the columns in the matrix with bins (i.e. set of intervals) to sparsify the features. (b) Combining the sparse columns based on their similarity in the original data. (c) Reducing the anomaly values using knee finding algorithms and normalizing the values across the rows.

unordered conjunctive (AND) logics. Each row cluster (i.e. cluster of instances in the dataset) represents a logic with the intersected column clusters (i.e. co-clusters related to the row cluster with high feature importance) as the important conditions (FACTS) describing the logics. As the row cluster contains instances with prediction outcomes, the probability distributions of these outcomes construct the consequent (RESULT) of the rule. Each probability outcome represents the *confidence* of a prediction in the rule and the number of instances in the logic represents the *support*. In the Iris dataset example in Figure 5.1, we can see that each column, which represents a range of the values, combines based on the feature vectors from LIME explanations to the dataset to explain the Random Forest classification model as logical visual expressions (e.g. The FACT petal width (cm) > 5.10 leads to the RESULT of P(virginica) = 1). Similarly, if the features are some analogies (e.g. visual prototypes) or categorical variables (e.g. texts), then the clauses inside the logical expression can be interpreted as similarity (e.g. The FACT similarity( dog prototype ) > 0.8 leads to the RESULT of P(dog) = 0.9 ) or existence (e.g. The FACTS exist('delicious') AND exist('great') lead to the RESULT of P(good) = 0.8).

## 5.2.4 Feature Engineering

In practice, the feature vectors generated by the explanation methods might not convey much insight when combined as a whole into an explanation matrix. This often happens when the features in the explanation are directly mapped from the original dimensions in the dataset. First, we describe two potential issues, when the explanation matrix is either overly *dense* or overly *sparse*. We provide example implementations for both scenarios in the Appendix. Then, we describe a way to smooth and normalize the matrix to further address some typical data issues in the matrix.

### 5.2.4.1 Sparsify the Explanation Matrix

In this situation, the features from the explanation method are homogenous across different instances. For example, it might output explanations like  $\{pedal\ length : 0.12, pedal\ width : 0.11, sepal\ width : 0.001\}$ , which states that the values of “*pedal length*” and “*pedal width*” in an input are important to its prediction outcome. However, this does not tell us what are the exact values of that input, which prevents a more in-depth understanding (e.g. how do different values inside a column affect the model behavior) when the inputs are clustered together in the matrix.

To address this problem, the usual way is to apply binning strategies to the explanation matrix with regards to the original values of the inputs (Figure 5.3(a)). For each feature in the explanation of input, instead of assigning the importance value to the original feature, the value is assigned to one of the *bins* of the feature that the input’s original value belongs to. These bins represent the ranges of values that explain the model’s behavior with a finer granularity. The binning function could be different kinds of discretizers (e.g. quartile, decile, or entropy), depending on the characteristics of the data. Such strategy is used in LIME’s implementation as well [148].

### 5.2.4.2 Densify the Explanation Matrix

In the opposite situation compared with the above, the features from the explanation method are too specific related to a small set of instances. For example,

explanations of document classification might output the important vocabularies that only exist in few documents (e.g., One document uses “great” to describe the food while another document uses “delicious” to express the same sentiment). In this case, the features are similar regarding the prediction outcomes they lead to. Thus, we can combine these sparse features into denser topics so that rows with the same topics can be clustered more easily.

To achieve the combination of features leading to similar outcomes, we can extract the importance of each feature to different classes by summing up the feature importances in the matrix by each class. This results in a low dimension matrix that encodes the influences of features on each prediction. The more similar in influence the features are, the more likely they convey the same meaning from the machine learning model. Then, we can apply straightforward clustering methods (e.g., K Means, DBScan, or Hierarchical Clustering) to group similar features. Finally, we group the values in the explanation matrix by summing them according to the cluster’s features. As a result, the densified explanation matrix will be more likely to form patterns for visual analysis. Such strategy is inspired by topic modeling approaches like LDA [189], but with a supervised set of labels.

#### **5.2.4.3 Smoothing and Normalizing the Explanation Matrix.**

Last but not least, we observe that sometimes the values of feature importance in the explanation matrix can be extremely high. This happens when the values of the input are located in the turning points of the machine learning model (i.e. gradient equals infinity). To ensure similar and important explanations can be clustered together, we can set the maximum values of the matrix to the knee point of the overall value distributions in the matrix using a knee finding algorithm [190]. Also, we normalize the matrix by the sum of each row to unify the understanding of input’s explanation as a percentage contribution to the prediction outcome (Figure 5.3(c)).

## 5.2.5 Information Theoretic Approach to Co-cluster the Explanation Matrix

We now present our information theoretic approach to co-cluster the explanation matrix such that it reveals the patterns like the visual analysis process illustrated in Figure 5.1. The approach provides the foundations to address two challenges: (1) unsupervised clustering, where the number of clusters does not need to be manually defined, and (2) interactive clustering, where the clustering results can be refined by visual analytics approaches.

### 5.2.5.1 Problem Definition

Let  $R$  and  $C$  be the set of row (instance) and column (feature) vectors in the explanation matrix  $E$  respectively such that  $E$  is equivalent to a joint distribution between  $R$  and  $C$  (i.e.  $p(R, C)$ ). Our goal is to find the optimal row and column clusters  $\hat{R}$  and  $\hat{C}$  so that they represent the logics in the explanation matrix like Figure 5.1.

The main idea of information theoretic approach is the use of measurement of data compression to find the best model selection [191]. The best model selection should satisfy the *Minimum Description Length* (MDL) principle: the best model should minimize its total description length  $L$ , which is made up of a model description length and description length of the original data with the help of the model:

$$L = L(M) + L(D|M) \quad (5.1)$$

In our problem setting,  $L(M)$  is proportional to the number of row and column clusters and  $L(D|M)$  is proportional to the compression loss by using clusters to represent the aggregated explanation matrix. By minimizing the total description length, we group similar instances and explanatory features simultaneously so as to balance compactness and information loss. For example, if we do not cluster any rows and columns at all,  $L(D|M)$  will be equal to zero but  $L(M)$  will be huge (i.e. many clusters to explore), whereas if we only have one cluster, the loss will be huge (i.e. the clusters poorly represent the information in the explanation matrix). Thus, we can translate it to a cost function  $T$  to find the best rows and columns

partition by minimizing it:

$$T(\hat{R}; \hat{C}) = \beta_R \|\hat{R}\| + \beta_C \|\hat{C}\| + D(\hat{R}, \hat{C}) \quad (5.2)$$

Here, the size of model can be represented by the number of rows and columns clusters  $\|\hat{R}\|$  and  $\|\hat{C}\|$  and the compression loss is defined as  $D(\hat{R}, \hat{C})$ , which is derived from the clustering result.  $\beta_R$  and  $\beta_C$  are user defined parameters to penalize large number of clusters so that users can increase the values to produce fewer clusters.

The next question is, how we should measure the compression loss? For example, consider the following synthetic explanation matrix below:

$$E = P(R, C) = \begin{bmatrix} .1 & .1 & 0 & 0 \\ .1 & .1 & 0 & 0 \\ 0 & 0 & .2 & .2 \\ 0 & 0 & 0 & .2 \end{bmatrix}$$

It is obvious to group the rows into two clusters:  $\hat{r}_1 = \{r_1, r_2\}$ ,  $\hat{r}_2 = \{r_3, r_4\}$  and the columns into two clusters:  $\hat{c}_1 = \{c_1, c_2\}$ ,  $\hat{c}_2 = \{c_3, c_4\}$ . The information theoretic definition of the resulting compression  $P(\hat{R}, \hat{C})$  and the approximation matrix recovered from the compression  $Q(\hat{R}, \hat{C})$  are as follows [192]:

$$P(\hat{R}, \hat{C}) = \begin{bmatrix} .4 & 0 \\ 0 & .6 \end{bmatrix}, Q(\hat{R}, \hat{C}) = \begin{bmatrix} .1 & .1 & 0 & 0 \\ .1 & .1 & 0 & 0 \\ 0 & 0 & .133 & .267 \\ 0 & 0 & .067 & .133 \end{bmatrix}$$

Each entry in the approximation matrix  $Q(\hat{R}, \hat{C})$  is calculated as follows:

$$Q(r, c) = P(\hat{r}, \hat{c}) \times \frac{p_R(r)}{p_{\hat{R}}(\hat{r})} \times \frac{p_C(c)}{p_{\hat{C}}(\hat{c})} \quad (5.3)$$

where  $p_R, p_{\hat{R}}, p_C, p_{\hat{C}}$  are the marginal probabilities of  $R, \hat{R}, C, \hat{C}$  respectively. For example,  $Q(3,4) = .6 \times (.4)/(.6) \times (.4)/(.6) = 0.267$ . Thus, the compression loss  $D(\hat{R}, \hat{C})$  can be expressed by the difference  $D(\hat{R}, \hat{C}) = \|E(R, C) - Q(R, C)\|$  for some matrix norm  $\|\cdot\|$ . In our implementation we use Frobenius norm as the matrix

norm.

### 5.2.6 The Melody Algorithm

We now present our MELODY (**M**achin**E** Learning **M****O****D**el Summar**Y**) algorithm. In the previous section we have created our goal to find the row and column clusters that minimize the cost function in Equation 5.2 among all possible number of clusters and all possible rows and columns combinations. Yet, the equation itself does not tell us how to reach the solution efficiently. Since the matrix can be considered as a graph where each entry is a weighted edge between a row node and a column node, we can use graph summarization [62] approach to provide a baseline solution (Algorithm 7). The overall idea is as follows:

1. Each row and column starts in its own cluster. Then, we put the row and column clusters into two separate lists (line 1-2).
2. We first fix the column cluster assignment. For the row clusters in the list, we randomly select one from the list (line 5).
3. We compare the selected row cluster with the remaining row clusters in the list as merge candidates (line 7-12): we try merging the selected cluster with each remained cluster and calculate the cost reduction by Equation 5.2 (line 8). We choose the candidate that produces the least cost.
4. If merging the selected cluster and its best candidate reduces the total cost, then we merge two clusters in the list (line 14-15). Otherwise, we remove the selected cluster in the list (line 17). Either way, the list will have one fewer item.
5. We repeat steps 2-4, but we fix the row clusters and merge the column clusters instead. The whole algorithm stops until there are no clusters remained in both lists.

Overall, in every iteration, a row (column) needs to measure the cost reductions with the remaining candidates in the list, which has the maximum size of  $\|R\|$  ( $\|C\|\$ ). Therefore, the time complexity of the basic algorithm is  $O(\|R\|^2 + \|C\|^2)$ . As a quadratic algorithm is infeasible for any moderately sized data for exploratory

**ALGORITHM 7:** MELODY (MachinE Learning MOdel SummarY)

```

Input :  $R, C$ - instances and explanatory features
         $\beta_R, \beta_C$  regularization terms
Output :  $\hat{R}, \hat{C}$ - row and column clusters

1  $R \leftarrow [\{r_1\}, \{r_2\}, \dots, \{r_m\}], \hat{R} \leftarrow \{\}$                                 /* initialize rows */
2  $C \leftarrow [\{c_1\}, \{c_2\}, \dots, \{c_n\}], \hat{C} \leftarrow \{\}$                             /* initialize columns */
3  $loss \leftarrow D(R, C)$                                                                /* initialize loss function */

4 while  $size(R) > 0$  and  $size(C) > 0$  do
5    $r_0 \leftarrow random\_pop(R)$                                                  /* randomly extract a cluster */
6    $\Delta L_{max} \leftarrow 0, r_{max} \leftarrow undefined$ 
7   for  $r \leftarrow R$  do
8      $\Delta L \leftarrow \beta_R - D(\{r \cup r_0\}, C \cup \hat{C})$ 
9     if  $\Delta L > \Delta L_{max}$  then
10       $\Delta L_{max} \leftarrow \Delta L, r_{max} \leftarrow r$ 
11    end
12  end
13  if  $\Delta L_{max} > 0$  then
14     $r_{max} \leftarrow \{r_{max} \cup r_0\}$                                          /* merge two clusters */
15  else
16     $\hat{R}.push(r_0)$                                                        /* push the cluster to final result */
17  end
18 end /* same procedure as for C... */                                         */

```

---

**ALGORITHM 8:** Top-k Nearest Neighbor Query

---

```

/* Initialize  $T_R \leftarrow build\_lsh\_table(R)$  and  $T_C \leftarrow build\_lsh\_table(C)$  after line 2 in Algo. 7 */
/* Replace R with  $query(r, R, T_R, k)$  in line 7 of Algo. 7 */
Input :  $v$  – query cluster
         $V, T_v$  – remaining clusters and LSH table
         $k$  – number of neighbors
Output :  $knn$ – top k nearest neighbors
1  $counter \leftarrow Counter()$                                 /* initialize counter */
2  $neighbors \leftarrow query\_lsh\_table(v, T_v)$            /* get collided entries */
3 for  $n \leftarrow neighbors$  do
4   for  $\bar{v}$  in  $V$  do
5     if  $n$  in  $\bar{v}$  then /* collision between the clusters */ 
6        $counter[\bar{v}] += 1/|\bar{v}|$ 
7       break
8     end
9   end
10 end
11  $knn \leftarrow counter.most\_common(k)$ 

```

---

visual analysis, we now propose a speed-up strategy to make our algorithm suitable for interactive performance.

#### 5.2.6.1 Speed Up Strategies With Data Sketches

While a randomized bottom-up algorithm scales linearly, Algorithm 7 is time-consuming as it needs an extra loop to compare all possible row or column clusters (line 7) in every iteration. However, if we look at the example matrix in Section 5.2.5, it is obvious that the first two rows (columns) are completely different from the last two rows (columns). Comparing candidates that are different indeed is of no use since they are unlikely to reduce the total cost. Thus, to speed up the algorithm, we propose a k-nearest neighbor query strategy with a novel use of locality sensitive hashing (LSH) [193] scheme to encode a row of column clusters. LSH defines a family of hash functions (i.e., sketches)  $[h_1(v_i), h_2(v_i), \dots, h_n(v_i)]$  for a vector  $v_i$  so that the probability of hash collisions between two vectors is proportional to their euclidean distances (i.e.,  $sim(v_i, v_j) \sim Pr[h_k(v_i) = h_k(v_j)]$ ). Vectors with similar values thus can be stored in the same buckets in an LSH table. Furthermore, we can extend this proportional to retrieve similar row (column) clusters. If two clusters have many similar vectors, then the number of hash collisions will be high. Therefore, the top-k clusters from the query will likely be similar neighbors.

The query algorithm is illustrated in Algorithm 8. First, an LSH table needs

to be built for rows and columns, respectively. Then, when a neighbor query is performed, we can use the hash keys from the query’s vectors to perform a table look-up to retrieve all the collided entries with the entries in the cluster ( subroutine *query\_lsh\_table* in line 2). We count the average number of collisions between the entries from the query cluster and the ones from the candidate clusters (line 5) and return the top k clusters with the highest number. This can drastically reduce the number of comparisons and the running time when the matrices are large (Section 5.4).

#### 5.2.6.2 Strategies Addressing Sparsity

Given a sparse explanation matrix, the bottom-up approach might face difficulties in cluster entries when the cost function is stuck a local minimum. Also, as the matrix is sparse, it is hard for the algorithm to know whether there are cluster structures at the beginning. These adversely affect the formation of significant clusters. To address this cold start problem, we reference from spectral graph partitioning [60] to create relatively smaller partitions of rows and columns using their singular vectors from SVD decomposition. Then, we can use our information-theoretic objective function to compress the matrix further.

#### 5.2.6.3 Interactive Clustering For Visual Analytics

Since the clustering result is based on the best outcome from the raw explanation matrix, the compactness and quality might be affected by the size and noise in the matrix. The noise here refers to corner cases of inputs where the model needs to use rare logics to make decisions on these inputs. Typically, to increase the visibility and reduce the visual complexity of a matrix, users are allowed to filter the sparse regions by removing the values by some thresholds like sizes of clusters or the densities of the clusters. In our case of understanding the decision logics of a machine learning model, the objective of visual analytics is to identify more general patterns of decision logics by removing complex decision paths. Therefore, we can run Algorithm 7 again on top of the current co-clustering result to further merge the clusters. The result of such interactive clustering can be illustrated in a real dataset in Figure 5.4. It is clear that without re-running the algorithm (Figure 5.4(a)), the similar patterns are not grouped together to convey an accurate

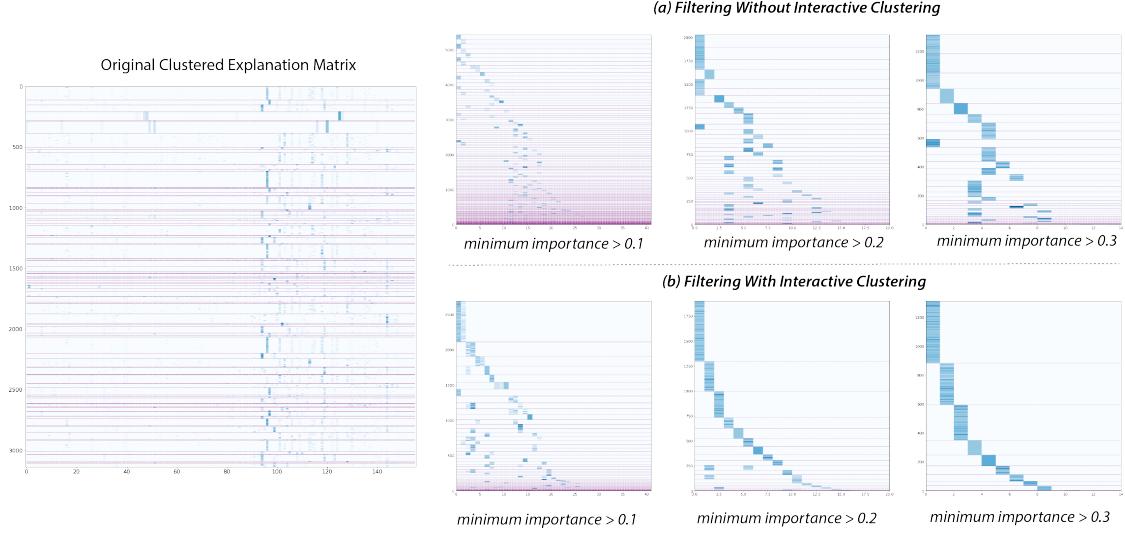


Figure 5.4: The result obtained by interactively clustering (purple lines showing only the row clusters for visual clarity) the explanation matrix upon visual analytics operations such as filtering on a real dataset. (a) Without interactive clustering, the similar patterns do not group together to provide the accurate sizes of each logical expressions. (b) By interactively clustering, the visual result conveys a much clearer and accurate information of the knowledge inside the model comparing with (a) with same information after filtering.

support of instances that belong to the same logic (Figure 5.4(b)), which defies the purpose of filtering the result.

### 5.3 Visual Analytics System: Melody Matrix

We now present the visual design to visualize the explanation matrix summary to help users understand the feature importance explanations from the machine learning models on the whole dataset. In the following discussion, we mainly describe the matrix visualization, MELODY MATRIX, and the interactions supported to facilitate analysis.

#### 5.3.1 Visualization Design

MELODY MATRIX mainly consists of three visual components: the *logics* view, the *facts* view, and the *instances* view. The logics view displays all the logics from the machine learning model that have been applied to the input dataset. The

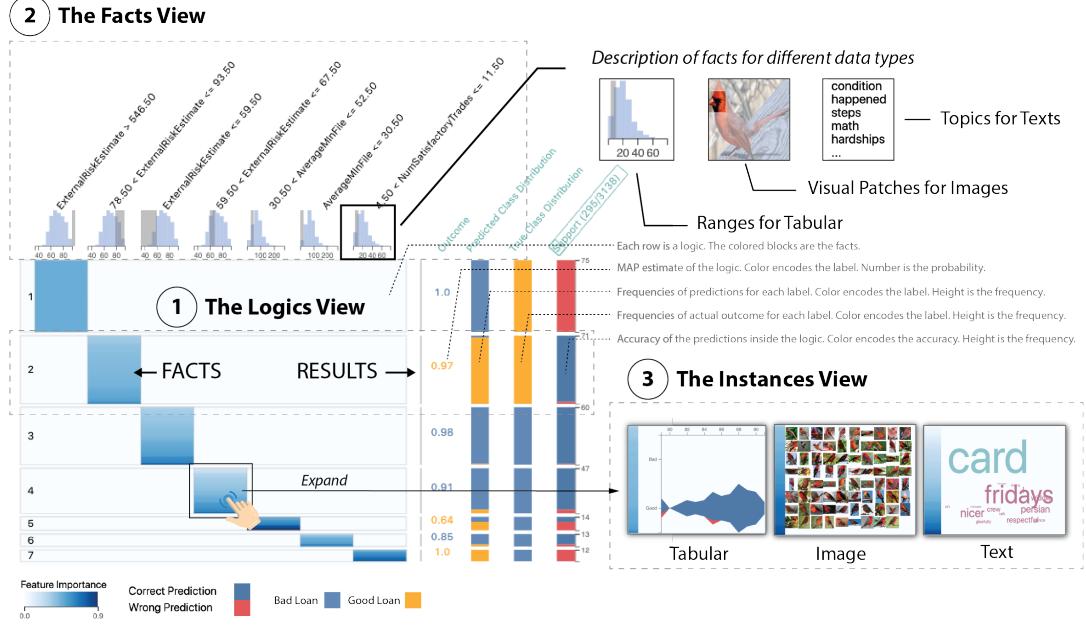


Figure 5.5: Main visualization of MELODY MATRIX. It consists of three visual components: (1) The Logics View displaying the row clusters obtained from the explanation matrix as rows with facts (i.e., explanatory features) as highlighted blocks and consequent statistics showing the right-hand side of each row. (2) The Facts View showing facts used among the logics as columns with feature description visualization. (3) The Instances View showing the original data information intersecting with the selected fact when clicked.

facts view describes what facts do the logics contain. The instances view provides the details of each fact in the logics. We provide slightly customized designs that convey similar meanings for tabular, image, and text data.

### 5.3.1.1 The Logics View

As illustrated in Figure 5.1, each row cluster is a logic applied to the input dataset from the machine learning model. The logics are obtained from the row partition results from the explanation matrix summarization discussed in Section 5.2.5. Thus, each logic is encoded as a row in the visualization (Figure 5.5(1)). Each row's height represents the support (i.e., the number of inputs from the dataset that falls into this logic). In each row, the highlighted positions represent the facts (i.e., columns) that exist in the logic. The facts are sorted according to the column partition results from summarization. The highlight of each fact in logic is a distribution of feature importance values across all logic

inputs. The values are sorted in increasing order and color-coded with a sequential scale. Overall, users can treat the combination of strongly highlighted blocks in the rows as a set of conjunctive (AND) conditions in the logic.

On the other hand, the consequent (RESULT) of the logics is displayed on the rows' right-hand side. First, the predicted outcomes and actual outcome of the inputs inside a logic can be perceived as probability distributions (i.e.,  $P(y|x)$  where  $y$  is the outcome, and  $x$  is the logic). Then, we can display the following properties:

1. The most probable prediction, which is the Maximum a posteriori estimation (MAP) of the logic. It is shown as a colored number where the color represents the output label and number showing the label's probability. When the number of output labels increases, this allows users to quickly identify the result of each logic (**Q2**).
2. The probability distributions of predictions and actual outcomes. They are shown as two individual vertical bar charts with color representing the labels and height representing the inputs' frequencies having the predicted and actual labels, respectively. It allows users to understand whether logic in general targets different classes, or it directly targets a specific outcome. It lets users understand whether there is a similar behavior between different classes as well (**Q1**).
3. The accuracy of the inputs' predictions inside the logic as a vertical bar chart showing the frequencies of correct predictions. With these numbers, the user can quickly identify the output labels by the colors and learn the model's accuracy within the similarly predicted inputs. It helps users identify the error regions of the model quickly (**Q4**).

**Design Considerations.** We consider different strategies to encode the facts inside the logics. Each fact in logic is a list of important values of each input regarding a fact (i.e., a column) in the logic (i.e., row cluster). The most important insight users should obtain from this list of values is that *is this fact important in the logic so that I can treat it as part of the rationale in the model?* (**Q1**) Therefore, we consider four encoding strategies: (1) sort and render each importance value in the list as pixel lines; (2) similarly render the values without sorting, (3) using an area chart to encode the values with sorting and (4) using an area chart without sorting. A pixel line represents each feature importance value as a line, or a small

	Low Support (i.e. Small Size)		High Support (i.e. Large Size)	
	Skewed Distribution	Unskewed Distribution	Skewed Distribution	Unskewed Distribution
<b>Pixel Line (With Sorting)</b>				
Pixel Line (Without Sorting)				
Area Chart (With Sorting)				
Area Chart (Without Sorting)				

Figure 5.6: Different visual encoding strategies to encode the facts (i.e. co-clusters) inside the logics of the matrix visualization. Our choice is the pixel line encoding with sorting the values inside the facts.

rectangle, with a sequential color encoding the value. To dive deeper into the possible visual outcomes, we can observe the outcomes under the combinations of two situations where the feature importance values inside a fact are skewed or not, and the support of the logic is small or large. Under these combinations, the visual representation of facts with different strategies can be seen in Figure 5.6. First, we discover that sorting is important to detect the sparsity of the chart. The sparsity of a fact lets the user indicate their certainty when formulating a logic within a row. Whether the user thinks the fact is important or not correlates with the accurate judgment of sparsity (**Q1**). Secondly, we find a pixel line chart more stimulating when the size of the logic is small. It is because the pixel line always occupies the full length in the chart. Therefore, it highlights the important features more easily in niche logic, which might be interesting special cases in the dataset.

### 5.3.1.2 The Facts View

As the columns of the explanation matrix are a set of facts that explain the model’s decision rationale to the users, these facts can be described visually or textually as well. Therefore, since they are displayed in fixed horizontal positions, the descriptions can be displayed on the columns of the matrix visualization (Figure 5.5(2)). Here, we describe different strategies to show this information on tabular, image, and text data. We can show the facts as a range that covers a data distribution of the range’s represented feature for tabular data. For example,

a fact  $0 < x < 5$  can be visualized as a gray box covering the interval between zero and five on the histogram of feature  $x$ . We can show the image patches that described the important (i.e. highly activated) visual representation learned from the model with a bounding box for image data. For text, we can show the top most important words in each topic. As a result, users can quickly understand each fact’s properties in logic, such as tightness of the interval, the color of the patches, and representative words (**Q1**).

### 5.3.1.3 The Instances View

When users click on a colored block (i.e., fact) in logic, the data that corresponds to the fact and logic are displayed inside the block (Figure 5.5(3)). For tabular data, we display the distribution of values across each output label with a stream plot with colors showing the distribution of data with different accuracies. For image data, we show the images of the logic sorted by the feature importance of the fact selected. For text data, we display the word weighted by feature importances as a word cloud with color indicating the output labels. While there are many ways to describe the original data, the key objective of the instances view is to keep the information displayed in the matrix visualization so that users can keep track of the matrix’s visual exploration. Within the same layout, users can inspect the data, logics, outcomes, and accuracies with the same alignment, which leverages the connections provided by matrix visualization when visualizing a large number of facts and logics [194].

## 5.3.2 Interactions

MELODY MATRIX supports four types of interactions: (1) interactive clustering to further increase the compactness of logics with user inputs; (2) filtering the matrix to reduce the cognitive burdens when visually exploring the large matrix; (3) querying the data to focus on specific goals after acquiring the overview of the logics, and; (4) details on demand.

### 5.3.2.1 Interactive Clustering

Summarizing the original explanation matrix might result in small pieces of logics illustrated in Figure 5.4. If the logic is too specific, users realize that the outcomes are similar for logics that share similar facts. They can remove the unimportant features from each input to consolidate the logic. In our implementation, we provide two types of filter: *filter by feature importance* and *filter by class*. Users can remove the input by class or remove the values inside the input by a minimum value threshold. Both results will lead to a reduction of values inside the explanation matrix. When Algorithm 7 is rerun, the co-clustering result will become more compact and provides better clarity of the matrix visualization. Also, it allows the user to focus on one output label only to investigate prediction logics for a particular output (**Q3**)

### 5.3.2.2 Filtering the Logics

Filtering the logics (i.e., rows) is useful when scalability issues exist when the number of logics is too high. For a complex model, it will easily happen. We provide two types of filter for this situation: *filter by support* and *filter by accuracy*. The first is to remove niche logics that only apply to a few inputs, and the second is to focus on error regions of the model by focusing on logics with low accuracies (**Q4**).

### 5.3.2.3 Querying the Data

Given a summarization result, users might want to focus on logics that are related to specific labels or facts. Thus, we provide two types of queries: *query by classes* or *query by facts*. The query will return logics that contain the selections, and the selected facts will be highlighted. These allow users to verify logics that are consistent with their domain knowledge (**Q3**).

### 5.3.2.4 Details on Demand

To provide a clear and effective visualization, we provide the details of each visual component only on the user's demand. The textual information of the logics, facts, labels, and accuracy can be obtained by hovering over the respective visual

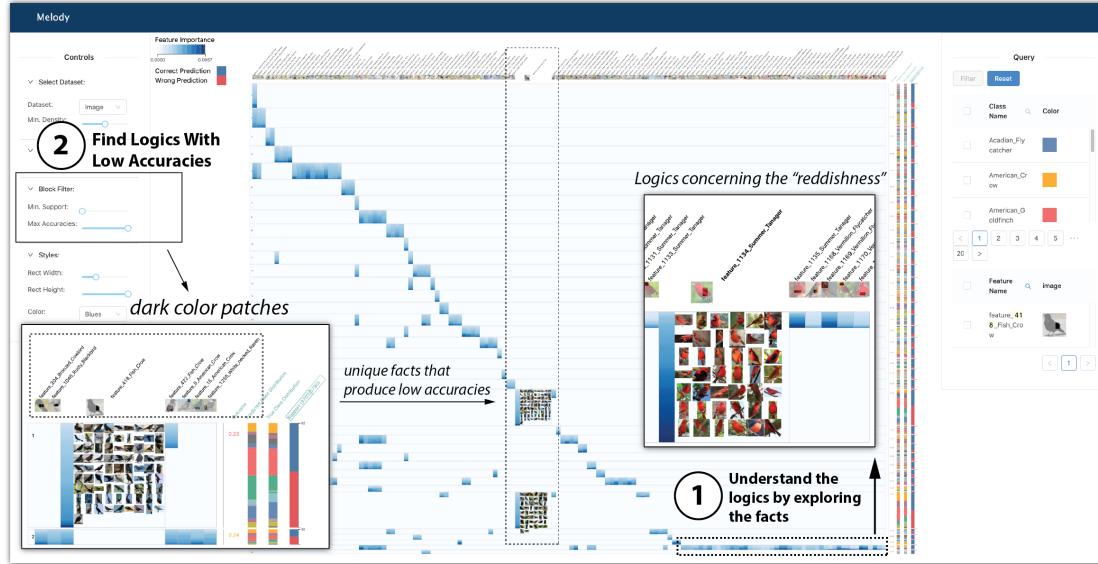


Figure 5.7: A matrix visualization summarizing the feature importance explanations obtained from a deep neural network that classifies birds’ images into 200 species. Each row represents a logic of a classification rationale and each column is a fact that explains the classification. (1) Users can understand the predictions by checking the facts in a logic. For example, when checking the highlighted facts, users understand the “reddishness” features that are used by the model for making predictions. (2) Through different interactions, users can focus on a part of the matrix to understand the model. For example, by filtering the logics to find logics with low accuracies, the matrix shows that the logics that are related to dark color patches are likely to produce wrong predictions.

elements. Also, the instances view (Figure 5.5(3)) are available only when the users click on the facts from the columns or the highlighted blocks.

## 5.4 Case Study

To evaluate the usefulness of MELODY and visualization, we perform three use case scenarios on various datasets.

### 5.4.1 Datasets and Models

The implementations are written in NumPy, and the experiments are run in a MacBook Pro with 2.4 GHz 8-Core Intel Core i9 CPUs and 32GB RAM. We use

Algorithm	Tabular	Image	Text
Baseline	33 mins	21 mins	> 7 hours
Baseline + LSH	<b>5s</b>	<b>13s</b>	<b>9s</b>

Table 5.1: Run time on datasets used in the case studies.

the following real-world datasets and ML models to conduct our experiments and use cases:

**Caltech-UCSD Birds-200-2011 Images.** The dataset includes 11,788 images with 200 species of birds. We use a Convolutional Neural Network (CNN) with a prototype layer [150] and achieves the highest test accuracy of 73.63%. The explanation matrix is extracted from the prototype layer, which has 1330 facts related to visual patches.

**Home Equity Line of Credit (HELOC).** It contains binary classifications of risk performance (i.e., good or bad) on 10,459 samples with even class distributions. We train a random forest classifier and achieves the highest test accuracy of 72.85%. We extract 167 facts and use SHAP [147] to construct our explanation matrix.

**US Consumer Finance Complaints.** The dataset contains 22,200 documents with ten classes (e.g., debt, credit card, and mortgage). We train an LSTM neural network model and achieve the highest test accuracy of 84.54%. We use IntGrad [146] to generate explanations for words in each document. We further combine the words by clustering their embeddings to generate 100 topics as facts.

**Run time performance.** We report the effect of the run time on the three use cases with the speedup strategies (Algorithm 8) in Table 5.1. The result clearly shows that by replacing the quadratic computation in the baseline approach (Algorithm 7), it becomes possible to produce results in interactive time. We also observe that the calculation of information loss is not linear in runtime since there are lots of data slicing operations to compute the approximation matrix ( $q(\hat{R}, \hat{C})$ ). It highlights the importance of limiting the number of candidate comparisons in the bottom-up process.

#### 5.4.1.1 Usage Scenario: Understanding an Image Classifier

We first describe a hypothetical walkthrough of understanding what a deep learning model has learned from a set of images (Figure 5.7). We use images as

examples because the visual presentations are intuitive to understand. Imagine Chris, an ornithologist, wants to study how birds' appearances distinguish their species. He downloads the data and runs the ML model to know how the machine learns the visual features.

**Understand the logics.** Chris uses MELODY and simplifies the matrix using the interactive clustering filter. After some iterations, he generates an explanation summary consisted of 37 logics covering around 70% of inputs in the dataset. Chris then inspects the facts in the column of the matrix. The visual patches are sorted by the clustering result and convey important visual features learned by the model. He realizes that the neural network learns to group birds with similar colors by dragging along the column. For example, the reddish birds fall into the same logic that is described by facts describing the reddish features among the birds (Figure 5.7(1)) (**Q1**).

**Understanding the limitation of the logic.** Chris checks the logics that have low accuracies to understand the limits of the model. He achieves this by filtering the logics with high accuracies. As a result, there exist two logics that have lower accuracy than 50%. By checking the facts and the instances within the logics, he realizes that the birds' dark color patches are the culprit of such a prediction outcome. Thus, Chris learns that the classification logics that rely on the color channel are not that effective when the birds are dark (**Q4**).

#### 5.4.1.2 Tabular Use Case: Understanding the Data Capability

We now present a use case about approaching the limit of predictability in training a dataset. Understanding how the learned knowledge helps to make predictions allows the financial worker to make improvements to the current credit system.

**Understand the summary.** The analyst obtains a matrix summary (Figure 5.8(1)). It shows four rows that both have decent support and facts with great feature importances (**Q2**). By inspecting the facts that appeared among these rows and the predicted outcomes. The analyst finds out the first three are concerned about the ranges within the feature "external risk estimate" (**Q1**). By inspecting the ranges in the column, it is clear that for many inputs in the dataset, the model can correctly classify them by the values of external risk estimates. If

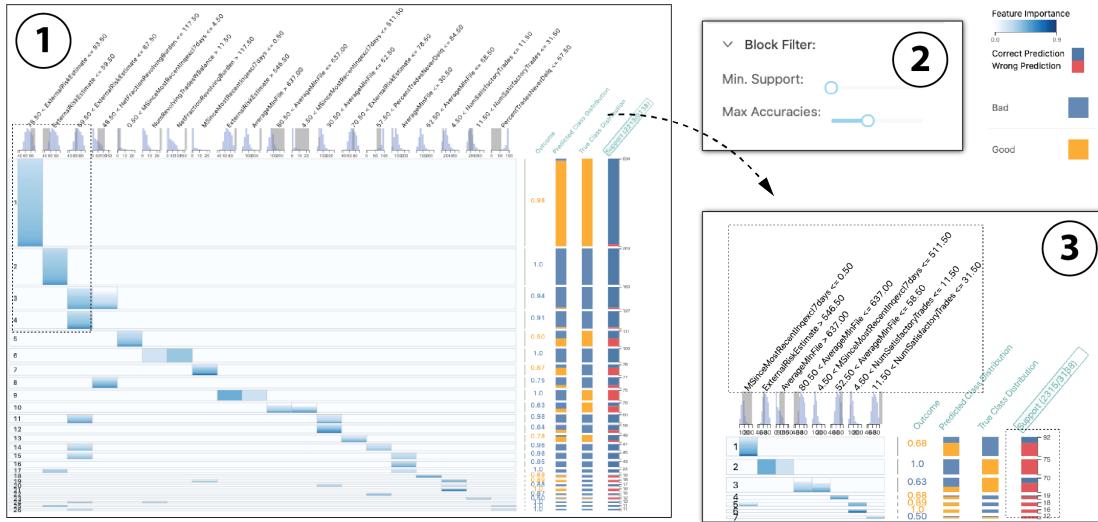


Figure 5.8: Use case of understanding a random forest classifier of credit risk classification trained on tabular data. (1) The most popular logics are displayed on the top left-hand side. The facts are mostly about the different ranges of a numeric feature (i.e., External Risks Estimates). (2) Filtering by accuracy allows the wrong logics to be emphasized. (3) By inspecting the facts from the inaccurate logics, it could be concluded that data quality issues are the main culprit of the misclassification.

the value is high, then the prediction would be “good” (i.e., yellow bar). Else, it would be “bad” (i.e., blue bar).

**Discovering the data quality issue.** By adjusting the matrix to screen out logics with high prediction accuracies (Figure 5.8(2)), the analyst discovers several logics that produce wrong predictions. When he inspects the facts among these logics, he finds extreme ranges among these facts (Figure 5.8(3)). The analyst remembers that he fills in the null values inside the data with a large number, so it means the null values play an essential role in the misclassification (**Q4**). To verify, he queries these ranges and discovers that these are the only logics that have such ranges in the model (**Q3**). Therefore, he concludes that the instances with null values do not contain an adequate rationale for the model to make predictions, in which he should discard them instead of filling in null values.

#### 5.4.1.3 Text Use Case: Predicting Customer Complaints

We present a use case of exploring a text classification model to understand different types of customer complaints. Understanding how customers complain can improve the call center's services. Our financial analyst first uses MELODY to acquire five logics and six facts. Then, he explores the logics and inspects the texts inside each fact in the logics. Overall, he sees that the words "card" and "money" dominates the facts learned from the model. It is understandable since customers usually enquire about their money on the bank services hotline and ask about the bank card for credit card issues. Interestingly, by inspecting the word frequencies in fact, the analyst observes some deviation of the use of words in the same logic. For example, there is the word "rush card" in the "card" topic for prepaid card services. Therefore, even though "card" is the keyword, it would add a "rush" beforehand if it is related to prepaid card services. The analyst concludes such a keyword predicts the prepaid card service inquiries when customers dial-in (**Q1**).

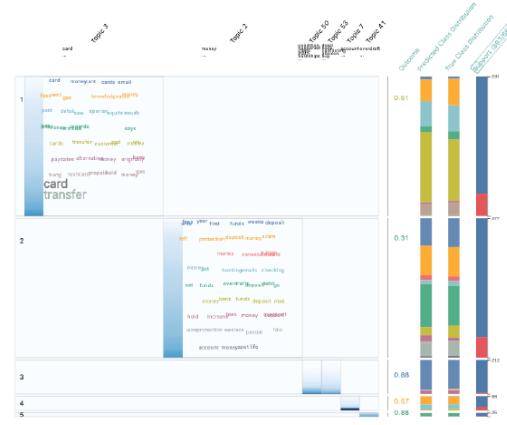


Figure 5.9: Explanation Summary of a text classifier.

## 5.5 User Study

We conduct a quantitative experiment to evaluate the effectiveness of MELODY MATRIX in helping users understand the behavior of the machine learning model. We aim at achieving two goals in the experiment:

1. We want to verify whether users can utilize matrix visualization and answer general machine learning tasks questions.
2. By observing how users use our tool to answer the questions, we distill a set of design lessons and limitations to suggest guidance in designing machine learning model explanation tools.

	Question	Task	Result
T1	Q2	Which of the following logic( <i>e.g. highlights on image_418 and image_315</i> ) has the highest support?	34/36
T2	Q1	Which of the following logics exists in the matrix?	35/36
T3	Q4	Which feature is likely to produce wrong predictions?	34/36
T4	Q3	With the following logic, what will likely to be the model's prediction?	33/36
T5	Q3	Which of the following features will produce prediction result of the following class label?	32/36

Table 5.2: Score of each task in the user study.

### 5.5.1 Participants

MELODY and MELODY MATRIX are designed for domain experts to understand machine learning models in an understandable human way. We recruited 18 participants (13 Male, 5 Female), aged between 21 to 42 (mean 28.1 with std. 5.09). All of them are graduate researchers from a large university with basic machine learning concepts (*e.g.*, knowing classification problems). The experiments were conducted through a 30-minute virtual interview session with screen sharing. We provided 20 USD amazon cards as gratitude upon completion.

### 5.5.2 Tasks

The study was divided into three steps. First, the interviewer went through a 10 minute tutorial with the participant while allowing them to explore the system freely. Then, they were asked to complete a list of tasks using MELODY MATRIX. During the tasks, they were encouraged to think and speak aloud their impressions. Last but not least, the participants were asked to complete seven questions regarding the system's usability in 7-point Likert-scale. We used the HELOC dataset in Section 5.4.1 as a demonstration during the tutorial. Then we use the Birds image neural network classifier in the formal study. The tasks are designed similar to understand a rule list model [174] which consist of answering five tasks regarding the design questions in the goals (**Q1-4**) (Table 5.2). Each question is a multiple choice of four with one correct answer, and two questions with the same formats were asked for each task (*i.e.*, 36 responses per task).

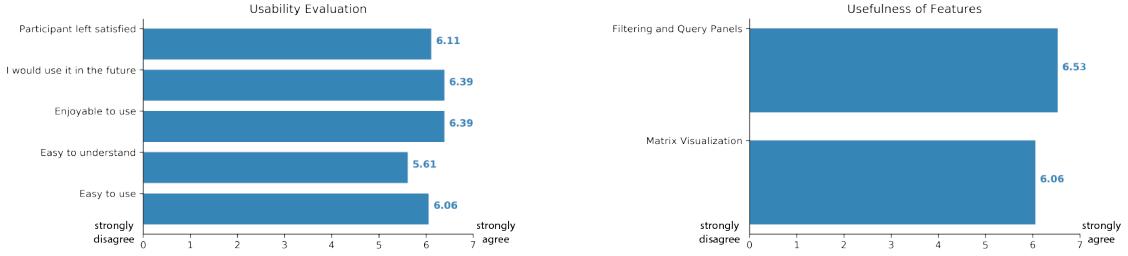


Figure 5.10: Average ratings from 18 participants regarding the usability of MELODY MATRIX.

### 5.5.3 Results and Design Lessons

The accuracies of the performed tasks are reported in Table 5.2. We did not record the exact task completion time. Table 5.2 shows that all participants performed the tasks correctly most of the time. For usability, the results are shown in Figure 5.10. All average Likert ratings were above six except “easy to understand”. Based on the feedback and participants’ observations when completing the tasks, we now describe the design lessons we learned when designing machine learning model explanation visualization.

**Visual complexity increases user interactions.** We observed a strong need for user querying and filtering to finish the task when the displayed matrix had many logics. For example, when asking users to answer which logic has the highest support, even though the logics were sorted in increasing support, almost all participants queried the logic listed in the multiple choices using the filter panel instead of inspecting the matrix directly. One participant explained this behavior, *“even though I can scroll to the top left corner and obtain the answer, I feel like I will get wrong by scrolling to the wrong region”*. Another responded, *“It is a neural network model. I know everything will be complex, so I plan to filter on anything I can before answering the questions.”* We believed that the observation suggested a strong need for visual analytics actions when solving tasks concerning complex logics. This observation also corroborated the fact that the filtering and querying panels received the highest usability score in Figure 5.10.

**Interpretable model explanations improve users’ confidence on visual results.** Interestingly, when asking participants to make predictions based on a set of facts, we observed an increased sophistication of rationale when answering the question. For example, many participants would take the color intensity of blocks

in the matrix as consideration and the prediction distribution of the logic at the same time. “*Because it directly shows what a neural network thinks, I’d think more deeply before giving any answer,*” one participant pointed out. Another participant gave an interesting opinion, “*Since everything displayed is truly what the neural network does and the matrix looks like having patterns inside, I am eager to look for directions to understand the model instead of just giving up and relying on tuning the parameters.*” Although the logics are inevitably complex, if it faithfully shows the model’s behavior, users are more likely to spend time using interactive visualization like MELODY MATRIX on model interpretation tasks.

**Counterfactuals of facts when evaluating the feature importance on a prediction.** We observed an interesting rationale for evaluating whether a fact is important to the logics that leads to a prediction result. The observation happened when we asked the participant to answer which logic is more likely to predict an outcome (T6). Some participants based their answers by the uniqueness of facts to the outcome like Figure 5.11.

This suggested a further need to visually summarize feature importance explanations like MELODY before using them to derive a logic that is only based on one input. To confirm whether a fact or a feature is the main rationale of an outcome, users may prefer inspecting whether the fact appears in other outcomes first.

**Intuitive meaning of explanations improves the rationale of answering further questions.** We observed that participants answered T4 very quickly (i.e., what will be the outcome of a logic). Some suggested that they started to get familiar with the birds’ images after answering the first three questions. Some of them responded, “I know the model sort of depends on the color to make predictions, so I don’t need to look at the blackbirds’ logics for a prediction of the red bird.” The ability to develop meaning on the explanatory feature could help to improve the judgment and efficiency of machine learning model interpretation, which highlights the importance of feature engineering on the explanation matrix in Section 5.2.4.

**Visual alignment improves searching in complex logic.** Some participants

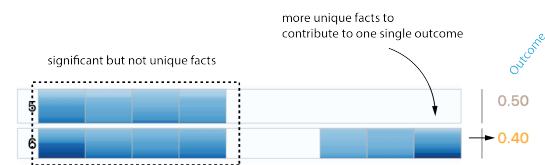


Figure 5.11: Evaluating feature importance by uniqueness.

recall the convenience of having the facts inside the logics aligned side by side inside the rows, which ease the burden of navigation. The alignment is the result of column partition from MELODY. We believe that when a logical model becomes more complex, clustering all the information that will be displayed (e.g., clustering inputs and important features in our case) is essential for scalable visual analysis.

# Chapter 6

## Topological Summary of Heterogeneous Model Explanations

Explainable Artificial Intelligence (XAI) is becoming increasingly popular, as more and more complex Machine Learning (ML) models are proposed in numerous industries such as healthcare, cybersecurity, and banking. While industries treasure the boost of performance provided by these predictive models, they also require these methods to provide clear explanations for the resulted predictions to cope with regulations such as GDPR [159]. There are many recent techniques providing local explanations to explain predictive models [148]. Given an input  $x$  and predictive model  $P$ , a local explanation method usually generates a real-valued feature vector  $e$  that represents the attributions of the input features to the predicted outcome. Since there are many local explanation methods available, it raises an important question – how to assess and compare these techniques? One popular method is Ablation Tests [195], which assesses the accuracy drops of an input by removing the important features identified by the explanation methods. However, this leads to a follow-up question—how to conduct the same assessments on the whole dataset?

In this paper, we consider a simple but useful global assessment of the local explanation techniques that take into account the entire dataset. We accomplish this through the analysis of the topological properties of the explanation space. Specifically, in this paper, we focus on explanation methods for binary classification.

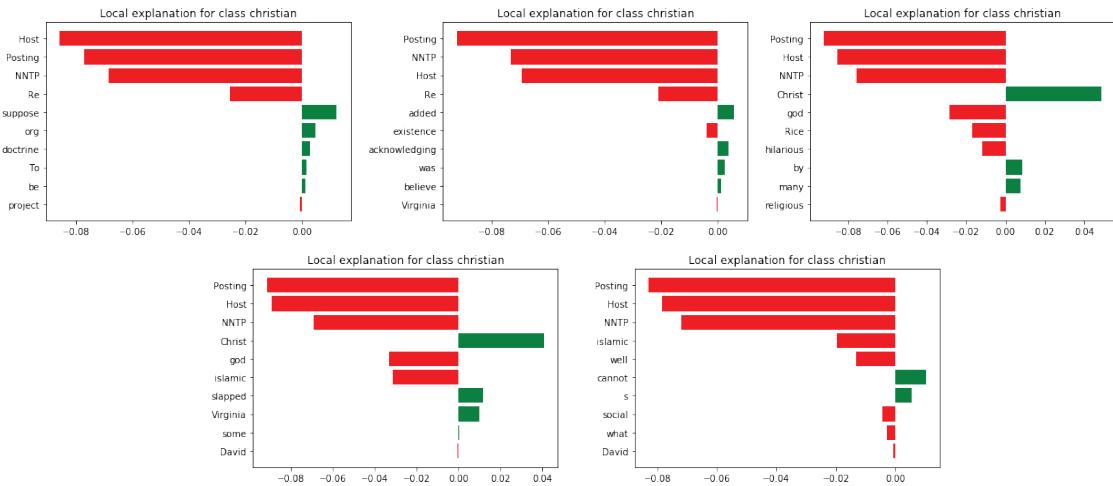


Figure 6.1: Five trials of LIME-generated explanations [148] with exactly the same parameters and inputs on sklearn’s 20 newsgroups dataset. The red bars indicate the words that contribute to the prediction of “Christian” news, and green bars indicate the words that contribute to the prediction of “theism” news. The lengths of the bars represent the feature importances.

We model a given explanation method as a scalar function to capture the relationship between the explanation space and the class prediction and compute a topological skeleton of this function. This skeleton is then used to compute a topological signature which is then used for comparing the explanation methods. Our technique is simple and easy to use and aims to address the following challenges arising from explanation technique assessments.

**Motivation 1: Explanation’s Stability..** Prior to comparing multiple local explanation methods, it is important to first understand a given explanation method, in particular, how it performs on any given dataset. Even using just a single explanation technique, it is common to observe different outcomes based on different parameters or simply different trials. For example, Figure 6.1 illustrates a scenario where an input’s explanation is different after simply executing the same commands multiple times. While it is easy to observe the differences between multiple local explanations through the bar charts, such analysis is only useful for analyzing the method for a single input. Such a visual approach becomes cumbersome when applied to the entire dataset. Moreover, it will be almost impossible to understand how stable the explanation method is for that data. Accomplishing this would necessitate generating a global measure or signature that

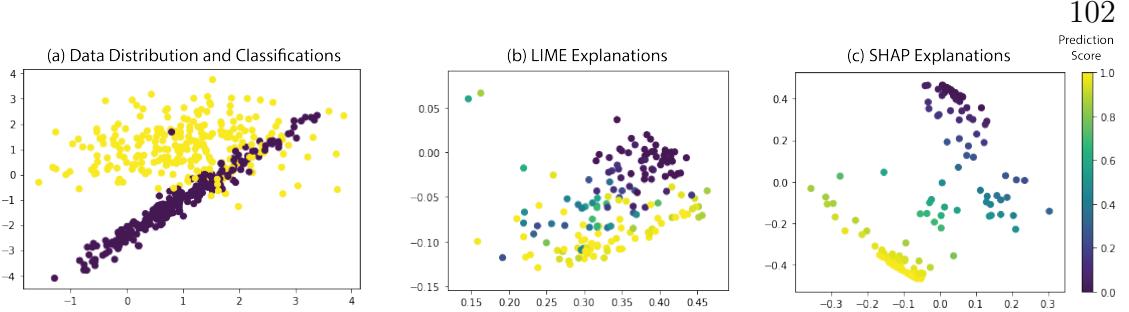


Figure 6.2: (a) A synthetic dataset with two features and two classes with its corresponding explanations generated by (b) LIME and (c) SHAP. For simplicity, both explanations are set with two features, which result in two different point clouds in two dimensions. The colors indicate the prediction scores of each point.

can capture the structure of the explanation space with respect to the given data. Our aim is to compute a topological signature corresponding to a given explanation method to accomplish this goal.

**Motivation 2: Comparisons Among Heterogeneous Explanations..** While comparing outcomes within the same explanation method can at least be done by measuring their distances (e.g., Euclidean distance) directly, explanations generated from different methods cannot be directly compared since they have different value ranges and dimensions. For example, in Figure 6.2, the LIME explanations and SHAP explanations have different ranges of values. Also, their values have different intrinsic meanings. To compare two different explanation methods as a whole, a straightforward way is to cluster the local explanations and compare the cluster similarities (e.g., Rand index) between them. However, there are two drawbacks to this approach. First, the values of explanations may not provide clear cluster structures (e.g., the points in Figure 6.2(b) appear together as a “blob”), making comparisons difficult to be effectively accomplished based only on the explanation values. Second, since explanations are inferred using the predictions of the ML model, the comparisons between different explanation methods are more faithfully measured by the relationships between the model predictions and the explanation values. For example, although the explanations in Figure 6.2(b) and Figure 6.2(c) have different value distributions, we can observe similar relationships between the values and the prediction scores (i.e., class 0 and 1 are roughly partitioned by points having prediction scores close to 0.5). As we show later, such properties are naturally captured by the shape, or more formally, the topological properties of the

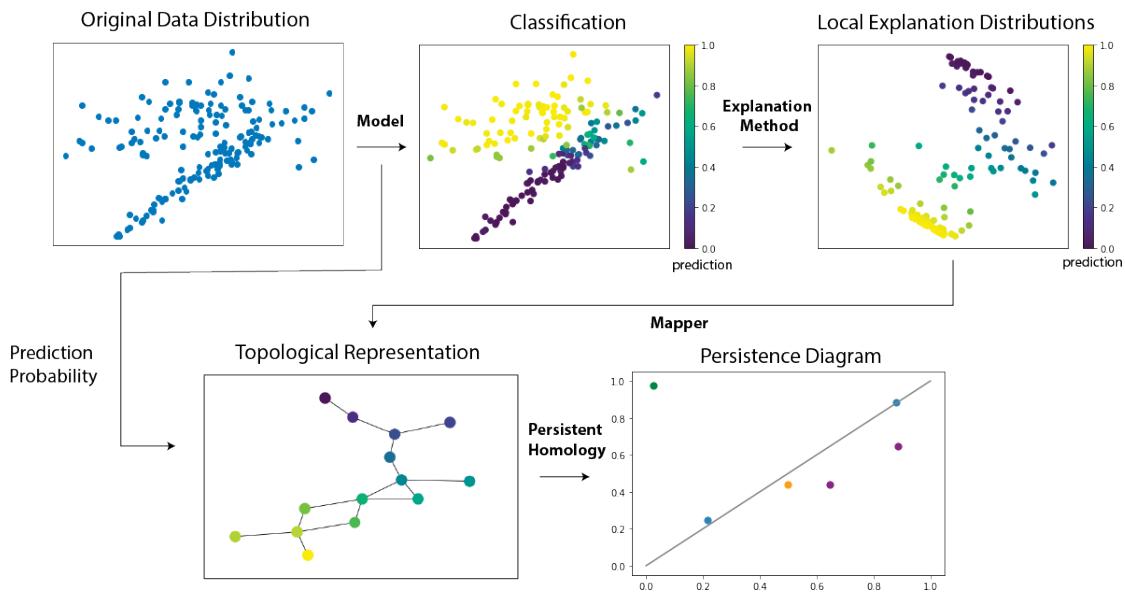


Figure 6.3: Pipeline of transforming the local explanations of a dataset in Figure 6.2(c) into a topological representation and signature for comparisons with other heterogeneous explanation methods.

explanation manifold. A crucial advantage of using topology for such an analysis is that it is agnostic to the geometric extent of the explanation space and thus allows direct comparison across different explanation methods.

**Approach and Contributions..** Our approach is illustrated in Figure 6.3. Given a dataset  $\mathcal{X} \subset \mathbb{R}^n$ , a binary classification model  $P : \mathcal{X} \rightarrow [0, 1]$  ( $P(x)$  denotes the probability that  $x \in \mathcal{X}$  belongs to class “1”), and an explanation method  $E : \mathcal{X} \rightarrow \mathbb{R}^d$ , our approach first models the relationship between the explanation approach and the classification model as a scalar function  $f : E(\mathcal{X}) \rightarrow [0, 1]$ . It then computes the topological skeleton of this function, in particular, an approximation of a *Reeb graph* [196], which is then used to compute a topological signature called the persistence diagram [197]. Analysis and comparison between the explanation methods are then accomplished by exploring the topological features as defined by the persistence diagrams and computing the distance between these topological signatures, respectively. To the best of our knowledge, our approach is the first to use computational topology for XAI.

The contributions of this work can be summarized as follows:

1. We propose a topology-based approach for assessing and comparing local expla-

nation techniques globally. By providing a domain-agnostic signature for the explanation techniques, our approach allows comparison across heterogeneous explanation approaches which is otherwise not possible.

2. We demonstrate the effectiveness of our approach through an extensive evaluation using synthetic as well as real datasets.

## 6.1 Related Work

To assess and compare the attributions from the local explanation methods, a straightforward way is to ablate the top K features ranked by the attributions and observe the decrease of the predicted output score. Varying the values of K and recording the output scores results in an ideally downslope curve. The lower the curve, the better the local explanation method since it shows that the explanation succeeds in identifying the important features. To avoid simply removing the top K features without considering the correlations among features, we can ablate the center of mass of the input instead [198]. Also, to avoid the issues of model extrapolation on the ablated inputs, we can retrain the models on the ablated data and measure the performance downgrade [195]. Furthermore, the local explanation methods can also be assessed by comparing their behavior between a randomly parameterized model and a trained model [199]. Besides ablation, we can measure the quality of explanations with metrics such as (in)fidelity and sensitivity under perturbations [200], or impact score that measures the feature importance on decision marking process [201]. If the apriori of feature importance of the dataset [202] is known, the feature importance of input across different models can also be assessed. In general, these techniques examine the classifiers' rationale on a single input, which motivates our work to propose a framework to examine the classifiers' behavior globally.

## 6.2 Topology Background

In this section, we briefly introduce the topological concepts on which our approach is built.

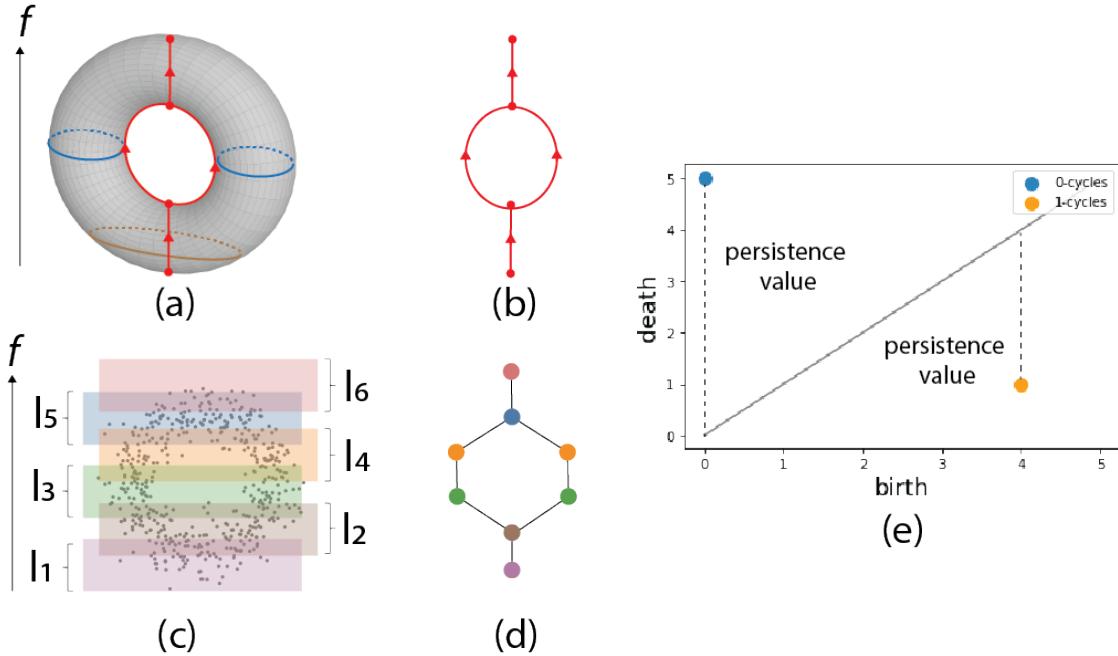


Figure 6.4: Approximate Reeb graph of a point cloud. (a) Height function defined on a torus. Level sets at two different heights are illustrated. (b) Reeb graph computed on the height function defined on the torus. (c) A point cloud sampled from the torus. (d) The approximate Reeb graph computed using the Mapper algorithm when the height function is divided into 5 intervals as shown in (c).

**Reeb graphs and the Mapper algorithm.** Consider a scalar function  $f : \mathbb{M} \rightarrow \mathbb{R}$ , that maps points from a manifold  $\mathbb{M}$  to a real value. The *level set*  $f^{-1}(a)$  at a given scalar value  $a$  is the set of all points that have the function value  $a$ . The *Reeb graph* [196] of  $f$  is computed by contracting to a single point each of the connected components of the level sets of  $f$ , resulting in a skeleton-like representation of the input. Figure 6.4(b) shows an example of the Reeb graph of the height function defined on a torus (Figure 6.4(a)).

A lot of real-world datasets are, however, available as sets of functions defined on a set of discrete high-dimensional points rather than as a continuous function. The *Mapper* algorithm [203] computes an approximation of Reeb graph of some user-defined function (often called *lens* or *filter* function) of such data. It basically divides the function range into a set of overlapping intervals and approximates the level sets to be the set of points that fall within each of these intervals. The connected components of these approximate level sets are then computed by clustering the

points that are part of a given interval. Each cluster then forms a node of the approximate Reeb graph, and an edge is present between two nodes if they share one or more input points. For example, consider the set of points in Figure 6.4(c) that are sampled from the torus in Figure 6.4(a). Assuming the height function, this is divided into a set of 5 intervals as shown in Figure 6.4(c). The connected components of the points that fall into these intervals are then used to generate the graph as shown in Figure 6.4(d).

In this work, we will be using the Mapper algorithm to compute the topological graph to understand and compare the space defined by the different explanation methods.

**Topological Persistence.** We now give an intuitive and informal description of the concepts about topological persistence. We refer the interested reader to the book by Edelsbrunner and Harer [197] for the formal definitions.

Given a scalar value  $a$ , the sublevel set  $f^{-1}((-\infty, a])$  is defined as the set of all points on the domain that have function value less than or equal to  $a$ . Consider a filtration of the input that sweeps the input scalar function  $f$  with increasing function values. As the function value increases, the topology of the sublevel sets changes at the *critical points* of the function (where its gradient is zero), and remains constant at other points. In particular, at a critical point, either a new topology is created, or some topology is destroyed. Here, topology is quantified by a class of  $k$ -dimensional cycles (or  $k$ -cycles). For example, a 0-dimensional cycle represents a connected component, a 1-dimensional cycle is a loop that represents a tunnel, and a 2-dimensional cycle bounds a void. A critical point is a creator if a new topology appears and a destroyer otherwise. Given a set of critical points  $c_1, c_2, \dots, c_m$ , one can pair up each creator  $c_i$  uniquely with a destroyer  $c_j$  which destroys the topology created at  $c_i$ . We say that a topological feature is born at  $c_j$  and it dies at  $c_j$ . The *topological persistence* [204] of this topological feature that is created at  $c_i$  is defined as  $f(c_j) - f(c_i)$ , which intuitively indicates the lifetime of this feature in this sweep.

The above notion of persistence allows features to have an “infinite” persistence, that is, there exist creators that are not paired with any destroyer. The notion of *extended persistence* [197, 205] extends the filtration to include a sweep over superlevel sets ( $f^{-1}([a, \infty))$ ), thus allowing pairing of the above mentioned creators.

Without loss of generality, for the remainder of this paper, we use the term persistence to mean extended persistence.

Since we are working with functions defined over discrete points, we use the graph computed by the Mapper algorithm to compute the topological persistence of the features of the input. Here, the filtration is defined on the nodes and edges of the graph as follows. Each node is assigned a function value equal to the mean of the function values of the clustered points represented by that node. The order of the nodes added during the filtration (or sweep) is defined by the function value of the nodes. An edge is added during the step of the filtration as soon as both its endpoint nodes are added. Note that the topological features, in this case, correspond to 0- and 1-cycles only.

**Persistence Diagram.** A persistence diagram plots the topological features as a 2-dimensional scatter plot. Each point in the plot corresponds to a single feature and has  $x$  and  $y$  coordinates equal to its birth and death values respectively obtained from the extended filtration. The persistence value of each figure is then the height of the corresponding point above (or below) the line  $x = y$ . Figure 6.4(e) shows the persistence diagram computed using the Mapper graph. Note that we plot all the features in the same plot, thus resulting in a single persistence diagram.

Persistence diagrams provide a useful mechanism to assess the structure of scalar functions. Moreover, it has also been shown that persistence diagrams are robust to noise [206]. In other words, persistence diagrams are stable under small irregular perturbations in the data, and the distance between two such diagrams is bounded. Hausdorff distance and bottleneck distance are two common measures used to compare two persistence diagrams. In this work, we use the *bottleneck distance* [206] for this comparison.

### 6.3 Topological Representation and Assessment of Local Explanation

Let  $\mathcal{X} \subset \mathbb{R}^n$  be a dataset and  $P : \mathcal{X} \rightarrow [0, 1]$  a binary classification model, where  $P(x) \in [0, 1]$  is the probability of  $x \in \mathcal{X}$  to belong to the “1” class. Given  $\mathcal{X}$  and the model  $P$ , a local explanation method can be seen as a mapping from the dataset to the explanation space  $E : \mathcal{X} \rightarrow \mathbb{R}^d$ , where  $E(x)$  provides the “importance” of the

different attributes for the classification of  $x \in \mathcal{X}$ . While  $d = n$  for most approaches,  $d$  can be greater than  $n$  if the explanation method outputs more than a scalar value for each attribute of the input.

The mapping  $\mathcal{X}' = E(\mathcal{X})$  gives rise to a point set in  $\mathbb{R}^d$  (the explanation manifold). We can define a function  $f : \mathcal{X}' \rightarrow [0, 1]$  as  $f(x') = P(x)$ , where  $x' = E(x), x \in \mathcal{X}$ . In this work, we use the function  $f$  as the lens function from which Mapper builds a summary representation for a given dataset and to compute the corresponding persistence diagram. Intuitively, this function captures the relationship between the explanations and the classification probabilities. Recall that our goal is to analyze and compare explanation methods. Since the explanation manifold (and space) can drastically vary across different methods and parameters, a direct comparison of the geometry of these manifolds is not possible. However, studying the topology of such functions allows us to analyze how different explanation methods are from a topological point of view, that is, we can make geometry agnostic comparisons. Furthermore, each topological feature (a point in the persistence diagram) can be easily mapped back to a set of input data points in  $\mathcal{X}$ , thus allowing us to also compare how the explanation space is spread across the input data.

**Mapper Parameters..** The Mapper algorithm used in this work requires two parameters: 1) the *resolution*  $r$  of the lens function that defines the number of intervals into which the scalar function range is divided; and 2) the *gain*  $g$ , which defines the percentage overlap between successive intervals. The value of these parameters determines the structure of the resultant graph, and hence the persistence diagram. In an ideal scenario (e.g., in a dense point set input), increasing the resolution and decreasing the overlap would result in the graph computed using the Mapper algorithm converging to the Reeb graph. However, in real-world data, such as the ones we are working with, having too high a resolution or too low a gain can result in the graph being just a set of disconnected nodes. Also, depending on the point distribution, small changes in the parameters could drastically change the resultant graph, and hence the persistence diagram (a trivial option would be to use a resolution of one, which however would simply be equal to clustering the points). We are therefore interested in identifying a set of parameters that results in a “stable” computation of these graphs but also having a relatively high resolution and low overlap.

To evaluate this stability, we use bootstrapping [207] to compute the confidence intervals for the generated graph in terms of the bottleneck distances. Consider a set of input explanation values  $E = \{e_1, e_2, \dots, e_n\}$  and the persistence diagram generated by its Mapper Graph  $D$ . For each iteration, we sample with replacement from  $E$  to construct  $E^* = \{e_1^*, e_2^*, \dots, e_n^*\}$  and compute the persistence diagram  $D^*$  for this input using the Mapper algorithm with random parameters. Then, we define the stability  $s$  as the bottleneck distance  $d_b(D, D^*)$ . The confidence region for  $s$ ,  $s_\alpha$ , can be computed as the quantity  $\hat{s}_\alpha$  defined by:

$$P(d_b(D, D^*) > \hat{s}_\alpha | E) = \alpha \quad (6.1)$$

where  $\alpha$  corresponds to the confidence interval.  $\hat{s}_\alpha$  can be estimated using Monte Carlo simulations. We compute this stability for varying resolution and gain values and choose an appropriate setting for the analysis. The Appendix illustrates this process for the experiments used in this paper.

In the following section, we show how our topological framework can be used to assess and compare different explanation methods.

## 6.4 Design Process For Visualization

In this section, we describe an iterative and user-centered design process with domain experts that results in the visualization shown in Figure 6.3. To be specific, we worked with a team of machine learning researchers in a financial institution for a year to study their needs and discuss various approaches with regards to model interpretability. To conceptualize the user needs, design rationale, and alternatives, we describe the whole journey of creating our topological summary visualization as a series of sequential considerations of the design factors in realizing a summary visualization [208]. The discussions we had with the domain experts can be categorized into four main areas: *task*, *data*, *method*, and *purpose*. First, we understand the tasks our experts want to achieve. These *tasks* are the needs for ML model interpretability from our experts and are not affected by the visualization and techniques. Then, we go through the available *data types* generated from the whole ML modeling pipeline and select the ones that are most suitable to the tasks.

Afterward, given the data and tasks, we can choose the data abstraction technique (*method*) that suits the domain tasks and problem characterization. Finally, once we have the data abstraction and tasks, we can identify and compare various visual encodings (*purpose*) to address the needs.

#### 6.4.1 Step 1: Tasks

To begin with, the ML experts require two clear motivations describe in Section 6, which are to compare the local explanations with different parameters and from different methods. The local explanations present the rationale of a classifier, and the experts would like to observe the differences among the rationale from different explanation outcomes. Therefore, in terms of data exploration, we can formulate the user needs as the following tasks:

- T.1 Compare different explanation results through the variations of the local explanations in the dataset.
- T.2 Unify the representations of different explanation techniques.
- T.3 Identify similar and dissimilar explanation results.

Although these tasks are derived from the domain-specific scenario, we can already develop several expectations for visualization. For high-level tasks, the tasks related to visual summarization can be categorized as high-level presentation intents, and visual discourses such as *associate*, *cluster*, *compare*, *distinguish*, and *rank* [209]. For model interpretations, the visual summary should provide the functionality for users to *compare* (T.1-2) and *distinguish* (T.3) different explanation outcomes. These directions help us to decide the preferred data, data abstractions, and visual encodings in the next steps.

#### 6.4.2 Step 2: Data Types

After we identify the tasks, the next step is to identify the useful data types from the ML modeling pipeline. The ingredients from the pipeline are as follows:

1. Original Data Distribution: raw tabular data providing all information of the input.

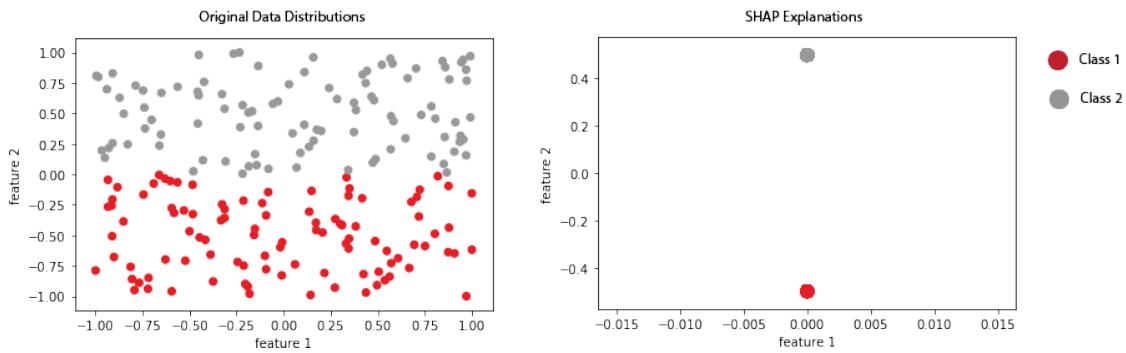


Figure 6.5: Synthetic data to determine the data inputs for visualization. For original data, there are chances that the variations of data variation of some dimensions are not used for predictions (i.e., feature 1). Showing these variations might hinder the users’ ability to understand the model’s rationale. For local explanations, the variations of not so useful features are significantly reduced.

2. Model Predictions:  $[0, 1]$  binary prediction score for each input indicating the likelihood of the predicted class.
3. Local Explanation: real-value feature vector indicating the attributions of each feature to result in the prediction from the classifier.

The core questions for determining the data for visualization is — Should we incorporate all information in the visualization? Which pieces of information are most applicable to the experts’ tasks? In order to answer these questions, we generate some simple synthetic data and present the outcomes for discussions (Figure 6.5). The synthetic data has two dimensions, and only one dimension is useful for the classification. If we compute the local explanations (i.e., SHAP) for the inputs in this dataset, we can see that there are no variations for the useless dimension (i.e., zero attributions for feature 1). Thus, for comparing explanation results (T.1), experts prefer the explanation space to original data space since the data variation reflects the model’s rationale (i.e., prediction probabilities), while the original data space might hinder the presentation of the model’s behavior when the number of dimensionality increases. At this stage, not only do we identify the domain-specific tasks but also the inputs for data abstractions and visualization.

For topological data summarization, the Mapper algorithm needs the user’s inputs on both the manifold and filter function, of which the choice will significantly affect the outcomes of the results. Thus, the argument above on the choice of data

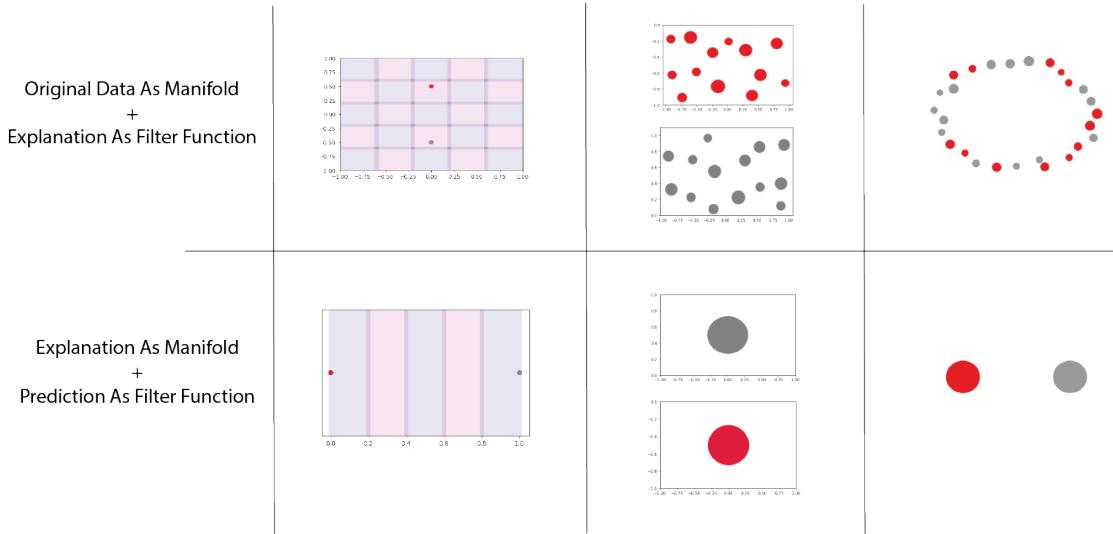


Figure 6.6: Demonstration of the choice of Mapper input with the synthetic data in Figure 6.5. Using explanation values or original data as the manifold will affect the variation on the final graph representation.

input will affect the Mapper result significantly as well. During our discussion with domain experts, we reflect the importance of the choice of data input with a demonstration of Mapper pipeline on the synthetic data (Figure ??). The filter function acts as a binning operation on the data, and clustering is applied to the binned data to reveal the nodes. Therefore, the variation of the manifold will significantly affect on the number of nodes in the final result due to the clustering algorithm. After understanding the usage of Mapper through the illustration, our experts comment that as it is important for the visualization to reflect the variation of the explanations (T.1), the choice of using explanation values as the manifold will suit better to the challenges.

### 6.4.3 Step 3: Data Abstraction

While this chapter is to present a topology-based framework to abstract the explanations from different explanation techniques, we also need to identify other possible alternatives to aggregate the explanations so as to justify the use of topology and consequently understand how to visually encode the topological representations to the users. Possible aggregation techniques include clustering and

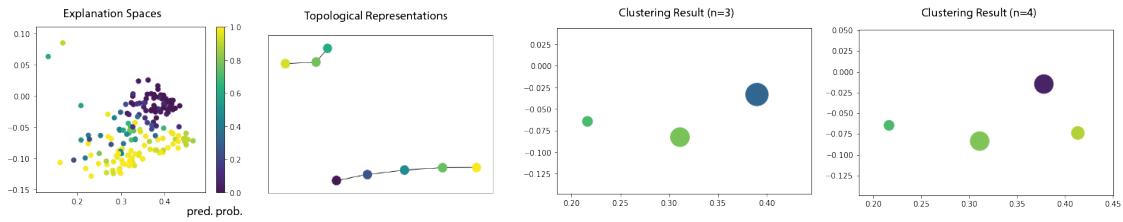


Figure 6.7: Design considerations for aggregating local explanations in the data abstraction level. The topological summary provides an additional partition constraint based on the prediction probability, which prevents the over aggregation of model’s classification results from clustering.

projections [210]. Apart from computation advantages that the topological method is consistent with explanation values with different ranges (T.2), the advantage of a topological representation also comes from its graph structure, which reveals the clear geometric structure that is based on the variety of the data. Clustering and projections are often combined together to generate summary visualization [211], but they in general favor similar instances over outliers. Also, the computations could not incorporate additional information (i.e., prediction probability), which is crucial to the model interpretation. To illustrate the challenges, we can visualize the topological representation result and clustering results with projections in the example LIME explanation in Figure 6.2. The main advantage of using the topology-based technique is the consideration of prediction probability in the aggregation process. Since clustering does not obtain the information of prediction probability in the process, inputs with different model’s decisions (i.e., classification results) are easily aggregated together, which results in the aggregation of prediction scores with high variances. If we use color hue to encode the model’s prediction as its rationale in the visualization, it will hinder much variety of the model’s behavior (T.1,3). Thus, topology-based summarization provides a more distinguishable model’s behavior due to its partition on the prediction values as well.

#### 6.4.4 Step 4: Visual Encodings

Once the topological summarization (method) is applied to the explanation outputs and prediction probabilities (data types) based on the tasks for explanation comparisons (tasks), the last step is to identify the presentation (purpose) of the topological graphs for the experts. We begin the usage of node-link diagrams

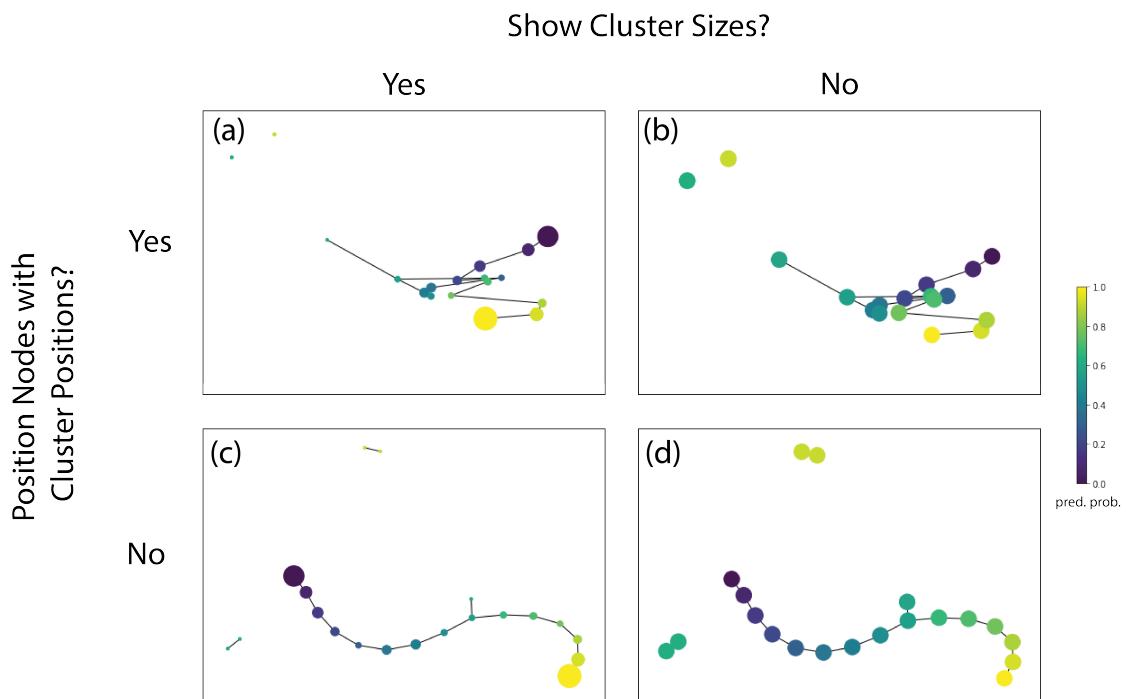


Figure 6.8: Design Alternatives for Topological Summary Visualization. (a) node-link diagram with layout determined by inputs' values and nodes encoded with sizes of clusters. (b) node-link diagram with layout determined by inputs' values and no encoding on nodes' sizes. (c) node-link diagram with layout determined by graph topology and nodes encoded with sizes of clusters. (d) node-link diagram with layout determined by graph topology and no encoding on nodes' sizes.

to encode the topological graphs since the experts are most familiar with such a visual representation. Also, the numbers of nodes and edges are limited due to the aggregation nature of the method. Thus, we do not discuss the usage of alternative graph visualization methods such as adjacency matrices. In node-link diagrams, we now discuss the layout for the nodes and also the encodings for the nodes.

For the layout algorithm, we can consider the layouts that are base on either the nodes' properties or the graphs' properties. For layout considering the nodes' properties (Figure 6.8(a)(b)), recall a node can be seen as a cluster of inputs, we can position the nodes based on the inputs' projections' coordinates. The advantage is that the explanations' proximities can be reflected, but such an advantage quickly diminishes when the dimensionality of the explanation increases. Also, as we can see in Figure 6.8(a)(b), by using the 2D values of the explanations (i.e., no projection technique is applied), the edges are already cluttered since topological graphs track the changes of prediction probabilities alongside the variations of the explanation spaces together. Thus, the edges that track the change of the model's rationale do not necessarily align with the variations of the explanation inputs. As a result, we use a graph-related layout algorithm ( force-directed layout), which takes the graph topology into considerations, and is more faithful to the topological representations.

Apart from the graph layout, the nodes in the node-link diagram can take different encodings (i.e., sizes and colors) into considerations. For color, we use the prediction probability to determine the color since it reflects the model's rationale and is the main consideration throughout our design process, from tasks to abstractions. The remaining encoding consideration is the size of nodes (Figure 6.8(c)(d)). Our rationale for the node's size is based on both the topological summary and tasks. For the topological representation, the nodes represent the critical points of tracking the prediction probabilities. Also, the Mapper computation also computes the set of nodes as overlapping clusters. Recall that the experts' tasks are to compare the model's rationale based on the variation of explanations in the dataset. Thus, we do not encode the nodes based on the clusters' sizes since the nodes are not disjoint (i.e., large nodes representing similar inputs can happen), and the experts are not interested in the redundancies in the dataset. Therefore, after the considerations of node sizes and layouts, we finalize with the visualization shown in Figure 6.8(d).

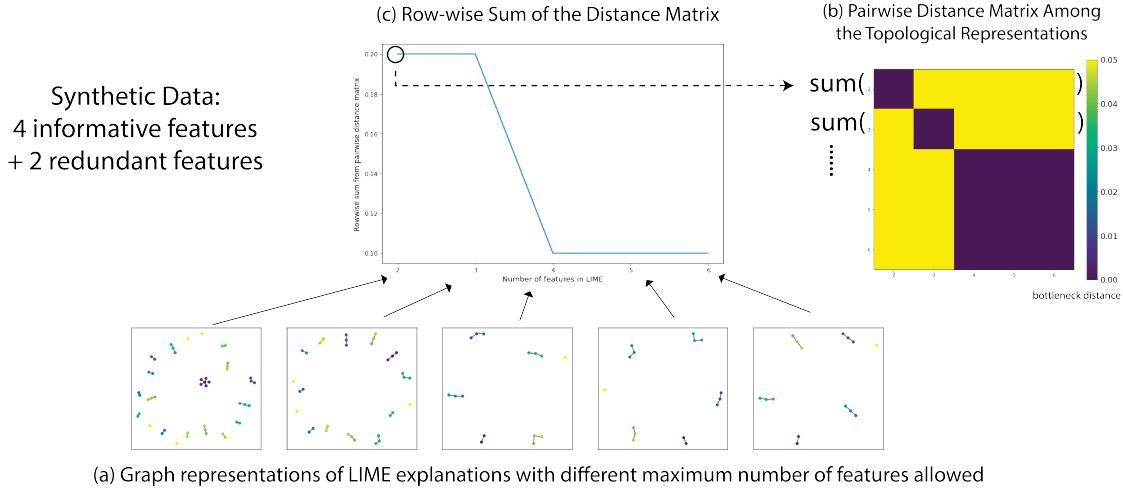


Figure 6.9: The topological representations and distance measurements of LIME explanations on a synthetic dataset with 4 useful features and 2 redundant features. (a) The graph representations of LIME explanations with different maximum number of features allowed in the regression. (b) The pairwise distance matrix among the topological representations. (c) The row wise sum of the distance matrix as the disagreement values among the explanation results.

## 6.5 Evaluation

In this section, we demonstrate the effectiveness of our approach to address the two challenges introduced in Section 6 (Motivations 1 and 2). For each of the challenges, our goal is to:

1. Demonstrate that our topological-based method captures the expected behavior of the explanation method under different settings. This is accomplished using synthetic datasets.
2. Provide usage scenarios demonstrating how our approach helps identify reasonable explanation methods and settings. This is accomplished using real-world datasets.

### 6.5.1 Experiment 1: Choosing Parameters in an Explanation Method

**Synthetic example.** In this experiment, we use synthetic data as ground truth to evaluate whether our approach correctly reflects the outcomes that is to be

expected from an explanation method’s parameter settings. We use LIME [148] as the example explanation technique for this purpose. One of the parameters in LIME is the *number of features*, which controls the maximum number of features allowed (i.e., sparsity) to train a linear classifier as the explanations. In theory, the ideal sparsity is the number of useful features that the classifier will use to predict the outcome. To test the effect of this parameter, we generate a synthetic dataset that contains 4 columns that are useful for predicting the outcome and 2 columns that are redundant [212]. We compute the LIME explanations on a Random Forest Classifier trained on these data, with varying number of features (varied from 2 to 6). This results in five different sets of explanation values. We first generate the topological skeleton using the Mapper algorithm as described earlier for each of these explanations (Figure 6.9(a)) and the corresponding persistence diagrams (topological signature). The pairwise distances between these persistence diagrams are then computed, which provides a way for quantifying the differences between the explanations (Figure 6.9(b)). It can be seen that these distances, when the number of features is smaller than 4, are different from when the number of features is greater than or equal to 4. Note that such an observation is consistent with the expected behavior of LIME—the regression needs 4 features to train a linear classifier. Once the maximum number of features becomes greater than or equal to 4, LIME’s behavior becomes consistent (i.e., resulting in small pairwise distances among the explanation outcomes under this parameter setting).

Additionally, we can further quantify the agreement and disagreement among different explanations as follows. We compute the disagreement between a given explanation with other explanations by summing up the pairwise bottleneck distances between that explanation and the others. This is essentially summing the row corresponding to the given explanation as shown in Figure 6.9(b). A low sum means that the pairwise distances are small, thus implying that the other explanations are similar to the given explanation. Similarly, a larger sum indicates disagreement with one or more explanations. Plotting these values as a line chart (Figure 6.9(c)) shows the trend that an increasing number of features leads to consistent explanation outcomes.

**Diabetes dataset..** We now demonstrate, using a real dataset, how our approach helps determine important parameter settings, such as the sampling rate, for SHAP

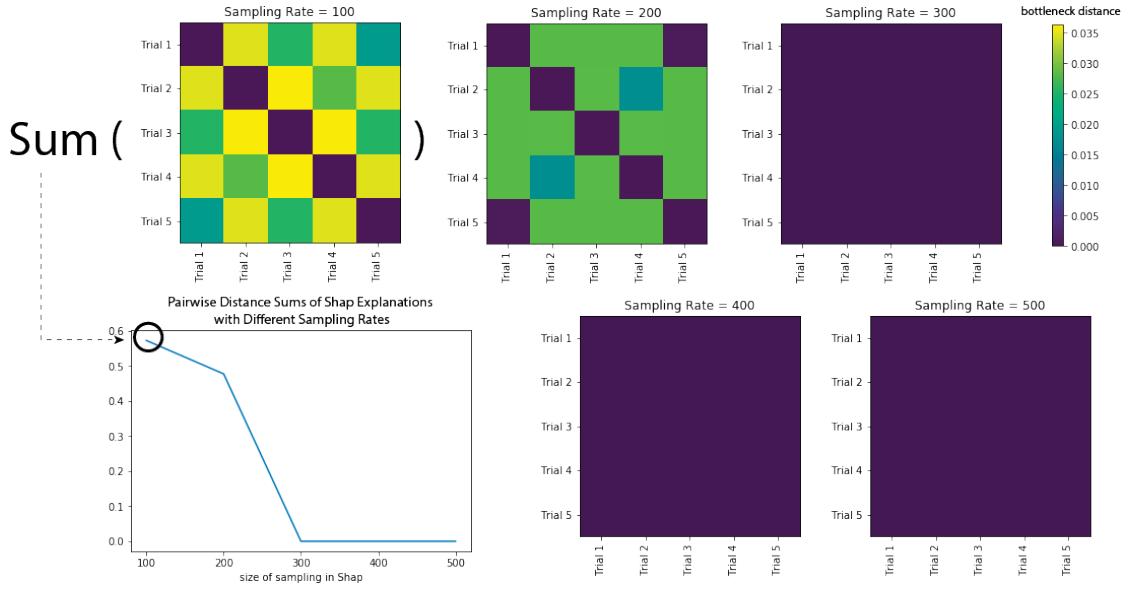


Figure 6.10: Using the bottleneck distances to measure the stability of SHAP explanation method under different sampling rates. We acquire the stability of each sampling rate by first running the explanation methods with a fixed sampling rate under 5 trials. Then we compute the pairwise distance matrix among the topological representations from the trials (i.e.,  $5 \times 5$  matrix). The stability can be reflected by summing up the whole matrix, with higher values meaning unstable outcomes. Plotting the values under different sampling rates reflects that SHAP explanations become consistent after having a sample size greater or equal to 300 (approx. half of the size of the dataset).

explanations [213]. Like other perturbation-based explanation methods, SHAP requires users to input the sampling rate for acquiring samples to train a local linear classifier. A poor sampling rate may lead to inconsistent explanation results like the example shown in Figure 6.1. However, having too large a sampling rate will be computationally expensive. We use the Pima Indian Diabetes dataset (PIDD) [214] for this experiment, which contains 614 inputs and 8 features, as an example. We train a Random Forest Classifier to predict the outcome and use perturbation based SHAP explanation method to acquire the explanation values. To understand the stability of explanation results at a dataset level, we perform multiple trials having a fixed sampling rate. We then compute the pairwise distances between the persistence diagrams corresponding to the explanations obtained from these trials (Figure 6.10).

A large sum of these pairwise distances implies that the explanation outcomes

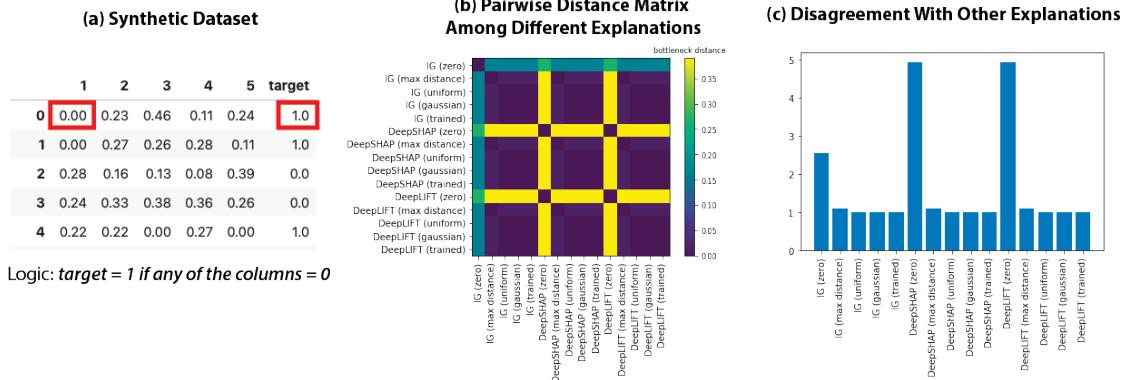


Figure 6.11: (a) Synthetic dataset with zero values being the decisive factor for predictions. (b) Pairwise distance matrix showing the significant bottleneck differences among explanation methods with zero baselines. (c) Highlighting the difference of the explanations among the zero baseline results with the disagreement measure (i.e. row wise sum of pairwise distances).

differ among multiple trials even when the parameter value is fixed. Figure 6.10 shows that the explanations become more consistent when the sampling rate increases to 300, which illustrates that stability are achieved with a higher sampling rate. While in this example, we obtain high stability (sum = 0) fairly quickly, this might not be the case when working on larger datasets. In such instances, one could trade-off the stability for other properties (e.g., limit on the sample size) and thus use the above approach to choose a parameter value that has as high stability as possible within the accepted limits.

### 6.5.2 Experiment 2: Comparing Different Explanation Methods

We now illustrate how our approach can be used to understand and compare different explanation methods. In particular, we aim to address a popular challenge in using local explanation techniques—what should the choice of baselines be for gradient based explanation methods. Baselines act as references to compare the relative importance of features in an input so that attributions can be calculated. We provide examples with both synthetic and real datasets to compare the behavior between different baselines. To be specific, we apply three explanation methods—Integrated Gradients [146], DeepLIFT [145], and SHAP [205] with five different baselines: (1) zero baseline (an input with all values being zeros), (2) maximum

distance baseline [215], (3) Gaussian baseline [216], (4) uniform baseline [215], and (5) a trained baseline [217], resulting in 15 explanation outputs for each experiment.

**Synthetic example.** Our goal in this experiment is to illustrate how our approach can be used to identify explanation methods that are expected to behave differently. To do so, we generate a synthetic dataset with 5 columns and determine the labels with an extremely simple logic—the input will be labeled as “1” if any of the columns contain a “zero” as value. Otherwise, they are labeled as “0” (Figure 6.11(a)). Under this setting, only zero values are important to the classification. Thus we expect the zero baseline is the only reference that is not a neutral input to the classifier. We train a neural network with three layers and achieve 100% test accuracy. We generate 15 topological representations for each of the explanation outputs. Note that we use the sigmoid function output as the prediction score for the lens function of the Mapper algorithm and hence to compute the persistence diagram. The pairwise distance matrix computed between the 15 persistence diagrams is shown in Figure 6.11(b). We can see that there are three rows and columns having much brighter colors (i.e. greater distances) than the others, meaning that these three explanation outcomes differ greatly as compared to the other explanations that result in quite similar signatures.

The three explanation outcomes are generated using all of the three methods used in the experiment applying the the zero baseline. Such observation is consistent with the fact that zero baselines produce different feature attributions by treating the important values as neutral references. Similar to Figure 6.10(b), we can compute the sum of pairwise distances between a given method and others to highlight the disagreement among the zero baseline results with the others (Figure 6.11(c)).

**Bank Marketing dataset.** We now repeat the above experiment using the Bank Marketing dataset [218] to compare the behavior of the different explanation methods. This dataset contains 45211 inputs with 20 columns (default, employment, occupations, etc., to name a few). Some columns have many zero values (Figure 6.12(a)), and it thus becomes important to understand whether these values affect the choices of baselines. We train a neural network with three hidden layers (87% test accuracy), and apply the 15 combinations of explanation methods and baselines to compare the explanation outcomes. We can observe results similar to the synthetic dataset scenario (Figure D.4 in Appendix), which shows that the zero

### (a) Banking Marketing Dataset

	age	job	marital	education	default	housing	loan
0	30	1.0	1.0	2.0	0.0	2.0	0.0
1	39	7.0	2.0	3.0	0.0	0.0	0.0
2	25	7.0	1.0	3.0	0.0	2.0	0.0
3	38	7.0	1.0	2.0	0.0	1.0	1.0
4	47	0.0	1.0	6.0	0.0	2.0	0.0

Criteria of choosing baselines for explanation methods  
- Are these zero values important to prediction?

### (b) Topological Representations of Different Explanations

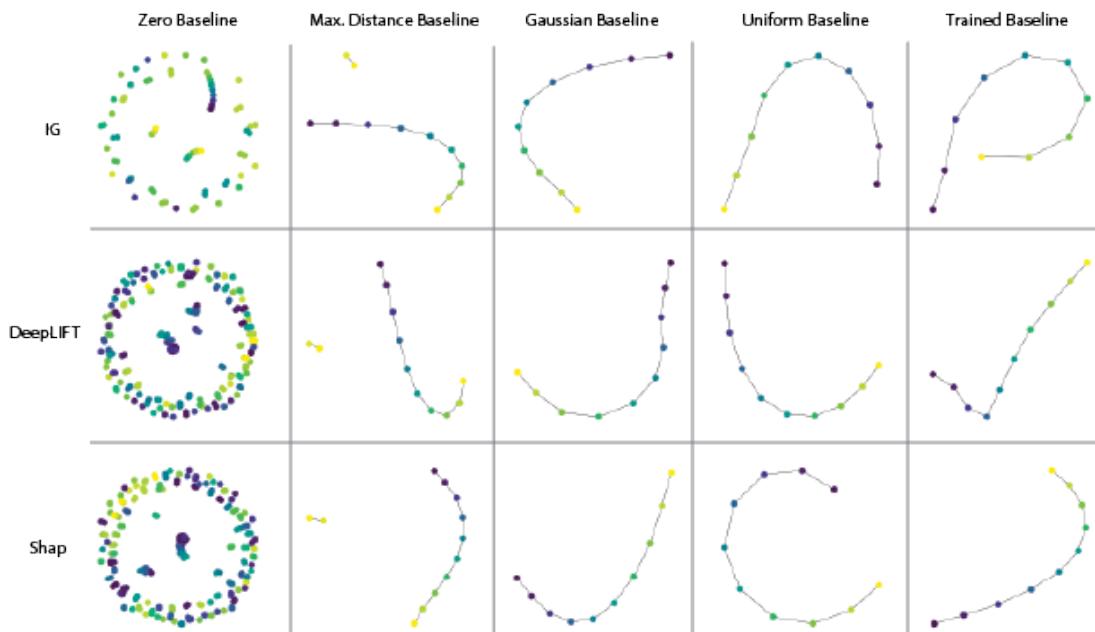


Figure 6.12: Identifying the influence of zero baselines in Bank Marketing dataset by comparing the topological representations of different combinations of explanation methods and baselines. (a) The dataset contains many zero values. Thus it is important to verify whether a zero baseline is appropriate. (b) Apart from using the distance matrix (similar to Figure 6.11), we can also visualize the topological skeleton of the explanation function to identify the differences.

baselines behave differently from the others. This behavior can also be visually seen from the topological skeletons corresponding to the explanations. As seen in Figure 6.12(b), the graphs corresponding to zero baselines have a topological structure significantly different from the others. When working in a real dataset scenario, it is normal for tabular datasets to contain zero values that provide important information. We demonstrate that with our topological representations, such influence can be revealed with the help of distance measures and visual comparisons.

## 6.6 Conclusion

In this chapter, we introduce a topology-based framework to summarize local explanations corresponding to an entire dataset. This is accomplished by first computing the topological skeleton of a scalar function that captures the relationship between the explanation space and the predictions. This is then used to compute a signature for each explanation in the form of a persistence diagram. This approach has the advantage that it is homogeneous and comparable even when used on explanation methods with heterogeneous formats. We demonstrate the validity and effectiveness of our approach through a set of experiments using both synthetic and real-world datasets.

While this chapter focused on binary classification, extending the approach to handle multi-class classifiers poses an important challenge—modeling a function similar to what is being done results in a multi-variate function, for which defining the appropriate filtration to compute the persistence diagrams becomes non-trivial. One possibility is to transform multi-class probabilities into scalar values, and we plan to investigate approaches for the same.

# Chapter 7

## Conclusions and Future Work

This dissertation presents four main contributions to data summaries for scalable visual analysis. The first contribution is data summaries for bipartite graphs, which summarize large-scale bipartite graphs with the Minimum Description Length Principles for interactive data analysis. The second contribution is data summaries for time series, which extract time series subsequence clusters as the summaries for exploring patterns in large-scale time series. The third contribution is the ML model summary, which summarizes feature-importance explanations to understand the model’s rationale to an input dataset. The fourth contribution is data summaries for comparing heterogeneous local ML model explanations, which summarize local explanations with heterogeneous dimensions into simplified topological representations.

While the contributions allow the interactive visual analysis of different large-scale datasets, there are still challenges that need to be addressed in future research. We now present some of these challenges and highlight interesting directions of research in the area of visual analytics.

**Scalability beyond Online Computations.** The contributions of this dissertation mainly focus on data summary generations by online computations. We assume the data fit into the main memory of a single laptop. However, real-world datasets are often stored in industrial DBMS, and billions of records are reasonable sizes for interactive data explorations as well. While being out of scope for this thesis, we have explored the options for interactive data summarization with the use of distributed systems [219]. If we assume precomputations are available, we

can generate data summaries using the parallelized platforms like MapReduce or Spark before users conduct the interactive analysis.

An interesting direction is the investigation of indexing for computing data summaries, where the indexes accommodate different users' parameters and return data structures that balance between online computations and offline data preparations.

**Actionable Computations Insights.** The contributions of this dissertation mainly focus on exploratory data analysis. Data summaries are also useful for automated predictions and classifications. Besides the contributions of the thesis, we have explored the use of data summaries for conducting predictive tasks after users' data explorations [220]. We are interested in deeper explorations of consequential tasks after the visual analysis with data summaries.

**Usability by Domain Experts.** The contributions of this dissertation provide interfaces for general explorations of common datasets. Yet, domain experts, who are common end-users for interactive data analysis, usually prefer their own pipeline for data analysis. For domain-specific data exploration, we have worked with more human-center design process [221] on datasets besides the introduction of data summarization, but it is of much importance to apply our data summaries on more applications and usages in the future.

## Appendices

# Appendix A

## Graph Summary

### A.1 Subroutines in BM-MDL Algorithm

---

**ALGORITHM 9:** *cost\_reduction\_for\_bundling((p,q), (p',q), R, S)*

---

**Input:**  $R \subseteq U \times V$ ,  $S \subseteq P \times Q$ ,  $p, p' \in P$  and  $q \in Q$   
**Output:** Cost reduction  $\Delta$  for bundling  $(p, q)$ ,  $(p', q)$

```

1   $\Delta = 0$ 
2  /* Calculate description length for  $(p, q)$  */  

3  if  $(p, q) \in S$  then
4       $\Delta += 1$ 
5       $\Delta += \alpha \cdot (\|p \times q\| - \|(p \times q) \cap R\|)$  /* Corrections: remove non-existing edges */
6  else
7       $\Delta += \alpha \cdot \|(p \times q) \cap R\|$  /* Corrections: add existing edges */
8  end
9  /* Calculate description length for  $(p', q)$  */  

10 if  $(p', q) \in S$  then
11      $\Delta += 1$ 
12      $\Delta += \alpha \cdot (\|p' \times q\| - \|(p' \times q) \cap R\|)$ 
13 else
14      $\Delta += \alpha \cdot \|(p' \times q) \cap R\|$ 
15 end
16 /* Calculate max description length reduction by bundling  $(p, q)$  and  $(p', q')$  */
17  $\Delta -= \min(1 + \alpha \cdot (\|(p \cup p') \times q\| - \|((p \cup p') \times q) \cap R\|), \alpha \cdot \|((p \cup p') \times q) \cap R\|)$ 
18 /* Output results */
19 return  $\Delta$ 

```

---

---

**ALGORITHM 10:**  $merge(p, p', R, \mathbb{S})$ 

---

**Input:**  $R \subseteq U \times V$ ,  $\mathbb{S} \subseteq P \times Q$  and  $p, p' \in P$ **Output:** Updated  $P, Q, \mathbb{S} \subseteq P \times Q$ 

- 1 remove  $p, p'$  from  $P, \mathbb{S}$
  - 2 remove all edges adjacent to  $p$  or  $p'$  from  $\mathbb{S}$
  - 3 add  $p_{new} = p \cup p'$  to  $P, \mathbb{S}$
  - 4 **for**  $q \in Q$  **do**
  - 5     **if**  $1 + \alpha \cdot (\|p_{new} \times q\| - \|(p_{new} \times q) \cap R\|) < \alpha \cdot \|(p_{new} \times q) \cap R\|$  **then**
  - 6         add edge  $(p_{new}, q)$  to  $\mathbb{S}$
  - 7     **end**
  - 8 **end**
-

## Appendix B

### Time Summary

#### B.1 Illustration of Synthetic Dataset Creation

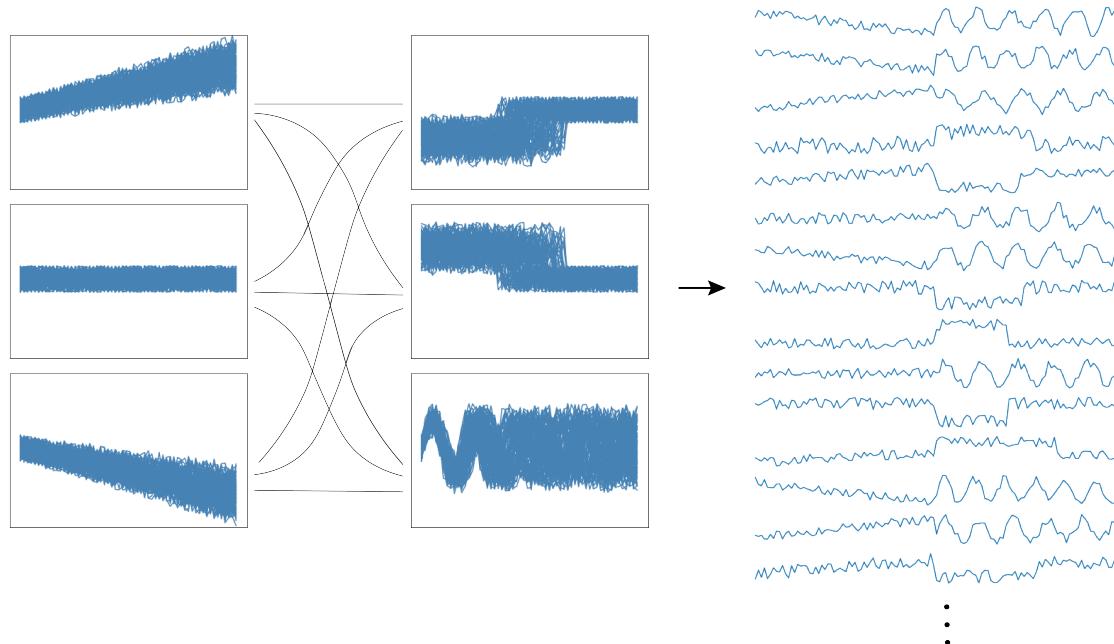


Figure B.1: Illustration of how the synthetic dataset used in Figure 4.5 is generated.

## B.2 Parameters Setting In Evaluation

For the result in Figure 4.5, we run our algorithm on three datasets with different combinations of durations of time series patterns. For each unique pattern and combination, there exists 100 time series. We fix the clustering strength to 1, minimum support to 50, and select the best results with recall greater than 0.95 among time window sizes of {25, 50, 100, 200} for three datasets.

For the results in Figure 4.7 and Figure 4.10, we set the time window size to 10% of the datasets' durations, strength to 10, and minimum support to 50.

## Appendix C

### Model Summary

#### C.1 Feature Engineering Implementation on Explanation Matrix

---

**ALGORITHM 11:** Sparsify Features in the Explanation Matrix

---

**Input :**  $X, E, S$  training data, explanation data, feature set

**Output :**  $\hat{E}$  – a sparsified explanation matrix

```

1 Initialize a dictionary of binning functions  $Bins$ 
2 Initialize a new sparse matrix  $\hat{E} : \hat{e}_i \mapsto \{\}$ 
3 for  $s$  in  $S$  do
4    $Bins[s] \leftarrow \text{BINNINGFUNCTION}(X[:, s])$  /* construct a binning function for each
      feature from its values in the training data */
5 end
6 for  $x_i, e_i$  in  $X, E$  do
7   for  $s$  in  $S$  do
8      $\hat{e}_i[Bins[s](x_i)] \leftarrow e_i$  /* rename the original feature as binned feature in the
        explanation matrix */
9   end
10 end
```

---

---

**ALGORITHM 12:** Densify Features in the Explanation Matrix

---

**Input :**  $E, pred, S$  - explanation data, predictions, feature set

**Output:**  $\hat{E}$  - a densified explanation matrix

```

1 Initialize a (number of features) × (number of prediction class) matrix  $F$ 
2 Initialize a new sparse matrix  $\hat{E} : \hat{e}_i \mapsto \{\}$ 
3 Partition  $E$  to  $E_{pred}$  according to  $pred$ 
4 for  $e_i$  in  $E_{pred}$  do
5    $| F[:, i] \leftarrow \text{sum}(e_i, \text{axis}=\text{"column"})$  /* get the sum of values for each feature according
     | to class i */
6 end
7  $\text{cluster\_labels} = \text{cluster}(F)$                                      /* cluster the features */
8 for  $e_i$  in  $E$  do
9   for  $label, members$  in  $\text{cluster\_labels}$  do
10     $| \hat{e}_i[\text{label}] \leftarrow \text{sum}(e_i[\text{members}])$  /* combine the values of features belonging to
      | the same cluster */
11  end
12 end

```

---

## Appendix D

# Topological Explanation Summary

### D.1 Tuning Parameters in the Mapper Algorithm

In this section we provide an overview of parameter tuning we performed for the Mapper algorithm. For the experiments in Section 6.5, we arrive at the values of resolutions and gains using the following processes (Figures D.1–D.4). First, we conduct the bootstrap method with varying resolutions and gains to acquire the confidence regions of the bottleneck distances. We return the 95% confidence region of each topological representation results so that for each combination of resolution and gain values, we have a value indicating difference between a representation and its similar copy under small perturbations. This results in a matrix heat map showing the confidence regions of resolutions and gains at which the topological representation is stable (i.e. the dark colored regions). After narrowing down the inspection, we explore the graph representations and explore the ones that are not too disconnected while at the same time is stable with a relatively high resolution and low overlap, as mentioned in Section ???. The graphs shown in the figures are the topological skeletons visualized using a force-directed layout, and the color encodes the average of the prediction probabilities of the inputs within the nodes.

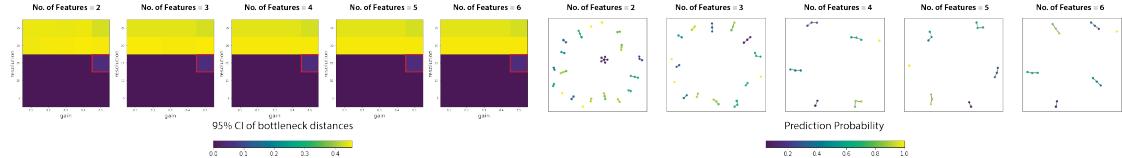


Figure D.1: Choosing the parameters for Mapper outputs in Figure 6.9. Left: Heat maps showing the confidence regions of different resolutions and gains for different explanation outputs. Right: graph representations for our choices of resolutions and gains (red rectangles in the heat maps).

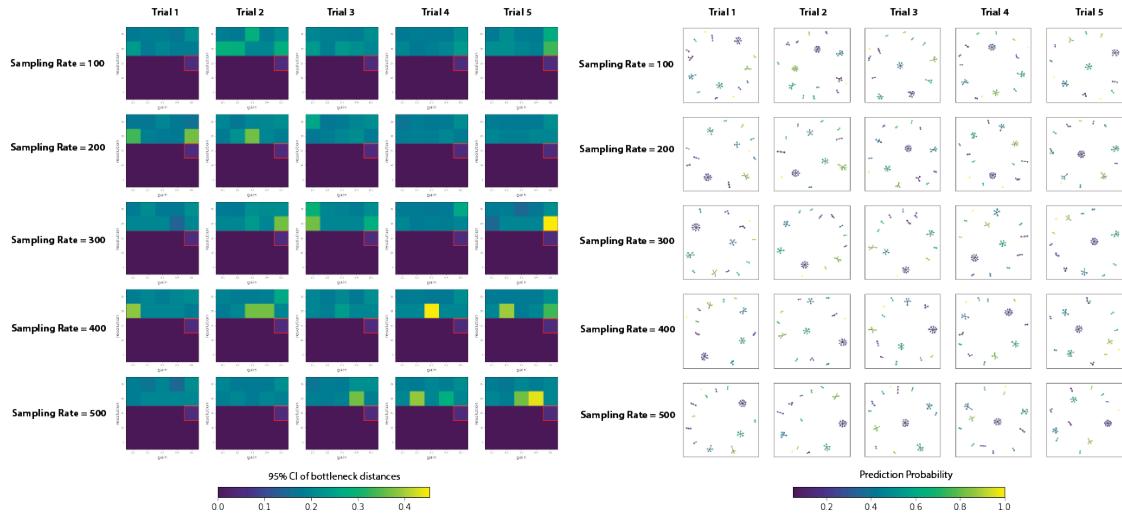


Figure D.2: Choosing the parameters for Mapper outputs in Figure 6.10. Left: Heat maps showing the confidence regions of different resolutions and gains for different explanation outputs. Right: graph representations for our choices of resolutions and gains (red rectangles in the heat maps).

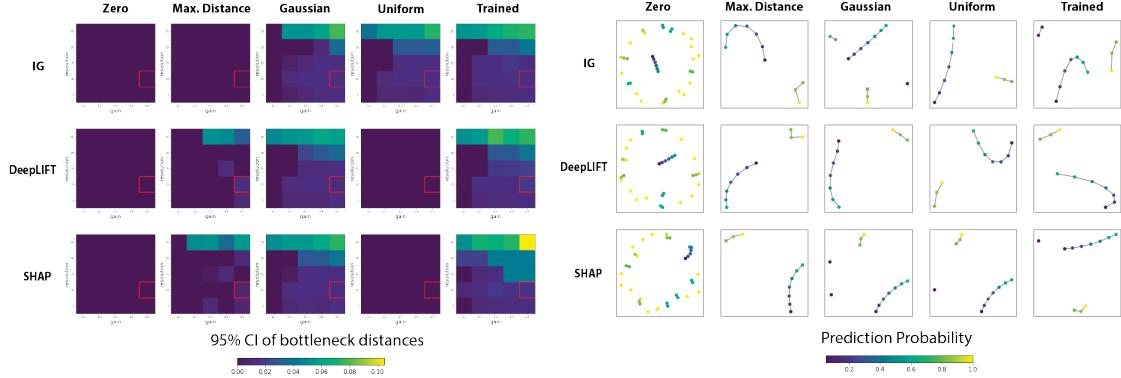


Figure D.3: Choosing the parameters for Mapper outputs in Figure 6.11. Left: Heat maps showing the confidence regions of different resolutions and gains for different explanation outputs. Right: graph representations for our choices of resolutions and gains (red rectangles in the heat maps).

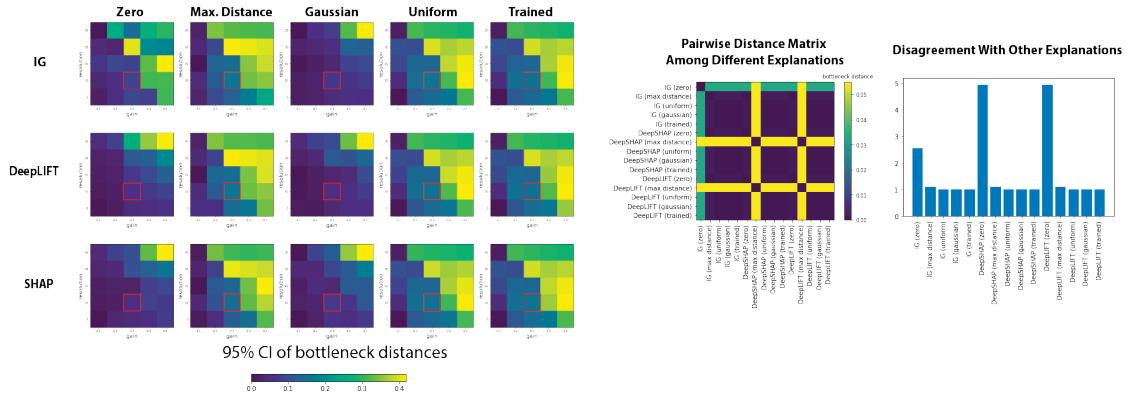


Figure D.4: Choosing the parameters for Mapper outputs in Figure 6.12 and the output of comparisons similar to Figure 6.11. Left: Heat maps showing the confidence regions of different resolutions and gains for different explanation methods. Right: Pairwise distance matrix showing the significant bottleneck differences among explanation methods with zero baselines, and bar charts highlighting the difference of the explanations among the zero baseline results with the disagreement measure (i.e. row wise sum of pairwise distances).

## Bibliography

- [1] John W Tukey et al. *Exploratory data analysis*, volume 2. Reading, Mass., 1977.
- [2] George A Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, 63(2):81, 1956.
- [3] Stephen G Eick and Alan F Karr. Visual scalability. *Journal of Computational and Graphical Statistics*, 11(1):22–43, 2002.
- [4] Yanhong Wu, Nan Cao, Daniel Archambault, Qiaomu Shen, Huamin Qu, and Weiwei Cui. Evaluation of graph sampling: A visualization perspective. *IEEE transactions on visualization and computer graphics*, 23(1):401–410, 2016.
- [5] Brian Duffy, Jeyarajan Thiyyagalingam, Simon Walton, David J Smith, Anne Trefethen, Jackson C Kirkman-Brown, Eamonn A Gaffney, and Min Chen. Glyph-based video visualization for semen analysis. *IEEE transactions on visualization and computer graphics*, 21(8):980–993, 2013.
- [6] Josua Krause, Adam Perer, and Enrico Bertini. Infuse: interactive feature selection for predictive modeling of high dimensional data. *IEEE transactions on visualization and computer graphics*, 20(12):1614–1623, 2014.
- [7] Jessica Lin, Eamonn Keogh, Stefano Lonardi, Jeffrey P Lankford, and Donna M Nystrom. Visually mining and monitoring massive time series. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 460–469, 2004.

- [8] Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *The craft of information visualization*, pages 364–371. Elsevier, 2003.
- [9] Kanit Wongsuphasawat, Zening Qu, Dominik Moritz, Riley Chang, Felix Ouk, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. Voyager 2: Augmenting visual analysis with partial view specifications. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 2648–2659, 2017.
- [10] Jimmy Johansson, Camilla Forsell, Mats Lind, and Matthew Cooper. Perceiving patterns in parallel coordinates: determining thresholds for identification of relationships. *Information Visualization*, 7(2):152–162, 2008.
- [11] Danyel Fisher. Big data exploration requires collaboration between visualization and data infrastructures. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, pages 1–5, 2016.
- [12] Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 631–636, 2006.
- [13] Bum Chul Kwon, Janu Verma, Peter J Haas, and Cagatay Demiralp. Sampling for scalable visual analytics. *IEEE computer graphics and applications*, 37(1):100–108, 2017.
- [14] Dominik Sacha, Hansi Seuratne, Bum Chul Kwon, Geoffrey Ellis, and Daniel A Keim. The role of uncertainty, awareness, and trust in visual analytics. *IEEE transactions on visualization and computer graphics*, 22(1):240–249, 2015.
- [15] Adrian Mayorga and Michael Gleicher. Splatterplots: Overcoming overdraw in scatter plots. *IEEE transactions on visualization and computer graphics*, 19(9):1526–1538, 2013.
- [16] Vahan Yoghoudjian, Tim Dwyer, Karsten Klein, Kim Marriott, and Michael Wybrow. Graph thumbnails: Identifying and comparing multiple graphs

- at a glance. *IEEE Transactions on Visualization and Computer Graphics*, 24(12):3081–3095, 2018.
- [17] Katy Börner, Andreas Bueckle, and Michael Ginda. Data visualization literacy: Definitions, conceptual frameworks, exercises, and assessments. *Proceedings of the National Academy of Sciences*, 116(6):1857–1864, 2019.
  - [18] Ben Shneiderman. Dynamic queries for visual information seeking. *IEEE software*, 11(6):70–77, 1994.
  - [19] Zhicheng Liu and Jeffrey Heer. The effects of interactive latency on exploratory visual analysis. *IEEE transactions on visualization and computer graphics*, 20(12):2122–2131, 2014.
  - [20] Lauro Lins, James T Klosowski, and Carlos Scheidegger. Nanocubes for real-time exploration of spatiotemporal datasets. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2456–2465, 2013.
  - [21] Haneen Mohammed. Continuous prefetch for interactive data applications. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 2841–2843, 2020.
  - [22] Dominik Moritz, Bill Howe, and Jeffrey Heer. Falcon: Balancing interactive latency and resolution sensitivity for scalable linked visualizations. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–11, 2019.
  - [23] Yuanzhe Chen, Panpan Xu, and Liu Ren. Sequence synopsis: Optimize visual summary of temporal event data. *IEEE transactions on visualization and computer graphics*, 2017.
  - [24] Dongyu Liu, Panpan Xu, and Liu 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, 2018.
  - [25] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender systems: an introduction*. Cambridge University Press, 2010.

- [26] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Rock: A robust clustering algorithm for categorical attributes. In *Data Engineering, 1999. Proceedings., 15th International Conference on*, pages 512–521. IEEE, 1999.
- [27] Clifford Carrubba, Matthew Gabel, and Simon Hug. Legislative voting behavior, seen and unseen: A theory of roll-call vote selection. *Legislative Studies Quarterly*, 33(4):543–572, 2008.
- [28] Yeuk-Yin Chan, Fernando Chirigati, Harish Doraiswamy, Cláudio T Silva, and Juliana Freire. Querying and exploring polygamous relationships in urban spatio-temporal data sets. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1643–1646. ACM, 2017.
- [29] Fernando Chirigati, Harish Doraiswamy, Theodoros Damoulas, and Juliana Freire. Data polygamy: the many-many relationships among urban spatio-temporal data sets. In *Proceedings of the 2016 International Conference on Management of Data*, pages 1011–1025. ACM, 2016.
- [30] Patrick Fiaux, Maoyuan Sun, Lauren Bradel, Chris North, Naren Ramakrishnan, and Alex Endert. Bixplorer: Visual analytics with biclusters. *Computer*, 46(8):90–94, 2013.
- [31] Marc Streit, Samuel Gratzl, Michael Gillhofer, Andreas Mayr, Andreas Mitterecker, and Sepp Hochreiter. Furby: fuzzy force-directed bicluster visualization. *BMC bioinformatics*, 15(Suppl 6):S4, 2014.
- [32] Panpan Xu, Nan Cao, Huamin Qu, and John Stasko. Interactive visual co-cluster analysis of bipartite graphs. In *Pacific Visualization Symposium (PacificVis), 2016 IEEE*, pages 32–39. IEEE, 2016.
- [33] Maoyuan Sun, Peng Mi, Chris North, and Naren Ramakrishnan. Biset: Semantic edge bundling with biclusters for sensemaking. *IEEE transactions on visualization and computer graphics*, 22(1):310–319, 2016.
- [34] Jian Zhao, Maoyuan Sun, Francine Chen, and Patrick Chiu. Bidots: Visual exploration of weighted biclusters. *IEEE transactions on visualization and computer graphics*, 2017.

- [35] Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.
- [36] Rafael Veras and Christopher Collins. Optimizing hierarchical visualizations with the minimum description length principle. *IEEE transactions on visualization and computer graphics*, 23(1):631–640, 2017.
- [37] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge university press, 2014.
- [38] Marcel Hlawatsch, Michael Burch, and Daniel Weiskopf. Visual adjacency lists for dynamic graphs. *IEEE transactions on visualization and computer graphics*, 20(11):1590–1603, 2014.
- [39] Shixia Liu, Weiwei Cui, Yingcai Wu, and Mengchen Liu. A survey on information visualization: recent advances and challenges. *The Visual Computer*, 30(12):1373–1393, 2014.
- [40] Ben Shneiderman and Aleks Aris. Network visualization by semantic substrates. *IEEE transactions on visualization and computer graphics*, 12(5):733–740, 2006.
- [41] Marian Dörk, Nathalie Henry Riche, Gonzalo Ramos, and Susan Dumais. Pivotpaths: Strolling through faceted information spaces. *IEEE transactions on visualization and computer graphics*, 18(12):2709–2718, 2012.
- [42] John Stasko, Carsten Görg, and Zhicheng Liu. Jigsaw: supporting investigative analysis through interactive visualization. *Information visualization*, 7(2):118–132, 2008.
- [43] Sohaib Ghani, Bum Chul Kwon, Seungyoon Lee, Ji Soo Yi, and Niklas Elmquist. Visual analytics for multimodal social network analysis: A design study with social scientists. *IEEE transactions on visualization and computer graphics*, 19(12):2032–2041, 2013.
- [44] Nan Cao, Jimeng Sun, Yu-Ru Lin, David Gotz, Shixia Liu, and Huamin Qu. Facetatlas: Multifaceted visualization for rich text corpora. *IEEE transactions on visualization and computer graphics*, 16(6):1172–1181, 2010.

- [45] Zeqian Shen, Kwan-Liu Ma, and Tina Eliassi-Rad. Visual analysis of large heterogeneous social networks by semantic and structural abstraction. *IEEE transactions on visualization and computer graphics*, 12(6):1427–1439, 2006.
- [46] Kazuo Misue. Drawing bipartite graphs as anchored maps. In *Proceedings of the 2006 Asia-Pacific Symposium on Information Visualisation - Volume 60*, APVis '06, pages 169–177, Darlinghurst, Australia, Australia, 2006. Australian Computer Society, Inc.
- [47] Bohyoung Kim, Bongshin Lee, and Jinwook Seo. Visualizing set concordance with permutation matrices and fan diagrams. *Interacting with computers*, 19(5-6):630–643, 2007.
- [48] Harri Siirtola and Erkki Mäkinen. Constructing and reconstructing the reorderable matrix. *Information Visualization*, 4(1):32–48, 2005.
- [49] Jacques Bertin. Semiology of graphics: diagrams, networks, maps. 1983.
- [50] Charles Perin, Pierre Dragicevic, and Jean-Daniel Fekete. Revisiting bertin matrices: New interactions for crafting tabular visualizations. *IEEE transactions on visualization and computer graphics*, 20(12):2082–2091, 2014.
- [51] Kazuo Misue and Qi Zhou. Drawing semi-bipartite graphs in anchor+ matrix style. In *Information Visualisation (IV), 2011 15th International Conference on*, pages 26–31. IEEE, 2011.
- [52] Bilal Alsallakh, Luana Micallef, Wolfgang Aigner, Helwig Hauser, Silvia Miksch, and Peter Rodgers. Visualizing sets and set-typed data: State-of-the-art and future challenges. In *Eurographics conference on Visualization (EuroVis)–State of The Art Reports*, pages 1–21, 2014.
- [53] Mohammed Javeed Zaki and C-J Hsiao. Efficient algorithms for mining closed itemsets and their lattice structure. *IEEE transactions on knowledge and data engineering*, 17(4):462–478, 2005.
- [54] Takeaki Uno, Tatsuya Asai, Yuzo Uchida, and Hiroki Arimura. An efficient algorithm for enumerating closed patterns in transaction databases. In *International Conference on Discovery Science*, pages 16–31. Springer, 2004.

- [55] Maoyuan Sun, Chris North, and Naren Ramakrishnan. A five-level design framework for bicluster visualizations. *IEEE transactions on visualization and computer graphics*, 20(12):1713–1722, 2014.
- [56] Maoyuan Sun, Lauren Bradel, Chris L North, and Naren Ramakrishnan. The role of interactive biclusters in sensemaking. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pages 1559–1562. ACM, 2014.
- [57] Mengchen Liu, Jiaxin Shi, Zhen Li, Chongxuan Li, Jun Zhu, and Shixia Liu. Towards better analysis of deep convolutional neural networks. *IEEE transactions on visualization and computer graphics*, 23(1):91–100, 2017.
- [58] Yosuke Onoue, Nobuyuki Kukimoto, Naohisa Sakamoto, and Koji Koyamada. Minimizing the number of edges via edge concentration in dense layered graphs. *IEEE transactions on visualization and computer graphics*, 22(6):1652–1661, 2016.
- [59] Yuval Kluger, Ronen Basri, Joseph T Chang, and Mark Gerstein. Spectral biclustering of microarray data: coclustering genes and conditions. *Genome research*, 13(4):703–716, 2003.
- [60] Inderjit S Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274. ACM, 2001.
- [61] Yao Ming, Shaozu Cao, Ruixiang Zhang, Zhen Li, Yuanzhe Chen, Yangqiu Song, and Huamin Qu. Understanding hidden memories of recurrent neural networks. *arXiv preprint arXiv:1710.10777*, pages 13–24, 2017.
- [62] Saket Navlakha, Rajeev Rastogi, and Nisheeth Shrivastava. Graph summarization with bounded error. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 419–432. ACM, 2008.
- [63] Jing Feng, Xiao He, Bettina Konte, Christian Böhm, and Claudia Plant. Summarization-based mining bipartite graphs. In *Proceedings of the 18th*

*ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1249–1257. ACM, 2012.

- [64] Deepayan Chakrabarti, Spiros Papadimitriou, Dharmendra S Modha, and Christos Faloutsos. Fully automatic cross-associations. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 79–88. ACM, 2004.
- [65] Mohammad Ghoniem, J-D Fekete, and Philippe Castagliola. A comparison of the readability of graphs using node-link and matrix-based representations. In *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*, pages 17–24. Ieee, 2004.
- [66] Jian Zhao, Zhicheng Liu, Mira Dontcheva, Aaron Hertzmann, and Alan Wilson. Matrixwave: Visual comparison of event sequence data. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, CHI ’15*, pages 259–268, New York, NY, USA, 2015. ACM.
- [67] Mark Harrower and Cynthia A Brewer. Colorbrewer. org: an online tool for selecting colour schemes for maps. *The Cartographic Journal*, 40(1):27–37, 2003.
- [68] Colin Ware. *Information visualization: perception for design*. Elsevier, 2012.
- [69] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4):19, 2016.
- [70] Tamara Munzner. A nested model for visualization design and validation. *IEEE transactions on visualization and computer graphics*, 15(6), 2009.
- [71] Juan P. Bello, Claudio Silva, Oded Nov, R. Luke Dubois, Anish Arora, Justin Salamon, Charles Mydlarz, and Harish Doraiswamy. Sonyc: A system for monitoring, analyzing, and mitigating urban noise pollution. *Commun. ACM*, 62(2):68–77, January 2019.

- [72] Harish Doraiswamy, Juliana Freire, Marcos Lage, Fabio Miranda, and Claudio Silva. Spatio-temporal urban data analysis: A visual analytics perspective. *IEEE computer graphics and applications*, 38(5):26–35, 2018.
- [73] Fabio Miranda, Marcos Lage, Harish Doraiswamy, Charlie Mydlarz, Justin Salamon, Yitzchak Lockerman, Juliana Freire, and Claudio T Silva. Time lattice: A data structure for the interactive visual analysis of large time series. 2018.
- [74] Tobias Schreck, Tatiana Tekušová, Jörn Kohlhammer, and Dieter Fellner. Trajectory-based visual analysis of large financial time series data. *ACM SIGKDD Explorations Newsletter*, 9(2):30–37, 2007.
- [75] Katsiaryna Mirylenka, Vassilis Christophides, Themis Palpanas, Ioannis Pefkianakis, and Martin May. Characterizing home device usage from wireless traffic time series. In *19th International Conference on Extending Database Technology (EDBT)*, 2016.
- [76] Pavlos Paraskevopoulos, Thanh-Cong Dinh, Zolzaya Dashdorj, Themis Palpanas, and Luciano Serafini. Identification and characterization of human behavior patterns from mobile phone data. *Proc. of NetMob*, 2013.
- [77] Themis Palpanas. Data series management: The road to big sequence analytics. *SIGMOD Record*, 44(2):47–52, 2015.
- [78] Themis Palpanas and Volker Beckmann. Report on the first and second interdisciplinary time series analysis workshop (ITISA). *SIGREC*, 48(3), 2019.
- [79] Edward Tufte and P Graves-Morris. The visual display of quantitative information., 1983.
- [80] Peter McLachlan, Tamara Munzner, Eleftherios Koutsofios, and Stephen North. Liverac: interactive visual exploration of system management time-series data. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1483–1492. ACM, 2008.

- [81] Dominik Moritz and Danyel Fisher. Visualizing a million time series with the density line chart. *arXiv preprint arXiv:1808.06019*, 2018.
- [82] Takafumi Saito, Hiroko Nakamura Miyamura, Mitsuyoshi Yamamoto, Hi-roki Saito, Yuka Hoshiya, and Takumi Kaseda. Two-tone pseudo coloring: Compact visualization for one-dimensional data. 2005.
- [83] Waqas Javed, Bryan McDonnel, and Niklas Elmquist. Graphical perception of multiple time series. *IEEE Transactions on Visualization & Computer Graphics*, (6):927–934, 2010.
- [84] Gautam Das, King-Ip Lin, Heikki Mannila, Gopal Renganathan, and Padhraic Smyth. Rule discovery from time series. In *KDD*, volume 98, pages 16–22, 1998.
- [85] Anne Denton. Kernel-density-based clustering of time series subsequences using a continuous random-walk noise model. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 8–pp. IEEE, 2005.
- [86] Dina Goldin, Ricardo Mardales, and George Nagy. In search of meaning for time series subsequence clustering: matching algorithms based on a new distance measure. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 347–356, 2006.
- [87] Eamonn Keogh and Jessica Lin. Clustering of time-series subsequences is meaningless: implications for previous and future research. *Knowledge and information systems*, 8(2):154–177, 2005.
- [88] Tim Oates. Identifying distinctive subsequences in multivariate time series by clustering. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 322–326, 1999.
- [89] Martin C Frith, Michael C Li, and Zhiping Weng. Cluster-buster: Finding dense clusters of motifs in dna sequences. *Nucleic acids research*, 31(13):3666–3668, 2003.

- [90] Craig G Nevill-Manning and Ian H Witten. Identifying hierarchical structure in sequences: A linear-time algorithm. *Journal of Artificial Intelligence Research*, 7:67–82, 1997.
- [91] Wolfgang Aigner, Silvia Miksch, Heidrun Schumann, and Christian Tominski. *Visualization of time-oriented data*. Springer Science & Business Media, 2011.
- [92] Benjamin Bach, Pierre Dragicevic, Daniel Archambault, Christophe Hurter, and Sheelagh Carpendale. A review of temporal data visualizations based on space-time cube operations. In *Eurographics Conference on Visualization, EuroVis 2014 - State of the Art Reports*, 2014.
- [93] Ming C Hao, Umeshwar Dayal, Daniel A Keim, and Tobias Schreck. Multi-resolution techniques for visual exploration of large time-series data. In *EUROVIS 2007*, pages 27–34, 2007.
- [94] Daniel A Keim, Mihael Ankerst, and Hans-Peter Kriegel. Recursive pattern: A technique for visualizing very large amounts of data. In *Proceedings of the 6th Conference on Visualization'95*, page 279. IEEE Computer Society, 1995.
- [95] Bruce J Swihart, Brian Caffo, Bryan D James, Matthew Strand, Brian S Schwartz, and Naresh M Punjabi. Lasagna plots: a saucy alternative to spaghetti plots. *Epidemiology (Cambridge, Mass.)*, 21(5):621, 2010.
- [96] Hartmut Ziegler, Marco Jenny, Tino Gruse, and Daniel A Keim. Visual market sector analysis for financial time series data. In *Visual Analytics Science and Technology (VAST), 2010 IEEE Symposium on*, pages 83–90. IEEE, 2010.
- [97] Anna Gogolou, Theophanis Tsandilas, Themis Palpanas, and Anastasia Bezerianos. Comparing similarity perception in time series visualizations. *IEEE transactions on visualization and computer graphics*, 2018.
- [98] Paolo Buono, Aleks Aris, Catherine Plaisant, Amir Khella, and Ben Shneiderman. Interactive pattern search in time series. In *Visualization and Data Analysis 2005*, volume 5669, pages 175–187. International Society for Optics and Photonics, 2005.

- [99] Paolo Buono, Catherine Plaisant, Adalberto Simeone, Aleks Aris, Ben Shneiderman, Galit Shmueli, and Wolfgang Jank. Similarity-based forecasting with simultaneous previews: A river plot interface for time series forecasting. In *2007 11th International Conference Information Visualization (IV'07)*, pages 191–196. IEEE, 2007.
- [100] Harry Hochheiser and Ben Shneiderman. Dynamic query tools for time series data sets: timebox widgets for interactive exploration. *Information Visualization*, 3(1):1–18, 2004.
- [101] Lior Berry and Tamara Munzner. Binx: Dynamic exploration of time series datasets across aggregation levels. In *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*, pages p2–p2. IEEE, 2004.
- [102] Jian Zhao, Fanny Chevalier, Emmanuel Pietriga, and Ravin Balakrishnan. Exploratory analysis of time-series with chronolenses. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2422–2431, 2011.
- [103] Theresia Gschwandtner, Wolfgang Aigner, Katharina Kaiser, Silvia Miksch, and Andreas Seyfang. Carecruiser: Exploring and visualizing plans, events, and effects interactively. In *Visualization Symposium (PacificVis), 2011 IEEE Pacific*, pages 43–50. IEEE, 2011.
- [104] Paul André, Max L Wilson, Alistair Russell, Daniel A Smith, Alisdair Owens, et al. Continuum: designing timelines for hierarchies, relationships and scale. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 101–110. ACM, 2007.
- [105] Dongning Luo, Jing Yang, Milos Krstajic, William Ribarsky, and Daniel Keim. Eventriver: Visually exploring text collections with temporal references. *IEEE Transactions on Visualization and Computer Graphics*, 18(1):93–105, 2012.
- [106] Raimund Dachselt, Mathias Frisch, and Markus Weiland. Facetzoom: a continuous multi-scale widget for navigating hierarchical metadata. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1353–1356. ACM, 2008.

- [107] Ragnar Bade, Stefan Schlechtweg, and Silvia Miksch. Connecting time-oriented data and information to a coherent interactive visualization. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 105–112. ACM, 2004.
- [108] James Walker, Rita Borgo, and Mark W Jones. Timenotes: a study on effective chart visualization and interaction techniques for time-series data. *IEEE transactions on visualization and computer graphics*, 22(1):549–558, 2016.
- [109] Markus Wagner, Djordje Slijepcevic, Brian Horsak, Alexander Rind, Matthias Zeppelzauer, and Wolfgang Aigner. Kavagait: Knowledge-assisted visual analytics for clinical gait analysis. *IEEE transactions on visualization and computer graphics*, 25(3):1528–1542, 2018.
- [110] Georgiy Shurkhovetskyy, N Andrienko, G Andrienko, and Georg Fuchs. Data abstraction for visualizing large time series. In *Computer Graphics Forum*, volume 37, pages 125–144. Wiley Online Library, 2018.
- [111] Karima Echihabi, Kostas Zoumpatianos, Themis Palpanas, and Houda Benbrahim. The lernaean hydra of data series similarity search: An experimental evaluation of the state of the art. *PVLDB*, 12(2):112–127, 2018.
- [112] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 2–11. ACM, 2003.
- [113] Jessica Lin, Eamonn Keogh, and Stefano Lonardi. Visualizing and discovering non-trivial patterns in large time series databases. *Information visualization*, 4(2):61–82, 2005.
- [114] Nicholas Ruta, Naoko Sawada, Katy McKeough, Michael Behrisch, and Johanna Beyer. Sax navigator: Time series exploration through hierarchical clustering. In *2019 IEEE Visualization Conference (VIS)*, pages 236–240. IEEE, 2019.

- [115] Ming C Hao, Manish Marwah, Halldór Janetzko, Umeshwar Dayal, Daniel A Keim, Debprakash Patnaik, Naren Ramakrishnan, and Ratnesh K Sharma. Visual exploration of frequent patterns in multivariate time series. *Information Visualization*, 11(1):71–83, 2012.
- [116] Prithiviraj K Muthumanickam, Katerina Vrotsou, Matthew Cooper, and Jimmy Johansson. Shape grammar extraction for efficient query-by-sketch pattern matching in long time series. In *Visual Analytics Science and Technology (VAST), 2016 IEEE Conference on*, pages 121–130. IEEE, 2016.
- [117] Yuan Li, Jessica Lin, and Tim Oates. Visualizing variable-length time series motifs. In *Proceedings of the 2012 SIAM international conference on data mining*, pages 895–906. SIAM, 2012.
- [118] Dominik Sacha, Matthias Kraus, Jürgen Bernard, Michael Behrisch, Tobias Schreck, Yuki Asano, and Daniel A Keim. Somflow: Guided exploratory cluster analysis with self-organizing maps and analytic provenance. *IEEE transactions on visualization and computer graphics*, 24(1):120–130, 2017.
- [119] Martin Steiger, Jürgen Bernard, Sebastian Mittelstädt, Hendrik Lücke-Tieke, Daniel Keim, Thorsten May, and Jörn Kohlhammer. Visual analysis of time-series similarities for anomaly detection in sensor networks. In *Computer graphics forum*, volume 33, pages 401–410. Wiley Online Library, 2014.
- [120] Matthew O Ward and Zhenyu Guo. Visual exploration of time-series data with shape space projections. In *Computer Graphics Forum*, volume 30, pages 701–710. Wiley Online Library, 2011.
- [121] Goethem Arthur Van, Frank Staals, Maarten Löffler, Jason Dykes, and Bettina Speckmann. Multi-granular trend detection for time-series analysis. *IEEE transactions on visualization and computer graphics*, 23(1):661–670, 2017.
- [122] Luka Stopar, Primoz Skraba, Marko Grobelnik, and Dunja Mladenic. Stream-story: exploring multivariate time series on multiple scales. *IEEE transactions on visualization and computer graphics*, 25(4):1788–1802, 2019.

- [123] Mike Sips, Patrick Kothur, Andrea Unger, Hans-Christian Hege, and Doris Dransch. A visual analytics approach to multiscale exploration of environmental time series. *IEEE Transactions on Visualization & Computer Graphics*, (12):2899–2907, 2012.
- [124] Patrick Köthur, Carl Witt, Mike Sips, Norbert Marwan, Stefan Schinkel, and Doris Dransch. Visual analytics for correlation-based comparison of time series ensembles. In *Computer Graphics Forum*, volume 34, pages 411–420. Wiley Online Library, 2015.
- [125] Tuan Nhon Dang, Anushka Anand, and Leland Wilkinson. Timeseer: Scagnostics for high-dimensional time series. *IEEE Transactions on Visualization and Computer Graphics*, 19(3):470–483, 2013.
- [126] Yifeng Gao and Jessica Lin. Efficient discovery of time series motifs with large length range in million scale time series. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 1213–1222. IEEE, 2017.
- [127] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment*, 1(2):1542–1552, 2008.
- [128] Katsiaryna Mirylenka, Michele Dallachiesa, and Themis Palpanas. Data series similarity using correlation-aware measures. In *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*, pages 11:1–11:12, 2017.
- [129] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994.
- [130] Philipp Eichmann and Emanuel Zgraggen. Evaluating subjective accuracy in time series pattern-matching using human-annotated rankings. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*, pages 28–37. ACM, 2015.

- [131] Miro Mannino and Azza Abouzied. Expressive time series querying with hand-drawn scale-free sketches. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 388. ACM, 2018.
- [132] Diego F Silva and Gustavo EAPA Batista. Speeding up all-pairwise dynamic time warping matrix calculation. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 837–845. SIAM, 2016.
- [133] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.
- [134] James Cheng, Yiping Ke, and Wilfred Ng. A survey on algorithms for mining frequent itemsets over data streams. *Knowledge and Information Systems*, 16(1):1–27, 2008.
- [135] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262, 2004.
- [136] Philippe Fournier-Viger, Cheng-Wei Wu, Antonio Gomariz, and Vincent S Tseng. Vmsp: Efficient vertical mining of maximal sequential patterns. In *Canadian conference on artificial intelligence*, pages 83–94. Springer, 2014.
- [137] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *icccn*, page 0215. IEEE, 2001.
- [138] Jianyong Wang and Jiawei Han. Bide: Efficient mining of frequent closed sequences. In *Proceedings. 20th international conference on data engineering*, pages 79–90. IEEE, 2004.
- [139] Dimitrios Gunopulos, Roni Khardon, Heikki Mannila, Sanjeev Saluja, Hannu Toivonen, and Ram Sewak Sharma. Discovering all most specific sentences. *ACM Transactions on Database Systems (TODS)*, 28(2):140–174, 2003.

- [140] Jarke J Van Wijk and Edward R Van Selow. Cluster and calendar based visualization of time series data. In *Proceedings 1999 IEEE Symposium on Information Visualization (InfoVis' 99)*, pages 4–9. IEEE, 1999.
- [141] Anja Struyf, Mia Hubert, Peter Rousseeuw, et al. Clustering in an object-oriented environment. *Journal of Statistical Software*, 1(4):1–30, 1997.
- [142] Marco Cuturi and Mathieu Blondel. Soft-dtw: a differentiable loss function for time-series. In *Proceedings of the 34th International Conference on Machine Learning- Volume 70*, pages 894–903. JMLR. org, 2017.
- [143] François Petitjean, Alain Ketterlin, and Pierre Gançarski. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition*, 44(3):678–693, 2011.
- [144] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Workshop at International Conference on Learning Representations*, 2014.
- [145] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning- Volume 70*, pages 3145–3153. JMLR. org, 2017.
- [146] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning- Volume 70*, pages 3319–3328. JMLR. org, 2017.
- [147] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*, pages 4765–4774, 2017.
- [148] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ” why should i trust you?” explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

- [149] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI*, volume 18, pages 1527–1535, 2018.
- [150] Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This looks like that: deep learning for interpretable image recognition. In *Advances in Neural Information Processing Systems*, pages 8928–8939, 2019.
- [151] Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [152] Yao Ming, Panpan Xu, Huamin Qu, and Liu Ren. Interpretable and steerable sequence learning via prototypes. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 903–913, 2019.
- [153] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):1–42, 2018.
- [154] Johan Huysmans, Karel Dejaeger, Christophe Mues, Jan Vanthienen, and Bart Baesens. An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. *Decision Support Systems*, 51(1):141–154, 2011.
- [155] Q Vera Liao, Daniel Gruen, and Sarah Miller. Questioning the ai: Informing design practices for explainable ai user experiences. *arXiv preprint arXiv:2001.02478*, 2020.
- [156] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1721–1730, 2015.

- [157] Cynthia Rudin and Berk Ustun. Optimized scoring systems: Toward trust in machine learning for healthcare and criminal justice. *Interfaces*, 48(5):449–466, 2018.
- [158] Chaofan Chen, Kangcheng Lin, Cynthia Rudin, Yaron Shaposhnik, Sijia Wang, and Tong Wang. An interpretable model with globally consistent explanations for credit risk. *arXiv preprint arXiv:1811.12615*, 2018.
- [159] Paul Voigt and Axel Von dem Bussche. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 10, 2017.
- [160] Fred Hohman, Minsuk Kahng, Robert Pienta, and Duen Horng Chau. Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE transactions on visualization and computer graphics*, 25(8):2674–2693, 2018.
- [161] F. . Tzeng and K. . Ma. Opening the black box - data driven visualization of neural networks. In *VIS 05. IEEE Visualization, 2005.*, pages 383–390, October 2005.
- [162] Hendrik Strobelt, Sebastian Gehrman, Hanspeter Pfister, and Alexander M Rush. Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE transactions on visualization and computer graphics*, 24(1):667–676, 2017.
- [163] Hendrik Strobelt, Sebastian Gehrman, Michael Behrisch, Adam Perer, Hanspeter Pfister, and Alexander M Rush. Seq2seq-viz: A visual debugging tool for sequence-to-sequence models. *IEEE transactions on visualization and computer graphics*, 25(1):353–363, 2018.
- [164] Mengchen Liu, Jiaxin Shi, Zhen Li, Chongxuan Li, Jun Zhu, and Shixia Liu. Towards better analysis of deep convolutional neural networks. *IEEE transactions on visualization and computer graphics*, 23(1):91–100, 2016.
- [165] Alsallakh Bilal, Amin Jourabloo, Mao Ye, Xiaoming Liu, and Liu Ren. Do convolutional neural networks learn class hierarchy? *IEEE transactions on visualization and computer graphics*, 24(1):152–162, 2017.

- [166] Mengchen Liu, Shixia Liu, Hang Su, Kelei Cao, and Jun Zhu. Analyzing the noise robustness of deep neural networks. In *2018 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 60–71. IEEE, 2018.
- [167] Fred Hohman, Haekyu Park, Caleb Robinson, and Duen Horng Polo Chau. Summit: Scaling deep learning interpretability by visualizing activation and attribution summarizations. *IEEE transactions on visualization and computer graphics*, 26(1):1096–1106, 2019.
- [168] Minsuk Kahng, Pierre Y Andrews, Aditya Kalro, and Duen Horng Polo Chau. Activis: Visual exploration of industry-scale deep neural network models. *IEEE transactions on visualization and computer graphics*, 24(1):88–97, 2017.
- [169] Kanit Wongsuphasawat, Daniel Smilkov, James Wexler, Jimbo Wilson, Dandelion Mane, Doug Fritz, Dilip Krishnan, Fernanda B Viégas, and Martin Wattenberg. Visualizing dataflow graphs of deep learning models in tensorflow. *IEEE transactions on visualization and computer graphics*, 24(1):1–12, 2017.
- [170] James Wexler, Mahima Pushkarna, Tolga Bolukbasi, Martin Wattenberg, Fernanda Viégas, and Jimbo Wilson. The what-if tool: Interactive probing of machine learning models. *IEEE transactions on visualization and computer graphics*, 26(1):56–65, 2019.
- [171] Mark Craven and Jude W Shavlik. Extracting tree-structured representations of trained networks. In *Advances in neural information processing systems*, pages 24–30, 1996.
- [172] David Martens, BB Baesens, and Tony Van Gestel. Decompositional rule extraction from support vector machines by active learning. *IEEE Transactions on Knowledge and Data Engineering*, 21(2):178–191, 2008.
- [173] Hongyu Yang, Cynthia Rudin, and Margo Seltzer. Scalable bayesian rule lists. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3921–3930. JMLR.org, 2017.
- [174] Yao Ming, Huamin Qu, and Enrico Bertini. Rulematrix: Visualizing and understanding classifiers with rules. *IEEE transactions on visualization and computer graphics*, 25(1):342–352, 2018.

- [175] Fred Hohman, Andrew Head, Rich Caruana, Robert DeLine, and Steven M Drucker. Gamut: A design probe to understand how data scientists understand machine learning models. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2019.
- [176] Thomas Mühlbacher, Lorenz Linhardt, Torsten Möller, and Harald Piringer. Treepod: Sensitivity-aware selection of pareto-optimal decision trees. *IEEE transactions on visualization and computer graphics*, 24(1):174–183, 2017.
- [177] Stef Van Den Elzen and Jarke J van Wijk. Baobabview: Interactive construction and analysis of decision trees. In *2011 IEEE conference on visual analytics science and technology (VAST)*, pages 151–160. IEEE, 2011.
- [178] Xun Zhao, Yanhong Wu, Dik Lun Lee, and Weiwei Cui. iforest: Interpreting random forests via visual analytics. *IEEE transactions on visualization and computer graphics*, 25(1):407–416, 2018.
- [179] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International Conference on Machine Learning*, pages 2668–2677, 2018.
- [180] Yao Ming, Panpan Xu, Furui Cheng, Huamin Qu, and Liu Ren. Proto-steer: Steering deep sequence model with prototypes. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):238–248, 2019.
- [181] Shoko Sawada and Masashi Toyoda. Model-agnostic visual explanation of machine learning models based on heat map. 2019.
- [182] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.
- [183] Quan Li, Kristanto Sean Njotoprawiro, Hammad Haleem, Qiaoan Chen, Chris Yi, and Xiaojuan Ma. Embeddingvis: A visual analytics approach to comparative network embedding inspection. In *2018 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 48–59. IEEE, 2018.

- [184] Shusen Liu, Peer-Timo Bremer, Jayaraman J Thiagarajan, Vivek Srikumar, Bei Wang, Yarden Livnat, and Valerio Pascucci. Visual exploration of semantic relationships in neural word embeddings. *IEEE transactions on visualization and computer graphics*, 24(1):553–562, 2017.
- [185] Nicola Pezzotti, Thomas Höllt, Jan Van Gemert, Boudewijn PF Lelieveldt, Elmar Eisemann, and Anna Vilanova. Deepeyes: Progressive visual analytics for designing deep neural networks. *IEEE transactions on visualization and computer graphics*, 24(1):98–108, 2017.
- [186] Paulo E Rauber, Samuel G Fadel, Alexandre X Falcao, and Alexandru C Telea. Visualizing the hidden activity of artificial neural networks. *IEEE transactions on visualization and computer graphics*, 23(1):101–110, 2016.
- [187] S. Xiang, X. Ye, J. Xia, J. Wu, Y. Chen, and S. Liu. Interactive correction of mislabeled training data. In *2019 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 57–68, October 2019.
- [188] Eric Wallace, Jens Tuyls, Junlin Wang, Sanjay Subramanian, Matt Gardner, and Sameer Singh. Allenlp interpret: A framework for explaining predictions of nlp models. *arXiv preprint arXiv:1909.09251*, 2019.
- [189] Jonathan K Pritchard, Matthew Stephens, and Peter Donnelly. Inference of population structure using multilocus genotype data. *Genetics*, 155(2):945–959, 2000.
- [190] Ville Satopaa, Jeannie Albrecht, David Irwin, and Barath Raghavan. Finding a “needle” in a haystack: Detecting knee points in system behavior. In *2011 31st international conference on distributed computing systems workshops*, pages 166–171. IEEE, 2011.
- [191] Kenneth P Burnham and David R Anderson. A practical information-theoretic approach. *Model selection and multimodel inference, 2nd ed. Springer, New York*, 2, 2002.
- [192] Inderjit S Dhillon, Subramanyam Mallela, and Dharmendra S Modha. Information-theoretic co-clustering. In *Proceedings of the ninth ACM SIGKDD*

- international conference on Knowledge discovery and data mining*, pages 89–98, 2003.
- [193] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388, 2002.
  - [194] Mohammad Ghoniem, Jean-Daniel Fekete, and Philippe Castagliola. On the readability of graphs using node-link and matrix-based representations: a controlled experiment and statistical analysis. *Information Visualization*, 4(2):114–135, 2005.
  - [195] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. *arXiv preprint arXiv:1806.10758*, 2018.
  - [196] Georges Reeb. Sur les points singuliers dune forme de Pfaff complètement intégrable ou d'une fonction numérique. *Comptes Rendus des séances de l'Académie des Sciences*, 222(847-849):76, 1946.
  - [197] Herbert Edelsbrunner and John Harer. *Computational topology: an introduction*. American Mathematical Soc., 2010.
  - [198] Amirata Ghorbani, Abubakar Abid, and James Zou. Interpretation of neural networks is fragile. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3681–3688, 2019.
  - [199] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *arXiv preprint arXiv:1810.03292*, 2018.
  - [200] Chih-Kuan Yeh, Cheng-Yu Hsieh, Arun Sai Suggala, David I Inouye, and Pradeep Ravikumar. On the (in) fidelity and sensitivity for explanations. *arXiv preprint arXiv:1901.09392*, 2019.
  - [201] Zhong Qiu Lin, Mohammad Javad Shafiee, Stanislav Bochkarev, Michael St Jules, Xiao Yu Wang, and Alexander Wong. Do explanations reflect decisions?

- A machine-centric strategy to quantify the performance of explainability algorithms. *arXiv preprint arXiv:1910.07387*, 2019.
- [202] Mengjiao Yang and Been Kim. Benchmarking attribution methods with relative feature importance. *arXiv preprint arXiv:1907.09701*, 2019.
  - [203] Gurjeet Singh, Facundo Mémoli, and Gunnar E Carlsson. Topological methods for the analysis of high dimensional data sets and 3d object recognition. *SPBG*, 91:100, 2007.
  - [204] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological Persistence and Simplification. *Discrete Comput. Geom.*, 28(4):511–533, 2002.
  - [205] Pankaj K. Agarwal, Herbert Edelsbrunner, John Harer, and Yusu Wang. Extreme elevation on a 2-manifold. *Discrete Comput. Geom.*, 36:357–365, 2006.
  - [206] David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete Comput. Geom.*, 37:103–120, 2007.
  - [207] Frédéric Chazal, Brittany Fasy, Fabrizio Lecci, Bertrand Michel, Alessandro Rinaldo, Alessandro Rinaldo, and Larry Wasserman. Robust topological inference: Distance to a measure and kernel distance. *The Journal of Machine Learning Research*, 18(1):5845–5884, 2017.
  - [208] Alper Sarikaya, Michael Gleicher, and Danielle Albers Szafir. Design factors for summary visualization in visual analytics. In *Computer Graphics Forum*, volume 37, pages 145–156. Wiley Online Library, 2018.
  - [209] Michelle X Zhou and Steven K Feiner. Visual task characterization for automated visual discourse synthesis. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 392–399, 1998.
  - [210] Niklas Elmquist and Jean-Daniel Fekete. Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines. *IEEE Transactions on Visualization and Computer Graphics*, 16(3):439–454, 2009.

- [211] John Wenskovitch, Ian Crandell, Naren Ramakrishnan, Leanna House, and Chris North. Towards a systematic combination of dimension reduction and clustering in visual analytics. *IEEE transactions on visualization and computer graphics*, 24(1):131–141, 2017.
- [212] Brian Barr, Ke Xu, Claudio Silva, Enrico Bertini, Robert Reilly, C Bayan Bruss, and Jason D Wittenbach. Towards ground truth explainability on tabular data. *arXiv preprint arXiv:2007.10532*, 2020.
- [213] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.
- [214] Jack W Smith, James E Everhart, WC Dickson, William C Knowler, and Robert Scott Johannes. Using the adap learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the annual symposium on computer application in medical care*, page 261. American Medical Informatics Association, 1988.
- [215] Scott M Lundberg, Gabriel G Erion, and Su-In Lee. Consistent individualized feature attribution for tree ensembles. *arXiv preprint arXiv:1802.03888*, 2018.
- [216] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.
- [217] Cosimo Izzo, Aldo Lipani, Ramin Okhrati, and Francesca Medda. A baseline for Shapely values in MLPs: from missingness to neutrality. *arXiv preprint arXiv:2006.04896*, 2020.
- [218] Sérgio Moro, Paulo Cortez, and Paulo Rita. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014.
- [219] Gromit Yeuk-Yin Chan, Fan Du, Ryan A Rossi, Anup B Rao, Eunyee Koh, Cláudio T Silva, and Juliana Freire. Real-time clustering for large sparse online visitor data. In *Proceedings of The Web Conference 2020*, pages 1049–1059, 2020.

- [220] Gromit Yeuk-Yin Chan, Tung Mai, Anup B Rao, Ryan A Rossi, Fan Du, Cláudio T Silva, and Juliana Freire. Interactive audience expansion on large scale online visitor data. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2021.
- [221] Gromit Yeuk-Yin Chan, Luis Gustavo Nonato, Alice Chu, Preeti Raghavan, Viswanath Aluru, and Cláudio T Silva. Motion browser: Visualizing and understanding complex upper limb movement under obstetrical brachial plexus injuries. *IEEE transactions on visualization and computer graphics*, 26(1):981–990, 2019.