Institutional Repository - Research Portal
Dépôt Institutionnel - Portail de la Recherche
researchportal.unamur.be
University of Namur

THESIS / THÈSE

DOCTOR OF SCIENCES

Integration of Constraints into Dimensionality Reduction Methods for Visualization

Vu, Viet Minh

Award date:
2021

Awarding institution:
University of Namur

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 12. Oct. 2024

Integration of Constraints into Dimensionality Reduction Methods for Visualization

Viet Minh Vu

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Science in Computer Science

Composition of the Jury:

Prof. Wim Vanhoof, President, *University of Namur, Belgium*
Prof. Benoît Frénay, Supervisor, *University of Namur, Belgium*
Dr. Adrien Bibal, Co-supervisor, *Université catholique de Louvain, Belgium*
Prof. Bruno Dumas, *University of Namur, Belgium*
Prof. John Lee, *Université catholique de Louvain, Belgium*
Prof. Michel Verleysen, *Université catholique de Louvain, Belgium*

December 2021

Cover design: © Presses universitaires de Namur

© Presses universitaires de Namur & Viet Minh Vu
Rue Grandgagnage 19
B – 5000 Namur (Belgium)

Reproduction of this book or any parts thereof, is strictly forbidden for all countries, outside the restrictive limits of the law, whatever the process, and notably photocopies or scanning.
Printed in Belgium.

ISBN: 978-2-39029-152-7
Registration of copyright: D/2021/1881/20

Contents

1 Introduction **1**
1.1 Context, Scope, and Motivation 1
1.2 Research Problems 3
1.3 Contributions 4

2 Background **7**
2.1 Background on Dimensionality Reduction 7
2.2 Background on Common DR Methods 12
2.2.1 PCA 12
2.2.2 MDS 13
2.2.3 SNE-based methods 16

3 A Survey of Integrating Constraints into DR Methods **21**
3.1 Scope and Methodology 22
3.2 Instance-level Constraints 25
3.2.1 Constraints on Fixed Positions of Points 25
3.2.2 Pairwise Constraints 32
3.2.3 Triplet Constraints 37
3.3 Dataset-level Constraints 38
3.3.1 Constraints on Features 39
3.3.2 Constraints from Labeled Data 40
3.3.3 Multi-Aspect Constraints 47
3.4 Discussion 51

4 Integration of Hierarchical Constraints into t-SNE **57**
4.1 The Need for Visualizations with Hierarchical Structures 59
4.1.1 Problems with t-SNE Visualizations 60
4.1.2 Hierarchical Structure in a Visualization 62
4.1.3 Our Contribution 64
4.1.4 Related Work 64
4.2 From Desired Hierarchical Structures to Hierarchical Constraints 66
4.2.1 Child-Parent Relationship in Hierarchical Constraints . . . 67
4.2.2 Sibling Relationship in Hierarchical Constraints 70
4.2.3 Usefulness of the Hierarchical Triplet Constraints 70
4.3 HCt-SNE: Hierarchical Constraints with t-SNE 72
4.3.1 Triplet Loss with Margin 72
4.3.2 Enforcing Hierarchical Constraints as a Regularization Term 74
4.3.3 Optimization through a Tree Traversal Algorithm 78
4.4 Experimental Results of HCt-SNE 81
4.4.1 Experimental Setup 81
4.4.2 Analysis of Qualitative Results 83
4.4.3 Analysis of Quantitative Results 88
4.5 Discussion 90

5 Integration of Fixed-position Constraints into Probabilistic DR Methods **95**
5.1 Probabilistic Approach in ML and in DR 98
5.1.1 Brief Introduction to a Probabilistic Approach in ML . . . 98
5.1.2 Probabilistic DR Methods 100
5.2 Proposed Unified Probabilistic DR Framework with Constraints . 101
5.2.1 General Workflow for Probabilistic Modeling 101
5.2.2 Constraint Integration using Informative Prior 105
5.3 Concrete Example 1: PMDS Model and interactive PMDS 107
5.3.1 Probabilistic Multidimensional Scaling 108
5.3.2 Interactive PMDS Model 114

5.4 Concrete Example 2: PPCA Model and interactive PPCA 120
5.4.1 Probabilistic Principal Components Analysis 120
5.4.2 Interactive PPCA Model 125
5.5 Experimental Results 127
5.5.1 Case Study 1: Interaction for Creating Transformation Effect 129
5.5.2 Case Study 2: Understandable Axes with Image Dataset . . 131
5.5.3 Case Study 3: Interpretable Axes with Tabular Dataset . . 132
5.6 Discussion 134
5.6.1 Usage of the Proposed Models 134
5.6.2 Limitations 136

6 Constraint-Preserving Score for Visualization Assessment 139
6.1 Quality Metrics for Visualization Assessment 142
6.1.1 How to Measure the Quality of a Visualization 143
6.1.2 How to Choose the Best Visualization 144
6.2 The Proposed Constraint-Preserving Score 147
6.2.1 Pairwise Constraints for Visualization Assessment 147
6.2.2 Derivation of the Constraint-Preserving Score 148
6.2.3 Advantages of f_{score} 150
6.3 Experimental Results with f_{score} 151
6.3.1 Experimental Setup 152
6.3.2 Comparison of f_{score} and Other Qualities Metrics 154
6.4 Empirical Characteristics of f_{score} 161
6.4.1 f_{score} as a Well-Behaved Function 161
6.4.2 Stability of f_{score} 162
6.4.3 Flexibility of f_{score} 164
6.5 Application of f_{score} for a Hyperparameter Tuning Problem . . . 167
6.5.1 Bayesian Optimization for Hyperparameter Tuning 168
6.5.2 Using f_{score} in Bayesian Optimization 169
6.6 Discussion 173

7 Discussion **175**
7.1 Discussion on Our Approach 175
7.2 Discussion on Our Limitations 176
7.3 Discussion on Constraint Acquisition 177
7.4 Discussion on Visualization Assessment 178
7.5 Discussion on Interactive Tools 179

8 Conclusion and Future Work **183**

9 Appendices **187**
9.1 iPDR Additional Results with iPPCA 187
9.1.1 Three case studies with iPPCA 187
9.1.2 Interaction with iPMDS with incomplete input data 189
9.2 Quality Metrics 192
9.3 Analysis of AUC Curves for Embeddings of SNE-based Methods . 194
9.4 Choosing the Regularization Coefficient α in HCt-SNE 195
9.5 Pairwise Constraints with DR Methods 197
9.5.1 Pairwise Constraints to Enhance Gram Matrix for PCA . . 197
9.5.2 Pairwise Constraints to Enhance the Affinities in t-SNE . . 198

Bibliography **201**

Abstract

When working with high-dimensional data, visualization techniques are useful tools to help us to discover patterns. However, the visualization results do not always match the users' expectations. An effective mechanism that allows human users to interact with machine learning systems is thus necessary. This thesis addresses the problem of combining users' feedback and dimensionality reduction (DR) techniques for visualization. From a machine learning viewpoint, users' feedback is considered as constraints. We focus on transforming users' constraints into adequate terms that can be used to enrich DR methods or to assess the visualization. Four contributions are made in this thesis.

First, we conduct a comprehensive survey on constraint integration in DR methods. This survey provides a new viewpoint to reorganize existing constrained DR methods by constraint types. Second, a long-standing problem of t-SNE, one of the most widely used visualization methods, is tackled. We propose to integrate a hierarchical tree into t-SNE embedding to enhance its global structure by the semantic information extracted from the tree. Third, fixed-position constraints are integrated into probabilistic models by our unified framework. Users' prior knowledge about the position of points is used to create informative prior distributions for a probabilistic model. Concrete interactive methods derived from this framework are used to allow users to manipulate the visualizations and to create understandable axes. Fourth and finally, we propose a novel way to use pairwise constraints for visualization assessment. Users can encode their expected structure in pairwise constraints of similar/dissimilar objects. Our proposed score measures how well these input constraints are preserved in order to determine the quality of a visualization.

Chapter 1

Introduction

The thesis is entitled "Integration of Constraints into Dimensionality Reduction Methods for Visualization". The constraints here are different kinds of users' feedback when they interact with the visualization produced by machine learning methods. The topic of this thesis is at the frontier of different domains: Human-Computer Interaction (HCI), Information Visualization (InfoVis), and Machine Learning (ML). Even though the topic in this thesis is approached from the lens of machine learning participants, technical terms will be explained in a simple language with as many visual examples as possible.

1.1 Context, Scope, and Motivation

More and more data are now produced, and we always want to exploit valuable information in those data. Traditional data processing system takes data and human-encoded rules to produce models that can help humans make decisions. New machine learning systems learn rules or patterns from data automatically. Different ML methods are proposed to process different kinds of structured tabular data, unstructured data like text or images, and even domain-specific data. Real-world data are often high-dimensional, for example, a small color image of size 128×128 has approximately 50000 pixels. A language model, in order to represent a word or a short text, also demands a very large number of dimensions that is equal to the size of the vocabulary of a language. A large dataset of such a high number of dimensions is thus difficult to process efficiently.

Moreover, when analyzing a dataset, people often want to quickly find out patterns in their data. They can apply exploratory data analysis methods, including visualization techniques, to get insights from data. Thanks to these methods, we can plot directly the whole dataset into a 2- or 3-dimensional space for easily observing and discovering visual patterns. However, how can we go from high dimensional (HD) data to a low dimensional (LD) visualization? Dimensionality reduction (DR) methods, a group of specific unsupervised machine learning techniques, aim to reduce dimensionality while preserving import characteristics and patterns in the data. DR methods for visualization are the main research topic of this thesis. A brief introduction to the DR problem and common visualization methods are presented in Chapter 2.

Nowadays, carefully designed and tested ML models can be deployed to process real-world data and make decisions without any intervention from humans. However, there are not so many such models. ML workflow is usually iterative. Researchers re-analyze their problems, re-design their ML algorithms, and re-evaluate their models on different kinds of data. When end-users or expert users apply these algorithms to their data in their domains, the algorithms do not always work *like a charm* nor solve their problem automatically. Much effort is taken to choose different algorithms for different problems, tune hyperparameters, and interpret results. What if the results of ML models do not match with the users' knowledge or their expectations? In the context of visualization methods, for example, the position of points in the visualization may not make sense, the similar data points of similar objects may be placed far apart and group patterns are not well revealed, etc. In these cases, it is necessary to have a mechanism allowing human users to interact directly with ML systems. In the scope of this thesis, we tackle the problem of integrating users' feedback into DR methods for visualization.

From a machine learning perspective, users' feedback, human prior knowledge, or domain expert knowledge can be considered as *constraints* for ML algorithms. If we want to build usable interactive visualization methods, the following questions in HCI and InfoVis domains cannot be ignored. Who are our users? How to design an interactive scenario? How do users interact with the visualization and how to collect their feedback? How to evaluate the quality of a visualization? We target expert users who know about DR methods with their pros and cons and want to try to incorporate domain knowledge into

their visualizations. We also target end-users who do not know the underlying mechanism of DR methods but want to interact with the visualization to give their feedback. We will answer the questions about the interactive tools that allow users to interact with the visualization, the task-based scenario used in our case studies, and the problem of visualization assessment in each chapter where these questions involve.

1.2 Research Problems

Three following research questions guide our research.

1. **RQ1**: *How to transform the users' feedback into constraints that can be used in DR methods?* This is the biggest question, for which one should consider what is a users' constraint and how to represent different kinds of users' constraints in the form of a term used by visualization methods.

2. **RQ2**: *After representing users' constraints in a suitable form, how to integrate these constraints into DR methods?* This question is usually considered together with the **RQ1**. A suitable representation of constraints can be applied for different DR methods. This question involves the problem of constraint optimization.

3. **RQ3**: *Can the users' opinions expressed in the term of constraints be used to assess the visualization?* When constraints are integrated into ML methods, we usually examine whereas these constraints are satisfied or violated. Inversely, we can also look at the points in the visualization and examine whereas these points satisfy or violate a set of pre-defined input constraints. This approach suggests a potential usage of constraint for visualization assessment.

The goal of this thesis is to combine human knowledge/users' feedback with widely used DR methods for visualization. We expect to enhance common visualization methods with new properties brought by user constraints. That can be done by integrating directly the user constraints into visualizations methods. We also want to consider users' opinions to evaluate the quality of a visualization. That can be done by measuring how well the users' constraints

are preserved in a visualization. With these ideas to respond to the research questions, our contributions in this thesis are summarized below.

1.3 Contributions

The *red thread* of this thesis is our solutions to combine users' constraints and DR methods. Two ideas are proposed for this combination: (i) integrating directly users' constraints into DR methods, and (ii) using these constraints as a method-agnostic measure for visualization assessment. With a focus on different kinds of users' constraints that can be used together with DR methods, we have made the following contributions.

First, a study about what can be considered as constraints and different constraint integration methods are conducted in our comprehensive survey presented in Chapter 3. This survey summarizes how the **RQ1** and **RQ2** are answered in the literature.

Second, a new kind of *hierarchical* constraints is integrated into t-SNE or others SNE-based methods. Chapter 4 introduces our proposed HCt-SNE method, in which users' knowledge about the semantic relation between data groups is described in the form of a tree. High-level abstraction in the relationship of tree nodes is translated into instance-level triplet constraints (**RQ1**), which are then formulated as a differentiable regularization term to be optimized together with t-SNE (**RQ2**). This work tackles an important issue of t-SNE that cannot preserve global structure in the visualization.

Third, we propose to integrate *fixed-position* constraints into probabilistic DR methods through a unified framework. When users observe a visualization, they can move points and place them to their desired position to form fixed-position constraints. Our idea is to consider users' prior knowledge about the position of points as an *informative prior distribution* for a probabilistic model (to target **RQ1**). We also propose a framework for integrating this kind of constraint for different probabilistic DR models (and thus solve **RQ2** by an inference algorithm). Chapter 5 introduces this idea as well as two novel methods named *interactive PPCA* and *interactive PMDS* derived from our proposed framework. The author of these methods experimented with the interactive case studies by himself.

Fourth and finally, pairwise constraints between similar and dissimilar objects are used to evaluate the quality of the visualizations produced by SNE-based methods (**RQ3**). Chapter 6 introduces our *constraint-preserving score*, a novel way to use users' constraints as an efficient and method-agnostic quality measure. The need of users is encoded in pairwise constraints. These constraints are then quantified to measure how well they are preserved in the visualizations. This measure has several good properties while still having a computational advantage. We also propose to use a model-based optimization approach (called Bayesian optimization) combined with our score to search through the space of all possible visualizations to find the best one.

Instead of focusing on one kind of constraint that works with one complex method, we aim to combine many kinds of constraints with simple and widely used DR methods. Advantages and limitations of our approach, as well as related topics regarding constraint acquisition and user-evaluation, are also discussed in Chapter 7. Chapter 8 concludes our work and opens new perspectives for future work. As a result, the following papers are produced during this thesis under the dedicated guidance of my supervisors:

1. Viet Minh Vu, Adrien Bibal, and Benoit Frénay. Integrating constraints into dimensionality reduction methods: a survey. *submitted to Neurocomputing*, 2021d

2. Viet Minh Vu, Adrien Bibal, and Benoit Frénay. Constraint preserving score for automatic hyperparameter tuning of dimensionality reduction methods for visualization. *IEEE Transactions on Artificial Intelligence*, 2:269–282, 2021a

3. Viet Minh Vu, Adrien Bibal, and Benoit Frénay. HCt-SNE: Hierarchical constraints with t-SNE. In *Proc. IJCNN*, 2021b

4. Viet Minh Vu, Adrien Bibal, and Benoit Frénay. iPMDS: Interactive probabilistic multidimensional scaling. In *Proc. IJCNN*, 2021c

5. Viet Minh Vu and Benoit Frénay. User-steering interpretable visualization with probabilistic principal components analysis. In *Proc. ESANN*, pages 349–354, 2019

Chapter 2

Background

This chapter summarizes the necessary background knowledge on the dimensionality reduction (DR) problem, provides clear definitions for the common technical terms, as well as an overview (with new insights) of common DR methods used in this thesis.

2.1 Background on Dimensionality Reduction

Machine learning methods are now solving problems that become harder and harder. The data that these methods work on are also more and more complex. Beyond the traditional UCI datasets with dozens of understandable features, real-world datasets often have a much higher number of dimensions, which makes it hard to process, analyze and visualize. When working with these data, we usually make an assumption that many dimensions in the original data are redundant, and the data in fact lie on a much lower-dimensional space. Dimensionality reduction (DR) techniques aim to reduce the dimensionality of the data while preserving several important characteristics.

Usage of Dimensionality Reduction Methods

We will go from DR methods in general to specific methods designed for visualization. Reducing the dimension of the data can facilitate subsequent classification, clustering, or visualization tasks. The reduced data in the low-

dimensional (LD) space are called the *embedding*. Since the embedding can retain important information in the original data while having much lower dimensions, the computation on this embedding can be more efficient. Therefore, DR methods, in general, can be used as a preprocessing method. When the embedding lies on a 2− or 3−dimensional space, it can be plotted in a scatter plot to help users easily observe patterns in the reduced data. The embeddings in 2D or 3D are not usually used for subsequent classification or clustering tasks but are used for visualization instead. In fact, more than 3 dimensions can be plotted at the same time on a scatter plot, in which three dimensions are used for the locations while additional dimensions can be represented by size, shape, color, transparency, marker, ... of the scatted point. However, it is not easy to identify patterns in the scatter plot with too much encoded information. For that reason, only the reduced data in 2D or 3D are represented in the scatter plot, with additional information about class labels are used to color the points to identify their group/class. Without confusion, in this thesis, when we mention DR methods without specifying their purpose, we imply DR methods for visualization. And also, the term *embedding* and *visualization* are used interchangeably to indicate the reduced data in 2D or 3D that can be visualized in a scatter plot.

Figure 2.1 shows an example when applying DR methods on synthetic 3D data. The first two plots (A0, A1) show a cloud of 3D points at two different angles. The underlying structure of these points is shown by the surface on the plots (B1, B2). Two 2D flat planes that attempt to approximate this surface are shown in the plots (C1, C2), but none of those seems to be adequate. However, we can still *'project'* this point cloud into a 2D plane. For example, the plots (B3) and (C3) show the results of a *nonlinear* and a *linear* method, respectively. This example shows that the 3D data can be somehow reduced to 2D data by different DR methods that we will introduce later. In the following section, we introduce two approaches of DR methods: a linear approach that projects the HD data into a lower-dimensional subspace, and a nonlinear approach that tries to find the underlying structures of the data.

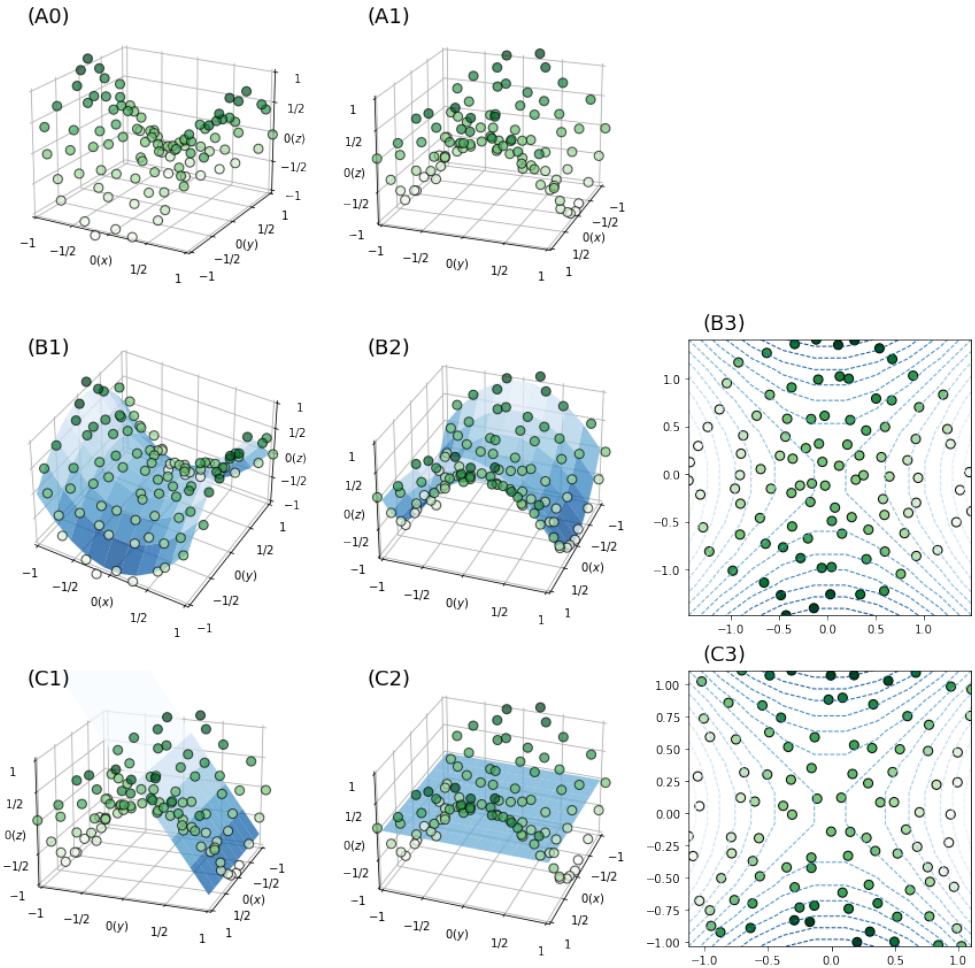


Figure 2.1: Example of 2D projections of 3D data. (A0, A1): a cloud of 3D data at two different angles. (B1, B2): the surface of the underlying structure of the data. (B3): the result of a nonlinear DR method that approximates the underlying surface in (B2). (C1, C2): two 2D flat planes to approximate the underlying surface. (C3): the result of a linear DR method.

Dimensionality Reduction Problem

One of most common approaches to tackle DR problems is to find a projection matrix $\boldsymbol{P}_{\mathrm{q}\times\mathrm{p}}$ that map the data $\boldsymbol{X}_{\mathrm{N}\times\mathrm{p}}$ from the original high-dimensional (HD) space of p dimensions to an LD space of $q \ll p$ dimensions. This approach leads

to a family of linear methods where the most famous unsupervised method is Principal Component Analysis (PCA - (Hotelling, 1933; Pearson, 1901)). The mapping is done through a projection $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{P}^T$, where each column of the embedding $\boldsymbol{Y}$ is a linear combination of every column of the original data. Not every projection matrix is a solution, the combination coefficients in the projection matrix $\boldsymbol{P}$ should indeed be learned from data (more details come later). This is the most intuitive and simplest way to see DR methods. However, we cannot always find a *good* linear projection to reduce our data. Let's put aside how to define a *good* criterion and focus on the problem of reducing the dimensionality of data. Chapter 6 will be devoted to explaining the problem of evaluating the quality of a visualization. Quality metrics for visualization assessment are also introduced in the context where they are used.

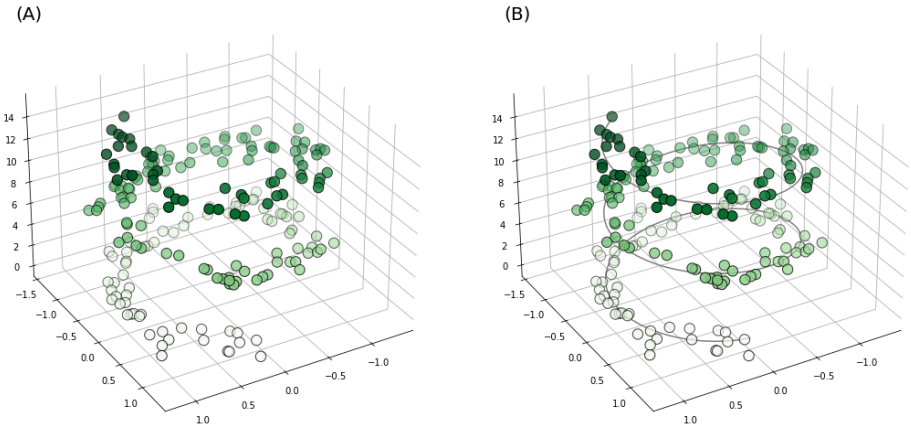


Figure 2.2: Example of a 1D manifold in a 3D plane.

Let us consider another approach for DR from the perspective under the *manifold hypothesis*, which states: "data tend to lie near a low dimensional manifold" (Fefferman et al., 2016). We are not going to state the strict mathematical definition of a manifold but explain it in a simple language. Let's look at a simple example in Figure 2.2 with three-dimensional data plotted on the left. Even though these points are represented in a 3D space, we can find that they lie on a spiral, shown on the right.

Based on this example, we will define several important terms. In fact, this spiral is a *line* where the data points lie on. The points are marked by

darkness: very dark points at one end of the line, very light points at the other end and the darkness of point reduces when traveling from one end to the other end. The position of points can be now approximately encoded by its darkness, which is the information represented by a scalar in a one-dimensional space. This line is, roughly speaking, called the *manifold* of the data points, which can be understood as an *intrinsic geometrical structure* of the data. We roll the line and *put this line on a 3D space* to form the spiral structure as shown in this figure. The process of *putting the line* on a 3D space is formulated as *embedding the manifold into another space.*

In order to understand what is a manifold for real data, let us take a look at the following pictures of a box captured at different angles from the COIL20 dataset(Nene et al., 1996)

As they are the same object, the only property to distinguish them is the rotation angle. We can imagine that these pictures lie on an LD manifold such as a line shown above or a line in a 2D space on the right.

If this LD manifold is embedded (is put) in another space of 128×128 dimensions, we can see that, these pictures are very specific samples in a huge space of all possible 256^{16384} gray-scale images. Given the COIL20 dataset of 128×128 gray-scale images, we would like to find the underlying structure of the rotated objects such as the manifold shown before. The manifold of the box in the above figure on the right is found by t-SNE, one of the most successful nonlinear DR methods.

Now we restate the manifold hypothesis: HD data are assumed to come from an LD manifold that is embedded in a much higher-dimensional space. The inverse problem of going from HD data to find the LD embedding manifold is thus called *manifold learning.* The LD representation of HD data is also called the *embedding* or the *manifold.* Due to the complexity in the HD data, a direct linear mapping may not be a robust way to find the LD representation. Since there may be no linear relationship between the HD and LD data, most

DR methods are nonlinear. Common nonlinear DR methods are reviewed and analyzed by Lee and Verleysen (2007), and empirically compared by van der Maaten, Postma, and van den Herik (2009).

2.2 Background on Common DR Methods

Three of the most widely used DR methods for visualization are introduced in this section, including PCA, MDS, t-SNE (and related methods in its family). They are also the fundamental methods, based on them, we develop new interactive DR methods that incorporate users' feedback/knowledge. Even though these methods are very common, we try to provide new insights and see them from different perspectives.

2.2.1 PCA

Principal Component Analysis (PCA), the most well-known DR method, has a long history of more than a century. It was originally introduced by Pearson (1901) as a problem of finding a *"best-fitting"* straight line or plane for a system of points in two- or higher-dimensional space. Until now, the core problem of PCA - a type of *eigenvalue* problem, is still being studied. Most recently, PCA is formulated as a competitive multi-agent game called *EigenGame* (Gemp et al., 2021). PCA is a long-standing and standard technique in economics, psychology, bioinformatics, and many other application domains thanks to its simplicity and efficiency for all kinds of real-world data. In these domains, the dataset usually has many variables and the users cannot understand them or the relationship between them. This will be more useful if users can focus on a few important (principal) features. PCA is a ubiquitous data pre-processing step to produce LD representations for both supervised and unsupervised machine learning problems. In technical terms, PCA *simplifies* data in the original feature space by producing a small number of features that retain trends (directions of dominant variances) and patterns (global structure) to help interpret data (Lever et al., 2017). For that reason, PCA is widely used for feature extraction, lossy data compression, and data visualization.

There are two equivalent definitions of the PCA problem. As defined as a linear projection by Pearson (1901), PCA minimizes the average projection error,

measured by squared distances between the data points and their projections:

$$\min_{\boldsymbol{W}} ||\boldsymbol{X}_{\mathrm{N}\times\mathrm{p}} - \boldsymbol{Z}_{\mathrm{N}\times\mathrm{q}}\boldsymbol{W}_{\mathrm{q}\times\mathrm{p}}^{T}||^2 = \min_{\boldsymbol{W}} \frac{1}{N}\sum_{i=1}^{N} ||\boldsymbol{x}_i - \boldsymbol{W}\boldsymbol{z}_i||^2, \tag{2.1}$$

where $\boldsymbol{Z}_{\mathrm{N}\times\mathrm{q}} = \boldsymbol{X}_{\mathrm{N}\times\mathrm{p}}\boldsymbol{W}_{\mathrm{p}\times\mathrm{q}}$ is the projection of the data matrix $\boldsymbol{X}$ onto the subspace formed by the columns of the projection matrix $\boldsymbol{W}$. Later, Hotelling (1933) defined PCA as an orthogonal projection onto a principal subspace that maximizes the variance of the projected data. The information in a dataset can be measured by how the instances spread in the data space (Deisenroth et al., 2020). For that reason, the variance is a good criterion to measure how well the reduced data can explain the original data.

PCA is a mature technique, however, it is still considered under new perspectives, for example in distributed environment (Balcan et al., 2014; Liang et al., 2013) and parallel paradigm (Gemp et al., 2021) in order to be applied for large scale datasets. We will also look at PCA under another perspective using a probabilistic approach as introduced by Tipping and Bishop (1999) for integrating with users' constraints (see Chapter 5).

2.2.2 MDS

Multidimensional Scaling (MDS) can be referred to as a family of techniques, traditionally used for finding psychological dimensions hidden in the data, for testing structural hypotheses, or generally for data exploration and visualization (Borg and Groenen, 2005b, Chap. 1). MDS can also be mentioned as a general problem of finding an LD embedding of the data in which the *proximities* between pairs of instances in this LD space respect the corresponding *proximities* in the data space. Metric MDS refers to the MDS methods when the proximity is measured by a metric distance such as the Euclidean distance (Torgerson, 1952). Non-metric MDS uses ranks instead of distances (Kruskal, 1964a,b).

We focus on metric MDS for visualization, in which the goal is to arrange points in a 2D space in such a way to approximate the HD Euclidean pairwise distances:

$$\text{Stress} = \sqrt{\sum_{1\leq i<j\leq N} \Big(d(\boldsymbol{x}_i, \boldsymbol{x}_j) - d(\boldsymbol{y}_i, \boldsymbol{y}_j)\Big)^2}, \tag{2.2}$$

where $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $d(\boldsymbol{y}_i, \boldsymbol{y}_j)$ are Euclidean distance between the points in the HD and LD space. Minimizing this stress function can be solved efficiently using iterative algorithms like SMACOF (Borg and Groenen, 2005a).

This method is simple and efficient for small datasets with few features. When the number of dimensions of the original data increases, MDS cannot well preserve all the pairwise distances due to a *"surprising behavior of distance metrics in HD space"* (Aggarwal et al., 2001). We will first introduce a common issue of the distance metrics and then analyze the pitfall of the distance-preserving approach in MDS.

The work of Aggarwal et al. (2001); Beyer et al. (1999) states that *"the ratio of the distances of the nearest and farthest neighbors in high dimensional space is almost 1 for a wide variety of data distributions and distance functions"*. Euclidean distance and other metric distances are intuitive in LD space but counter-intuitive in high dimensions stated by (Domingos, 2012) as *"intuition fails in high dimensions"*. The cause of this issue is attributed to the *curse of dimensionality*, a cornerstone problem in machine learning/data mining introduced by Bellman (1966). A simple explanation of this problem can be found in (Bishop, 2006, Sec. 1.4), saying that most of the mass of a multivariate Gaussian does not concentrate near the mean but around its "shell" - distant from the center. In a very low dimensional space, sampled points from a Gaussian locates near the mean that gives us a familiar bell shape of its probability density function. In contrast, sampled points from a highly multivariate Gaussian lie near the surface of a hypersphere. This phenomenon makes the points in an HD space are almost distant from each other and thus their pairwise distances are nearly the same.

In order to illustrate this phenomenon, we sample points in three different multivariate Gaussian distributions of 2, 100, and 200 dimensions and calculate the pairwise Euclidean distances between the points in each Gaussian. The histograms of these distances in three-dimensional spaces are plotted in Figure 2.3. In 2D, there are always close pairs with distance values close to zero. In 100D and 200D, the distances concentrate on large values and there is no close pairs[1]. Through theoretical work and empirical data shown above, one

[1]This distance concentration phenomenon for uniformly distributed data is also introduced in Figure 2.9 in the Ph.D. dissertation of François (2007) and in the keynote lectures of Verleysen (2020) at IJCCI2020. A similar finding is shown in the lecture notes of Dmitry

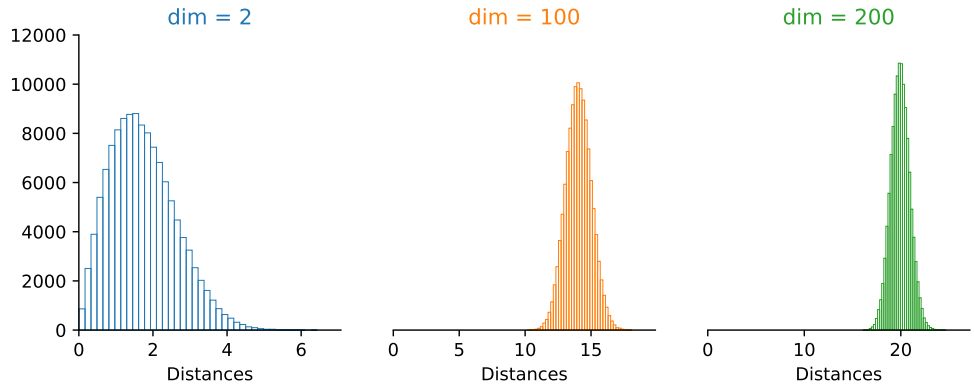


Figure 2.3: Distributions of pairwise distances between 500 points sampled from each multivariate Gaussian distribution of 2, 100, and 200 dimensions. In 2D, the value of pairwise distances ranges from 0, indicating that there are many close pairs. In higher dimensional space, the distance values concentrate on very high values, while there are absolutely no close pairs.

should notice two issues. First, there is no *small distance* in an HD space as shown in Figure 2.3 due to the curse of dimensionality[2]. Second, the pairwise distances in an HD space concentrate on very high values due to the points being almost equal-distant and far from each other.

From the above arguments, minimizing directly the distance distortion between the HD and LD spaces as done by MDS may not be adequate for preserving pairwise distances of HD data. The distances in an LD space can be *discriminated*, meaning that there are always pairs of very small and very large distances. In contrast, distances in an HD space do not discriminate anymore (Verleysen, 2020). Moreover, the distribution of pairwise distances in two spaces is different as visually inspected in Figure 2.3. Therefore, direct distance matching by the stress in Equation 2.2 may not work for HD data.

This issue is addressed in a probabilistic approach, which reformulates the distance-preserving problem. Instead of matching two sets of pairwise distances using the distortion, one can learn a probabilistic transformation to map the distributions of distances in the HD and LD space. The distribution of distances

Kobak, publicly available at https://github.com/dkobak/dkobak.github.io.

[2]An exponentially large number of points is needed to fill the HD space. With a finite random sampling process, it is rare to sample two points that are close in Euclidean distance.

in an LD space can be formulated mathematically while the distribution of distances in a data space can be measured from observed data. Using a standard *change of variables of probability density functions* can help to learn a transform for matching two distributions. This solution is derived in Section 5.3.2. A better solution is to give up preserving distances and try to preserve neighborhood information instead. This idea leads to a family of neighbor-preserving methods introduced right below.

2.2.3 SNE-based methods

Neighbor-preserving methods, starting with Stochastic Neighbor Embedding (SNE) (Hinton and Roweis, 2003) and its successors t-SNE, LargeVis, UMAP, aim to preserve neighborhood information (local structures) in the HD data. These methods generally consist of two main steps. First, a neighborhood graph is constructed from the HD data. This step requires an hyperparameter that determines the set of k-nearest neighbors (kNN), called n_neighbors in UMAP and *perplexity* in t-SNE and LargeVis. This kNN graph is weighted in different ways to transform proximities in the data into probabilities of being neighbors. Second, these probabilities are used to find the position of the data points in an LD space while preserving the neighborhood information.

Constructing the kNN graph requires computing pairwise distances between all N instances and has a complexity of $\mathcal{O}(N^2)$. t-SNE (Maaten and Hinton, 2008) constructs the exact kNN graph and thus cannot scale with large datasets. LargeVis (Tang et al., 2016) approximates a very accurate kNN graph by using random projection trees to obtain neighborhood candidates for each instance. In t-SNE and LargeVis, edges in the kNN graph are weighted by an isotropic Gaussian kernel with an adapted bandwidth derived from the perplexity parameter. UMAP (McInnes et al., 2018a) has a different theoretical foundation and uses a more sophisticated topological data analysis to model local connectivity by a fuzzy topological structure.

In the embedding space, all three methods create a neighborhood graph and transform it to neighborhood probabilities using the Student's t-distribution (UMAP uses a similar but more general function). A graph layout problem must then be solved to match the neighborhood probabilities in HD and LD. t-SNE solves it by minimizing their Kullback-Leibler divergence. LargeVis models the

probability of obtaining an edge between neighborhood nodes in the LD space and maximizes the log-likelihood of this model. UMAP considers the graphs in the HD and LD spaces as fuzzy sets and minimizes the cross-entropy between them. All three methods use gradient descent for optimization.

The quality of the output embedding depends heavily on the hyperparameters of these methods, which control the construction of the kNN graph in the HD space. The perplexity/n_neighbors determines the approximate number of neighbors for each instance: small values reveal more local structures, while large values reveal more global structures in the data. UMAP also uses another hyperparameter (min_dist) to determine the minimum distance between points in the embedding in order to directly control how tight the groups are formed in the visualization.

Table 2.1: Definition of basic notations in SNE-based methods.

	In HD space	In LD space
Affinity	$\nu_{j\|i} = \exp\left(-\frac{\|\|\boldsymbol{x}_i - \boldsymbol{x}_j\|\|^2}{2\sigma_i^2}\right)$	$\omega_{ij} = \exp\left(-\|\|\boldsymbol{y}_i - \boldsymbol{y}_j\|\|^2\right)$
Conditional Probability	$p_{j\|i} = \frac{\nu_{j\|i}}{\sum_k^N \nu_{k\|i}}$	$q_{j\|i} = \frac{\omega_{ji}}{\sum_k^N \omega_{ki}}$
(Symmetric) Probability	$p_{ij} = \frac{p_{j\|i} + p_{i\|j}}{2N}$	$q_{ij} = \frac{w_{ij}}{\sum_k^N \sum_l^N w_{kl}}$

t-SNE, LargeVis, and UMAP follow a similar approach of stochastic neighbor embedding but use different notation. The core formulae of these methods are rewritten using the same notations defined as follows. The affinity between two points $\boldsymbol{x}_i, \boldsymbol{x}_j$ in the HD space, which is used to measure similarity, is denoted by $\nu_{j|i}$. This is the core idea of SNE-based methods that quantify this similarity by a Gaussian kernel with an adaptive bandwidth σ_i. This bandwidth is found by a binary search to adapt with the perplexity hyperparameter. The affinity is then transformed into a (conditional) probability $p_{j|i}$ by a normalization. This term is interpreted as a probability of a point $\boldsymbol{x}_j$ to be neighbor of the given point $\boldsymbol{x}_i$. Similarly, the affinity and the probabilistic in the LD space are defined in the same way, as summarized in Table 2.1.

Table 2.2: Objective functions of SNE-based methods.

SNE-based Methods	Objective function
Asymmetric SNE	Using $p_{j\|i}, q_{j\|i}$, $$\mathcal{L} = \sum_i^N \sum_j^N p_{j\|i} \log \frac{p_{j\|i}}{q_{j\|i}} \quad (2.3)$$
Symmetric SNE, t-SNE	Using p_{ij}, q_{ij}, t-SNE uses $\omega_{ij} = \frac{1}{1+\|\|\boldsymbol{y}_i - \boldsymbol{y}_j\|\|^2}$, known as the Student's t-distribution with one degree of freedom, $$\mathcal{L} = \sum_i^N \sum_j^N p_{ij} \log \frac{p_{ij}}{q_{ij}} \quad (2.4)$$
LargeVis	Using p_{ij} and ω_{ij} as in t-SNE, $$\mathcal{L} = \sum_{(i,j)\in E} p_{ij} \log \omega_{ij} + \gamma \sum_{(i,j)\in \bar{E}} \log(1-\omega_{ij}) \quad (2.5)$$
UMAP	Using ν_{ij}, ω_{ij} but with different definitions $$\mathcal{L} = \sum_{ij} \left[\nu_{ij} \log\left(\frac{\nu_{ij}}{\omega_{ij}}\right) + (1-\nu_{ij}) \log\left(\frac{1-\nu_{ij}}{1-\omega_{ij}}\right)\right] \quad (2.6)$$

The objective functions of different SNE-based methods are compared in Table 2.2. We try to highlight the symmetric/asymmetric terms used in these objective functions and point out the difference between modern methods like LargeVis and UMAP. LargeVis is based on graph theory and defines the set E containing the edges in the neighbor graph with non-zero weight. The objective

function of LargeVis in Equation 2.5 is explicitly defined as a combination of an attractive force (the first term) and a repulsive force (the second term) with a balancing factor γ. UMAP does not use the probabilities directly as in t-SNE and LargeVis but uses the symmetric affinities instead. The affinities in both HD and LD spaces are defined differently in Equation 2.6 (based on a fuzzy topological structure), but the affinities of t-SNE and LargeVis can be derived as a special case.

Let us take a closer look at SNE and t-SNE, the core methods in this thesis. The proximities (or affinities) of the original data are transformed into probabilities $\boldsymbol{P} = [p_{ij}]$ via softmax function, which is also known as Boltzmann distribution or Gibbs distribution in statistical physics (Landau and Lifshitz, 1980). A similar transformation (with a different kernel) is applied to the points in the LD space that gives a probability $\boldsymbol{Q} = [q_{ij}]$ of being neighbors in the visualization. A Kullback-Leibler divergence loss function (KL loss) sets a high cost for putting close neighbors far away in the LD space. Minimizing the KL loss by gradient descent method has a useful effect: it makes close points in the same neighborhood attract each other (by a so-called *attractive force*) and generally makes all points repulse each other (by a *repulsive force*). This is an analogy to the N-body problem in physics, for which our collegues Delchevalerie, Mayer, Bibal, and Frénay (2021) propose to use an efficient solution called the *Particle-Mesh* algorithm to speed up t-SNE by a large factor. Even though t-SNE is very useful and widely used, it has a quadratic computation complexity. Many works focus on improving this aspect. The attractive force involving the softmax similarities $\boldsymbol{P}$ and can be accelerated thanks to approximate nearest neighbor algorithms (that speeds up the graph construction). The repulsive force involves $\boldsymbol{Q}$ and can also be accelerated by physics-inspired methods, for example, Barnes-Hut acceleration (van der Maaten, 2014a) ($\mathcal{O}(N \log N)$), fast Fourier transform (FFT)-accelerated interpolation-based method (Linderman et al., 2019) ($\mathcal{O}(N)$), or Particle-Mesh (Delchevalerie et al., 2021) ($\mathcal{O}(M \log M)$ with $M \lll N$ is the number of points in a grid for discretizing space).

Another type of problem of t-SNE is analyzed by (Wattenberg et al., 2016) focusing on how to use this method effectively. The same problems are also encountered with UMAP[3]. The SNE-based methods are very sensitive to the neighborhood size (perplexity or n_neighbors), which is often hard to choose

[3] https://pair-code.github.io/understanding-umap/

for both end-users and experts. In t-SNE, the Gaussian kernel width for computing the similarities $\boldsymbol{P}$ is adapted to achieve the input perplexity. Lee et al. (2014, 2015) suggest using multiple perplexities to calculate *multiscale softmax similarities* that frees users from choosing this hyperparameter. This multiscale approach has a complexity of $\mathcal{O}(N^2 \log N)$ but has been recently reduced to $\mathcal{O}(N(\log N)^2)$ by a randomized subsampling process combined with the VP-tree and Barnes-Hut algorithm (de Bodt et al., 2020). The similarities $\boldsymbol{P}$ can also be refined by filtering out points in the neighborhood that are not in the same class, which is proposed in the novel class-aware t-SNE method (de Bodt et al., 2019). These similarities are even constructed by a *Student's t* kernel to remove the dependency on the perplexity, which is proposed in the *twice Student tt-SNE* method (de Bodt et al., 2018). We also contribute a solution to this problem of finding the best hyperparameters for neighbor-preserving methods, which is introduced in Chapter 6. Moreover, t-SNE is claimed to preserve local structures (neighbors) while often struggles to reveal global structure in the visualization. Multiscale approach (Lee et al., 2014, 2015) solves this problem by preserving at all scales, both local and global neighborhood information. We look at the global structure preservation problem from another viewpoint involving relative distances between clusters in the visualization raised by Wattenberg et al. (2016), and propose to integrate global hierarchy into t-SNE by a new method presented in Chapter 4.

In summary, this chapter gives a brief introduction to the DR problem and common DR methods with some new insights. We focus on the aspect of finding a manifold of data embedded in an HD space, instead of viewing the DR problem as finding a projection to reduce dimensionality. Three of the most widely used DR methods are summarized from different perspectives. The traditional PCA problem can be seen as maximizing variances or minimizing construction error, as a competitive eigen-game between agents, or a latent variable model. Through the problem of MDS, we try to convince that there are no small distances in an HD space where all pairs of points are equal-distant, and we should go from distance-preserving to neighbor-preserving approach. Several issues of the successful t-SNE method are also mentioned, which motivates us to correct these problems using users' constraints. In the next chapter, we will start with an introduction to different kinds of users' constraints and different approaches to integrating them into DR methods.

Chapter 3

A Survey of Integrating Constraints into DR Methods

This chapter is a study of existing methods for integrating various types of constraints into DR methods, summarized in a comprehensive survey.

Contents

3.1 Scope and Methodology **22**
3.2 Instance-level Constraints **25**
3.3 Dataset-level Constraints **38**
3.4 Discussion **51**

This chapter is based on our working paper: "Integrating Constraints into Dimensionality Reduction Methods: a Survey" (Vu, Bibal, and Frénay, 2021d) (submitted to Neurocomputing).

3.1 Scope and Methodology

Through this thesis, we study how users' feedback can be used together with DR methods for visualization. DR methods aim to preserve important characteristics and to find a faithful representation of data in an LD space. However, the patterns revealed by DR methods may not always match users' expectations, especially when users have prior knowledge about their data. DR methods objectively learn from data, while the users' needs are often subjective. In order to make DR methods more useful, many works have been devoted to integrating users' feedback into base DR methods. Sacha et al. (2017b) look at this integration as a "human-in-the-loop" process and investigate how users interact with DR techniques. We propose a new viewpoint: considering users' feedback as *constraints* that DR methods should respect besides their main objective of faithfully representing the data. This survey aims to reorganize the existing literature for constraint integration for DR.

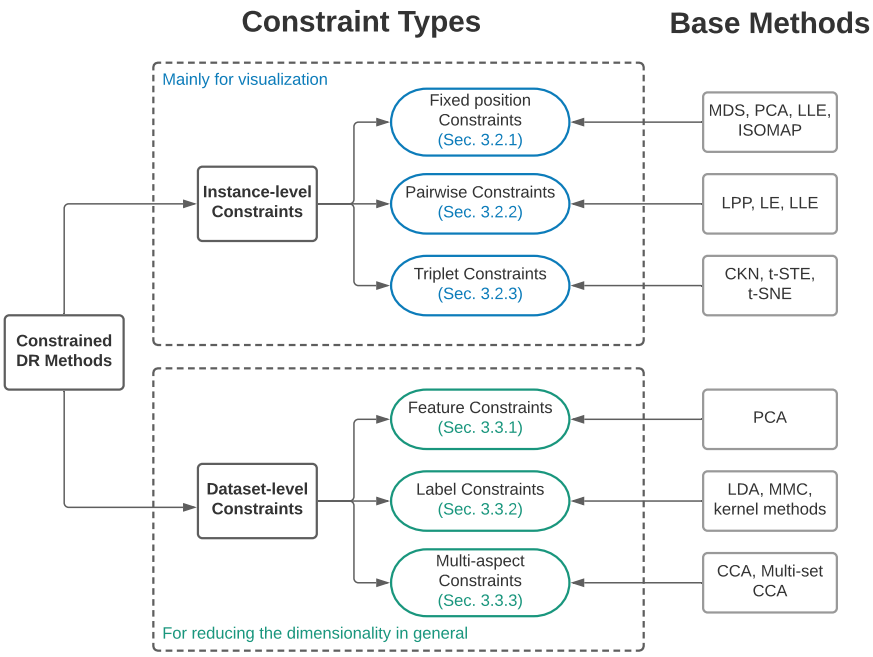


Figure 3.1: The structure of this survey, categorized by constraint types.

The most important point in our survey method is the categorization of constraints. Before conducting this survey, we first look for what kind of information can be considered as constraints. Based on recent surveys in unsupervised methods (Chao et al., 2019; Davidson and Basu, 2007; Grossi et al., 2017), many kinds of *side information* can be considered as constraints. Side information includes all kinds of additional information that can be used to describe the data. DR methods are thus required not only to preserve information in the original data but also to preserve the additional information if available. The additional information can come from users' feedback or domain experts' knowledge. It can also come from other sources outside the given dataset, such as external features that describe the data points by different characteristics or additional labels that group the points by a different criterion than the original class labels.

The new information from labels or from users' feedback is indeed a kind of *supervision* information. We do not make a clear distinction between (semi-) supervised and unsupervised DR methods but focus on the constraint representation instead. For example, graph-based DR methods always construct a neighbor graph to capture global structure in the data. When supervision information such as a partial or full set of labels is available, it can be used to modify the graph, for example, to increase weights of the edges connecting data points in the same class. Prior knowledge of users or their feedback when they observe the data can also be collected in the form of pairs between similar/dissimilar objects. Even though this kind of feedback is integrated into an unsupervised DR method, it has exactly the same effect to enhance the neighbor graph as the above supervision information. Within the scope of this survey, in a narrow sense, we suggest calling these methods (unsupervised methods with additional constraints or (semi-) supervised methods) as *constrained methods*. Therefore, one central topic of this survey is constraint representation.

Based on this definition of constraints, we collected papers of DR methods that work with different kinds of constraints and propose the following categorization, which is summarized in Figure 3.1. The first two research questions posed previously (on the constraint representation - **RQ1** and constraint optimization - **RQ2**) help us to define this comprehensive categorization. Typically in machine learning, when talking about constraints such as a *must-link* or *cannot-link* constraints, we assume that they are the *hard constraints*. These

constraints are used to modify internal data representation in the DR algorithms such as the covariance matrix or the neighbor graph, which can not be violated (a.k.a. hard constraints). However, when these constraints are represented differently, for example, in the form of a regularization term that is independent of the objective function of base DR methods, they are allowed to be violated (a.k.a. soft constraints). For another example, let us consider the users' constraints on the fixed position of points. If these constraints are encoded as a fixed block of the embedding matrix, they are considered as hard constraints. However, they can also be considered as soft constraints in an interactive method, which takes into account the uncertainty in the users' feedback. Therefore, considering a constraint as soft or hard depends on the constraint representation. Moreover, recent constrained DR methods like minimum-distortion embedding (MDE) (Agrawal et al., 2021) can preserve both hard constraints (orthogonal and orthonormal constraints on the columns of the embedding matrix) and soft constraints (a constraint on fixed position indicated by users) at the same time. For that reason, we also do not distinct hard and soft constraints in this survey.

The first category of this survey consists of constraints at the level of individual data point, called the *instance-level constraints* (Section 3.2). Methods in this category are mainly used for visualizations, and the constraints are typically collected from users' feedback or prior knowledge about the relationship between data instances. Similar to the perspective of *"what you see is what you can change"* (Sacha et al., 2017a), the first subgroup consists of methods that allow users to move points in the visualization to the desired position to form *fixed position constraints*. Moreover, users can also express their ideas about the similarity/dissimilar between individual objects to form *pairwise constraints*. Another kind of constraint constructed from tuples of three instances to highlight a contrastive relationship among them is called *triplet constraints*.

The second category consists of constraints on the whole dataset that can affect systematically every single data point, called the *dataset-level constraints* (Section 3.3). For example, *feature constraints* allow users to change the weights of each feature in their dataset to observe the features' importance and their effect on the embedding. Under our definition of constraints, class labels can be considered as constraints since the labels can guide DR methods to discriminate data points by different classes. (Semi-)supervised DR methods can be placed

in this subgroup of *label constraint* methods. Data points can also be described simultaneously by different sets of features or can possess different sets of labels associated with them. *Multi-aspect constraints* help to produce a unique embedding for this kind of data while preserving the information encoded in different views or label sets. In each subgroup, the constrained DR methods are grouped according to the optimization approach or the problem formulation of the base DR methods.

3.2 Instance-level Constraints

Instance-level constraints denote constraints on individual data points. We consider three kinds of instance-level constraints. (i) Fixed-position constraints apply to only one specific data point to specify its location in the LD space. (ii) Pairwise constraints apply to a pair of two points to indicate the similar/dissimilar relationship between them. (iii) Triplet constraints apply to a tuple of three points to indicate that the first point is closer to the second point than to the third point. This third kind of constraint can be considered as a more general case of pairwise constraints when a similar constraint (between the first and the second points) and a dissimilar constraint (between the first and the last one) are considered at the same time. These constraints can be defined on the data points in the original space or in the reduced LD space. This section presents how to encode these types of constraints into DR methods and different approaches to solve such constrained problems.

3.2.1 Constraints on Fixed Positions of Points

Constraints in the form of fixed positions of points are typically encountered in interactive visualization applications. In these applications, users, with their prior knowledge about data, may want to fix the position of several points or groups of points of interest in a scatter plot since they found a mismatch between their prior knowledge and the visualization. Through this setup, users can observe an initial visualization of their data, and then interact with the visualization to give their feedback. The user feedback or prior knowledge on the position of points is considered as *fixed position constraints* for DR methods.

The exact position of points fixed by users can be considered directly as

constraints for DR methods (Kim et al., 2017; Leman and Endert, 2010; Yang et al., 2006). When users already have an initial idea about the position of several points in their dataset, they may want to move the points or groups of points in the visualization to the desired positions. For example, a 2D map of cities built by an MDS-based method can be different from the reference geographical map because of the orientation-invariance of MDS visualizations. Users can correct the embedding by moving the points of several chosen cities to the geographic position of a reference map.

The relative position of interacted points can also be used to indicate the relationship between these points. For example, users can move several points close together/far apart to indicate that these points are similar/different and that they should stay close to/far away from each other in the embedding. This type of relationship can be used to learn the correlation of features that make points close or distant (Endert et al., 2011; House et al., 2015).

In the next section, we review how the fixed position constraints are integrated into DR methods, as well as three different approaches for solving the constrained DR problem. The advantages and disadvantages of these approaches are discussed and several common usages of this type of constraint, particularly in visual analytic, are highlighted.

3.2.1.1 Direct Approach

The most straightforward approach to integrating the fixed position constraints into DR methods is to set them explicitly in the matrix $\boldsymbol{Y}$ of embedded points. Let $\boldsymbol{Y}_0$ denote the fixed position of points known in advance as prior information. Let $\boldsymbol{Y}_1$ denote the position of the remaining points that we need to find. The output matrix $\boldsymbol{Y}$ is represented by two separated blocks of fixed points and unknown points $\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{Y}_0 \\ \boldsymbol{Y}_1 \end{bmatrix}$.

The basic optimization in Locally Linear Embedding (LLE) (Roweis and Saul, 2000), Isometric feature mapping (ISOMAP) (Tenenbaum et al., 2000), and Local Tangent Space Alignment (LTSA) (Zhang and Zha, 2004) is

$$\begin{aligned} &\min_{\boldsymbol{Y}} \boldsymbol{Y}\boldsymbol{M}\boldsymbol{Y}^T \\ &\text{subject to } \boldsymbol{Y}^T\boldsymbol{Y} = \boldsymbol{I}_{\mathrm{q}\times\mathrm{q}}, \end{aligned} \tag{3.1}$$

where $\boldsymbol{M}$ is a pairwise weight matrix (in LLE) or alignment matrix (in LTSA). A general semi-supervised nonlinear DR framework (SS-NDR) (Yang et al., 2006) is used to develop semi-supervised versions of these above methods (SS-LLE, SS-ISOMAP, SS-LTSA). Under this framework, the above problem in Equation 3.1 becomes

$$\boldsymbol{Y}_1 = \arg\min_{\boldsymbol{Y}_1} \begin{bmatrix} \boldsymbol{Y}_0 \\ \boldsymbol{Y}_1 \end{bmatrix} \begin{bmatrix} \boldsymbol{M}_{00} & \boldsymbol{M}_{01} \\ \boldsymbol{M}_{10}^T & \boldsymbol{M}_{11} \end{bmatrix} [\boldsymbol{Y}_0^T \quad \boldsymbol{Y}_1^T], \tag{3.2}$$

where the matrix $\boldsymbol{M}$ is represented by different blocks based on the fixed and the unknown points, and is constructed in different ways for the three nonlinear DR methods above. The unknown coordinates in $\boldsymbol{Y}_1$ appear in a system of linear equations derived from Equation 3.2 and can be easily solved since $\boldsymbol{Y}_0$ and $\boldsymbol{M}$ are both fixed.

This representation of $\boldsymbol{Y}$ is not only used for eigen-based DR methods but also used for iterative methods like t-SNE and MDS. XGVis(Buja et al., 2001, 2008), an interactive toolkit based on MDS, uses the absolute position of fixed points or groups of points in $\boldsymbol{Y}_0$ as anchor points to find the configuration of other points that minimizes the stress criteria of metric and non-metric MDS. Following the same idea, the PIVE framework (Kim et al., 2017) introduces an interactive pipeline that allows users to interact in the early iterations with the intermediate visualization results of t-SNE or MDS after each optimization iteration. The users can move points or groups of points in the visualization at the intermediate state of the optimization process to construct the matrix $\boldsymbol{Y}$, which is now used as a new initial embedding to continue to run t-SNE or MDS. This direct approach is simple, easy to optimize, and fast enough to be integrated into visualization methods in an interactive context.

Nonlinear DR methods, in general, do not produce stable results due to their random initialization. Constraints on fixed positions are claimed to improve the stability for MDS (Buja et al., 2008), t-SNE (Kim et al., 2017) and other nonlinear DR methods like LTSA and LLE (Yang et al., 2006). However, in this direct approach, the constraint in $\boldsymbol{Y}_0$ is processed independently to the unknown blocks $\boldsymbol{Y}_1$.

3.2.1.2 Probabilistic Approach

The probabilistic approach proposes a novel way to integrate the fixed position constraints into probabilistic DR methods like probabilistic PCA (PPCA) (Tipping and Bishop, 1999), probabilistic MDS (PMDS) (Hefner, 1958; MacKay and Zinnes, 1986; Zinnes and MacKay, 1983), and Generative Topographic Mapping (GTM) (Bishop et al., 1998). In probabilistic DR methods, the input data points are called the observed variables, and the unknown embedded points are called the latent variables. Each type of observed and latent variable is modeled by a probability distribution. The goal is to find the distribution of the latent variables, i.e., to estimate the parameters of their distributions that best represent the observed data. The constraints on the fixed position are first transformed into probabilistic terms, which are then used by the model to estimate the unknown parameters.

Bayesian Visual Analysis (*BaVA*) (House et al., 2015) is a framework combining two research areas: visual analytic and Bayesian statistics, and can be applied for PMDS, PPCA (Endert et al., 2011; House et al., 2015) and GTM (Endert et al., 2011; Han et al., 2016). BaVA follows a probabilistic modeling process with a focus on the iterative loop to update the model when receiving users' feedback. This process is also known as a *sequential Bayesian updating*, and thus the Bayesian part comes in its name. Simply speaking, in this sequential updating, we start with an initial prior belief and update our belief after observing data to derive a posterior. In the next iteration, the old posterior becomes a new prior belief, and we continue updating this belief when observing new incoming data to obtain a new posterior. This is the key idea of BaVA to allow experts to incorporate feedback and new information to update the model. BaVA transforms the users' *cognitive feedback* $f^{(c)}$ (e.g. fixed position of points) into *parametric* feedback $f^{(p)}$ and models this transformation by a distribution $\pi(f^{(p)} \mid f^{(c)}, \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is model's unknown parameters. The BaVA framework is flexible to tailor for different probabilistic DR methods. For example, *User-guided PPCA* (Endert et al., 2011), a variant of PPCA under BaVA framework, allows users to move points: either drag them far apart or close together to form the fixed position constraints. In that way, BaVA models the features that reflect the relationship of the fixed points and learns a better covariance matrix for PPCA.

Using the same probabilistic approach, interactive PPCA (iPPCA) (Vu and Frénay, 2019) and interactive Probabilistic MDS (iPMDS) (Vu et al., 2021c) propose the novel idea of encoding the constraints of the fixed positions directly into a prior distribution. In general, the positions of the unknown (unobserved) points are represented by the latent variables $\mathbf{Z} = [\mathrm{z}_1, \ldots, \mathrm{z}_n]$ with a standard normal prior distribution $\mathrm{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$. If the position of the point $\boldsymbol{y}_i$ is known with a certainty level modeled by a variance σ_{fixed}, the prior distribution for the fixed point is represented by $\mathrm{z}_i \sim \mathcal{N}(\boldsymbol{y}_i, \mathbf{1}\sigma^2_{fixed})$. The positions of the other unknown points are found by MAP estimation.

The main advantage of the probabilistic approach is that (i) it can handle noisy and missing input data, and (ii) it can output an embedding with an estimation of uncertainty for each point. Moreover, the uncertainty in the users' feedback about the position of fixed points can also be modeled in the prior distribution of a probabilistic model (Vu and Frénay, 2019; Vu et al., 2021c). Another advantage of this approach is the separation of the modeling and the inference step. We typically focus efforts on the modeling step to represent both the input data and the constraints in a faithful way. Once the model is defined, different inference algorithms like MLE, MAP, Markov Chain Monte Carlo, or Variational Inference can be applied. This separation is useful because the inclusion of constraints generally is performed during the modeling step.

3.2.1.3 Non-probabilistic Approach

In contrast to the direct and probabilistic approaches, DR methods can also be constrained in two separate steps. First, the constraints of fixed points are used to learn a better representation for the input data, such as a weighted distance function to measure the similarity of the input data. Second, the original optimization problem is handled using the learned representation.

Visual to Parametric Interaction (V2PI) (Leman et al., 2013), a non-probabilistic version of BaVA, is a typical example of this approach. In this framework, a fixed point constraint is considered as cognitive feedback since it is a particular representation of human knowledge about the dataset. This type of constraint is transformed into a parametric form integrated into the covariance matrix for PCA or in a custom distance function for MDS. *User-guided MDS* (Endert et al., 2011; Leman et al., 2013) is an example of applying V2PI to

the MDS problem. This method is based on *Weighted MDS* (Carroll and Chang, 1970), which uses a normalized p-dimensional weight vector $\boldsymbol{w} = [w_1, \dots w_p]$ to represent the importance of each feature. The weighted Euclidean distance between two p-dimensional data points is defined as

$$d_w^2(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{k=1}^{p} w_k \left(x_i^{(k)} - x_j^{(k)} \right)^2, \text{ where } \sum_{k=1}^{p} w_k = 1. \tag{3.3}$$

In order to faithfully reflect the pairwise distances between the fixed points, the subset of fixed points $\boldsymbol{Y}_0 = [\widetilde{\boldsymbol{y}}_i]$ is used to learn an adaptive weight vector

$$\boldsymbol{w} = \arg\min_{\boldsymbol{w}} \sum_{\tilde{\boldsymbol{y}}_i, \tilde{\boldsymbol{y}}_j \in \boldsymbol{Y}_0} \left| d_w^2(\boldsymbol{x}_i, \boldsymbol{x}_j) - d^2(\tilde{\boldsymbol{y}}_i, \tilde{\boldsymbol{y}}_j) \right|.$$

The coordinates of the remaining unknown points can be found by solving the traditional MDS problem (Torgerson, 1952) of minimizing the stress function $\sum_{1 \le i < j \le N} \left(d(\boldsymbol{x}_i, \boldsymbol{x}_j) - d(\boldsymbol{y}_i, \boldsymbol{y}_j) \right)^2$, using the weighted distance $d_w(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

This adaptive weighted distance function can also be applied for other kinds of distance such as the geodesic distance in ISOMAP, a variant of metric MDS. ISOMAP uses the shortest-path distance in a connected neighborhood graph to faithfully represent distances in the manifold. However, this geodesic distance is sensitive to the neighborhood size due to a *short-circuiting* problem, in which the algorithm falsely assumes that two points are close in the geodesic distance when they are not (Tenenbaum et al., 2000). ISOMAP under V2PI framework uses the fixed point constraint to rescale the measure of the short-circuiting edges to make sure these edges will not be chosen in the shortest paths (Leman and Endert, 2010).

The V2PI and BaVA frameworks are designed to transform the user feedback into a parametric form used by PCA, MDS, ISOMAP, or their alternative probabilistic versions. However, these frameworks have to tailor the representation of the fixed point constraints according to different base DR methods. The constrained DR methods in this approach are simpler than the probabilistic methods in Section 3.2.1.2 but are more complex than the direct methods in Section 3.2.1.1 in the optimization process.

3.2.1.4 Discussion on Fixed Point Constraints

In summary, the idea of the reviewed (both linear and non-linear) methods in this section falls under the umbrella of five unified frameworks. The general semi-supervised nonlinear DR framework (Yang et al., 2006) introduces a simple and efficient way to encode the fixed point constraint directly in the embedding matrix. The PIVE framework (Kim et al., 2017) introduces an interactive pipeline that uses the modified intermediate state of the embedding to rerun t-SNE or MDS. The probabilistic BaVa framework (House et al., 2015) and its non-probabilistic version V2PI (Leman and Endert, 2010) allow DR methods to extract useful information from the constrained points to learn global parameters like covariance matrix or feature weights used in the base DR methods. The last framework introduces the idea of encoding directly the position of fixed points with some level of uncertainty into prior distribution for the latent positions in PPCA and PMDS (Vu and Frénay, 2019; Vu et al., 2021c). While we focus on the representation of the fixed point constraints, the reviewed methods in this group cover both common linear (User-guided PCA, iPPCA) and non-linear (SS-LLE, SS-ISOMAP, MDS-based,. . .) methods.

The fixed point constraint is mainly used for interactive visualization methods where the users want to express their needs about the desired position for their selected points. The DR methods enhanced by this type of constraint are easy to use for the end-user. For example, by moving the points to the target position, users can test their hypotheses about the structure or pattern they would like to see in the visualization (House et al., 2015). By fixing the position of several anchors points, users can also manipulate (rotate, translate or flip) the visualization (Buja et al., 2008). Moreover, by taking into account the relative relationship between fixed points, the users can select examples to describe the meaning of their desired axes in the visualization (Vu and Frénay, 2019; Vu et al., 2021c).

However, how to choose the fixed points effectively is still a hard problem(De Silva and Tenenbaum, 2004). The SS-NDR framework (Yang et al., 2006) introduces a sensitivity analysis, but it is not applicable for a wide range of nonlinear DR methods. This is the main reason why these methods are mainly used for interactive visual analytic, where the subjective feedbacks of users are used, and the visualizations are generally assessed by human judgment.

3.2.2 Pairwise Constraints

In unsupervised learning, where the labeled data are not available, pairwise constraints between points in a small subset can be used to provide information about the underlying structure of the data. Pairwise constraints have been applied successfully in many clustering methods (Davidson and Basu, 2007). This section reviews different approaches to integrate pairwise constraints into DR methods. There exist two kinds of pairwise constraints. A must-link (or positive) constraint between two points indicates that they are similar, while a cannot-link (or negative) constraint between two points indicates that they are not similar. Notice that the similarity can either be provided through a predefined metric measurement or through human perception. Let $\mathcal{M}$ and $\mathcal{C}$ be the sets of must-link and cannot-link constraints. If two data points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are connected by a must-link or a cannot-link, we denote them by $ML(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{M}$ or $CL(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{C}$ respectively. We also write $(i, j) \in \mathcal{M}$ or $(i, j) \in \mathcal{C}$ for short. These must-link and cannot-link constraints are also called *equivalence constraints* (Bar-Hillel et al., 2005; Cevikalp et al., 2008) since they can be inferred or propagated by the transitivity and entailment rule[1]. Cannot-link constraints are not transitive and are therefore not as informative as must-link constraints (Bar-Hillel et al., 2005). However, in almost all reviewed methods, both types of constraints are used and have proved their efficiency.

This section focuses on two approaches to integrate the must-link and cannot-link constraints into DR methods. The graph-based approach directly uses the additional pieces of information implied in the pairwise constraints to modify the graph representation of the input data. In contrast, the discriminant-based approach considers the must-link and cannot-link constraints to distinguish similar and dissimilar points.

3.2.2.1 Graph-based Approach

In general, the graph-based DR methods represent the input data in the HD space in the form of a (connected) graph. Each point is connected with their K nearest neighbors or with their neighbors in an ϵ-ball in the HD space. The

[1]Transitivity rule applied for must-link constraints: $ML(\boldsymbol{x}, \boldsymbol{y}), ML(\boldsymbol{y}, \boldsymbol{z}) \rightarrow ML(\boldsymbol{x}, \boldsymbol{z})$. Entailment rule for generating all cannot-link constraints from two sets of must-link constraints: $ML(\boldsymbol{a}, \boldsymbol{b}), ML(\boldsymbol{x}, \boldsymbol{y}), CL(\boldsymbol{a}, \boldsymbol{x}) \rightarrow CL(\boldsymbol{a}, \boldsymbol{y}), CL(\boldsymbol{b}, \boldsymbol{x}), CL(\boldsymbol{b}, \boldsymbol{y})$.

hidden patterns in the data can be represented through the different structures in the neighborhood graph. The goal of graph-based DR methods is thus to reveal these hidden structures.

The commonly encountered optimization problem in graph-based methods such as Locality Preserving Projections (LPP) (He and Niyogi, 2004), Laplacian Eigenmaps (LE) (Belkin and Niyogi, 2003) or Locally Linear Embedding (LLE) (Roweis and Saul, 2000) is closely related to the eigenvector problem in Equation 3.2. The goal is to find a projection matrix $\boldsymbol{P}_{\mathrm{q}\times\mathrm{p}}$ that map the data from the p-dimensional space to the lower q-dimensional space $\boldsymbol{Y}_{\mathrm{n}\times\mathrm{q}} = \boldsymbol{X}_{\mathrm{n}\times\mathrm{p}}\boldsymbol{P}^T{}_{\mathrm{p}\times\mathrm{q}}$. Since there exist many solutions for the projection matrix, we force the orthogonal constraint ($\boldsymbol{P}^T\boldsymbol{P} = \mathbf{1}_{\mathrm{p}\times\mathrm{p}}$) for the columns of $\boldsymbol{P}$, i.e., their columns should be orthogonal and have unit norm. The problem is reduced to minimize the pairwise distances between projected points in the LD space and can be achieved using the graph Laplacian matrix $\boldsymbol{L}_{\mathrm{n}\times\mathrm{n}} = \boldsymbol{D} - \boldsymbol{W}$.

$$\begin{aligned} \min_{\boldsymbol{P}} \ & \mathrm{trace}(\boldsymbol{Y}^T\boldsymbol{L}\boldsymbol{Y}) = \mathrm{trace}(\boldsymbol{P}\boldsymbol{X}^T\boldsymbol{L}\boldsymbol{X}\boldsymbol{P}^T) \\ \text{subject to} \ & \boldsymbol{Y}^T\boldsymbol{D}\boldsymbol{Y} = \mathbf{1}. \end{aligned} \tag{3.4}$$

In graph theory, this Laplacian matrix $\boldsymbol{L}$ is the most natural way to characterize a graph represented by a weighted adjacency matrix $\boldsymbol{W}$, where $\boldsymbol{D}$ is a diagonal matrix of degree determined by the number of neighbors of each point (Davidson, 2009). Each point i in the diagonal matrix $\boldsymbol{D}$ represent the degree of the node i in the neighborhood graph: $D_{ii} = \sum_j W_{ij}$.

Based on this fundamental optimization problem, the main idea of the constrained methods is to modify the Laplacian matrix $\boldsymbol{L}$ to encode the pairwise constraints. This can be done by changing the values in the input weighted adjacency matrix using the given pairwise constraints. Several constrained DR methods (Davidson, 2009; Tang and Zhong, 2007) can be generalized using the expansion of the general graph spectral problem in Equation 3.4. The most important idea of these methods is that, for a pair (i, j), the corresponding projected points $(\boldsymbol{y}_i, \boldsymbol{y}_j)$ should be close if $(i, j) \in \mathcal{M}$ and they should be distant

if $(i, j) \in \mathcal{C}$. Let us rewrite this problem in the form of quadratic terms as

$$\begin{aligned} \boldsymbol{P} &= \arg\min_{\boldsymbol{P}} \operatorname{trace}(\boldsymbol{P}^T \boldsymbol{X}(\boldsymbol{D} - \boldsymbol{W})\boldsymbol{X}^T \boldsymbol{P}) \\ &= \arg\min_{\boldsymbol{P}} \sum_{i \neq j} ||\boldsymbol{P}^T \boldsymbol{x}_i - \boldsymbol{P}^T \boldsymbol{x}_j||^2 W_{ij}. \end{aligned} \tag{3.5}$$

This objective function in Equation 3.5 is also called *graph preserving criteria* (Yan et al., 2007; Yan and Wang, 2009). The weighted adjacency matrix $\boldsymbol{W}_{\mathrm{n}\times\mathrm{n}}$ is the key point to encode the pairwise constraints. Various methods are proposed to modify this matrix in two ways: (i) assigning discrete values to W_{ij} to denote the connection between nodes, or (ii) assigning continuous values to W_{ij} to weight the edges of the neighborhood graph.

In the first case, only the graph structure is modified by the constraints since W_{ij} can only take the values in $\{0, 1, -1\}$ (Davidson, 2009; Tang and Zhong, 2007). $W_{ij} = 0$ when the two points are not in the neighborhood of each other, $W_{ij} = +1$ when $(i, j) \in \mathcal{M}$, and $W_{ij} = -1$ when $(i, j) \in \mathcal{C}$. This simple formulation for the connected graph can capture the structure encoded in the pairwise constraints efficiently. However, there may be similar points that are not in the same neighborhood. Constrained Locality Preserving Projection (CLPP) (Cevikalp et al., 2008) proposes to enrich the set of constraints by propagating must-links to strengthen the similarity information and propagating cannot-links to remove the weak dissimilar links.

In the second case, W_{ij} is used as a weight for every pair of points in the dataset. In that case, the must-link and cannot-link pairs can be weighted by different normalized constants adapted to the number of must-links $|\mathcal{M}|$ and cannot-links $|\mathcal{C}|$ as

$$W_{ij} = \begin{cases} \frac{1}{2n^2} + \frac{\alpha}{2|\mathcal{M}|} & \text{if } (i, j) \in \mathcal{M}, \\ \frac{1}{2n^2} - \frac{\beta}{2|\mathcal{C}|} & \text{if } (i, j) \in \mathcal{C}, \\ \frac{1}{2n^2} & \text{otherwise,} \end{cases}$$

where α, β are the scalar parameters for balancing the contribution of each constraint type (Zhang et al., 2007a). Robust CLPP (YU et al., 2010) benefits from this weighted matrix and applies the path-based similarity (Fischer et al., 2004) to find the intrinsic geometric structure of the graph.

The general idea of the constrained methods in this group is based on traditional graph-based DR methods like Locality Preserving Projections (LPP) (He and Niyogi, 2004), Laplacian Eigenmaps (LE) (Belkin and Niyogi, 2003) or Locally Linear Embedding (LLE) (Roweis and Saul, 2000). Several constrained methods are also extended with kernels to learn a projection from a kernel space to an LD space (Cevikalp et al., 2008; Meng et al., 2017; Yan et al., 2012). These constrained methods consist of two steps. The graph is first constructed using the neighborhood information and input constraints. The graph spectral optimization is then solved to find the best mapping that reveals the graph structure in the LD space. This mapping should preserve the locality by minimizing the distances between the projected points in such a way that close/distant points in the HD space stay close/distant in the LD space. Since this objective is in line with the similar/dissimilar constraints, the pairwise constraints can easily be integrated into graph-based methods through a unified formulation in Equation 3.5. The optimization in this equation is a standard eigenvalue problem and has a closed-form solution. However, the main issue is how to determine the neighborhood size in the graph construction. Different solutions exist to tackle this problem, such as a parameter-free method for graph construction (Yan and Wang, 2009) or an LLE-based method that learns this parameter automatically (Saxena et al., 2004).

3.2.2.2 Discriminant Criteria-based Approach

In contrast to the graph-based approach that uses pairwise constraints directly, another approach is based on Linear Discriminant Analysis (LDA) and uses pairwise constraints to enhance the discriminant criteria. Methods in this group aim to maximize the *between-group* similarity and to minimize the *within-group* similarity simultaneously (Long et al., 2020; Sanodiya et al., 2020). This criterion enhances the similarity of points within the same group or points connected by must-links (similar links) and degrades the similarity of points in different groups or points connected by cannot-links (dissimilar links).

Several methods following this approach propose to enhance the discriminant information by enriching the set of input constraints. In a semi-supervised setting, the constraints can be generated from partial class labels if available. However, data points in the same class may belong to different subgroups.

The *semi-supervised DR for multiple sub-classes* method (Tong et al., 2012) tackles this problem by introducing the concept of inter-subclass must-link and intra-subclass must-link constraints to better represent the discriminant. *Semi-supervised discriminant ISOMAP* (Huang et al., 2019) redefines must-link constraints as links between points with the same label and introduces the new *likely-link* constraints as links between points in the same neighborhood. The other way to enhance the discriminant is to exploit the relative relation between constrained points. The *similarity order preservation* method introduced in (Hu et al., 2021) finds a projection in such a way that points with less similarity have larger distances in the LD space, while points with greater similarity have smaller distances in the LD space. This idea is similar to the triplet constraints in Section 3.2.3.

LDA-based is one of the most common approaches for supervised DR methods. However, using the pairwise constraints is not a direct way to present the discriminant. Another approach that uses partial labels based on semi-supervised LDA that provides better solutions is presented in Section 3.3.

3.2.2.3 Discussion on Pairwise Constraints

Most of the DR methods with pairwise constraints are graph-based methods. Pairwise constraints can be easily generated from the available labels in the dataset, or from groups of similar/dissimilar points collected from user knowledge. The constrained methods in this group integrate pairwise constraints into the neighborhood graph. This graph can both represent the data and preserve the constraints at the same time. Useful patterns in the graph are extracted thanks to the graph spectral optimization method. The methods in this group are usually used as a pre-processing step to reduce the dimensionality of data for the downstream tasks (Tang and Zhong, 2007; Zhang et al., 2007a). Moreover, these methods are not iterative and hard to be used for visual analysis tasks. Another usage of pairwise constraints is to assess the quality of the embedding for several visualization methods (Vu et al., 2021a)). Since the pairwise constraints reflect the structures in the data, we can measure how well these constraints are preserved in the visualization. This constraint-preserving score can also be used to tune the hyperparameter of visualization methods (t-SNE, UMAP) automatically.

3.2.3 Triplet Constraints

Triplet constraints are a particular type of constraint with three observations. They are extremely useful when the exact measurement of the (dis)similarity is not available. These constraints are in the form *"$\boldsymbol{x}_i$ is more similar to $\boldsymbol{x}_j$ than to $\boldsymbol{x}_k$"*. For convenience purposes, a triplet is denoted as a tuple (i, j, k) and can be modeled by a probability p_{ijk} of how likely $\boldsymbol{x}_i$ is more similar to $\boldsymbol{x}_j$ than to $\boldsymbol{x}_k$. This representation is typically used in common triplet embedding methods such as Crowd Kernel Learning (CKL) (Tamuz et al., 2011), Stochastic Triplet Embedding (STE), t-distributed STE (van der Maaten and Weinberger, 2012), and TriMap (Amid and Warmuth, 2019). Since these methods directly learn the position of points in the LD space, they represent p_{ijk} using the distance of the embedded points $||\boldsymbol{y}_i - \boldsymbol{y}_j||$ and $||\boldsymbol{y}_i - \boldsymbol{y}_k||$. The embedding $\boldsymbol{Y}$ can be optimized using gradient-based method (as in TripMap) or maximum likelihood $\max_{\boldsymbol{Y}} \sum_{(i,j,k)\in\mathcal{T}} \log p_{ijk}$, where $\mathcal{T}$ is the set of all triplet constraints.

Triplet constraints between three objects can also be generalized to two pairs in order to represent the rank or the order $d(\boldsymbol{x}_i, \boldsymbol{x}_j) < d(\boldsymbol{x}_k, \boldsymbol{x}_l)$, which is naturally used in non-metric MDS (Agarwal et al., 2007). Moreover, triplet constraints among individual objects can be generalized to be applied to more abstract objects like *group of points* or *semantic concepts of object*. *t-SNE with Hierarchical Constraints* (HCt-SNE) (Vu et al., 2021b) transforms the hierarchical structure into triplet constraints between groups in the hierarchy tree, which are encoded into a regularization term for t-SNE. *SNE-and-Crowd-Kernel Embedding* (SNaCK) (Wilber et al., 2015) combines t-SNE with t-STE, a triplet embedding method(van der Maaten and Weinberger, 2012). SNaCK is used to reveal the semantic concepts that already exist in the data that visualization methods like t-SNE do not capture. For example, in a food dataset, the model can easily create a visualization in which the food dishes are arranged according to their colors while it is much harder to group the dishes by taste (delicious, spicy, flavor, etc.). To discover these semantic categories, models need to be guided by human hints.

Discussion on Triplet Constraints

Triplet constraints are more informative than fixed-position and pairwise constraints but have a more complex formulation. Triplet constraints can be

generated from pairwise constraints, from groups of similar and dissimilar points, or being queried from users. Triplet constraints can be integrated into well-known DR methods (Vu et al., 2021b; Wilber et al., 2015) or can be used to learn the embedding directly from triplets, as in triplet embedding methods (Amid and Warmuth, 2019; Tamuz et al., 2011; van der Maaten and Weinberger, 2012). Moreover, in interactive machine learning, *"triplets are one of the most flexible options in practical use because they do not rely on prior knowledge, are invariant to scale, and are stable between and within subjects"* (Wilber et al., 2015). For that reason, crowd-sourcing methods such as CKL (Tamuz et al., 2011) and SNaCK (Wilber et al., 2015) are usually used to find the most informative constraints to ask users for their feedback. With triplet constraints, these methods can capture the relationships between objects that may not appear in class labels.

The methods using triplet constraints are related to a group of metric learning methods using triplet loss. This kind of loss is called contrastive loss (Arora et al., 2019), which is used to learn a representation function $f_\theta(\cdot)$ that make the anchor point $\boldsymbol{x}_i$ closer to the positive point $\boldsymbol{x}_j$ than to the negative point $\boldsymbol{x}_k$. The similarity between points mapped by the function $f_\theta(\cdot)$ is measure by $d(f_\theta(\boldsymbol{x}_i), f_\theta(\boldsymbol{x}_j)), d(f_\theta(\boldsymbol{x}_i), f_\theta(\boldsymbol{x}_k))$, where $d(\cdot)$ can be any similarity measure such as a Gaussian heat kernel or a dot product. Triplet loss and contrastive loss in general are recently used for visual representation (Chen et al., 2020), face recognition (Schroff et al., 2015) or image retrieval (Ge, 2018).

3.3 Dataset-level Constraints

The previous section considers constraints at the level of individual data points, which are only applied to small ensembles of selected instances in the dataset. In this section, we discuss *dataset-level* constraint, a new kind of constraint that generally affects the whole dataset instead of individual instances. For example, if users want to give greater importance to selected features in their dataset, they can manually increase the weights for them and observe how the visualization changes accordingly. These constraints on feature weights are detailed in Section 3.3.1. Labels can also be considered as constraints for dimensionality reduction techniques. Indeed, because DR is unsupervised, when labels are available, this additional source of information can enrich the

visualization. We consider these additional labels as constraints that can be used to enhance the separation between different classes in the dataset in Section 3.3.2. Moreover, if we have another data source such as a supplementary dataset with new features providing additional information to the original dataset, a new embedding can be created to explain the features in both datasets. In this case, the whole supplementary dataset is considered as a multi-view constraint, a type of multi-aspect constraint that is discussed in Section 3.3.3.

3.3.1 Constraints on Features

One of the most fundamental tasks in machine learning is to compare the similarity between data points. This can be done by computing the distance between these points. In the Euclidean distance, all the features are considered to have the same weight (or the same importance), which may not always be adequate for all problems. Distance metric learning methods (Bellet et al., 2013; Kulis et al., 2012; Yang and Jin, 2006) aim to learn an adaptive distance function such as the Mahalanobis distance to account for the correlation between features. Metric learning automatically learns a metric from data or task-specific distance functions when supervision is available. Another much simpler solution is to use a weighted Euclidean distance in which each feature is assigned a particular weight to indicate its importance. Although the feature weights can also be learned from data, users can still set weights according to their knowledge about the dataset.

Several interactive methods such as iPCA (Jeong et al., 2009, 2015) and iLDA (Choo et al., 2010) allow users to examine how changing feature weights affect the visualization. In these methods, the changes in each feature/dimension are reflected by the changes in the position of data points in the visualization. From that, users can interpret the importance of each feature. Bounded PCA (Giordani and Kiers, 2007) allows experts to set lower and upper bounds on the values of the features of interest and produces an optimal projection that makes the projected data satisfy the predefined bounds.

However, users do not always know in advance the importance of features, and it is hard to evaluate automatically the learned feature weights. It can be more intuitive if users can interact with the visualization to deduce the features' importance. In linear DR methods like PCA, changes in the feature space

directly translate to changes in the visualization. Inversely, changes in position in the visualization can also be traced back to the feature space (Cavallo and Çağatay Demiralp, 2017). In this case, the changes that users make in the visualization act as constraints to adapt feature weights. This idea is also applied to other kinds of visualization, such as the star-coordinates display for analyzing the dependencies of features in PCA (Molchanov and Linsen, 2014).

3.3.2 Constraints from Labeled Data

When available, class labels can also be considered as additional constraints for DR methods. They can be used to generate instance-level constraints such as pairwise or triplet constraints in Section 3.2. However, in this section, we focus on DR methods that directly use the available labels to reinforce the discriminant information, e.g., the class structure in the visualization. For instance, in the embeddings, all the points of the same class should be grouped close together while points in different classes are separated. This section reviews several supervised and semi-supervised DR methods and focuses on how to represent the discriminative information available through the class labels. From a methodological perspective, semi-supervised DR methods are considered as special cases of supervised methods since their objective functions are similar (Chao et al., 2019). This survey brings another viewpoint by considering the supervised and semi-supervised methods as constrained methods. Indeed, these DR methods should reveal the structure of unlabeled data while satisfying the label constraints at the same time. We highlight important ideas of extracting discriminant information in supervised DR methods and their extensions in the semi-supervised setting.

To motivate the use of supervision in DR methods, Figure 3.2 shows an example of the structure discovered by two common unsupervised (PCA) and supervised (LDA) methods. PCA looks for a projection in such a way that the projected data explains the variance in the original data as much as possible. The information in a dataset can be measured by how the instances spread in the data space (Deisenroth et al., 2020). However, the directions selected by PCA shown in Figure 3.2(a) only reflect one global aspect of the spread without considering potential information about data classes. This projection is thus not useful due to a large overlap of the two groups. The reason for this lack of clear

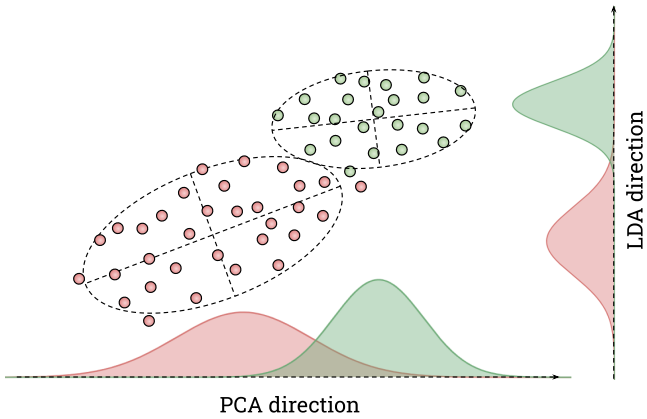


Figure 3.2: A pictorial illustration for the difference between PCA and LDA projections.

separation is that the two features of the data are highly correlated. Hence, only one eigenvector of the covariance matrix is needed, which corresponds to the direction of the largest spread to recover the data.

By opposition, Linear Discriminant Analysis (LDA) finds the projection in Figure 3.2(b) that better distinguishes the two groups. LDA finds a linear combination of features that preserves the class separability. Instead of only one covariance matrix, LDA uses two unnormalized covariance matrices (called the *scatter matrices*). It reveals the discriminant information encoded in the class labels by modeling the separation between instances from different classes and the coherence between instances of the same class. These two criteria are called the between-class and within-class separations and are measured by the between/within-class scatter matrices, which are defined as

$$\begin{aligned} \boldsymbol{S}_b &= \sum_{c=1}^{C} n_c \left(\boldsymbol{\mu}^{(c)} - \boldsymbol{\mu}\right)\left(\boldsymbol{\mu}^{(c)} - \boldsymbol{\mu}\right)^T \\ \boldsymbol{S}_w &= \sum_{c=1}^{C} \sum_{j|y_j=c} \left(\boldsymbol{x}_j - \boldsymbol{\mu}^{(c)}\right)\left(\boldsymbol{x}_j - \boldsymbol{\mu}^{(c)}\right)^T, \end{aligned} \tag{3.6}$$

where C is the number of classes in the dataset, n_c and $\boldsymbol{\mu}^{(c)}$ are the number of instances and the mean vector of class c, and $\boldsymbol{\mu}$ is the mean vector of the whole dataset. Fisher's Discriminant Analysis (FDA) (Fisher, 1936; Fukunaga, 2013) is a specific case with binary classes. In general, LDA finds a projection

$\boldsymbol{P}$ that maximizes the *between-class separation*, and minimizes the *within-class separation*:

$$\boldsymbol{P} = \underset{\boldsymbol{P}\in\mathbb{R}^{q\times p}}{\arg\max}\,\text{trace}\left(\frac{\boldsymbol{P}\boldsymbol{S}_b\boldsymbol{P}^T}{\boldsymbol{P}\boldsymbol{S}_w\boldsymbol{P}^T}\right). \tag{3.7}$$

This optimization can be solved in three ways: by an eigendecomposition approach, by a graph-based approach, or by a combination of both approaches in a least-squares formulation, named the manifold regularization approach (Belkin et al., 2006). In addition to the scatter matrices, each approach can use additional terms (such as a regularization or adjacency matrix) to model the label constraints.

3.3.2.1 The Eigendecomposition Approach

The traditional method to solve the LDA problem in Equation 3.7 is to rewrite it as a generalized eigenvalue problem

$$\boldsymbol{S}_b\boldsymbol{v} = \boldsymbol{\lambda}\boldsymbol{S}_w\boldsymbol{v}, \tag{3.8}$$

and to find all eigenvectors $\boldsymbol{v}_k$ corresponding to nonzero eigenvalues λ_k to form the projection matrix $\boldsymbol{P} = \boldsymbol{V}_{1..k} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k]$ with $k \leq C-1$. This form helps us to see why labels can be considered as a kind of discriminant constraint. It arises when solving the lagrangian of the constraint problem equivalent to the one in Equation 3.7

$$\begin{aligned} \boldsymbol{v} = \underset{\boldsymbol{v}\in\mathbb{R}^p}{\arg\min} \quad & -\frac{1}{2}\boldsymbol{v}^T\boldsymbol{S}_b\boldsymbol{v} \\ \text{s.t.} \quad & \boldsymbol{v}^t\boldsymbol{S}_w\boldsymbol{v} = 1, \end{aligned} \tag{3.9}$$

where the unit constraint $\boldsymbol{v}^t\boldsymbol{S}_w\boldsymbol{v} = 1$ eliminates the invariance to rescaling of the projection vectors $\boldsymbol{v}$.

LDA is considered a global method that focuses on the global separation between classes. However, local structures in the data can be revealed through the relationship between data points captured in the neighborhood graph. Another simple way to obtain the local structures, even when the data distribution of each class is non-Gaussian, is introduced in Local FDA (LFDA) (Sugiyama, 2006). The label constraints in LFDA are represented by the *local scatter matrices*, which are the between- and within-class matrices weighted by the

affinities between instances. The weights are enhanced or reduced depending on whether the instances belong to the same or different classes[2].

When both the global structure brought by PCA and the discriminant information from partially labeled data brought by LFDA are desired, these criteria can be combined, for example, in the semi-supervised version of LFDA (SELF) (Sugiyama et al., 2008). The covariance matrix used in PCA is an unnormalized *total scatter matrix*, and is proved being equivalent to the sum of between- and within-class scatter matrices (Fukunaga, 2013): $\boldsymbol{S}_t = \sum_{i=1}^{n} (\boldsymbol{x} - \boldsymbol{\mu}) (\boldsymbol{x} - \boldsymbol{\mu})^T = \boldsymbol{S}_b + \boldsymbol{S}_w$. SELF combines the scatter matrices for both the labeled and unlabeled points as:

$$\begin{aligned} \overline{\boldsymbol{S}_b} &= (1-\beta)\boldsymbol{S}_b + \beta \boldsymbol{S}_t \\ \overline{\boldsymbol{S}_w} &= (1-\beta)\boldsymbol{S}_w + \beta \boldsymbol{I}_p, \end{aligned} \tag{3.10}$$

where $\beta \in [0, 1]$ is a trade-off parameter. SELF collapses to PCA when $\beta = 1$ and to LFDA when $\beta = 0$. LFDA can also be combined with Locality-Preserving Projection (LPP) (He and Niyogi, 2004) in the same fashion (Sugiyama et al., 2008), however, LFDA and LPP both focus on local structures.

In the eigendecomposition approach, the compact form of the objective function in Equation 3.7 is also known as the trace-ratio optimization problem (Jia et al., 2009). This general problem arises when dealing with linear DR methods in supervised (Wang et al., 2007), unsupervised (Wang et al., 2014), and semi-supervised settings (Sanodiya et al., 2020; Zhang et al., 2013). Since a closed-form for these problems exists, the eigendecomposition is an effective approach for small and moderate size datasets.

3.3.2.2 The Graph-based Approach

The original LDA, like PCA, is designed to preserve the global structure of the data manifold while ignoring local structures like the neighborhood information. The graph-based approach can be used to tackle the preservation of local structures in LDA. In the semi-supervised setting, when both labeled and

[2]The affinity $\boldsymbol{A}_{ij} = \frac{\exp(-||\boldsymbol{x}_i - \boldsymbol{x}_j||^2)}{\sigma_i \sigma_j}$ is calculated by a Gaussian kernel with an adaptive Gaussian bandwidths σ_i, σ_j selected heuristically based on the distances to the surrounding neighbors.

unlabeled data are available, the graph can be constructed in two different ways, with or without the labeled data. In the first case, the adjacency graph $\boldsymbol{A}$ is constructed from the nearest neighbor of each point in the whole dataset of both labeled and unlabeled instances (Cai et al., 2007a). In the second case, only the labeled data are used to construct the graph in the same manner. We can observe that the points in the neighborhood of a point $\boldsymbol{x}_i$ can have the same or a different label with respect to $\boldsymbol{x}_i$. Therefore, the neighborhood $N(\boldsymbol{x}_i)$ can be separated into two disjoint sets of neighbors, the ones with the same class (within-class) $N_w(\boldsymbol{x}_i)$ and those with a different class (between-class) $N_b(\boldsymbol{x}_i)$. Locality Sensitive Discriminant Analysis (LSDA) (Cai et al., 2007b) proposes to construct a within-class graph G_w between the neighbors of the same class based on N_w, and a between-class graph G_b between the neighbors of different classes based on N_b. LSDA maximizes the margin between the local area around each point and other points of different classes. In the semi-supervised setting, the adjacency graph of unlabeled data can be used together with these two discriminant graphs (called within-manifold and between-manifold scatter in (Song et al., 2008a)) to achieve both global and local preservation.

After obtaining the adjacency matrix $\boldsymbol{A}$, the structure of the graph can be revealed via the Laplacian matrix $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$, where $\boldsymbol{D}$ is the diagonal matrix of node degrees. The optimization problem in Equation 3.7 can be reformulated with the total scatter matrix $\boldsymbol{S}_t = \boldsymbol{S}_b + \boldsymbol{S}_w$. The generalized eigenvalue problem in Equation 3.8 can be modified to integrate the Laplacian graph as

$$\boldsymbol{S}_b \boldsymbol{v} = \boldsymbol{\lambda}(\boldsymbol{S}_t + \alpha \boldsymbol{X} \boldsymbol{L} \boldsymbol{X}^T)\boldsymbol{v}. \tag{3.11}$$

The Laplacian of the whole dataset (Cai et al., 2007a) or of the within/between-class graphs (Cai et al., 2007b) is integrated into the objective function as a regularization term to help the projection capture the manifold in the graph. Additional information such as the data density can also be integrated into the weighted adjacency matrix (Sun et al., 2017) to enrich the Laplacian matrix.

3.3.2.3 The Manifold Regularization Approach

Both approaches presented for the LDA problem focus on the representation of the labeled and unlabeled data points using the between/within-scatter matrices and the graph Laplacian matrix. LDA-based methods assume that the linear

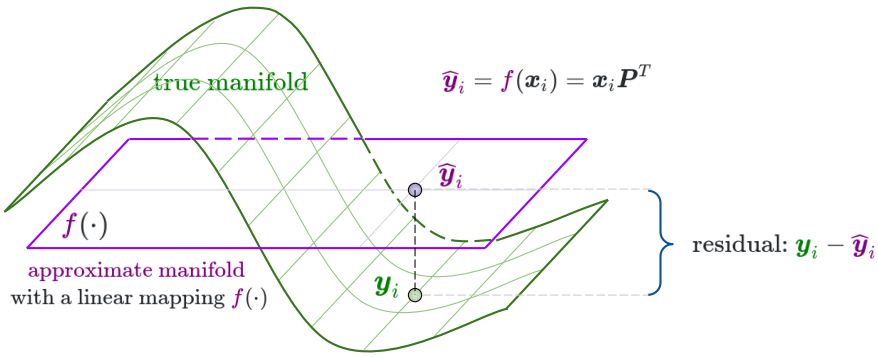


Figure 3.3: The DR method is expected to find an LD subspace that is as close to the unknown intrinsic manifold as possible. From this viewpoint, the DR objective can be reformulated under a regression framework. This figure is based on the original illustration of Nie et al. (2010).

projection, i.e., the projected points in the hyperplane $f(\boldsymbol{X}) = \boldsymbol{X}\boldsymbol{P}^T$, can faithfully represent the HD data. However, HD data are usually assumed to lie on a lower-dimensional manifold that can be highly nonlinear, as illustrated in Figure 3.3. Various nonlinear DR methods have been designed to reveal such nonlinear manifolds (Lee and Verleysen, 2007). Even so, the linear methods are still useful thanks to their simple and elegant formulation (as in Equation 3.8), their closed-form solution, and their ability to project new unseen data. This section presents a new class of linear DR methods that are aware of the nonlinear manifold. These methods are developed based on the idea of manifold regularization (Belkin et al., 2006).

The idea of the manifold regularization approach is to reformulate the LDA problem as a least-squares problem and add regularization terms to control the projection matrix. This formulation is also called the spectral regression LDA problem (Shu et al., 2012). The unified framework for semi-supervised linear DR method introduced in (Song et al., 2008b) can be generalized to various frameworks (Chen et al., 2007; Nie et al., 2010; Shu et al., 2012; Sindhwani et al., 2005), and can be summarized as

$$\boldsymbol{P} = \arg\min_{\boldsymbol{P}} \frac{1}{n} \sum_{i=1}^{n} ||f(\boldsymbol{x}_i) - \boldsymbol{y}_i||^2 + \alpha_T ||f||_T^2 + \alpha_M ||f||_M^2, \tag{3.12}$$

where f is the mapping from the data space to a reduced space of at most C

dimensions, where C is the number of classes. In LDA-based methods, $f(\boldsymbol{x}_i) = \boldsymbol{x}_i\boldsymbol{P}^T$ can be interpreted as a label prediction, $\boldsymbol{y}_i$ being a one-hot vector of the true label of the input data point $\boldsymbol{x}_i$. The first term in Equation 3.12 is thus the prediction error that should be minimized. The mapping f is a function of both the projection matrix $\boldsymbol{P}$ and the input data $\boldsymbol{X}$. Therefore, the regularization terms are thus based on $\boldsymbol{P}$ and $\boldsymbol{X}$. The first regularization term is usually the Euclidean (ℓ_2) norm of the projection matrix $||\boldsymbol{P}||_T^2 = \boldsymbol{P}\boldsymbol{P}^T$, which is also called the Tikhonov regularization, and controlled by α_T. The second regularization term is the manifold regularization, based on the input data $\boldsymbol{X}$ and controlled by α_M. This term is usually defined as $||f||_M^2 = \text{trace}(\boldsymbol{P}\boldsymbol{X}\boldsymbol{L}\boldsymbol{X}^T\boldsymbol{P}^T)$ (Sindhwani et al., 2005; Song et al., 2008b) since the Laplacian matrix can approximate the Laplace–Beltrami operator[3] and control the manifold smoothness (Belkin et al., 2006)). The main advantage of this combination under the least-squares framework is that the regularized methods can preserve both global and local structures (Chen et al., 2007) and better cope with the data sampled from a nonlinear manifold (Nie et al., 2010).

3.3.2.4 Discussion on Label Constraints in LDA-based Methods

Two main directions for integrating constraints into DR methods have been considered so far. First, the objective function of an unsupervised DR method can be adapted to satisfy constraints. Methods in Section 3.2 rely on this first direction. Second, Section 3.3.2 presents semi-supervised DR methods in which partially labeled data are used as constraints that DR methods have to satisfy while projecting the unlabeled data into an LD space. In this case, the discriminant information in the constraints is optimized with a (semi-)supervised objective to preserve additional structures in the unlabeled data or in the whole dataset. The idea of maximizing the between-class separation and minimizing the within-class separation in this section is also applied directly to distance metric learning, including several methods based on LDA such as Discriminant Neighborhood Embedding (DNE) (Zhang et al., 2007b).

Although LDA-based methods are simple and efficient, they have several disadvantages. The main disadvantage is that when the number of instances in each class is small (smaller than the number of dimensions), the scatter/covariance

[3]The Laplace–Beltrami operator is the divergence of the gradient of the data manifold.

matrix is ill-conditioned and cannot be estimated accurately. This issue can be encountered in the semi-supervised setting when the number of labeled instances is usually small. A regularization technique (RDA) (Friedman, 1989; Guo et al., 2007) or two-stage PCA+LDA (Belhumeur et al., 1997; Ye and Li, 2005) are proposed to tackle this singularity problem. Another potential solution is to produce more labels from the available original labels based on label propagation techniques (Shao et al., 2020).

It should be noted that LDA is a linear method that may not be a suitable solution to capture the intrinsic structure when the underlying manifold of the data is highly nonlinear. In order to overcome this issue, linear methods can be extended with a kernel for both global (Cai et al., 2007a) and local (Song et al., 2008a) LDA-based methods. When using a kernel, the input adjacency matrix and the weighted matrix of labeled points are still computed in the original space. Data are mapped to the kernel space, and the eigendecomposition is solved in this space. Supervised nonlinear DR techniques such as supervised ISOMAP (Geng et al., 2005) can also help in this situation.

LDA-based methods only work when at least a small number of labels are available. However, when the labels are not available, the idea of LDA can still be applied for other kinds of constraints. For example, instances connected by must-link constraints can form a group of points sharing the same unknown label, called the *chunklets* in Relevant Component Analysis (RCA) (Bar-Hillel et al., 2005). RCA uses the points (of unknown class) belonging to the same group to construct a *within-chunklet* covariance matrix and uses the whitening transform of this matrix as the final projection matrix.

3.3.3 Multi-Aspect Constraints

When working with DR algorithms, the input is a dataset $D = \{\boldsymbol{X}, \boldsymbol{Y}\}$ consisting of the input data $\boldsymbol{X}$ and possibly the labels $\boldsymbol{Y}$. Several DR methods reduce the dimensions of X while respecting the constraints between instances, constraints on the features, and constraints in the form of labels in $\boldsymbol{Y}$ that have been reviewed. However, in a real-world setting, the input data can be collected in different unusual forms. Different sets of features (called a multi-view dataset) characterizing the same data instances can exist. These views can also be of different nature. For instance, a heterogeneous dataset can contain texts and

images, in which each picture has an associated textual description (called a multi-modal dataset). Within a dataset, each instance can have one or more labels associated with it (called a multi-label dataset). The important point in this categorization is that each data point can have multiple aspects associated with it. The ith data point can appear in the two sets of features denoted as $\boldsymbol{x}_i^{(1)}, \boldsymbol{x}_i^{(2)}$ or can have two labels $y_i^{(1)}, y_i^{(2)}$. The DR methods applied to these kinds of datasets should find an embedding that represents all the aspects available for a dataset. Therefore, different aspects of a dataset are considered as constraints for the DR methods, which we call *multi-aspect constraints*.

3.3.3.1 Multi-view Constraints

A multi-view dataset is denoted $D = \{\boldsymbol{X}^{(1)}, \boldsymbol{X}^{(2)}, \ldots, \boldsymbol{Y}\}$, where $\boldsymbol{X}^{(1)}, \boldsymbol{X}^{(2)}, \ldots$ are different feature sets that explain different aspects of the data. For instance, a researcher can collect data on the performance of college students and have two separate sets of psychological variables (such as *motivation level* or *locus of control*), as well as academic variables (such as the standard test scores on mathematics, writing, and reading). The researcher might be interested in finding a small number of (unknown) variables that can explain the association between the two sets of variables. When users only want to visualize their data using only one feature set, they can use the other external variables to explain the visualization (Bibal et al., 2018, 2021; Marion et al., 2019). When all the feature sets should be considered, the DR problem of projecting the data points into a subspace is formulated as a constrained problem for which we have to preserve the hidden correlation across multiple views.

In statistics, Canonical Correlation Analysis (CCA) (Hotelling, 1936) is used to find the relationship between two sets of variables of the same instances. From a probabilistic perspective, the two observed datasets $\boldsymbol{X}^{(1)} = \{\boldsymbol{x}_i^{(1)} \in \mathbb{R}^{p1}\}, \boldsymbol{X}^{(2)} = \{\boldsymbol{x}_i^{(2)} \in \mathbb{R}^{p2}\}$ are generated from unknown latent variables. Therefore, $\boldsymbol{X}^{(1)}$ and $\boldsymbol{X}^{(2)}$ could probably share a common variation. CCA helps to remove the noise or data-specific variation in each dataset (the variation that is not present in the other dataset) and only keeps the shared variation that captures the common latent variables. The two datasets are first projected to a LD space using two projection matrices $\boldsymbol{U} = [\boldsymbol{u}_k] \in \mathbb{R}^{q \times p1}$ and $\boldsymbol{V} = [\boldsymbol{v}_k] \in \mathbb{R}^{q \times p2}$. CCA seeks for q pairs of projection vectors $\{\boldsymbol{u}_k, \boldsymbol{v}_k\}_{k=1}^{q}$

(also called the *canonical variate pairs*) to mutually maximize the correlations between $\boldsymbol{X}^{(1)}\boldsymbol{u}_k^T$ and $\boldsymbol{X}^{(2)}\boldsymbol{v}_k^T$. As CCA aims to project the data in two different views into a shared low dimension space, it is used mainly for reducing the data dimensions for subsequent tasks. This LD space is consistent with the latent variables in the CCA model and the reduced data by CCA do not lose any information for performing subsequent tasks (Foster et al., 2008). The first canonical variate pair $\{\boldsymbol{u}_1, \boldsymbol{v}_1\}$ can be used to project the data from the two views onto these two canonical variables. By doing this, a scatter plot in which each axis captures the most representative information (variance) in each dataset can be obtained.

In order to understand why different views are considered as constraints, we reformulate the CCA problem as a constrained problem. Since the projection vectors are scale-invariant, the correlation maximization

$$\max_{\boldsymbol{u},\boldsymbol{v}} \rho\left(\boldsymbol{X}^{(1)}\boldsymbol{u}^T, \boldsymbol{X}^{(2)}\boldsymbol{v}^T\right) = \frac{\boldsymbol{u}^T\boldsymbol{X}^{(1)}\boldsymbol{X}^{(2)T}\boldsymbol{v}}{\sqrt{\left(\boldsymbol{u}^T\boldsymbol{X}^{(1)}\boldsymbol{X}^{(1)T}\boldsymbol{v}\right)}\sqrt{\left(\boldsymbol{u}^T\boldsymbol{X}^{(2)}\boldsymbol{X}^{(2)T}\boldsymbol{v}\right)}} \tag{3.13}$$

can be rewritten as

$$\begin{aligned} \max_{\boldsymbol{u},\boldsymbol{v}} \quad & \boldsymbol{u}^T\boldsymbol{X}^{(1)}\boldsymbol{X}^{(2)T}\boldsymbol{v} \\ \text{s.t.} \quad & \begin{cases} \boldsymbol{u}^T\boldsymbol{X}^{(1)}\boldsymbol{X}^{(1)T}\boldsymbol{v} = 1, \\ \boldsymbol{u}^T\boldsymbol{X}^{(2)}\boldsymbol{X}^{(2)T}\boldsymbol{v} = 1. \end{cases} \end{aligned} \tag{3.14}$$

The covariance matrices which appear in this formulation can help to integrate the discriminant information into CCA.

Similar to other linear DR methods, CCA is efficient for small datasets thanks to its linearity and its closed-form solution via eigendecomposition. However, it has two main disadvantages. First, CCA is not suitable for datasets with highly nonlinear manifolds. A kernel version of CCA is introduced to tackle this issue (Akaho, 2006). CCA is also combined with the idea of nonlinear graph-based DR methods such as locality preserving projections (LPP) to better preserve the nonlinear intrinsic structures of the data (Sun and Chen, 2007). Second, CCA is designed to capture the correlation between only two datasets. A famous extension to deal with more datasets is called Multi-set CCA (MCCA) (Kettenring, 1971). The basic idea of MCCA and its variants

is to maximize the correlation between multiple datasets defined through the concept of inter-set correlation, which is the ratio of the *between-set* and the *within-set* covariances. An idea of *kernel matching* is also applied to measure the dependencies between multiple views (Zhang et al., 2017).

3.3.3.2 Multi-label Constraints

In supervised learning, each data point has only one label. In multi-label datasets, a data point may have multiple labels. For example, a document can belong to different topics, or an image of a beach can belong to scene classes such as beach, sea, land, or sky. Since these kinds of data are encountered in machine learning, various multi-label learning algorithms are developed for the multi-label classification (Zhang and Zhou, 2014) or multi-output regression problems (Borchani et al., 2015). However, under the unsupervised setting, very few DR methods tackle this problem. A DR method applied to multi-label datasets should reduce the dimensions of the data while respecting the complex relationship of each data point when they are constrained by a set of multiple labels.

The first approach for multi-label DR methods uses the idea of LDA to reveal the discriminant information in the class labels. Since one data point is assigned to several labels, it is hard to directly estimate the between/within-class scatters. Multi-label LDA takes advantage of label correlation and introduces the *class-wise scatter matrices* to avoid the label ambiguity (Wang et al., 2010). Moreover, the unlabeled data point can be assigned a *soft label*, that is the label information propagated from its neighbors (Guo et al., 2016). In contrast to this approach that relies on the correlation between multiple labels, another approach exploits the relationship between the features and each label. Since a data point is assigned to different labels, there may exist different feature sets in the feature space that correlate to each label. Therefore, a DR method can project the data to a subspace on which the dependence between the projected features and the target labels are maximized (Mikalsen et al., 2019; Zhang and Zhou, 2008).

3.3.3.3 Discussion on Multi-aspect Constraints

Multi-label and multi-view are very specific settings in machine learning and are considered as a specific kind of constraint between different aspects of a dataset. These settings require that different views or labels must be associated with the same set of instances, i.e., the same data point should appear in different feature sets or have multiple labels. The constrained DR methods introduced in this section are mainly extended from LDA and CCA to deal with multi-label and multi-view constraints. These methods cannot be applied when different datasets or label sets are not associated with each other. However, a different approach using transfer learning (Wang et al., 2008) allows applying LDA-based techniques on different non-associated datasets in the same domain. For example, one can have a dataset containing faces of unknown people and another labeled face datasets such as the Yale or the AT&T datasets. LDA or its variance is first applied on the labeled face dataset to learn important features to discriminate visages. The learned mapping can be then applied to the unknown faces for extracting features from this unlabeled dataset.

3.4 Discussion

This survey studies (i) how to integrate different kinds of constraints into DR methods, and (ii) how to solve the constrained problem. The former is about constraint representation, the latter is about constraint optimization. The constraint optimization problem can be solved by fixed traditional approaches including eigendecomposition methods, iterative/gradient-based methods, or inference algorithms for probabilistic models. The constraint representation is much more varied. Sometimes, the constraint representation is tied to the optimization method such as the block-matrix representation (Section 3.2.1.1) or the constraint encoding via covariance matrices (Section 3.3.2.1). Other times, the constraint representation is completely independent of the constraint optimization, for which a great example is a probabilistic approach (Section 3.2.1.2) where different inference algorithms can be used to optimize the same model.

After reviewing many traditional constrained DR methods, this survey provides several guidelines to choose them in different situations, with a focus on their usefulness and application.

- Case 1: The users start from scratch and do not know which DR method and which type of constraints to use. They can follow Figure 3.1 to navigate the specific section in this survey.
- Case 2: The users are currently using a specific DR method. However, they encountered problems with their DR method, and they would like to use constraints/side information in order to fix these issues. They can follow the summary in Table 3.1.
- Case 3: The users have specific requirements, which are the criteria that the DR methods must preserve. For example, they want to preserve both global and local structures in their data. Assuming that the users can collect the required constraints, they can follow Table 3.2 to choose the suitable DR methods that match their preservation criteria.

Table 3.1: Several common issues with the traditional DR methods and the solutions with constraints.

Base	Issues/Needs	Solution with Constraints	Constrained Methods	Ref.
MDS	Rotation invariance and lack of axes interpretation	Knowing in advance the position of several points	XGVis (Buja et al., 2001, 2008)	3.2.1.1
			iPMDS (Vu et al., 2021c)	3.2.1.2
			V2PI-MDS (Endert et al., 2011; Leman et al., 2013)	3.2.1.3
		Using external features	BIR (Bibal et al., 2018; Marion et al., 2019), BIOT (Bibal et al., 2021)	3.3.3.1
PCA	Lack of interactive controls to evaluate the importance of features	Interactively setting feature weights;	iPCA (Jeong et al., 2009, 2015)	3.3.1
		Using two-way interaction tools	Forward-Backward Projection (Cavallo and Çağatay Demiralp, 2017), Star-coordinates Interaction (Molchanov and Linsen, 2014)	
	Lack of interactive way to set constraints of feature values	Using domain knowledge of value bounds for specific features	Bounded PCA (Giordani and Kiers, 2007)	3.3.1
	Lack separation in the visualization	Knowing in advance the position of several points	V2PI-PCA (Endert et al., 2011)	3.2.1.2
	Lack of local structures	Combining with a local discriminant method	SELF (Sugiyama et al., 2008)	3.3.2.1
PPCA	Rotation invariance or lack of separation in the visualization	Knowing in advance the position of several points	iPPCA (Vu and Frénay, 2019)	3.2.1.2
			User-guided PPCA (House et al., 2015)	

Continuation of Table 3.1

LDA/ FDA	Lack of interactive controls	Interactively setting feature weights	iLDA (Choo et al., 2010)	3.3.1
	Lack of intrinsic geometric / local structures	Using local scatter matrices, possibly in a semi-supervised setting	Local FDA (Sugiyama, 2006), SELF (Sugiyama et al., 2008) LFDA + iLPP (Sugiyama et al., 2008)	3.3.2.1
		Combining with graph-based methods to reveal intrinsic structures	SDA (Cai et al., 2007a), LSDA (Cai et al., 2007b), CGMSDR (Sun et al., 2017)	3.3.2.2
	A data point can have multiple labels	Extending to multi-label setting	Multi-label LDA (Wang et al., 2010), SSMLDR (Guo et al., 2016)	3.3.3.2
LDA/ MMC	Not using unlabeled data when available	Using the graph constructed from unlabeled data	S3MPE (Song et al., 2008a)	3.3.2.2
	Lack of mechanism to add regularization	Reformulating the LDA problem in a least squares framework	(Spectral) LocLDA (Shu et al., 2012), LapLDA (Sindhwani et al., 2005), LSLDA (Chen et al., 2007), FME (Nie et al., 2010)	3.3.2.3
	Depending on the available labels	Using pairwise constraints to form groups	RCA (Bar-Hillel et al., 2005)	3.3.3.2
	Cannot work well with the underlying nonlinear manifold	Extending with kernels	SDA (Cai et al., 2007a), S3MPE (Song et al., 2008a)	3.3.3.2
CCA	Can handle only two views	Extending to multi-view	MCCA (Kettenring, 1971)	3.3.3.1
		Using kernel matching	FMDR (Zhang et al., 2017)	
	Cannot handle nonlinear data	Extending with kernel or combining with a nonlinear graph-base method	KCCA (Akaho, 2006) Locality CCA (Sun and Chen, 2007)	3.3.3.1
Isomap	Lack of interactive controls	Knowing in advance the position of several points	SS-Isomap (Yang et al., 2006)	3.2.1.1
			V2PI-Isomap (Leman and Endert, 2010)	3.2.1.3
	Lack of discriminant information in the embedding	Using class labels with pairwise constraints	SSD-Isomap (Huang et al., 2019)	3.2.2.2
LPP	Similar points not in the same neighborhood	Collecting pairwise constraints	GCDR-LP (Davidson, 2009), CLPP (Cevikalp et al., 2008), Robust CLPP (YU et al., 2010)	3.2.2.1
	Lack of discriminant structures	Combining with a local discriminant method	LFDA + iLPP (Sugiyama et al., 2008)	3.3.2.1
t-SNE	Lack of global structure	Collecting triplet constraints	t-STE (van der Maaten and Weinberger, 2012), SNaCK (Wilber et al., 2015)	3.2.3
		Using prior knowledge of hierarchy	HCt-STE (Vu et al., 2021b)	

Continuation of Table 3.1

Cannot see evolution during each iteration	Setting position of points in early iteration	PIVE-*t*-SNE (Kim et al., 2017)	3.2.1.1

Table 3.2: Several common criteria of constrained DR methods.

Constrained DR Methods	Ref.	Preservation of			
		Local Structure	Global Structure	Discriminant Info.	Distance
XGVis (Buja et al., 2001, 2008)	3.2.1.1		✓		✓
SS-LTSA (Yang et al., 2006)	3.2.1.1		✓		
PIVE-*t*-SNE (Kim et al., 2017)	3.2.1.1	✓			
SS-LLE (Yang et al., 2006)	3.2.1.1		✓		
SS-Isomap (Yang et al., 2006)	3.2.1.1		✓		✓
V2PI-PCA (Endert et al., 2011)	3.2.1.2		✓		
User-guided PPCA (House et al., 2015)	3.2.1.2		✓		
iPPCA (Vu and Frénay, 2019)	3.2.1.2		✓		
iPMDS (Vu et al., 2021c)	3.2.1.2		✓		✓
V2PI-Isomap (Leman and Endert, 2010)	3.2.1.3		✓		✓
V2PI-MDS (Endert et al., 2011; Leman et al., 2013)	3.2.1.3			✓	
CLPP (Cevikalp et al., 2008)	3.2.2.1	✓			
Robust CLPP (YU et al., 2010)	3.2.2.1	✓			
GCDR-LP (Davidson, 2009)	3.2.2.1	✓			
SSD-Isomap (Huang et al., 2019)	3.2.2.2		✓	✓	
t-STE (van der Maaten and Weinberger, 2012)	3.2.3	✓			✓
SNaCK (Wilber et al., 2015)	3.2.3	✓			✓
HC*t*-STE (Vu et al., 2021b)	3.2.3	✓	✓		✓
iPCA (Jeong et al., 2009, 2015)	3.3.1		✓		
Forward-Backward Projection (Cavallo and Çağatay Demiralp, 2017)	3.3.1		✓		
Star-coordinates Interaction (Molchanov and Linsen, 2014)	3.3.1		✓		
Bounded PCA (Giordani and Kiers, 2007)	3.3.1		✓		
iLDA (Choo et al., 2010)	3.3.1		✓	✓	
SELF (Sugiyama et al., 2008)	3.3.2.1	✓	✓	✓	
LFDA + iLPP (Sugiyama et al., 2008)	3.3.2.1	✓		✓	

Continuation of Table 3.2

		Local Structure	Global Structure	Discriminant Info.	Distance
Local FDA (Sugiyama, 2006)	3.3.2.1	✓	✓	✓	
SDA (Cai et al., 2007a)	3.3.2.2	✓	✓	✓	
LSDA (Cai et al., 2007b)	3.3.2.2	✓	✓	✓	
CGMSDR (Sun et al., 2017)	3.3.2.2	✓	✓	✓	
S3MPE (Song et al., 2008a)	3.3.2.2	✓	✓	✓	
(Spectral) LocLDA (Shu et al., 2012)	3.3.2.3	✓	✓	✓	
LapLDA (Sindhwani et al., 2005)	3.3.2.3	✓	✓	✓	
LSLDA (Chen et al., 2007)	3.3.2.3	✓	✓	✓	
FME (Nie et al., 2010)	3.3.2.3	✓	✓	✓	
BIR (Bibal et al., 2018; Marion et al., 2019), BIOT (Bibal et al., 2021)	3.3.3.1		✓		✓
MCCA (Kettenring, 1971)	3.3.3.1		✓		
FMDR (Zhang et al., 2017)	3.3.3.1		✓		
KCCA (Akaho, 2006)	3.3.3.1		✓		
Locality CCA (Sun and Chen, 2007)	3.3.3.1	✓	✓		
RCA (Bar-Hillel et al., 2005)	3.3.3.2	✓	✓		
Multi-label LDA (Wang et al., 2010)	3.3.3.2		✓	✓	
SSMLDR (Guo et al., 2016)	3.3.3.2		✓	✓	

Table 3.3: Analysis of three principal approaches for solving constrained DR problems.

Eigen-based	Graph-based	Regularization-based
+ Elegant formulation, closed-form solution. + Fast and efficient for small and medium datasets	+ Taking advantage of the global information in the graph Laplacian + Can be reformulated as a generalized eigenvalue problem to obtain a closed-form solution	+ Flexible in representing the constraints + Can be solved by an eigendecomposition or a gradient-based method + Can be scaled for large datasets
− Cannot be scaled for large datasets due to the cubic complexity of the eigendecomposition procedure	− The neighbor graph is sensitive to the number of nearest neighbors or the radius of the neighborhood zone	− Complex formulation involving the manifold regularization

Using Table 3.1, we change the perspective from the aspect of the integration of constraints into DR methods to a *problem-solution* perspective. From this new aspect, we look at important issues of traditional DR methods and the solution brought by constrained methods. After finding potential constraint DR methods for the given problem, users can also look at the general pros and cons of the underlying approach of these methods. Table 3.3 summarizes the advantages and disadvantages of three common approaches (eigen-based, graph-based, and regularization-based). It should be noted that some computational aspects of the constrained DR methods are not reviewed, including running time, complexity, scalability, reproducibility, and code availability.

This survey includes our own proposed constrained DR methods. Through this survey, we found three interesting findings, which motivate us towards our proposed methods. First, a probabilistic approach for DR is not popular and there are very few works that model the constraints in a probabilistic manner. Constraints can be integrated into unsupervised and (semi-)supervised methods, dominated by eigendecomposition and graph-based approaches. For that reason, we propose new methods for integrating constraints into probabilistic DR methods to fill this gap in the literature (see Chapter 5). Second, most reviewed methods start from available constraints or from a *human-in-the-loop* process to find out the way to integrate users' feedback/constraints. We propose to look at these methods from a *problem-solution* perspective, where we focus on the problems of commonly used DR methods and find out solutions realized by constraint integration. Under this perspective, we analyze a long-standing problem of global structure preservation of SNE-based methods and propose a solution using hierarchical constraints (see Chapter 4). Finally, all the reviewed methods use constraints to enhance DR methods. We have not extended the scope of this survey to find other usages of constraints, however, directly injecting constraints into a base DR method is the most common approach. We contribute to enriching the literature of constraint usage with a novel way to use the users' constraints to evaluate the quality of a visualization (see Chapter 6).

Chapter 4

Integration of Hierarchical Constraints into t-SNE

This chapter presents a new idea of integrating hierarchical structure in the form of a tree into t-SNE visualizations with a method called **HCt-SNE**.

Contents

4.1 The Need for Visualizations with Hierarchical Structures 59
4.2 From Desired Hierarchical Structures to Hierarchical Constraints 66
4.3 HCt-SNE: Hierarchical Constraints with t-SNE 72
4.4 Experimental Results of HCt-SNE 81
4.5 Discussion 90

This chapter is based on our publication titled "HCt-SNE: Hierarchical Constraints with t-SNE" (Vu, Bibal, and Frénay, 2021b).

Traditional DR methods like PCA or LDA work well for small or moderate datasets. These methods are designed for general DR tasks and used as a pre-processing step. When analyzing large and more complex datasets like images or textual documents, people not only want to have a good embedding in an LD space but also want to visualize their data for some exploratory analyses. For that reason, specific DR methods are designed only for visualizing the data. t-distributed stochastic neighbor embedding (t-SNE) was introduced by Maaten and Hinton (2008), and through time becomes one of the most widely used visualization methods. Even though t-SNE is more computationally demanding than other traditional DR methods, it can work with large datasets of different data types. Groups of similar points are what we usually look for when observing a visualization. By focusing on neighborhood relationships in the original data, t-SNE produces group structures in the embedding as a way to preserve the neighborhood information.

However, t-SNE does not always work well with complex data like the high-dimensional raw input pixels of color images. t-SNE was also found to have several issues when representing the global structure in the data (Wattenberg et al., 2016). This chapter is devoted to explaining our contribution to enhance t-SNE visualization with global hierarchical structure via the proposed **HC*t*-SNE** method (Vu et al., 2021b). Our general idea is summarized in Figure 4.1. Starting with the commonly used CIFAR10 dataset (Krizhevsky, 2009), t-SNE produces a useless visualization (crowded and difficult to analyze) shown in the top-right. This problem of t-SNE and the motivation for a better visualization is analyzed in Section 4.1. We propose an original idea of injecting hierarchical structure in the form of a tree into t-SNE embeddings. Section 4.2 explains step-by-step through examples how to transform the hierarchical tree into **hierarchical constraints**, which can be used in t-SNE. Section 4.3 details the algorithms and the optimization process in our method HCt-SNE. Quantitative and qualitative comparisons between HCt-SNE and other SNE-based methods are introduced in Section 4.4. More in-depth analysis and discussion of the algorithm will be given in Section 4.5.

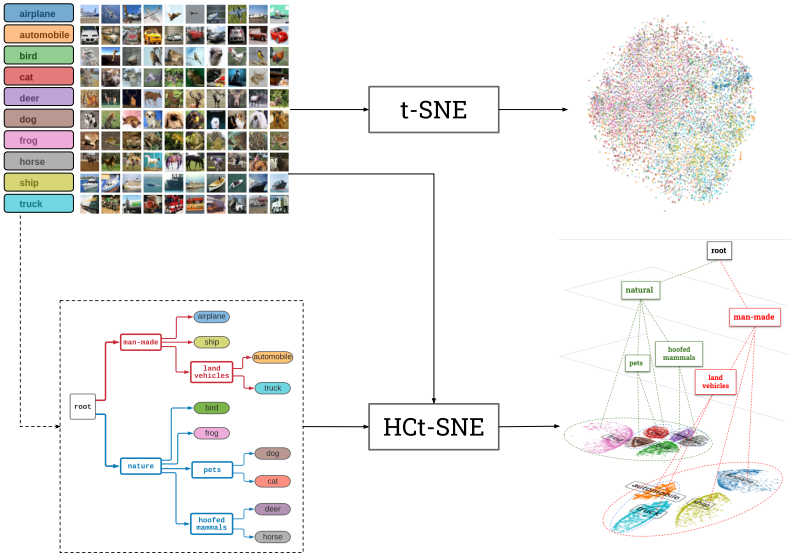


Figure 4.1: An overview of the idea of HCt-SNE. For a complex dataset of color images, t-SNE does not always perform well since the global structure in the data may not be preserved. We propose HCt-SNE, a new idea to integrate hierarchical constraints in the form of a hierarchy tree. In that way, we can reveal the hierarchical structure in the visualization like the one shown in the bottom-right.

4.1 The Need for Visualizations with Hierarchical Structures

Exploratory Data Analytic techniques help us to reveal the *story behind data* (Spiegelhalter, 2019). They make use of data visualization methods to summarize the main characteristics of the data. Several basic graphical techniques like histogram, box plot or scatter plot can help us to understand the data features and the relationship between them (Spiegelhalter, 2019). Scatter plots produced by t-SNE with two or three dimensions learned from neighborhood information extracted from the HD input data are one of the most effective ways to communicate the finding in data, specifically in bioinformatics (Kobak and Berens, 2019; Kobak and Linderman, 2021; Linderman et al., 2019; Narayan et al., 2021). Most of the time, the interesting finding revealed in t-SNE embeddings is the group structures where similar data points are grouped close together to form clusters in the visualization. This kind of

structure is usually very pleasing to the human eye. However, t-SNE still has several drawbacks as pointed out by Wattenberg et al. (2016). In this chapter, we focus on two issues related to the global structure preservation problem and try to propose a new solution. Other prior works addressing these issues will also be discussed.

4.1.1 Problems with t-SNE Visualizations

t-SNE does not always produce faithful visualizations, not only for complex HD datasets like color images but also for simple LD datasets (Wattenberg et al., 2016). The overall problem is that t-SNE does not always preserve global structures in the data. In this section, we consider *global structure* from two aspects, which can be seen as two main issues of t-SNE.

Issue with global *semantic* structure

Figure 4.2: Background effect on a t-SNE embedding of CIFAR10. Images with the same background are placed close together regardless of the object in each image.

An example embedding of t-SNE for the CIFAR10 dataset in the top-right corner of Figure 4.1 shows the overlap and blending of clouds of points. In this case, it is impossible to identify groups even when the points are colored using the ground-truth class labels. Let us look closely at the visualization and display

the corresponding image for each data point. Figure 4.2 shows a grid of sample images in the 2D embedding space of t-SNE. Images with a similar background but not a similar object, are placed close together. The color density pattern may not be what we are looking for, and many other DR methods reveal this type of pattern for CIFAR10 when the raw pixels are used. The reason for this phenomenon is that t-SNE and other methods use the Euclidean distance ($L2$-pixel distance) to measure the similarity between images. Even though the Euclidean distance is not intuitive in the HD space, it usually works in practice for the normalized numerical features of tabular datasets. For image datasets where small objects often appear on a large background, this kind of distance puts a strong focus on the background. Therefore, it is hard for DR methods to discover the *semantic* information in the input images. Let us consider the three images , , and from the CIFAR10 dataset for an illustration. The airplane in the blue sky is around three times closer to the bird in the blue background than to the airplane on a gray background. This effect leads to a poor neighborhood estimation (around 38% accuracy for KNN classification in the LD space with $K = 10$). Since t-SNE tries to preserve this poor neighborhood structure, its embedding cannot reveal the real semantic groups in the dataset.

Issue with *relative* distances between groups

In contrast to the complex image dataset in the first example, in this case we consider a simple artifical datasets of three two-dimensional Gaussian with 200 data points each (Wattenberg et al., 2016). Figure 4.3 shows the original dataset on the left with three groups in which the orange and green groups are 5 times more distant than the orange and blue ones. t-SNE is run with different values of perplexity ranging from a small value of 5 to a large value of 100. None of the visualizations on the right of Figure 4.3 reveal the relative distances between the three input Gaussians. The perplexity of t-SNE is said to be used to control the trade-off when preserving local vs. global structure (Maaten and Hinton, 2008). However, we do not see the true global pattern with any perplexity. The perplexity in this example simply controls the compactness of local groups. Wattenberg et al. (2016) claims that most of the time, "distances between well-separated clusters in a t-SNE plot may mean nothing".

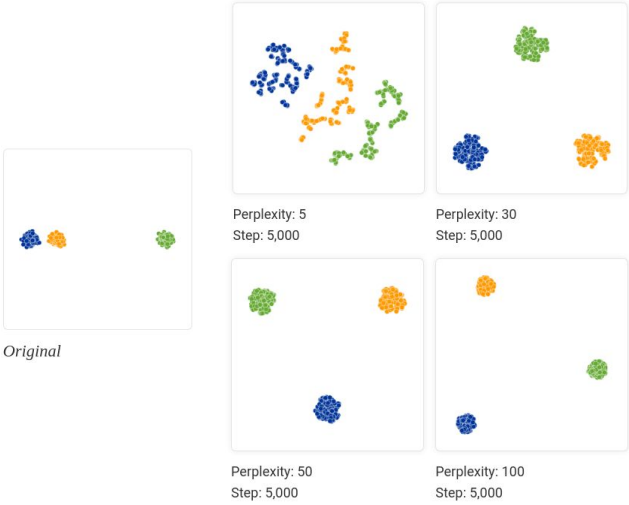


Figure 4.3: Example where t-SNE does not preserve the distance between separated clusters in several visualizations, reproduced from (Wattenberg et al., 2016): semantic information may not be taken into account.

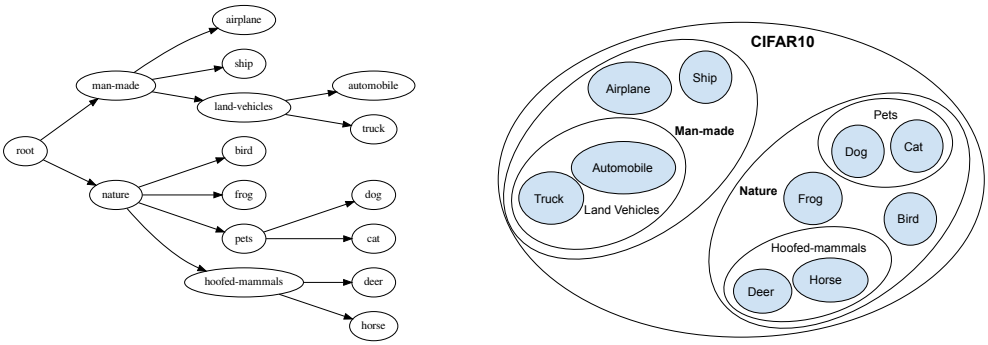
The above issues motivate us towards a goal of enhancing global structure in t-SNE embeddings. In order to tackle both the problems of highlighting semantic structure and relative relationship between groups, we propose to incorporate a *global hierarchical structure* into t-SNE.

4.1.2 Hierarchical Structure in a Visualization

Throughout this thesis, we aim at integrating prior knowledge into widely used DR methods. We aim to integrate knowledge expressed as a hierarchical structure into the 2D visualization of t-SNE. Here we assume that users have prior knowledge about the hierarchical structure in their data. This kind of structure can be very simple, for example, the user may just know a general fact that there are 10 classes in the dataset. A more detailed hierarchical structure can be obtained when the user has more specific knowledge or assumptions such as abstract groups that contain the concrete groups.

An example of hierarchical structure that the user may have for the CIFAR10 dataset is shown in Figure 4.4a. By looking at the available label names of 10 classes in the dataset, human users can easily identify two abstract groups

based on the semantic meaning of the class labels: a group of *man-made* objects (containing images of vehicles) and another group of *nature* objects (containing images of animals). In each group, we can also define other groups containing subgroups, and so on and so forth. This kind of structure can help us to have a global view of the semantic of the data. By specifying the child-parent relationship in the hierarchy, the *relative relationship* between the concrete groups in the dataset is more clearly revealed. A tree in Figure 4.4a is a simple and efficient way to represent the hierarchical structure by its nature, which is very intuitive and easy to understand for humans.



(a) Hierarchical constraints expressed as a tree by the user.

(b) Hierarchical structure as enforced in the visualization.

Figure 4.4: Example of hierarchical constraints to embed human knowledge in a visualization of the CIFAR10 dataset.

Our goal is to embed this kind of tree structure into t-SNE visualizations. Figure 4.4b shows a pictorial illustration of the desired visualization enhanced by the hierarchical structure in Figure 4.4a. In this figure, the boundaries around the groups are one way that users may use to interpret the hierarchical structure. This figure shows an idea of the desired visualization where groups are separated well enough to be easily distinguished. Likewise, child groups under the same parent group should be closer to each other to enhance the child-parent relationship. Now we see the importance of the relative distances between groups that can help to reveal the hierarchical structure in a 2D embedding. Unfortunately, t-SNE does not preserve this kind of distance as shown previously. In the following sections, we will develop the idea of enhancing the relative distances between semantic groups in order to get close to the desired visualization in Figure 4.4b.

4.1.3 Our Contribution: Bringing Hierarchical Structure into t-SNE Embeddings

We have pointed out the problems of t-SNE and the need for a visualization that can reveal both global hierarchical structure and semantic information in the data. We propose to embed the user-defined hierarchical structure in the form of a tree directly into t-SNE visualizations with a new method called HCt-SNE- Hierarchical Constraints with t-SNE (Vu et al., 2021b). Using a tree is a natural way for human to express their knowledge about the hierarchy, as pointed out by cognitive scientists (Hirtle, 1995). Therefore, we choose to use a tree structure to encode the knowledge of users and consider this tree as hierarchical constraints to preserve. In brief, our solution provides a global view to look at the dataset as a whole before diving into the detail of each subgroup. It also enhances semantic information in the dataset by focusing on the relative relationship between child groups and their parent groups as well as their sibling groups.

Even though a tree structure is a rich representation for humans, it is just an abstract data structure for computers. User knowledge is represented by a notation of semantic nodes in the tree but not the concrete data points, this kind of information is not usable for machine learning algorithms. The important step here is to transform the user knowledge in the form of a tree to a numerical representation that can be used by DR methods. Based on the user knowledge encoded in a hierarchical tree, we introduce a new concept called **hierarchical constraints**, which are the *triplet* constraints extracted from the nodes in the tree. Section 4.2 provides more details and examples to explain what triplet constraints are, how the hierarchical constraints look like, how they are constructed, and why they are useful.

4.1.4 Related Work

As discussed above, t-SNE has two major issues: the lack of global structure preservation and the lack of consideration of the semantic. Prior works have tackled these issues and can be categorized into four groups.

The first group of methods addresses the problem of global structure in t-SNE embeddings. Den-SNE (Narayan et al., 2021) is a density-preserving

approach that takes into account the cluster size, i.e., the density represented by the number of points in each group of similar data points. Uniform manifold approximation and projection (UMAP) (McInnes et al., 2018a) claims to preserve global structures better than t-SNE. Its n_neighbors hyperparameter controls the trade-off between global and local structures, while min_dist controls the appearance of groups in the embedding. Global t-SNE (Zhou and Sharpee, 2018) combines the original KL loss with a global cost function, focuses on large distances in both the HD and LD spaces, and shows an improvement on small and simple datasets.

The second group of methods contains supervised DR methods that include class labels in the DR process to address the lack of semantic in the visualization for image datasets. This includes neighborhood component analysis (Goldberger et al., 2005), UMAP in a supervised setting (McInnes et al., 2018a) and class-aware t-SNE (de Bodt et al., 2019), which all use class labels to consider the semantic information in the embedding.

The third group includes methods that combine visual analytic techniques to discover hierarchical structures in the embedding, such as hierarchical stochastic neighbor embedding (HSNE) (Pezzotti et al., 2016). This interactive method for real-time analysis is used for massive datasets like cytometry data (van Unen et al., 2017). HSNE incorporates the principle of *Overview-First, Details-On-Demand* by constructing the hierarchical representation of the data based on the user's given landmarks at different scales.

The fourth group uses the triplet loss to obtain similar representations for similar points and vice versa. The triplet loss is widely used in deep learning for face recognition (Schroff et al., 2015), image retrieval with deep metric learning (Ge, 2018) or self-supervised visual representation learning (Chen et al., 2020). For visualization, the triplet loss can be used to directly update the embedding, such as in t-distributed stochastic triplet embedding (t-STE) (van der Maaten and Weinberger, 2012) and TriMap (Amid and Warmuth, 2019). t-STE uses a heavy-tailed Student-t kernel (that focuses on local similarities) to measure triplet satisfaction. TriMap uses a custom contrastive loss based on triplet constraints weighted by pairwise distances in the HD space.

Although the above methods enhance t-SNE, none of them solve the lack of semantic and global structure at the same time. Moreover, they do not allow users to express directly the semantics they expect in the visualization, except

HSNE. Yet, it only offers an overview of the global structure, and feedback is given on separate sub-parts of the embedding, whereas our method provides a global solution.

Our method differs from TriMap and STE in two points. First, we focus on a small number of informative triplets that encode the hierarchy in the input constraints instead of sampling all possible triplets. Second, we combine the property of preserving both global and local structures instead of focusing only on local structure as in t-STE or only on global structure as in TriMap. In the next section, we will go from the need for visualization with a hierarchical structure to the construction of hierarchical constraints and introduce a new visualization method that preserves this kind of constraint.

4.2 From Desired Hierarchical Structures to Hierarchical Constraints

As introduced before, the user's desired hierarchical structure is expressed as a tree, an abstract representation to capture the semantic relationship between data groups (represented as tree nodes). We propose to extract the hierarchical information in the tree using *hierarchical constraints* between data instances in the tree nodes. In fact, a hierarchical constraint is a *triplet constraint* between data instances, which is constructed at different levels of abstraction according to the corresponding tree nodes. The root node is considered to have the highest level of abstraction and contains the entire dataset. When going down the tree, the level of abstraction decreases as groups are split until the leaf nodes. A triplet constraint is represented as a tuple (y, y^+, y^-) of an anchor y, a positive example y^+ and a negative example y^-. This is a compact way to indicate that y should be closer to y^+ than to y^-.

When reading a tree, we can follow a particular branch to capture the child-parent relationship between the tree nodes (depth-first search tree traversal). We can also observe all the child nodes at each tree level before going down to the nodes of the lower level (breadth-first search tree traversal). We employ these two ways of reading a tree to exploit two different sets of constraints that encode the child-parent and the sibling relationships.

Supposing that a tree as in Figure 4.4a is available for the CIFAR10 dataset.

Each leaf node corresponds to one class in the dataset. The intermediate nodes are the abstract groups defined by users containing all the data points in their child groups. For example, the *land-vehicles* group has two children: the *automobile* and the *truck* groups, this group will thus contain all the images in the *car* and *truck* classes in the dataset. Let us take a car object as a running example to see what are the triplet constraints related to this object (in the form of (, positive example, negative example)).

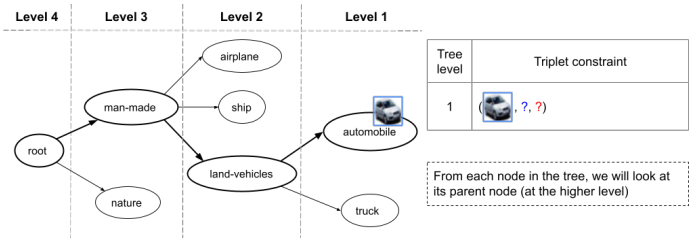
4.2.1 Child-Parent Relationship in Hierarchical Constraints

As mentioned before, we can extract information from a tree by walking through a particular branch. In this example, the car object belongs to the *automobile* group. We will start from this group at the lowest level of the tree and go up toward the root to collect the constraints that enhance the child-parent relationship. Table 4.1 illustrates step-by-step how the hierarchical constraint can be extracted from the tree.

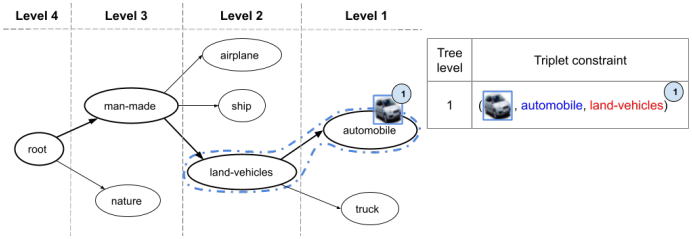
We only need to focus on the tree branch containing the *automobile* group shown in the preparation step in this table. Starting from the leaf node of *automobile* group, we go up one level at a time. At step 1, is now in the *automobile* group, which is a child of *land-vehicles* group. This should be closer to *every image* in the *automobile* group than to *any image* in the *land-vehicles* group. The reason is that the *land-vehicles* contains also other images of trucks, and should be only placed close to car images but not truck images. This information can be summarized by a triplet (, *automobile*, *land-vehicles*). Going up in the tree, at step 2, is now in the *land-vehicles* group, which is a child of the *man-made* group. A constraint (, *land-vehicles*, *man-made*) expresses that should be closer to every image in the *land-vehicles* group than to any image in the *man-made* group since in the *man-made* group also contains images of airplanes and ships. Similarly, at step 3, we can extract a constraint (, *man-made*, *root*), where *root* is the root node of the tree containing all images in the dataset. This last constraint indicates that should be closer to every image in the *man-made* group than to any image in the whole dataset because the dataset also has images of natural objects such as animals.

Table 4.1: Step-by-step example of extracting triplet constraints to reinforce child-parent relationships. The child and parent groups are annotated on the tree by closed curves. The corresponding triplets are shown on the right.

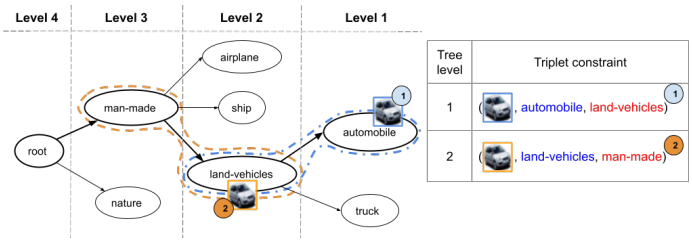
Step 0: Preparation for generating triplet constraints.



Step 1: is now in the *automobile* group, it should be closer to every image in this group rather than to any image of its parent *land-vehicles* group.



Step 2 is now in the *land-vehicles* group, it should be closer to every image in this group rather than to any image of its parent *man-made* group.



Step 3 is now in the *man-made* group, it should be closer to every image in this group rather than to any image in the entire dataset.

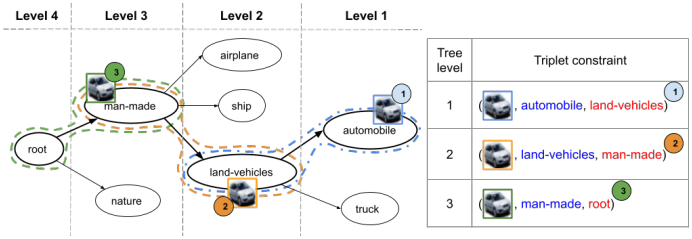
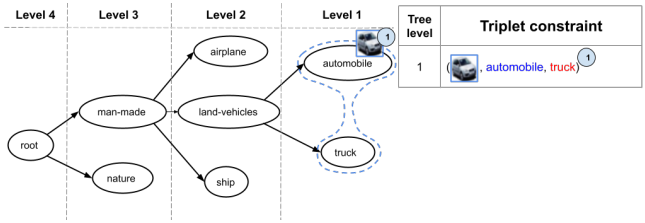
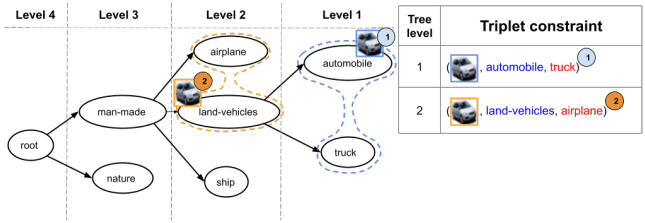


Table 4.2: Examples of triplet constraints to distinguish different child groups of the same parent group. Pairs of sibling groups at each level are annotated by closed curves. The corresponding triplets are shown on the right.

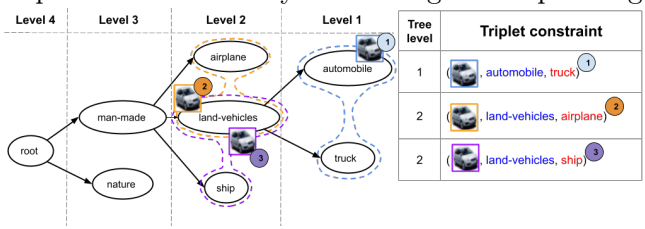
Step 1: is now in the *automobile* group, it should be closer to every image in this group rather than to any other image of its parent group.



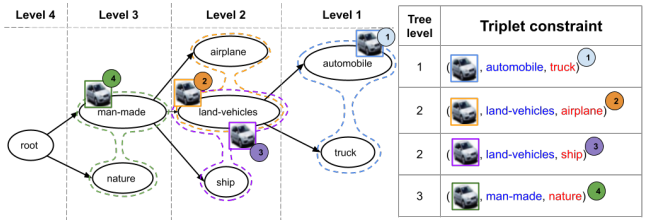
Step 2 is now in the *automobile* group, it should be closer to every image in this group rather than to any other image of its parent group.



Step 3 is now in the *automobile* group, it should be closer to every image in this group rather than to any other image of its parent group.



Step 4 is now in the *automobile* group, it should be closer to every image in this group rather than to any other image of its parent group.



4.2.2 Sibling Relationship in Hierarchical Constraints

Besides the information from the child-parent relationship, we can also extract the relationship between groups at the same level of abstraction in the tree. When a parent node has several child nodes, the information between sibling nodes can help to distinguish the groups represented by these sibling nodes. Using the same example of object, Table 4.2 summarizes the steps performed to extract hierarchical triplet constraints to enhance sibling relationships.

Similar to the process of extracting triplet constraints in the above example, at step 1, we start with in the *automobile* group at the lowest level in the tree. Since the *automobile* and *truck* are the only two children of the *land-vehicles* group, at this level we have only one sibling pair of these two child groups. The constraint (, *automobile*, *truck*) indicates that should be closer to every image in the *automobile* group than to any image in the *truck* group. Going up one level toward the root, at level 2, is now in the *land-vehicles* group. At this level, this group has two sibling pairs. In step 2 in Table 4.2, a triplet (, *land-vehicles*, *airplane*) is extracted to indicate that should be closer to every image in the *land-vehicles* group than to any image in the sibling *airplane* group. Similarly, in step 3, we extract a triplet (, *land-vehicles*, *airplane*). In the last step, when reaching level 3, we obtain a constraint (, *man-made*, *nature*) indicating that should be closer to every image of *man-made* objects than to any image of *nature* objects.

4.2.3 Usefulness of the Hierarchical Triplet Constraints

The above triplet constraints are extracted when processing in the *automobile* group. This process can be applied for every image in the *automobile* group. That will give us a set of triplet constraints that makes the *automobile* group be distinguished from other groups in the dataset. By traveling the tree at each level of abstraction (going from the lowest level to higher levels), the extracted triplet constraints naturally encode the hierarchical information. For instance, in the first example in Section 4.2.1, we make *automobile* distinct from other groups in *land-vehicles*, then make *land-vehicles* distinct from other groups in *man-made* and finally make *man-made* distinct from other groups in the entire dataset. The same effect has been shown when considering the sibling relationship in the second example in Section 4.2.2.

However, we can wonder why two different ways to extract the hierarchical constraints (called two **rules**) are needed when they give a similar effect of distinguishing groups? Are the sibling-enhanced constraints and the child-parent-enhanced constraints redundant? In fact, the effects of these two rules are the same only in the case where a node has exactly two children. In a general case where a node has multiple child nodes, the effects of these rules are different, depending on the number of data points in each group and the number of child groups in each parent group. More analysis based on the gradient when optimizing these rules is introduced in the next section.

For now, intuitively, the first rule for enhancing the child-parent relationship only allows us to consider the child node and its parent on the same branch of the tree. In contrast, the second rule considers all nodes at the same level of abstraction regardless of whether these nodes lie on the same or different tree branches. In the above example, we only show the sibling pairs in the *man-made* branch. However, with a specific tree traversal algorithm (called *Level-Order traversal*, detailed later), we can consider sibling nodes crossing branches to enrich the set of triplet constraints. Moreover, with the combination of the two rules, we do not need to enumerate all combinations of nodes in the tree to generate triplet constraints. Traveling through the tree in level-order as demonstrated in the above examples helps us to construct systematically a set of informative hierarchical triplet constraints.

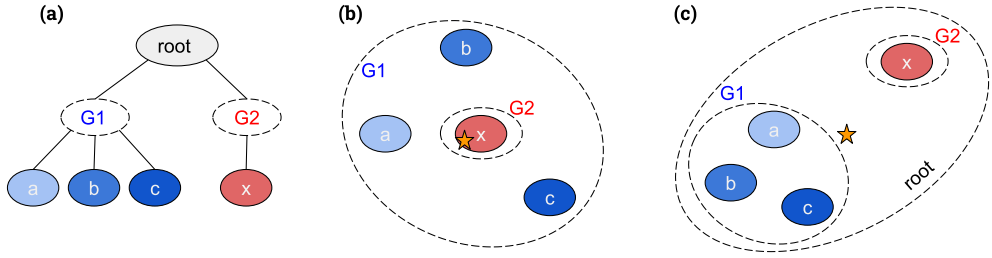


Figure 4.5: Explanation of why we need two rules for extracting the hierarchical triplet constraints from a simple tree in (a). The start symbol denotes the centroid of the whole dataset represented by the root node. If only the second rule is used for enhancing the sibling relationship between the red group (x) and the blue groups (a), (b), (c), we might end up with the result in (b). If the first rule is also used together with the second rule, we can obtain the result in (c) where the hierarchical structure is clearly revealed.

For more intuition, the schema in Figure 4.5 explains why both rules are needed. Supposing that we have a simple dataset with four groups in Figure 4.5(a): three blue groups belonging to the abstract group G1 and the red group belonging to the abstract group G2. Let us consider the target group (x), which has three sibling groups (a), (b), (c) at the same level. Only applying the second rule for enhancing the sibling relationship, we can end up with a result in Figure 4.5(b) where the target group (x) is placed at the center to make sure that it is distant from all three surrounding blue groups. If we combine with the first rule (applied for the abstract groups G1 and G2), we can make the target group (x) far away from all other child groups of the root node (the orange start symbol ★).

At this point, we have an initial idea of hierarchical triplet constraints extracted from an abstract tree. The obtained constraints reflect both semantic information and the hierarchical structure encoded in the data. It should also be noted that in the notation of the triplet constraint, the name of the group is used to represent all images in this group. A formal formulation for the constraints will be introduced in our algorithm presented in the next section.

4.3 HC*t*-SNE: Hierarchical Constraints with *t*-SNE

The proposed HC*t*-SNE method (Vu et al., 2021b) is a combination of a machine learning method with a tree traversal algorithm. HC*t*-SNE assumes that the user constraints are expressed in the form of a tree, which can be constructed based on class labels or user prior knowledge about the dataset. Previously, we have shown how the hierarchical tree is transformed into hierarchical triplet constraints through intuitive examples. This section formulates the optimization problem in *t*-SNE using these triplet constraints. We will first introduce the basic idea of quantifying triplet constraints with a **triplet loss**, a simple loss function that measures how well the points in the triplet are separated.

4.3.1 Triplet Loss with Margin

A triplet constraint $(\boldsymbol{y}_i, \boldsymbol{y}_i^+, \boldsymbol{y}_i^-)$ is said to be satisfied when the anchor point $\boldsymbol{y}_i$ and the positive point $\boldsymbol{y}_i^+$ are close to each other, while this anchor point and the negative point $\boldsymbol{y}_i^-$ are far apart. The notion of *close* or *far* is simply

measured by the traditional squared Euclidean distance between $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$, denoted as $d(\boldsymbol{y}_i, \boldsymbol{y}_j) = ||\boldsymbol{y}_i - \boldsymbol{y}_j||^2$. Our goal is to produce the embedding $\boldsymbol{Y}$ in such a way that the triplet constraints are satisfied as much as possible, i.e., $d(\boldsymbol{y}_i, \boldsymbol{y}_i^+) < d(\boldsymbol{y}_i, \boldsymbol{y}_i^-)\ \forall i$. In practice, in order to make the difference between these two distances more significant, we want the distance $d(\boldsymbol{y}_i, \boldsymbol{y}_i^+)$ to be much smaller than the distance $d(\boldsymbol{y}_i, \boldsymbol{y}_i^-)$, at least by a predefined *margin*. That means, we expect that $d(\boldsymbol{y}_i, \boldsymbol{y}_i^+) + \mathit{margin}$ is still $\leq$ than $d(\boldsymbol{y}_i, \boldsymbol{y}_i^-)$. This idea is illustrated in Figure 4.6.

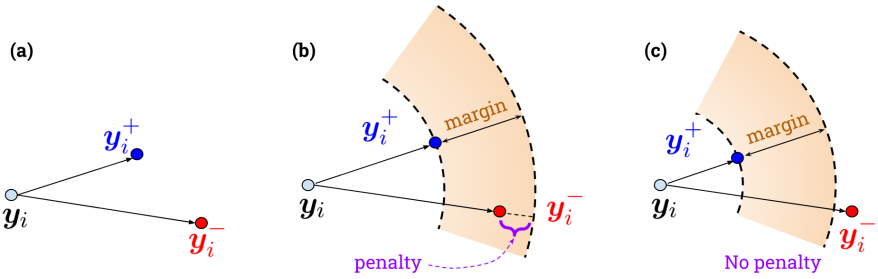


Figure 4.6: Illustration of triplet constraint with margin.

The triplet shown in Figure 4.6(a) is a valid triplet but is not a good one because the distance $d(\boldsymbol{y}_i, \boldsymbol{y}_i^-)$ is not much larger than the distance $d(\boldsymbol{y}_i, \boldsymbol{y}_i^+)$. Since the positive point $\boldsymbol{y}_i^+$ is not close enough to the anchor $\boldsymbol{y}_i$, if we consider a *margin* in Figure 4.6(b), the distance $d(\boldsymbol{y}_i, \boldsymbol{y}_i^+) + \mathit{margin}$ is now larger than $d(\boldsymbol{y}_i, \boldsymbol{y}_i^-)$. In other words, the triplet (a) is no longer a valid triplet under this margin. And thus, the triplet loss is defined as

$$\ell(\boldsymbol{y}_i, \boldsymbol{y}_i^+, \boldsymbol{y}_i^-) = d(\boldsymbol{y}_i, \boldsymbol{y}_i^+) + \mathit{margin} \ - \ d(\boldsymbol{y}_i, \boldsymbol{y}_i^-).$$

If the positive point is closer to the anchor as in Figure 4.6(c) (or the negative point is farther away), under the same margin, this new triplet is still a valid one and thus has no penalty. Therefore, the following general form for a triplet loss is used to *quantify* a triplet constraint under a predefined *margin*:

$$\ell(\boldsymbol{y}_i, \boldsymbol{y}_i^+, \boldsymbol{y}_i^-) = \left[d(\boldsymbol{y}_i, \boldsymbol{y}_i^+) - d(\boldsymbol{y}_i, \boldsymbol{y}_i^-) + \mathit{margin}\right]_+, \tag{4.1}$$

where $[x]_+ = max(0, x)$. Based on the triplet loss, we can manage the relative distance between groups using the centroids of groups as positive and negative points in a triplet. More details are given in the next section.

The *margin* hyperparameter can be tunable to control the separation between positive and negative examples. We can imagine that, when minimizing this loss for the whole set of triplet constraints, we can separate the positive and negative groups. The pattern in the visualization is thus heavily dependent on the quality of the input triplet constraints. In the previous section, we have introduced a specific way to generate informative triplet constraints that can capture semantic and hierarchical information. The integration of these triplet constraints into t-SNE is detailed in the following section.

4.3.2 Enforcing Hierarchical Constraints as a Regularization Term

The semantic information and hierarchical structure in the tree can be interpreted as the relationship between tree nodes (aka the groups or classes in the dataset). We call the relationship between nodes as *group-level* hierarchical constraints. The process of extracting triplet constraints from the tree is a transformation of group-level constraints into another form so-called *individual-level* hierarchical constraints. These triplet constraints are then represented by a differentiable regularization term in order to be optimized alongside the objective function of t-SNE.

Let us denote the embedding $\boldsymbol{Y} = \{\boldsymbol{y}_i\}_{i=1}^{N}$. To generalize the triplet introduced in the last section, let us denote a triplet related to the embedded point $\boldsymbol{y}_i$ as $(\boldsymbol{y}_i, G^+, G^-)$, where G^+, G^- denote the group of positive/negative examples for the anchor $\boldsymbol{y}_i$. This triplet indicates that $\boldsymbol{y}_i$ should be closer to every data point in G^+ than to any data point in G^-. In order to obtain a compact representation for the concept of *every data point* in group G_k, we use the centroid $\boldsymbol{c}_k$ (center of gravity) as a representative point for this group. Using this idea, a triplet now represents the relation between the anchor point and the two centroids of its positive and negative groups.

Let us denote a group and one of its siblings as G_k and $G_{k'}$. The centroids of G_k and its parent group are denoted as $\boldsymbol{c}_k$ and $\boldsymbol{p}_k$. We will redefine the rules used to generate the triplet constraints introduced in the last section.

Rule 1 for creating triplet that enhances child-parent relationship:

A point $\boldsymbol{y}_i \in G_k$ should be closer to the centroid $\boldsymbol{c}_k$ of its group than to the centroid $\boldsymbol{p}_k$ of its parent group.

This rule means that $\boldsymbol{y}_i$ should be close to points in G_k rather than being fused in the parent group. It thus prevents child groups from concentrating in the center of the parent group as explained in Figure 4.5. In this rule, only the points in the parent group (including all the points in G_k) are used, the focus is thus put on the intra-distances within the parent group. Applying the formulation of triplet loss in Equation 4.1 for all points in the group G_k according to this rule, we have the penalty (the loss) for the first kind of constraints as

$$\mathcal{L}_{intra} = \frac{1}{|G_k|} \sum_{\boldsymbol{y}_i \in G_k} \Big[d(\boldsymbol{y}_i, \boldsymbol{c}_k) - d(\boldsymbol{y}_i, \boldsymbol{p}_k) + \underbrace{m \cdot d(\boldsymbol{y}_i, \boldsymbol{p}_k)}_{margin} \Big]_+, \tag{4.2}$$

where $|G_k|$ is the number of instances in G_k and the distance function $d()$ is a squared Euclidean distance $d(\boldsymbol{y}_i, \boldsymbol{y}_j) = ||\boldsymbol{y}_i - \boldsymbol{y}_j||^2$. The $margin = m \cdot d(\boldsymbol{y}_i, \boldsymbol{p}_k)$ is not a chosen value as in Equation 4.1, but it is controlled by a positive hyperparameter $m \in [0, 1]$, called a *relative margin* instead. Since the position of points in the embedding is not normalized, the distance value is not normalized either. There is thus no evidence to choose a good margin value that works well for different datasets. We introduce a relative margin $m \in [0, 1]$ to facilitate the choice of margin. When $m = 0$, no margin is used. When $m = 1$, only the positive point $\boldsymbol{c}_k$ is used. Every point is now forced to move closer to the center of its group.

When the point $\boldsymbol{y}_i$ satisfies the related triplet constraint defined by Rule 1, no penalty is applied. Otherwise, if the point $\boldsymbol{y}_i$ is too close to $\boldsymbol{p}_k$, it violates Rule 1 and contributes to the loss $\mathcal{L}_{intra}$. In order to correct the violated point, we update its gradient as

$$\frac{\partial \mathcal{L}_{intra}}{\partial \boldsymbol{y}_i} = \frac{2}{|G_k|} \Big[(\boldsymbol{y}_i - \boldsymbol{c}_k) - (1 - m)(\boldsymbol{y}_i - \boldsymbol{p}_k) \Big]. \tag{4.3}$$

As consequence, the point $\boldsymbol{y}_i$ will be moved far away from $\boldsymbol{p}_k$. Figure 4.7

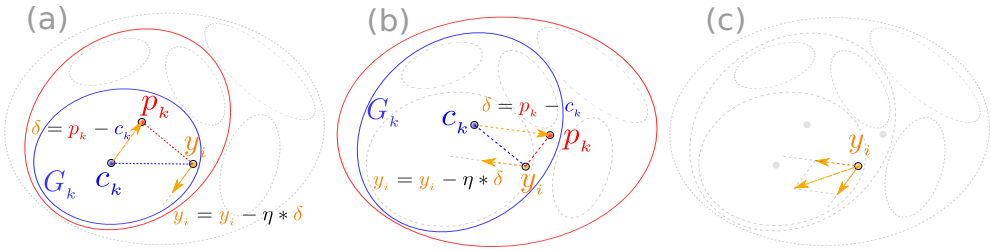


Figure 4.7: Illustration for the gradient of triplets in Rule 1 (Equation 4.2 - child-parent relationships) at two levels: (a) a leaf node and its parent and (b) one level higher. For the sake of simplicity and to avoid cluttering the figure, margin m is set to zero to get a simplified gradient $\frac{\partial \mathcal{L}_{intra}}{\partial \boldsymbol{y}_i} = 2(\boldsymbol{p}_k - \boldsymbol{c}_k)$. At each level, the instance to constrain $\boldsymbol{y}_i$ is compared to the centroid $\boldsymbol{c}_k$ of its own group and the centroid $\boldsymbol{p}_k$ of its parent group. The adaptation with a learning rate η is $-\eta\,\boldsymbol{\delta}$ where $\boldsymbol{\delta} = \boldsymbol{p}_k - \boldsymbol{c}_k$. The sum of the gradients for $\boldsymbol{y}_i$ at the two levels is shown in (c).

illustrates the gradient without margin ($m = 0$) for the sake of simplicity.

> Rule 2:
>
> A point $\boldsymbol{y}_i \in G_k$ should be closer to the centroid $\boldsymbol{c}_k$ of its group than to the centroid $\boldsymbol{c}_{k'}$ of its sibling group $G_{k'}$.

This rule helps to distinguish between different child groups in a larger parent group. This time, we focus on distances between child groups, which is called the *inter-distance* between groups. Applying the triplet loss for every point $\boldsymbol{y}_i$ in a group G_k, we have the following loss function

$$\mathcal{L}_{inter} = \frac{1}{|G_k|} \sum_{\boldsymbol{y}_i \in G_k} \Big[d(\boldsymbol{y}_i, \boldsymbol{c}_k) - d(\boldsymbol{y}_i, \boldsymbol{c}_{k'}) + \underbrace{m \cdot d(\boldsymbol{y}_i, \boldsymbol{c}_{k'})}_{margin} \Big]_+ . \tag{4.4}$$

The gradient for updating a point $\boldsymbol{y}_i$ if it violates the constraint is

$$\frac{\partial \mathcal{L}_{inter}}{\partial \boldsymbol{y}_i} = \frac{2}{|G_k|} \Big[(\boldsymbol{y}_i - \boldsymbol{c}_k) - (1 - m)(\boldsymbol{y}_i - \boldsymbol{c}_{k'}) \Big]. \tag{4.5}$$

Figure 4.8 illustrates the gradient of the point that violates Rule 2 in the case without margin.

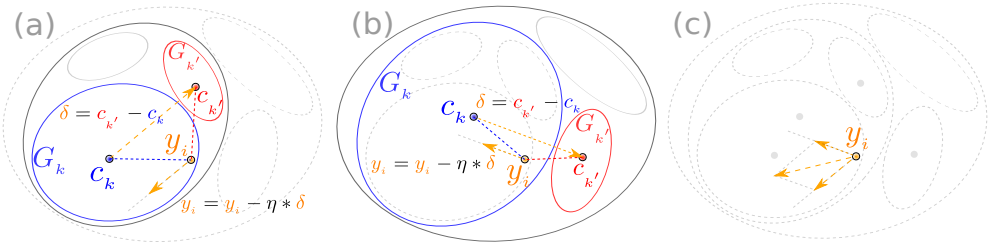


Figure 4.8: Illustration for the gradient of triplets in Rule 2 (Equation 4.4 - sibling relationships) with the same conventions as in Figure 4.7, except that $\boldsymbol{y}_i$ is compared to the centroid $\boldsymbol{c}_k$ of its own group and the centroid $\boldsymbol{c}_{k'}$ of its sibling group. Again, $m = 0$ simplifies the gradient to $\frac{\partial \mathcal{L}_{inter}}{\partial \boldsymbol{y}_i} = 2(\boldsymbol{c}_{k'} - \boldsymbol{c}_k)$. Here, there is only one sibling group, but in more complex cases, one has to sum the contributions of all siblings.

When observing closely at the direction of the gradients illustrated in Figures 4.7 and 4.8, we can notice that these two rules update the gradient for the point related to the violated triplets in a similar direction. We have claimed that these two rules are not redundant. If we look at the input hierarchy as a tree with different branches, we see that Rule 1 makes the groups in the same branch distinguishable, while Rule 2 can cause the same effect for *cross-branches* groups (i.e., groups in different branches). It should be noted that a point $\boldsymbol{y}_i$ can be related to a violated triplet constraint according to Rule 1, or Rule 2, or both of these two rules. If only one rule is used, we can miss useful relationship information encoded by the other rule. When both two rules are used, for example, in the simple case shown in Figures 4.7 and 4.8, these rules update the gradient consistently. The updated gradient for points related to violated triplet constraints depends on the nature of the related group (that the point belongs to) and the hierarchical structure related to that group. In practice, we show that using both these two rules can accelerate the convergence of the algorithm (see Section 4.5).

After defining these two above rules,we integrate the penalties defined in Equations 4.2 and 4.4 into the objective function of t-SNE. The additional penalties play a role as a regularization that modifies the position of the points related to the violated triplet constraints. This approach can take advantage of t-SNE to discover the group structure of the data in an unsupervised manner and enhance the global semantic structure in the embedding thanks to our regularization term. The most important advantage of this regularization term

is that it is differentiable. The gradient with respect to the point in the violated constraints is easy and efficient to calculate (Equations 4.3 and 4.5).

4.3.3 Optimization through a Tree Traversal Algorithm

Algorithm 1: The proposed HCt-SNE algorithm.

Input : High-dimensional data $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^{N}$,
Original t-SNE embedding $\boldsymbol{Y}_0 = \{\boldsymbol{y}_i\}_{i=1}^{N}$,
Hierarchical constraints in the form of a tree $\mathcal{T}$,
Weight for each rule $\omega_1 = 0.5$ and $\omega_2 = 0.5$,
Relative margin m, Learning rate η,
Coefficient of the regularization term α.

Output : HCt-SNE embedding $\boldsymbol{Y}$

1 Initialize $\boldsymbol{Y} = \boldsymbol{Y}_0$
2 **for** $iter \leftarrow 1$ **to** n_iter **do**
  /* Call the tree traversal routine */
3 $\mathcal{L}_{intra}, \mathcal{L}_{inter}, \frac{\partial \mathcal{L}_{intra}}{\partial \boldsymbol{Y}}, \frac{\partial \mathcal{L}_{inter}}{\partial \boldsymbol{Y}} =$ `LevelOrderTreeTraversal` (
4 $\boldsymbol{Y}$, // the current embedding, which changes after each iteration
5 $\mathcal{T}$, // the fixed input tree, which does not change in every iteration
6 ω_1, ω_2, m // parameters to weight and control the regularization
7)
  /* Update the new loss and gradients */
  /* α balances the original t-SNE loss and the new regularization */
8 $\mathcal{L} = KL_{loss} + \alpha\Big(\mathcal{L}_{intra} + \mathcal{L}_{inter}\Big)$
9 $\frac{\partial \mathcal{L}}{\partial \boldsymbol{Y}} = \frac{\partial KL_{loss}}{\partial \boldsymbol{Y}} + \alpha\Big(\frac{\partial \mathcal{L}_{intra}}{\partial \boldsymbol{Y}} + \frac{\partial \mathcal{L}_{inter}}{\partial \boldsymbol{Y}}\Big)$
  /* Update the current embedding by gradient descent. */
  /* Notice: in practice, momentum is used. */
10 $\boldsymbol{y} = \boldsymbol{y} - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{Y}}$
11 **end**
12 **return** $\boldsymbol{y}$

Algorithm 1 summarizes the proposed HCt-SNE algorithm. The novel idea lies in the construction of the penalty terms using the two proposed rules in Equations 4.2 and 4.4: these penalty terms are calculated at each level of abstraction in the tree. This step is named `LevelOrderTreeTraversal`

Algorithm 2: Level-order tree traversal routine to calculate the regularization and the corresponding gradients.

Input : Current embedding $\boldsymbol{Y} = \{\boldsymbol{y}_i\}_{i=1}^{N}$,
Hierarchical constraints in the form of a tree $\mathcal{T}$,
Weight for each rule $\omega_1 = 0.5$ and $\omega_2 = 0.5$,
Relative margin $m = 0.5$.

Output : Regularization terms $\mathcal{L}_{intra}, \mathcal{L}_{inter}$,
Corresponding gradients $\frac{\partial \mathcal{L}_{intra}}{\partial \boldsymbol{Y}}, \frac{\partial \mathcal{L}_{inter}}{\partial \boldsymbol{Y}}$

```
1  L_intra = L_inter = 0                                   // Two new loss terms
2  ∂L_intra/∂Y = ∂L_inter/∂Y = 0⃗                           // Gradient of each new loss
3  for level l ← 1 to height(T) do
       /* Optimize Rule 1                                                    */
4      foreach node G_k at level l do
5          c_k = G_k.centroid
6          p_k = G_k.parent.centroid
7          loss1 = OptimizeRule1(G_k, c_k, p_k, y, m)      // Loss from Equation 4.2
8          L_intra = L_intra + ω_1 loss1
9          ∂L_intra/∂Y = ∂L_intra/∂Y + ω_1 ∂loss1/∂Y        // Gradient from Equation 4.3
10     end
       /* Optimize Rule 2                                                    */
11     foreach sibling pair (G_k, G_k') at the same level l do
12         c_k = G_k.centroid
13         c_k' = G_k'.centroid
14         loss2 = OptimizeRule2(G_k, c_k, c_k', y, m)     // Loss from Equation 4.4
15         L_inter = L_inter + ω_2 loss2
16         ∂L_inter/∂Y = ∂L_inter/∂Y + ω_2 ∂loss2/∂Y        // Gradient from Equation 4.5
17     end
18 end
19 return L_intra, L_inter, ∂L_intra/∂Y, ∂L_inter/∂Y
```

subroutine and is highlighted in Algorithm 1. When the new regularization terms and their corresponding gradients are calculated in each iteration, they are combined with the original KL divergence objective function of t-SNE. The learning rate η can be chosen by a heuristic, such as the one suggested by Kobak and Berens (2019), which sets $\eta = N/12$. HCt-SNE can take an old

t-SNE visualization as an initialization for a quick convergence. In general, any random initialization or PCA initialization can also work.

The most important hyperparameter in our algorithm is the contribution of the hierarchical regularization α. However, tuning α for a good balance between the unsupervised objective of t-SNE and the supervised objective in the regularization is a difficult task. We have not defined a suitable metric to measure the compromise between these two objectives, and only measure the overall quality of the final visualization. A simple strategy to manually find an acceptable α is a trial-and-error approach of choosing an α that makes the new loss of HCt-SNE constantly decreased. We can also look at the value of the regularization term to make sure that the chosen α can make both the regularization and the new HCt-SNE loss decreased at the same time (as described in Appendix 9.4). Notice that in this work, we have to manually choose an acceptable α for each dataset. More work is needed to tune this hyperparameter automatically to achieve a good balance between the two unsupervised and supervised objectives.

It should also be noted that t-SNE is not designed to preserve distances between points. Its objective function minimizes the KL divergence between the probabilities (of being neighbors) and does not involve directly the distances. In opposite, the triplet loss in the proposed regularization (and its gradient) is based only on the distances between points in the LD space. Minimizing this triplet loss can also be interpreted as preserving distances between specific points (anchor points, positive and negative points) created by the input tree.

As shown before, the proposed rules enhance the semantic relationship between groups. And thus, when they are applied at different levels of the tree, they can enhance the hierarchical structure in the embedding. For that reason, we propose to visit the nodes of the tree in *level-order*, from the lowest level of leaf nodes to the highest level of the root node. At each level, the penalty terms according to each rule are constructed using the instances in the groups at that level and the corresponding parent groups (for Rule 1). Algorithm 2 details step-by-step how to accumulate the penalty terms at each level of the tree. In this algorithm, the two functions `OptimizeRule1` and `OptimizeRule2` apply Equations 4.2 and 4.4 for every point $\boldsymbol{y}_i$ in the group G_k. The order in which these two rules are applied does not matter since the loss and gradient of related points in the violated constraints will be accumulated at the end

of each iteration. The triplet constraints generated by each rule are weighted respectively by $\omega_1 = \omega_2 = 0.5$. These weights can be modified according to the need to focus on each specific rule.

In practice, our algorithm can be easily integrated into accelerated variants of t-SNE like Barnes-Hut t-SNE (van der Maaten, 2014a), FIT-t-SNE (Linderman et al., 2019) or other SNE-based methods like UMAP (McInnes et al., 2018a). The triplet loss has a fast gradient update since the centroids of each group remain fixed. The relative margin prevents us from manually tuning the margin value for each dataset. As shown in the following experiments, a relative margin $m = 0.5$ works well for all experimented datasets. This hyperparameter can also be tuned easily since its effect can be observed visually (more details come in the next section).

4.4 Experimental Results of HCt-SNE

In our experiments, two questions are addressed.

1. How are the global structure and the hierarchical information represented in HCt-SNE visualizations?

2. How do HCt-SNE embeddings compare with the ones of other methods?

HCt-SNE is compared with the original unsupervised t-SNE with Barnes-Hut acceleration (van der Maaten, 2014b) and two other supervised DR methods: UMAP in a supervised setting (McInnes et al., 2018a) and class-aware t-SNE (*cat*-SNE (de Bodt et al., 2019)). We first compare the visualization results of these methods qualitatively. A quantitative evaluation with several visualization quality scores is also performed.

4.4.1 Experimental Setup

We perform experiments on three standard image datasets MNIST (LeCun et al., 2010), Fashion-MNIST (Xiao et al., 2017) and CIFAR10 (Krizhevsky, 2009). Input pixels are first normalized in $[0, 1]$, and PCA is then applied to keep 95% of variance. Like t-SNE, HCt-SNE can work with other kinds of data. However, we use image datasets since users can create hierarchical constraints

visually by looking at the images. It should be noted that the hierarchical tree can be created using examples that are manually selected to form leaf nodes of the tree. Users can create more abstract groups by grouping similar leaf nodes according to their semantic meaning. *cat*-SNE and supervised UMAP use the available labels in the dataset. We also use the class information of each dataset in our experiments to construct the corresponding leaf nodes, each leaf node is one class. The intermediate nodes are created manually according to the desired hierarchical structure of users. For instance, in CIFAR10, *truck* and *automobile* are grouped into *land-vehicles*; this higher-level group is then grouped with *ship* and *airplane* to get *man-made*, etc.

In our experiments, HC*t*-SNE takes a *t*-SNE embedding as the initial state and does not apply the exaggeration phase. This setting allows us to compare directly a *t*-SNE visualization without constraints (initial state) and an HC*t*-SNE visualization with hierarchical constraints. In *t*-SNE, the exaggeration phase is important to form the global structure (Kobak and Berens, 2019). In contrast, HC*t*-SNE does not need this phase since the triplet constraints have already regularized the objective function to enhance the hierarchical structure.

Preliminary experiments are performed with (Barnes-Hut)*t*-SNE, UMAP, and *cat*-SNE to find the good hyperparameters for the experimented dataset. A perplexity of 50 is used for *t*-SNE, n_neighbors=10, and min_dist=0.1 are used for supervised UMAP. *cat*-SNE is used with its suggested setting with $\theta = 0.9$ to expand the neighborhood size in the HD space to capture at least 90% of data points with the same label. HC*t*-SNE uses the same hyperparameter values as *t*-SNE and has two additional hyperparameters. First, the relative margin m determines the separation of the groups in the visualization and can be set to 0.5 to make sure that the groups are neither too close nor too far away. Second, α determines the contribution of the hierarchical constraints to the new loss. α depends on the specific hierarchical tree of each dataset and can be easily tuned by observing the value of the regularization term ($\mathcal{L}_{intra} + \mathcal{L}_{inter}$), and then by trying several values to make sure this term decreases consistently. The reported results are calculated from the following values of α: 7.5×10^{-4} for MNIST and Fashion-MNIST, and 5×10^{-3} for CIFAR10. Our implementation is based on openTSNE (Poličar et al., 2019) with Barnes-Hut acceleration.

In order to quantitatively assess the visualizations, three different scores are used. The co-ranking-based score $AUC[R_{NX}]$ (Lee and Verleysen, 2009, 2010)

measures how well the neighborhood information in the HD space is preserved in the LD space. The KNN-gain score $AUC[G_{NN}]$ (de Bodt et al., 2019) measures how much we gain in terms of KNN accuracy when using the embedding in the LD space instead of the original data in the HD space. $AUC[R_{NX}]$ and $AUC[G_{NN}]$ are in the range of $[-1, 1]$, in which 1 is the best, -1 is the worst, and 0 means that there is no gain (or loss) in the neighborhood preservation of the KNN accuracy with the embedding in LD space. It should be noted that, for these scores, a small positive value is acceptable while a negative value is bad. These two metrics have a complexity of $\mathcal{O}\left(N^2 \log N\right)$, where N is the number of instances in the dataset. Because of this complexity, we use a subset of 10k data points for each dataset to facilitate the computation of these metrics. It also helps us to make a fair comparison with *cat*-SNE, since it is not optimized for very large datasets. Finally, the simple KNN score (with $K = 10$) suggested by (Belkina et al., 2019; Kobak and Berens, 2019) is used to measure how useful the 2D embedding is for a classification task.

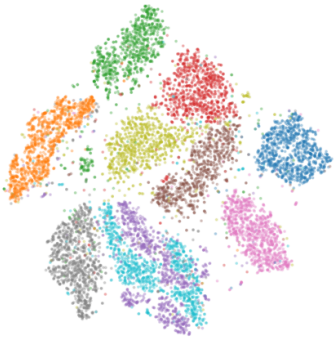
4.4.2 Analysis of Qualitative Results

Qualitative results for visual assessment for three datasets MNIST, Fashion-MNIST and CIFAR10 are shown in Tables 4.3, 4.4 and 4.5, respectively. The first four visualizations in each table show the results of the original *t*-SNE, *cat*-SNE, our method HC*t*-SNE, and supervised UMAP. The name of each class in the visualization is shown in the annotation. In the last row of each table, we show the input hierarchical tree and a proposed interpretation of the structures in HC*t*-SNE. The color of the leaf nodes in the tree and the groups in the visualizations are matched to help to identify them easily. In the last figure in each table, we manually annotated the abstract groups in the visualization result of HC*t*-SNE. Our goal is to show how the structures revealed by HC*t*-SNE match the input hierarchical tree.

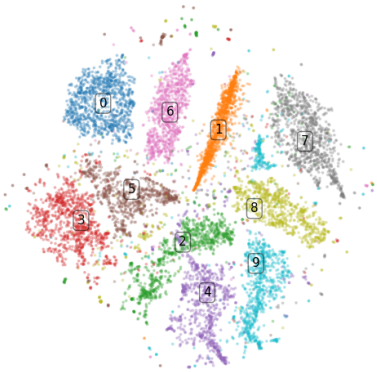
For MNIST, *t*-SNE, *cat*-SNE, and UMAP give good visualizations with clear separated groups. Since the supervised information is used for UMAP, we see the groups are more contracted. However, the position of the groups in the visualizations of these methods is arbitrary due to the random initialization. In contrast, HC*t*-SNE gives a stable result in which we can easily interpret the structure in its visualization as shown in the last figure of Table 4.3.

Table 4.3: Qualitative comparison of the embeddings for the MNIST dataset.

Unsupervised *t*-SNE



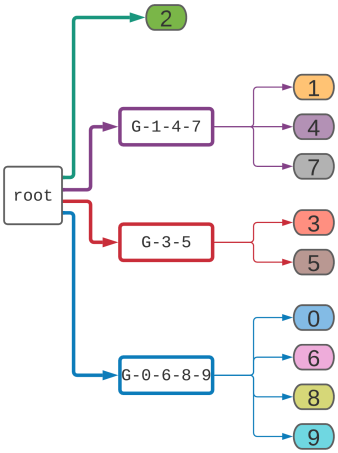
cat-SNE



HC*t*-SNE (our)

UMAP

Input hierarchy for HC*t*-SNE



Interpretation of structures in HC*t*-SNE

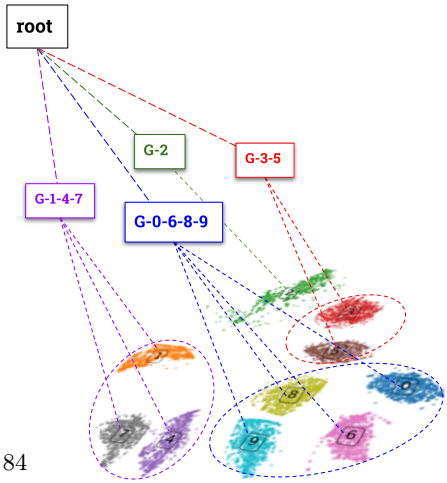


Table 4.4: Qualitative comparison of the embeddings for the Fashion-MNIST dataset.

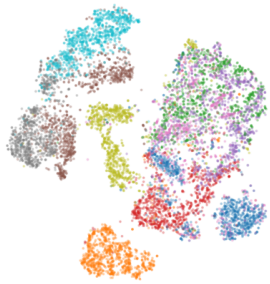
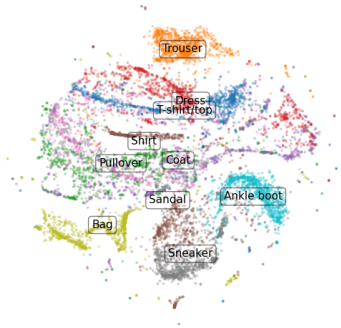
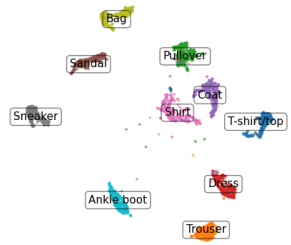
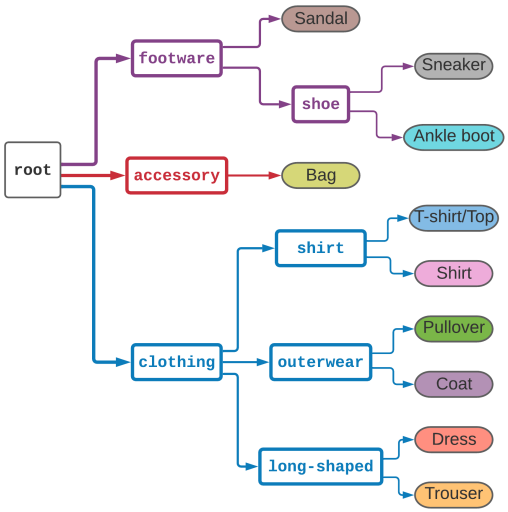
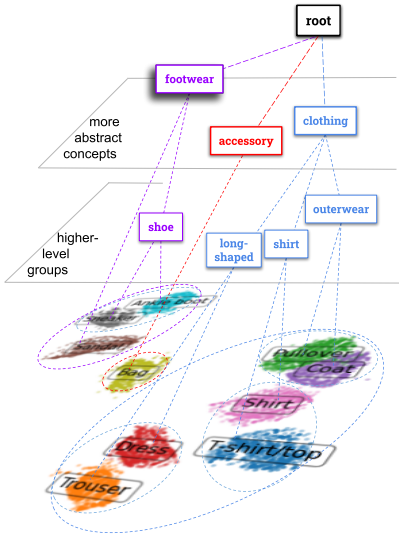
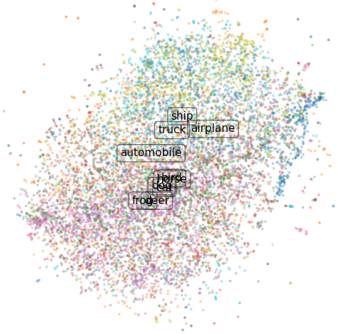
Unsupervised t-SNE	*cat*-SNE
	
HCt-SNE (our)	UMAP
	
Input hierarchy for HCt-SNE	Interpretation of structures in HCt-SNE
	

Table 4.5: Qualitative comparison of the embeddings for the CIFAR10 dataset.

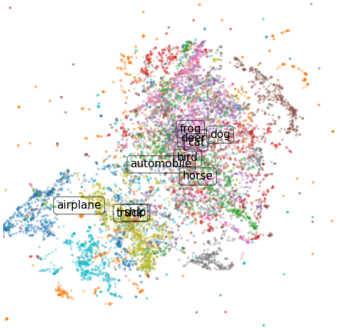
Unsupervised t-SNE

cat-SNE

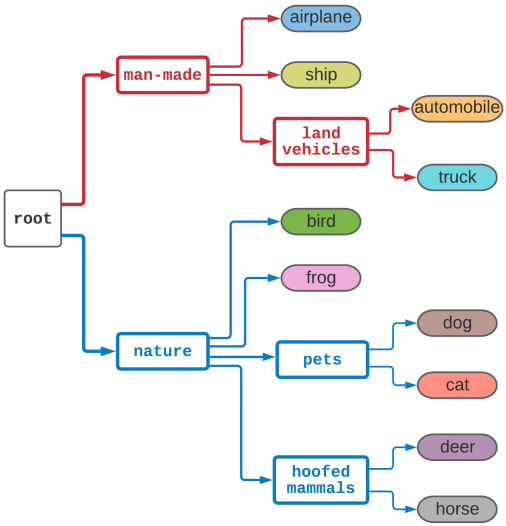


HC*t*-SNE (our)

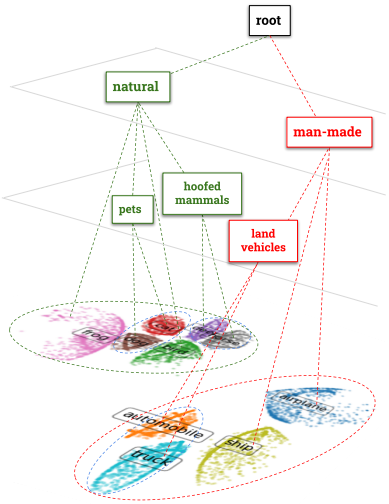
UMAP



Input hierarchy for HC*t*-SNE



Interpretation of structures in HC*t*-SNE



For Fashion-MNIST, the visualizations of t-SNE and *cat*-SNE in Table 4.4 are not very useful since there are always large overlaps between groups. UMAP gives a better result with well-separated groups. However, the same problem of global structure and relative distance between groups still exists in the UMAP result. For example, the *Sandal* - *Sneaker* groups are placed close to the *Bag* group but not the *Ankle boot* group. Similarly, the *Sandal* group is close to the *Pullover* - *Shirt* group. For this reason, it is not easy to reason about a large, abstract group of footwear containing sandals, shoes, and boots. HCt-SNE can solve this problem thanks to the guided input structure. The most important result of HCt-SNE is that it preserves the given input structure constraints.

Lastly, for CIFAR10, all competing methods (t-SNE, *cat*-SNE, and UMAP) fail to reveal distinct groups, as shown in the Table 4.5. With these methods, it is hard to reveal both the group structures and the global structure for hard datasets like CIFAR10. In contrast, HCt-SNE reveals separated groups and gives a global view corresponding to the input hierarchical tree. Even when the user creates several small semantic groups such as the *pets* group of *dog*s and *cat*s, the *hoofed mammals* group of *deer*s and *horse*s, and the *land-vehicles* group of *automobile*s and *truck*s, HCt-SNE can always reveal this semantic information by placing the related sub-groups close together.

Users can also control the separation of groups in the visualization of HCt-SNE with the relative margin hyperparameter. With different values of the relative margin $m \in [0, 1]$, HCt-SNE reveals the same global hierarchical structure but with different focuses on local group structures. With a large margin, we expect the high-level (abstract) groups are expected to be well separated as shown in Figure 4.9.

The qualitative results in this section serve as a comparison between HCt-SNE and other methods to show that HCt-SNE can create more useful visualizations, and to verify that the structure revealed by HCt-SNE matches the given input hierarchical tree. We have these advantages of HCt-SNE with a small additional computation (of linear complexity as discussed later). In order to evaluate the proposed method subjectively, we also assess the HCt-SNE visualization with quantitative metrics.

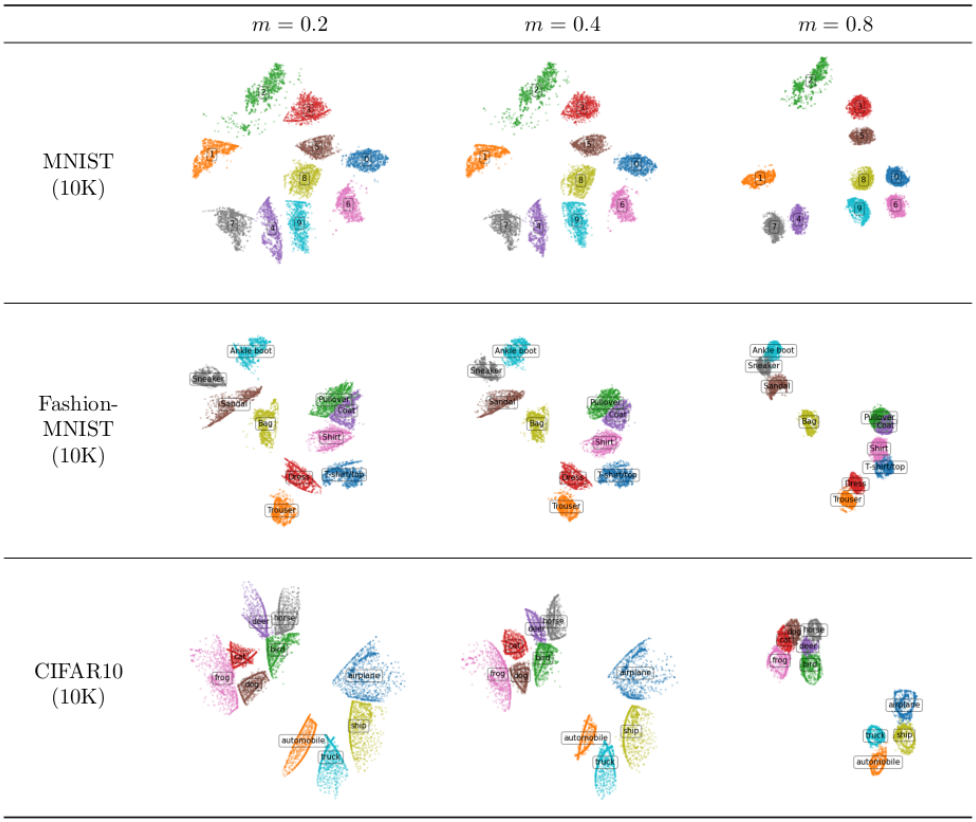
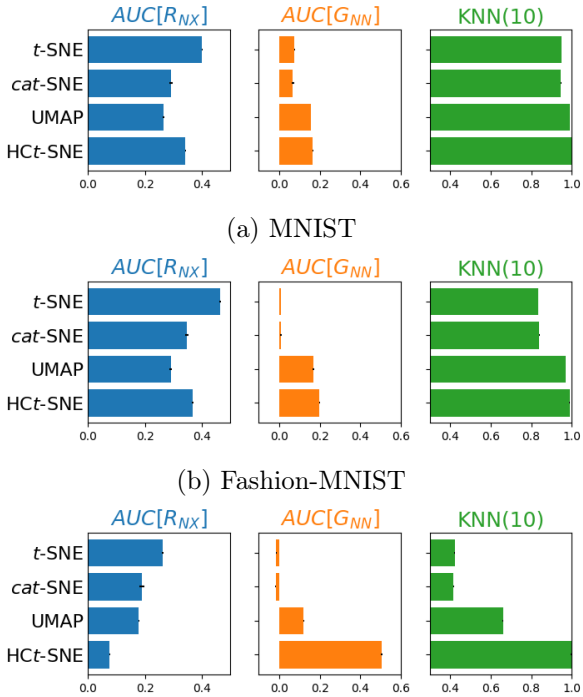


Figure 4.9: Effect of margin parameter.

4.4.3 Analysis of Quantitative Results

Figure 4.10 shows quantitative results in terms of three introduced metrics: $AUC[R_{NX}]$, $AUC[G_{NN}]$, and KNN score (see Section 4.4.1). Each method is run 10 times with different random initialization. The average score and the 95% confidence interval are shown. The hyperparameters and the input hierarchical tree for these methods are kept unchanged during different runs. Due to the very small variances in the results, we can conclude that the difference between different methods is significant.

In MNIST and Fashion-MNIST, gray-scale images have the same background, and thus there is no background effect. For both datasets, our method performs similarly to supervised UMAP and outperforms t-SNE and *cat*-SNE. On the



(a) MNIST

(b) Fashion-MNIST

(c) CIFAR10

Figure 4.10: Average scores of different metrics for three datasets. All methods are run 10 times with different random initialization. The scores are calculated for each resulting visualizations. The mean values are reported in the bar chart and 95% confidence intervals are shown in the black bars. It should be noted that the scores are stable, and confidence intervals are small, which shows that differences are significant.

contrary, CIFAR10 contains color images for which the background effect becomes a real problem, as presented in Figure 4.2. In a neighborhood, the images do not necessarily belong to the same class. Therefore, the neighborhood preserving score $AUC[R_{NX}]$ and the KNN-based score $AUC[G_{NN}]$ behave oppositely. More specifically, HCt-SNE has a low $AUC[R_{NX}]$ score since it does not preserve the neighborhood structure in the HD data. As we have shown that the neighborhood information calculated from pixel-wise Euclidean distances poorly reflects the semantic information in the class labels. HCt-SNE is designed to take into account the semantic information and thus cannot gain in terms of $AUC[R_{NX}]$ score since it breaks the neighborhood information in HD space. However, it outperforms t-SNE and *cat*-SNE by a large margin in

terms of $AUC[G_{NN}]$ and KNN scores since it has well exploited the class label information encoded in the hierarchical constraints.

4.5 Discussion

This section discusses several technical details in the design and analysis of HCt-SNE, its usage, its limitations, and perspectives to extend this method.

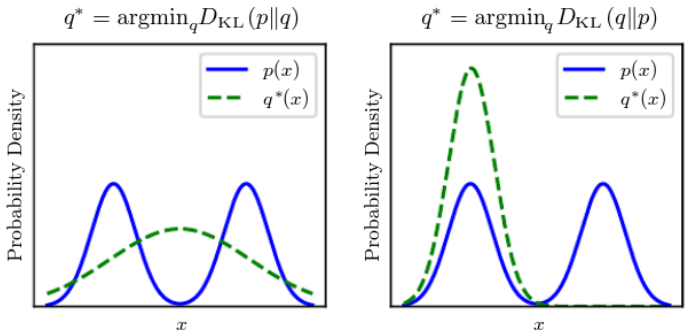


Figure 4.11: Illustration for two kinds of forward $KL[\boldsymbol{P} \mid \boldsymbol{Q}]$ (left) and backward $KL[\boldsymbol{Q} \mid \boldsymbol{P}]$ (right) KL divergence. This figure is taken from Figure 3.6 in the public lecture slides[1] of the Deep Learning book (Goodfellow et al., 2016).

We have shown how HCt-SNE works with the real data. We now discuss why our approach with HCt-SNE works and what is the theory behind HCt-SNE. For the hard (complex) datasets like CIFAR10, if we have better features such as the features extracted from a convolutional neural network, the t-SNE visualization can be improved. However, when the raw input pixels are used, a simple regularization-based approach like HCt-SNE can be useful. In fact, in t-SNE, groups that are semantically different can still be placed close together. The reason is that t-SNE focuses too much on the *attractive* force to make similar objects clumped together while less focus on *repulsive* force to push dissimilar objects far apart (Wattenberg et al., 2016).

t-SNE uses a forward KL divergence $KL[\boldsymbol{P} \mid \boldsymbol{Q}]$ from $\boldsymbol{Q}$ to $\boldsymbol{P}$, which has a *mean-seeking* behavior (Goodfellow et al., 2016, Sec. 3.13). Figure 4.11 shows an illustration of the behavior of the forward $KL[\boldsymbol{P} \mid \boldsymbol{Q}]$ and backward $KL[\boldsymbol{Q} \mid \boldsymbol{P}]$

[1] https://www.deeplearningbook.org/slides/03_prob.pdf

divergences. We focus on the $KL[\boldsymbol{P} \mid \boldsymbol{Q}]$ in t-SNE, which encourages putting mass of $\boldsymbol{Q}$ on whether $\boldsymbol{P}$ has some mass. That means, for every close pair (i, j) for which $\boldsymbol{P}$ has mass (since p_{ij} is non-zeros), t-SNE will try to make the corresponding pair as close as possible (by putting mass on q_{ij}). This behavior of the forward KL divergence distributes the mass of $\boldsymbol{Q}$ widely without focusing on any mode in the distribution of original data as illustrated in Figure 4.11. t-SNE may thus fail in the cases where we have multiple modes and large overlapping of different classes/groups in the distribution of the HD data. Combining the two kinds of divergence (Venna et al., 2010) or using the generalized Jensen–Shannon divergence (Lee et al., 2013, 2015) can overcome this problem and better preserve small neighborhoods (corresponding to small perplexity values). HCt-SNE introduces another way to tackle this problem using regularization terms to modify the behavior of the KL loss. This new loss (based on the triplet loss with relative margin) enhances the repulsive force that makes the negative examples far away from the anchor point. Better divergence-based objective function suggested by Lee et al. (2013) may have more effect in discovering multiple modes in the HD data than a simple regularization-based approach. Our proposed regularization term has a computation advantage since it is cheap to compute (both the loss and its gradient) and can be interpreted as a secondary distance-preserving objective for t-SNE.

In theory, the triplet loss can be considered as an energy function that minimizes the *compatible settings* (i.e., the anchor and the positive point) and maximizes the *incompatible settings* (i.e., the anchor and the negative point) (LeCun et al., 2007). This is the same idea as metric learning methods, which try to learn a distance function to make similar points close together and dissimilar points far apart (Kulis et al., 2012). We can thus extend HCt-SNE by replacing the triplet loss with another contrastive loss such as the following one introduced in (Arora et al., 2019):

$$\mathcal{L} = -\mathbb{E}_{\boldsymbol{X}} \left[\log \frac{\exp\left(f_\theta(\boldsymbol{x})^T f_\theta(\boldsymbol{x}^+)\right)}{\exp\left(f_\theta(\boldsymbol{x})^T f_\theta(\boldsymbol{x}^+)\right) + \sum_{j=1}^{N-1} \exp\left(f_\theta(\boldsymbol{x})^T f_\theta(\boldsymbol{x}_j^-)\right)} \right], \quad (4.6)$$

where $f_\theta(\boldsymbol{x})$ is a representation function. The triplet constraints are represented by a regularization term that preserves global and hierarchical structures, and thus could also be integrated into other DR methods like UMAP.

While we can obtain a visualization with global hierarchical structures, we only have to pay a small additional computation cost for the regularization term, which depends on the number of triplet constraints. The number of triplets in the worst case is $\mathcal{O}(KHN) \ll \mathcal{O}(N^3)$, where K is the number of pairs of sibling nodes and H is the height of the tree. In a dataset of C classes, the number of triplet generated by using a naive combination is $N \times \frac{N}{C} \times \frac{(C-1)N}{C} = \mathcal{O}(N^3)$. In this worst-case where all the leaf nodes have the same depth of H, the number of triplets between child nodes and parent nodes is $\mathcal{O}(HN)$, the number of triplets between sibling nodes is $\mathcal{O}(KHN)$. In total, the number of triplets in the worst case is $\mathcal{O}(HN) + \mathcal{O}(KHN) = \mathcal{O}(KHN)$.

However, our approach has some limitations. When addressing the problem of global structure in the visualization, we should consider the shape of the groups and the relative distances between them. HC*t*-SNE currently tackles the second aspect while ignoring the first one. Revealing and preserving the shape of groups with neighborhood embedding methods is still an open problem. Also, in our experiments, we did not consider how to embed new data points. Another potential issue with HC*t*-SNE is that the quality of the embedding depends on the quality of the input hierarchical tree. When the tree is not well-structured, or the leaf nodes do not contain enough data points, the proposed regularization may not work. The reason is that the regularization term is in fact a triplet loss, which only uses one positive and one negative example. Although this loss is constructed at different abstract levels of the tree, we have not exploited all the available *negative pairs*. As pointed out in recent works on contrastive learning, negative pairs between samples have a large impact on the quality of the representation (Arora et al., 2019; Schroff et al., 2015; Sohn, 2016). Using a better contrastive loss such as an *N-pair loss* (Sohn, 2016) can help to increase the number of triplets when the input tree does not have enough data points. Lastly, HC*t*-SNE, like *t*-SNE, is more suitable for visualization but not for general dimensionality reduction tasks. Even though the experimental results show a promising KNN-based score with our method, we still cannot conclude that HC*t*-SNE can be used for extracting features for subsequent tasks of classification or clustering. We need to do more experiments and comparisons with a baseline hierarchical classification method. However, this is not in the scope of this thesis that focuses only on constraint integration.

From a usage aspect, HC*t*-SNE is an accessible method for end-users who want to easily explore datasets with the support of a hierarchical representation of constraints for groups of instances. For example, this can be useful when clusters are not clearly separated or even overlap in the visualization computed by *t*-SNE. It may happen when the distance metric in the HD space is not satisfactory or when the user does not have an appropriate neural network for feature extraction. In this case, if the labels are available, a simple tree with one root node and all the class labels as leaf nodes can also help to separate the groups in the visualization without requiring complex hierarchical information. Even if the users do not have any idea about the hierarchical structure of their data, they can use a hierarchical clustering method like HDBSCAN (Campello et al., 2013) to obtain a hierarchical tree to pass to HC*t*-SNE. The additional semantic information in the hierarchical constraints can come from external sources or the data themselves. Therefore, HC*t*-SNE can be used in a purely unsupervised two-stage algorithm as follows. First, a hierarchical clustering method is used to group the data into small clusters, while the hierarchical tree is extracted from the dendrogram of the clustering result. Second, HC*t*-SNE is used to visualize the discovered clusters. This combination gives us a new usage of HC*t*-SNE that is more accessible for end-users as they do not need to provide any hierarchical constraints. This information can be obtained automatically in an unsupervised manner thanks to (hierarchical) clustering methods.

Our future work will focus on the constraints expressed by users for DR methods. While *t*-SNE and UMAP are widely used, their results lack global structure preservation, and users have no means to inject their knowledge into the visualization. We plan to work on interactive ways to visually build the hierarchical tree by, e.g., selecting sample images to create nodes. This tree will be passed to HC*t*-SNE to create a meaningful and useful visualization. Besides, we will also tackle the limitations of our method to expand its usage. For example, the constraints learned from the training set could also help to project new points correctly into their groups.

Chapter 5

Integration of Fixed-position Constraints into Probabilistic DR Methods

This chapter presents a unified framework to inject users' prior knowledge about the position of points directly into the prior distribution of probabilistic DR models such as PPCA or PMDS.

Contents

5.1 Probabilistic Approach in ML and in DR 98
5.2 Proposed Unified Probabilistic DR Framework with Constraints 101
5.3 Concrete Example 1: PMDS Model and interactive PMDS . . . 107
5.4 Concrete Example 2: PPCA Model and interactive PPCA . . . 120
5.5 Experimental Results 127
5.6 Discussion 134

This chapter is based on our two publications: "iPMDS: Interactive Probabilistic Multidimensional Scaling" (Vu, Bibal, and Frénay, 2021c) and "iPPCA: User-steering Interpretable Visualization with Probabilistic Principal Components Analysis" (Vu and Frénay, 2019).

In the previous chapter, we introduced a t-SNE-based method with hierarchical constraints. That kind of constraint can be constructed from class labels of the dataset with human annotation to form a tree structure. This tree can be constructed beforehand since it requires only the human general knowledge about the semantic groups in the dataset. In this chapter, we work with another type of constraint, that is more interactive and user-centric.

When users observe the visualization result of a DR method, they can find that the visualization is not what they expected. Figure 5.1 shows an example of the visualization produced by MDS. In this example, the traditional MDS method takes the pairwise distances between ten cities in the US (shown in the heatmap on the top left) and returns a 2D visualization that represents the positions of these cities in a 2D map. The US map shown in the background of the plot on the bottom left is for reference. Besides the mismatch between the positions produced by MDS and the true position in the reference map, the biggest problem of MDS is the orientation of the visualization. This is a simple case where the user's knowledge can be used to correct the error of DR methods. An end-user can know in advance (or can look at the reference map and find out) that, *Olympia* is at the north of the West coast and *Washington D.C.* is at the East coast of the US. He/she can give feedback to the algorithm by fixing the position of two points corresponding to these two cities on the 2D map. This feedback is considered as a kind of *constraint on the fixed position of points of interest indicated by users*. Our proposed framework can integrate this constraint into a probabilistic DR method (such as probabilistic MDS or probabilistic PPCA), and produce a more reasonable visualization like the one shown in the bottom right of Figure 5.1.

The user's feedback on the position of points is used frequently in interactive visual analytic (Sacha et al., 2017b). In the scope of this thesis, this kind of feedback is considered as *fixed-position constraints* to be integrated into DR methods. These constraints can be considered as prior knowledge of users on the position of several particular points. This viewpoint gives rise to our choice of the probabilistic approach that connects the users' prior knowledge to the prior distribution of a probabilistic model. Section 5.1 gives a brief introduction to the probabilistic approach in machine learning and in dimensionality reduction. Based on this foundation of probabilistic modeling, in Section 5.2, we introduce our *unified probabilistic framework* for integrating fixed-position constraints

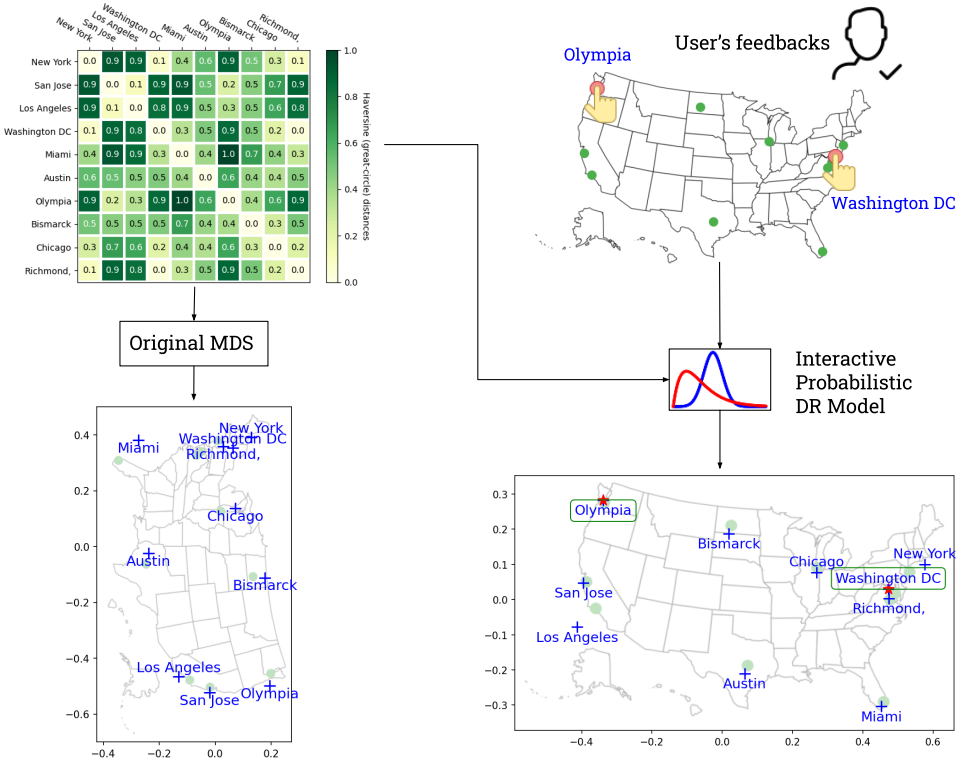


Figure 5.1: Overview of the proposed interactive probabilistic DR model (iPDR).

into several well-known probabilistic DR methods. A concrete example of this framework applied to a distance model for solving the MDS problem is called iPMDS (Vu et al., 2021c), which is introduced in Section 5.3. Similarly, applying this framework for PPCA, we introduce an interactive, user-steering model called iPPCA (Vu and Frénay, 2019) in Section 5.4. Experimental results with the proposed methods through various interactive case studies are presented in Section 5.5 (and additional results in Section 9.1). Finally, Section 5.6 summarizes and discusses the pros and cons of our approach, as well as perspective for interactive probabilistic DR models.

5.1 Probabilistic Approach in ML and in DR

Before diving into the integration of constraints into probabilistic DR methods, we first introduce the probabilistic approach in machine learning and dimensionality reduction. This brief introduction helps us to clarify why we choose the probabilistic approach. When understanding the probabilistic modeling process, it will be more natural to introduce our idea for constraint integration in probabilistic DR models.

5.1.1 Brief Introduction to a Probabilistic Approach in ML

Machine learning *methods* (that can also be called *techniques* or *algorithms*) often define sequence of steps to solve a specific problem. For example, in order to find the principal components that maximize the variances of the data, we do 2 steps: (1) constructing a covariance matrix from the centered data, and then (2) finding the eigenvectors corresponding to the top largest eigenvalues of this covariance matrix. This process is one way to formulate PCA, where these eigenvectors are called the principal components. However, the relationship between the projected data on the top eigenvectors and the criteria of maximizing the data variance is not always obvious. In general, these steps only tell us what to do in order to solve the problem, but do not tell us why. Machine learning partitioners often have to learn a bunch of well-studied methods for solving a wide range of problems. When tackling a new problem, their focus is on mapping their problem to one of these existing methods, i.e., choosing the right algorithm.

A *probabilistic approach* in machine learning opens another perspective to look at the traditional machine learning problems. Under this approach, the *unknown quantities* such as the values in the projection matrix of a linear DR method, or the unknown position of points in an LD space of a nonlinear DR method are represented as random variables drawn from parameterized probabilistic distributions (Murphy, 2021). Many well-known machine learning methods can be considered as probabilistic models (Khan and Rue, 2021). We can always find (or formulate) the corresponding probabilistic model for a wide range of machine methods, from simple logistic regression model (Friedman et al., 2001, Chap. 3) to complex deep neural network (Wilson and Izmailov,

2020). Standard unsupervised DR methods like PCA and MDS both have the corresponding probabilistic versions, which are the center topic of this chapter.

We will now put more focus on describing the problem rather than finding or designing a method to solve the problem. In general, explicitly the domain knowledge can be used to construct a probabilistic model that explains the observed data (Bishop, 2013). A model is made up of a set of assumptions, expressed in a precise mathematical form (Winn and Bishop, 2018). Probabilistic theory gives us a formal language to describe the relationship between the unknown variables we need to find and the observed variables we can observe through the input data. The mapping from hidden variables to the observed quantities is described through a *generative process*. By carefully choosing the variables and their corresponding probabilistic distributions, as well as defining an appropriate generative process, and quantifying how the real data match the defined model, we can understand how the observed data are generated. Thanks to the ability to fully control the model and the generative process, we can integrate the users' constraints into the model to discover how the constraints affect the output of the model.

A model is often parameterized by the parameters that characterize the probabilistic distributions of the variables. Once having a defined model, the next step is to measure how well this model describes the observed data, which is quantified by a *likelihood function*. The likelihood function is called the *likelihood of the parameters given the data* (MacKay, 2003). The name of this function suggests that it measures how likely the parameterized model reflects the observed data. Naturally, we expect to find a set of model parameters that maximizes this likelihood. This is the basic idea of the *maximum likelihood estimation (MLE)*. Until now, we see two important steps in probabilistic modeling. First, we have to posit a probabilistic model using the language of random variables drawn. We then try to infer the parameters of these underlying probabilistic distributions. The inference step is often done by using MLE, MAP (maximum a posteriori), or other advanced methods like variational inference (VI) or Markov chain Monte Carlo (MCMC).

5.1.2 Probabilistic DR Methods

Various basic DR methods like PCA and MDS have corresponding probabilistic versions. For example, probabilistic PCA (PPCA) (Bishop, 1999; Tipping and Bishop, 1999) is a typical example of *latent variable models* that formulates a generative process for mapping a *latent point* in an LD space to a point in an HD space. The *inverse mapping* of this model can be used to *infer* the latent position given the observed data in the HD space. That is exactly the problem of PCA, however, rather than focusing on the construction of the projection matrix, we focus on how the data are generated through a noisy linear mapping. More details of PPCA and our extended interactive version of PPCA will be presented in Section 5.4.1. Another example, probabilistic MDS (PMDS) (MacKay, 1983; MacKay and Zinnes, 1986; Zinnes and Griggs, 1974; Zinnes and MacKay, 1983) solves the same distance-preserving problem of MDS while producing an LD configuration of the HD data. However, this model does not measure the distortion between the pairwise distances in the LD and HD space but tries to model how the pairwise distances are generated. The PMDS model reformulates the distance preservation problem as an inference problem. It learns the latent position of points in such a way that the observed input distances look likely to be generated from the model. Details about this model and our proposed interactive extension will be presented in Section 5.3.2.

The PPCA and PMDS models are two examples of basic latent variable models, which focus on the representation of the unknown position of points in an LD space. More complex probabilistic models designed for dimensionality reduction tasks are also introduced. In opposite of PPCA, a linear latent variable model, *generative topographical mapping* (GTM - (Bishop et al., 1998)) is a nonlinear latent variable model used mainly for visualization tasks. The nonlinear mapping (from the data space to the latent space) is obtained through a set of basic functions that are similar to the kernels in the kernel methods (Hofmann et al., 2008). This model defines an explicit density of the data by a *constrained* mixture of Gaussian (Hinton et al., 1991), in which each Gaussian center is related to the position of latent points via the nonlinear mapping function. Another well-known nonlinear probabilistic model is the *Gaussian process latent variable model* (GP-LVM - (Lawrence, 2005)). This model is strongly related to not only nonlinear probabilistic models (GTM, PPCA, and factor analysis) but also classical kernel PCA and MDS methods.

As in PCA, we need to define a linear mapping function (which is corrupted by noise in PPCA) from the LD space (latent space) to HD space (data space). In GP-LVM, this mapping function is not restricted to be linear and can be learned automatically. Gaussian process arises as a natural solution since it is a specific kind of probabilistic model that defines distributions over function spaces (Rasmussen, 2003, Chap. 1, 2). Instead of defining/choosing the basic functions (kernels) as in GTM, GP-LVM can learn both linear and nonlinear kernels that produce a flexible mapping function.

In the scope of this thesis, we do not focus on the over-complex models like GTM or GP-LVM, but we aim to model both the data and the constraints given by users by a simple and intuitive model. For that reason, we develop a general framework that allows integrating users' constraints directly into any probabilistic DR models. We demonstrate the application of this framework by applying it to the original PPCA and PMDS model to derive two new interactive methods, which are detailed in the next section.

5.2 Proposed Unified Probabilistic DR Framework with Constraints

The main idea of our proposed framework for integrating fixed-position constraints into any probabilistic DR method is based on a concept of the *informative prior*. Basically, the prior knowledge of users that are represented in the form of fixed-position constraints is encoded into the prior distributions of the probabilistic model. Therefore, the constraints can be integrated through by the choice of the prior distributions and the way we modify these priors to encode the users' feedback. Before going into details of the constraint integration, let us break down the important components of a probabilistic model.

5.2.1 General Workflow for Probabilistic Modeling

A general workflow for probabilistic modeling is summarized in Figure 5.2. Mathematical notations corresponding to the quantities that appear in each step in this workflow are shown on the right.

The first step is to analyze the problem in order to describe formally the

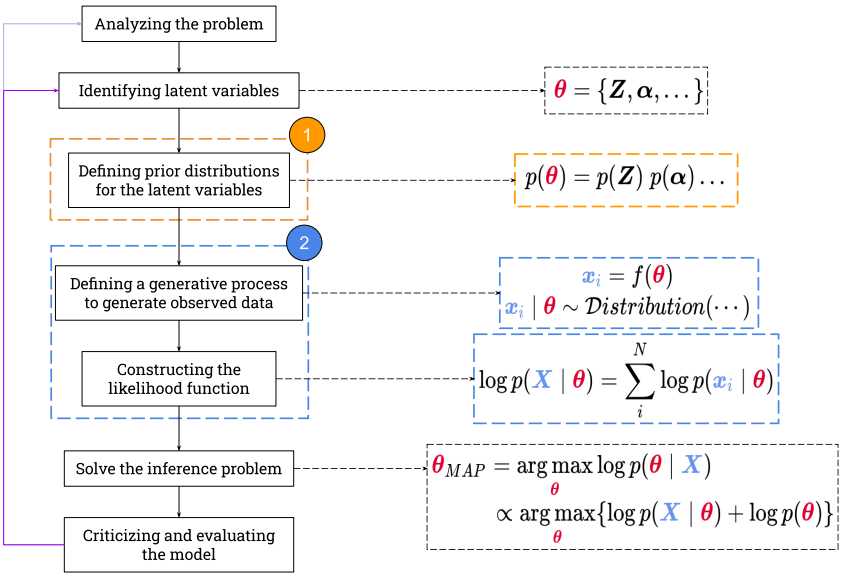


Figure 5.2: The general workflow for probabilistic modeling and integrating fixed-position constraints. The corresponding mathematical notations for the quantities aprear in each step are summarized on the right. Our contribution for the constraint integration lies on the block (1), which is detailed in Section 5.2.2. The block (2) involves in defining a generative process for each particular model, which is explained through two derived methods in Section 5.3 and Section 5.4, respectively.

problem in a probabilistic language (using the notations of probabilities and distributions). In the context of a dimensionality reduction problem, under the viewpoint of a probabilistic approach, we consider the desired embedding consisting of unknown points represented by *latent variables* in an LD latent space. The input data are represented by *observed variables* in an HD space.

The second step is to identify all the unknown quantities, i.e., all the latent variables. For example, a variable that represents the position of a point in the LD latent space is called the latent position variable. Besides, we can have other latent variables that control the distributions of other quantities in the model. All these latent variables, including the latent position variables are denoted as $\boldsymbol{\theta}$.

The third step is to define prior distributions for the latent variables. These prior distributions describe our assumptions about the value of each variable. For example, the latent position variable can be modeled by a multivariate

standard normal distribution. Since the position of a point can be represented by a vector of real values, using a normal distribution to represent real values is a reasonable choice. Indeed, the statistical data show that the IQ scores, the blood pressure, the height of (a subset of) the population, (which are all real values) follow the normal distribution (Spiegelhalter, 2019). Our contribution is to integrate the fixed-position constraints into these prior distributions, which will be detailed in the next section.

The fourth step is to define a generative process to map the point from an LD latent space to an HD space. Simply speaking, we want to represent a point $\boldsymbol{x}_i$ in an HD space as a function $f(\cdot)$ of the corresponding latent position variable and other latent variables in the model. This function can be a linear mapping corrupted by noises like in the PPCA model or a complex nonlinear mapping like in the PMDS model. By manipulating the probability distributions of different variables involved in the generative process, we can find out the conditional probability of the observed variable given the latent variables $p(\boldsymbol{x}_i \mid \boldsymbol{\theta})$.

The fifth step comes naturally when we factorize the model of the whole dataset into the likelihood function involving each individual data point as $\log p(\boldsymbol{X} \mid \boldsymbol{\theta}) = \sum_{i}^{N} \log p(\boldsymbol{x}_i \mid \boldsymbol{\theta})$. The likelihood function is indeed a function of all unknown quantities that tell us how likely the observed data can be generated given particular sampled values for the unknown quantities. For that reason, if we maximize this likelihood, we can find an estimation for the set of latent variables that explain the observed data the best.

However, MLE does not use the prior distributions. In the sixth step, the unknow quantities are estimated using the maximum a posteriori (MAP) algorithm. The set of latent variables is inferred as

$$\begin{aligned} \boldsymbol{\theta}_{MAP} &= \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta} \mid \boldsymbol{X}) \\ &\propto \arg\max_{\boldsymbol{\theta}} \{\log p(\boldsymbol{X} \mid \boldsymbol{\theta}) + \log p(\boldsymbol{\theta})\}. \end{aligned} \tag{5.1}$$

The log-prior appears in the above equation indicating that we take into account the prior knowledge when defining the model. This quantity also encodes the users' constraints and thus can be used to regularize the model to respect the input constraints. The optimization problem in Equation 5.1 can be solved

efficiently by a gradient-based algorithm like the standard gradient descent algorithm or Adam algorithm (Kingma and Ba, 2015). It should be noticed that the complexity[1] of the gradient-based algorithm depends on the number of parameters. For the specific latent variable models in this chapter, the number of parameters in vector $\boldsymbol{\theta}$ is proportional to the number of data points. Therefore, the complexity of the gradient update is $\mathcal{O}(N)$. Let us assume the complexity for calculating the log posterior $\log p(\boldsymbol{\theta} \mid \boldsymbol{X})$ and its gradient is $\mathcal{O}(f(N)) > \mathcal{O}(N)$. The update loop of a gradient descent procedure with T iterations is summarized in the following pseduo-code. The complexity of the

```
for iter ← 1 to T do
    Calculate the log posterior log p(θ | X).                    // Complexity O(f(N))
    Calculate the gradient ∇_θ log p(θ | X).                     // Complexity O(f(N))
    Update the model parameters with a learning rate η:
      θ = θ − η∇_θ log p(θ | X).                                 // Complexity O(N)
end
```

whole process is $T\left(\mathcal{O}(f(N)) + \mathcal{O}(f(N)) + \mathcal{O}(N)\right) = \mathcal{O}(Tf(N))$. We keep the factor T since in practice, we often need hundreds of iterations to converge to the optimal solution, which is a considerable number in the case of a medium-sized dataset. In the following section, we will analyze the complexity $\mathcal{O}(f(N))$ of the log posterior of each concrete model.

In our framework, we propose to use this optimization-based approach to solve the inference problem. In fact, the inference step in a probabilistic model can be tied to the modeling step. For example, the prior distributions can be chosen to be in the same family with the likelihood (which is called the conjugate prior) in order to make the posterior distribution have a closed-form solution. We opt to use a general (gradient-based) optimization procedure to allow our framework to work with any probabilistic model without modifying the inference algorithm. Thanks to this convenient property, we do not have to focus on the optimization aspect for each model. Instead, we spend time

[1] When analyzing gradient-based optimization algorithms, we usually care about the convergence analysis. For example, the convergence of Adam and similar adaptive methods are proved to be bounded (Kingma and Ba, 2015) and thus gradient-based methods are guaranteed to converge. We discuss the computational complexity in order to see if the proposed methods are fast enough to run in an interactive context.

and effort on the modeling step to adapt well-known DR models to work with fixed-position constraints.

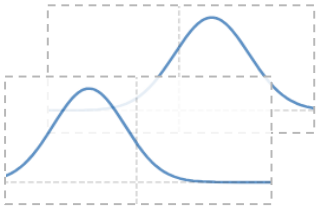
The last step is to criticize and evaluate the model. The result of our framework is an estimation of latent position for the points in an LD space that reflect the users' input constraints. Section 5.5 shown how to assess quantitatively and qualitatively the visualization results produced by our interactive methods. Moreover, this workflow is an iterative process, where we can turn back to the beginning of the modeling to redefine the prior distributions. Users when interacting with the model can also provide different sets of feedbacks and observe the corresponding results in an interactive loop.

The above framework can work with different kinds of probabilistic DR models. In Figure 5.2, the blue block (2) is tied to each particular DR model. Section 5.3 and Section 5.4 are devoted to explain the nonlinear PMDS and the linear PPCA models, as well as their interactive versions derived from our framework. The core idea for constraint integration through prior distribution in the orange (1) in Figure 5.2 will be detailed below.

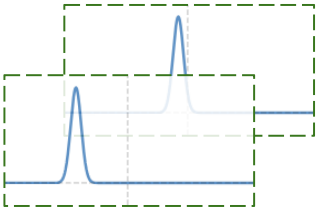
5.2.2 Constraint Integration using Informative Prior

In our approach using probabilistic latent variable models, we have to specify the latent position variables that represent the position of points in the LD space. Each observed data point in the HD space has one corresponding latent variable. We can control each particular latent variable through its prior distribution. As discussed above, the latent position of a point can be modeled by a multivariate normal distribution. For example, in a visualization task, we model the unknown position of points in a two-dimensional space.

Each latent position variable is represented by two Gaussians (corresponding to its two dimensions) as shown in the figure on the right. Each Gaussian has different means and has the same large variance to model the uncertainty about the position of the point.



If users have prior knowledge about the position for this point, they can move it to the desired position in a 2D plane. The two new coordinates for this point are recorded and encoded as the new means for each Gaussian of the two latent dimensions.



Since the exact location of points in the latent space is unknown, each dimension of this point can take any real value, or in other words, $\boldsymbol{\mu}_i$ can be drawn from a multidimensional uniform distribution $\mathcal{U}(a, b)$, with $a, b \in (-\infty, +\infty)$, for example. However, this does not give us any useful information about the unknown location, and thus this uniform distribution is also called an *uninformative prior*. In contrast, this location variable can be modeled by a more *informative prior*, such as a multivariate standard normal distribution, a reasonable choice for the point in a vector space. Learning the model, as shown in Equation 5.1, involves evaluating the *log prior*. In the case of an uninformative uniform prior (in a finite range), the log prior is a constant and does add any information to regularize the log-likelihood. In the case of an informative prior such as a standard normal, the log prior is a function of latent variables, which plays a role as a regularization term in the objective function.

Notice that, when the position of a point is unknown, the variances of the corresponding Gaussian are used to capture the uncertainty. When the point is fixed by users, the uncertainty about its position is reduced. However, we also set a small variance around each new coordinate to indicate the certainty in the feedback of users. Figure 5.3 show an overall view of the modified prior distributions for every point indicated by users. The common assumption about the independence of the latent position variables allows us to encode the fixed position of different points at the same time.

In our framework, after the step of defining prior distributions, there is no difference between the model with or without users' interaction. Since the users' feedback is integrated into the prior distributions in a general form, the modified prior and the original (non-modified) are treated in the same way. The inference step is thus unchanged for different models and different sets of prior distributions. In order to demonstrate how this framework works with different probabilistic DR models, we use two well-known DR models in this framework to derive their interactive versions.

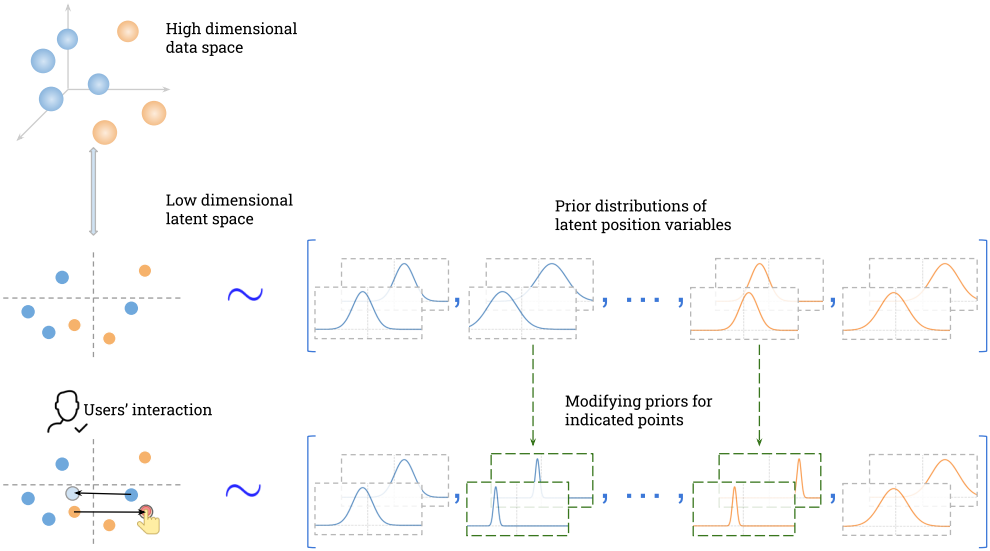


Figure 5.3: Idea of modifying prior distributions for the points indicated by users.

5.3 Concrete Example 1: PMDS Model and interactive PMDS

Multidimensional Scaling (MDS) refers to a family of techniques that transform the information in the proximities between N data points in the HD space into a *configuration* of N points in an LD Euclidean space (Borg and Groenen, 2005b, Chap. 3). In the metric MDS problem, the proximities arise from a metric space, i.e., they can be calculated using the Euclidean distances in the HD space. The traditional metric MDS method solves this problem by minimizing the *stress* defined as a residual sum of squares of the distortions between all pairs of distances in the low- and high-dimensional spaces (Kruskal, 1964a; Torgerson, 1952). In our proposed framework, we consider a probabilistic approach for solving this *distance preserving* problem. When the probabilistic MDS model (PMDS) (Hefner, 1958; MacKay and Zinnes, 1986) is used in our framework, we have a concrete method called *interactive PMDS* (iPMDS) (Vu et al., 2021c).

5.3.1 Probabilistic Multidimensional Scaling

We can think of Probabilistic Multidimensional Scaling (PMDS) as a probabilistic formulation for solving the metric MDS problem. The foundation of this formulation started from a classical *Hefner distance model* (Hefner, 1958), which is revised in many applications such as in geology (MacKay, 1983), in economics (MacKay and Zinnes, 1986), or in psychology (MacKay, 1989; Zinnes and Griggs, 1974; Zinnes and MacKay, 1983). In the PMDS model, the focus is put on describing the problem. We will first describe the characteristics of the pairwise distances, then formulate the distance-preserving problem using the distribution of distances found in the first step. Even though this model was widely used in statistical modeling, it was often introduced in purely mathematical terms. Our contribution in this section is to reveal the intuitive idea of this model in a visual form and highlight the connection between a distance model and the dimensionality reduction problem of MDS.

Hefner Distance Model

In statistics, when studying random variables, the standard analysis is to find what are the distributions of these variables. Hence, in order to characterize the Euclidean distances between any pair of data points in a vector space, we may think about the distribution of these distance values.

Supposing that each point $\mathbf{z}_i = [\mathrm{z}_{i1}, \ldots, \mathrm{z}_{ir}]$ lies on an r-dimensional space. (These notations are used to keep consistency with the main references for the text in this section (Hefner, 1958; MacKay and Zinnes, 1986; Zinnes and MacKay, 1983).) If the two data points $\mathrm{z}_i, \mathrm{z}_j$ are determined, that means we can observe or measure every dimension of each data point, the Euclidean distance between them is calculated as $D_{ij} = \sqrt{\sum_{k=1}^{r} (\mathrm{z}_{ik} - \mathrm{z}_{jk})^2}$. However, if the points $\mathbf{z}_i, \mathbf{z}_j$ are not fully observed or measured, that means we cannot access the dimension $\mathrm{z}_{ik}, \mathrm{z}_{jk}$, how can we estimate the distance D_{ij}? Another way to ask this question is how to characterize the Euclidean distance D_{ij} (or the squared distance D_{ij}^2) between any two points $\boldsymbol{z}_i, \boldsymbol{z}_j$ without knowing exactly the coordinate of each data point?

A short answer: the *standardized squared distance* is a random variable that follows a *noncentral chi-squared distribution*, characterized by a degree of

freedom and a *noncentrality* parameter (Hefner, 1958). Knowing the distribution of this variable helps us to construct a model for describing the distance-preserving problem. The rest of this section details the construction of this distribution.

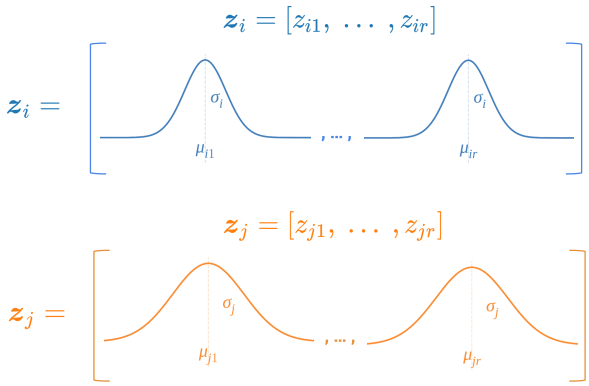


Figure 5.4: Hefner's representation of a data point in a vector space.

For each data point $\boldsymbol{z}_i$, Hefner (1958) proposed to model each unknown coordinate z_{ik} by a simple one-dimensional Gaussian located at the position μ_{ik} with a variance σ_i^2. In that way, a multidimensional point $\boldsymbol{z}_i = [\mathrm{z}_{ik}]_{k=1}^{r}$ can be represented by a multivariate Gaussian where each element is a Gaussian $\mathcal{N}(\mu_{ik}, \sigma_i^2)$. The distribution of each coordinate differs from each other at the location (the mean parameter μ_{ik}) but shares the same[2]variance σ_i^2. Since the exact coordinates of $\boldsymbol{z}_i$ is unknown, we can assume that the expected coordinates can be captured by the *location variable* $\boldsymbol{\mu}_i = [\mu_{i1}, \ldots, \mu_{ir}]$. Figure 5.4 shows a visual schema for the representation of the unknown points in the Hefner model. The variance σ_i^2 (also called the *variability parameter*) represents the uncertainty about the exact location (in every direction) of the corresponding point $\boldsymbol{z}_i$. Different points have different variances.

Let us consider the squared distances D_{ij}^2. The goal of the Hefner model is to characterize this squared distance by finding how the scalar values of this distance are distributed. In order to do that, we have to answer two questions. (1) How to calculate the (squared) Euclidean distance between two points $\boldsymbol{z}_i, \boldsymbol{z}_j$ that are represented by the multivariate Gaussians as shown in Figure 5.4? (2)

[2]Each dimension of a point can have different variances. However, it is not necessary since later we will normalize the coordinates to make them to have unit variance.

Supposing that we can calculate the pairwise distance based on the parameters $\boldsymbol{\mu}_i, \boldsymbol{\mu}_j, \sigma_i^2, \sigma_j^2$, how to find an appropriate distribution for this distance?

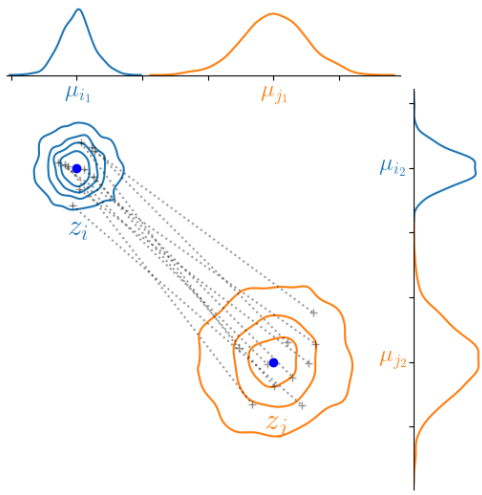


Figure 5.5: Illustration for a distance model. Two points $\mathbf{z}_i$ and $\mathbf{z}_j$ are represented by their locations $\boldsymbol{\mu}_i, \boldsymbol{\mu}_j$ and their variances shown by the contours around their locations. The distance between $\mathbf{z}_i$ and $\mathbf{z}_j$ is a random variable that can be estimated by sampling the first point from the distribution of $\mathbf{z}_i$, sampling the second point from the distribution of $\mathbf{z}_j$ and calculating the Euclidean distance between the two sampled points. The dotted lines represent the sample distances. The Hefner distance model (Hefner, 1958) characterizes the distribution of the sample distances represented by these dotted lines.

In this formulation, the distance D_{ij} is a random variable since it can be calculated from the sampled points from the two multivariate Gaussian distributions of $\mathbf{z}_i, \mathbf{z}_j$. More specifically, we first sample a point $\widetilde{\boldsymbol{z}}_i$ from $\mathcal{N}(\boldsymbol{\mu}_i, \sigma_i^2)$, and then sample a second point $\widetilde{\boldsymbol{z}}_j$ from $\mathcal{N}(\boldsymbol{\mu}_j, \sigma_j^2)$. A sampled distance is the Euclidean distance $d(\widetilde{\boldsymbol{z}}_i, \widetilde{\boldsymbol{z}}_j)$ between these two sampled points. This sampling process is repeated many times. Figure 5.5 illustrates these sampled distances for two points in a 2D space ($r = 2$).

The squared term $(\mathrm{z}_{ik} - \mathrm{z}_{jk})^2$ arises in the calculation of the squared distance D_{ij}^2. Since each element $\mathrm{z}_{ik}, \mathrm{z}_{jk}$ in this term is a Gaussian random variable, the difference $\mathrm{z}_{ik} - \mathrm{z}_{jk}$ is also a Gaussian random variable:

$$\mathrm{z}_{ik} \sim \mathcal{N}(\mu_{ik}, \sigma_i^2) \tag{5.2a}$$

$$\mathrm{z}_{jk} \sim \mathcal{N}(\mu_{jk}, \sigma_j^2) \tag{5.2b}$$

$$\Longrightarrow \mathrm{z}_{ik} - \mathrm{z}_{jk} \sim \mathcal{N}(\mu_{ik} - \mu_{jk}, \sigma_i^2 + \sigma_j^2). \tag{5.2c}$$

We have now observed that the squared distance D_{ij}^2 is a sum of squared terms $(\mathrm{z}_{ik} - \mathrm{z}_{jk})^2$, where each term (when being standardized) follows a normal distribution. The following classical results in statistics define the distribution of the sum of squares of normal random variables.

Formula 1. Let $\{\mathrm{t}_1, \ldots, \mathrm{t}_k, \ldots \mathrm{t}_r\}$ be r independently and normally distributed random variables with mean μ_k and unit variance $\sigma_k^2 = 1$. A new random variable $\mathbf{t} = \sum_{k=1}^{r} \mathrm{t}_k^2$ is distributed by a noncentral chi-squared distribution $\chi^2(r, \lambda)$ with r degrees of freedom and a noncentrality $\lambda = \sum_{k=1}^{r} \mu_k^2$.

$$\begin{aligned} &\mathrm{t}_k \sim \mathcal{N}(\mu_k,\ \sigma_k^2 = 1) \\ &\quad \Big\Downarrow \ \mathbf{t} = \mathrm{t}_1^2 + \cdots + \mathrm{t}_k^2 + \cdots + \mathrm{t}_r^2 \\ &\mathbf{t} \sim \chi^2\left(r,\ \lambda = \sum_{k=1}^{r} \mu_k^2\right). \end{aligned}$$

Here is the analogy in our distance model to the variables in this formula: the squared distance variable D_{ij}^2 plays a role of $\mathbf{t}$, where each of r elements $(\mu_{ik} - \mu_{jk})$ plays a role of t_k. The only obstacle preventing us from applying this formula is that, each variable t_k is required to have unit variance while the corresponding normal variable $(\mu_{ik} - \mu_{jk})$ has variance of $\sigma_i^2 + \sigma_j^2$. In order to transform any normal variable to have unit variance, we apply the following *standardization* formula.

Formula 2. If $\mathbf{x}$ is a normal random variable with mean μ and variance σ^2, then the distribution of $\dfrac{\mathbf{x} - \mu}{\sigma}$ will have a standard normal distribution (zero mean and unit variance):

$$\mathbf{x} \sim \mathcal{N}(\mu, \sigma^2) \quad \Longrightarrow \quad \mathbf{z} = \frac{\mathbf{x} - \mu}{\sigma} \sim \mathcal{N}(0, 1). \tag{Formula 2a}$$

Applying this formula for standardizing a normal random variable without subtracting the mean, we obtain the following consequence: If $\mathbf{x}$ is a normal

random variable with mean μ and variance σ^2, the random variable $\mathbf{z} = \dfrac{\mathbf{x}}{\sigma}$ is normally distributed with mean μ and unit variance:

$$\mathbf{x} \sim \mathcal{N}(\mu, \sigma^2) \quad \Longrightarrow \quad \mathbf{z} = \frac{\mathbf{x}}{\sigma} \sim \mathcal{N}\left(\frac{\mu}{\sigma}, 1\right). \tag{Formula 2b}$$

We now can apply Formula 2b for the distribution of $(\mu_{ik} - \mu_{jk})$ in Equation 5.2c with $\sigma = \sqrt{\sigma_i^2 + \sigma_j^2}$:

$$\begin{aligned} \mathrm{z}_{ik} - \mathrm{z}_{jk} &\sim \mathcal{N}(\mu_{ik} - \mu_{jk}, \sigma_i^2 + \sigma_j^2) \\ \frac{\mathrm{z}_{ik} - \mathrm{z}_{jk}}{\sqrt{\sigma_i^2 + \sigma_j^2}} &\sim \mathcal{N}\left(\frac{\mu_{ik} - \mu_{jk}}{\sqrt{\sigma_i^2 + \sigma_j^2}},\ 1\right). \end{aligned} \tag{5.3}$$

From this result in Equation 5.3, applying Formula 1 with the following substitutions $\mathrm{t}_k = \dfrac{\mathrm{z}_{ik} - \mathrm{z}_{jk}}{\sqrt{\sigma_i^2 + \sigma_j^2}}$ and $\mu_k = \dfrac{\mu_{ik} - \mu_{jk}}{\sqrt{\sigma_i^2 + \sigma_j^2}}$, we obtain

$$\sum_{k=1}^{r} \frac{(\mathrm{z}_{ik} - \mathrm{z}_{jk})^2}{\sigma_i^2 + \sigma_i^2} \sim \chi^2\left(r, \lambda = \sum_{k=1}^{r} \frac{(\mu_{ik} - \mu_{jk})^2}{\sigma_i^2 + \sigma_i^2}\right) \tag{5.4a}$$

$$\Longrightarrow \quad \frac{D_{ij}^2}{\sigma_{ij}^2} \sim \chi^2\left(r, \frac{d_{ij}^2}{\sigma_{ij}^2}\right), \tag{5.4b}$$

where $d_{ij}^2 = \sum_{k=1}^{r} (\mu_{ik} - \mu_{jk})^2$ and $\sigma_i^2 + \sigma_j^2$ is denoted as σ_{ij}^2 for short. Hence, the standardized squared pairwise distances $\dfrac{D_{ij}^2}{\sigma_{ij}^2}$ follow a noncentral chi-squared distribution with r degrees of freedom and a noncentrality parameter $\lambda = \dfrac{d_{ij}^2}{\sigma_{ij}^2}$. This distribution is parameterized by $\boldsymbol{\mu} = [\mu_{ik}]$ with $i = 1..N$ and $k = 1..r$, and $\boldsymbol{\sigma}^2 = [\sigma_i^2]$ with $i = 1..N$.

The result from Equation 5.4 gives us the probabily distribution of $\frac{D_{ij}^2}{\sigma_{ij}^2}$. An additional transformation is needed to obtain the target probability distribution of D_{ij}. Let us denote G the cumulative distribution function (CDF) of a noncentral chi-squared distribution of $\frac{D_{ij}^2}{\sigma_{ij}^2}$, and F the CDF of the target quantity

D_{ij}. Since the CDF of a noncentral chi-squared is strickly increasing on its support $[0, +\infty)$, the *change of variable* formula[3] gives us

$$F(D_{ij}) = G\left(\frac{D_{ij}^2}{\sigma_{ij}^2}\right). \tag{5.5}$$

Taking the derivative of these CDFs gives us the probability density function (PDF)

$$f(D_{ij}) = \frac{2D_{ij}}{\sigma_{ij}^2} g\left(\frac{D_{ij}^2}{\sigma_{ij}^2}\right), \tag{5.6}$$

where g is the PDF of a noncentral chi-squared random variable.

For complete details, the PDF $g(\cdot)$ of a random variable $\mathbf{x}$ distributed by a noncentral chi-squared distribution (with a degree of freedom k and a noncentrality parameter λ) is defined as

$$g(\mathbf{x}; k, \lambda) = \chi^2(r, \lambda) = \frac{1}{2} \exp \frac{-(\mathbf{x} + \lambda)}{2} \left(\frac{\mathbf{x}}{\lambda}\right)^{\frac{k}{4} - \frac{1}{2}} I_{\frac{k}{2}-1}\left(\sqrt{\lambda \mathbf{x}}\right), \tag{5.7}$$

where $I_\nu(\mathbf{y})$ is a modified Bessel function of the first kind of degree ν

$$I_\nu(\mathbf{y}) = \left(\frac{\mathbf{y}}{2}\right)^\nu \sum_{i=0}^{\infty} \frac{(\mathbf{y}^2/4)^i}{i!\, \Gamma(\nu + i + 1)}. \tag{5.8}$$

In summary, Hefner model characterizes the pairwise Euclidean distance in an r-dimensional space by the density function $f(D_{ij})$ defined in Equation 5.6. This representation is used to construct the likelihood function in the PMDS model for the distance-preserving problem.

Probabilistic Distance Preserving Model

In the Hefner model, the data points are assumed to lie on an r-dimensional space where the pairwise distances D_{ij} are constructed. This model also assumes that the data are not fully observed, but we can measure the pairwise distance between data points. That means the pairwise distance matrix $\boldsymbol{D} = [D_{ij} =$

[3] A graphical explanation for the change of variable rule for increasing functions can be found at https://online.stat.psu.edu/stat414/lesson/22/22.2.

$d(\boldsymbol{x}_i, \boldsymbol{x}_j)]$ is available while the data points $\boldsymbol{x}_i, \boldsymbol{x}_j$ are unknown. The forward direction of calculating $\boldsymbol{D}_{\mathrm{N}\times\mathrm{N}}$ from the HD data is trivial. The backward direction of inferring the configuration of N data points from $\boldsymbol{D}$ is much harder. Using the Hefner model, we can model the unknown data points in a very low-dimensional space ($r = 2$ or 3) in such a way that the pairwise distances reconstructed from this space approximate the true measured/observed input distances in $\boldsymbol{D}$. This is exactly the problem of metric MDS.

Using Hefner model for describing pairwise distances, the construction of a probabilitis MDS model (MacKay and Zinnes, 1986; Zinnes and MacKay, 1983) is as follows.

1. As the true pairwise distances can be collected beforehand, they are called observed variables $D_{ij} \in \mathbb{R}$.

2. As the original data points are unknown, each point $\boldsymbol{z}_i$ is modeled by a multivariate (r-dimensional) Gaussian located at the location $\boldsymbol{\mu}_i \in \mathbb{R}^r$ with a scalar variance $\sigma_i \in \mathbb{R}$.

3. We expect to reconstruct the true distances $\boldsymbol{D}_{\mathrm{N}\times\mathrm{N}} = [D_{ij}]$ by describing the distribution of these distances by the PDF defined in Equation 5.6. That means we expect to approximate the true input distance D_{ij} by the distance $d_{ij} = \sqrt{\sum_{k=1}^{r}(\mu_{ik} - \mu_{jk})^2}$ in the LD space through the relationship defined in the probability distribution in Equation 5.4.

4. Finally, the likelihood function $p(D_{ij}|d_{ij})$ tells us how well the approximated distance d_{ij} matches the observed distance D_{ij}. The locations and variances $(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ are parameters of the model and are estimated by maximum likelihood. As a result, the locations $\boldsymbol{\mu} \in \mathbb{R}^{N\times r}$ are the representation of N points in an LD space.

This model is used in the modeling step of our framework, which leads to an interactive model called *interactive PMDS* (iPMDS).

5.3.2 Interactive PMDS Model

The unified framework in Section 5.2 provides a mechanism to integrate users' constraints on fixed-position into the prior distributions. When the PMDS model

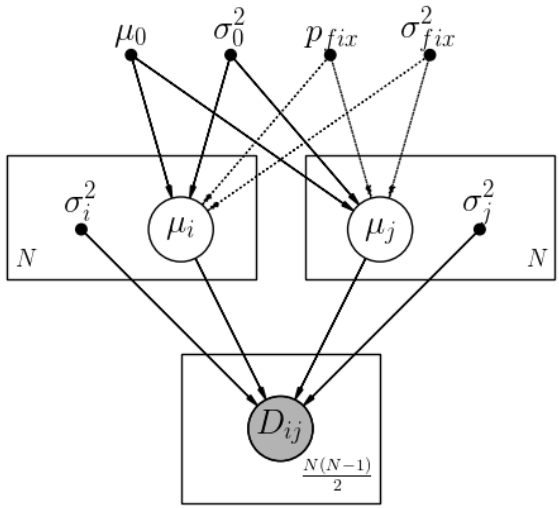


Figure 5.6: Graphical model for the proposed latent variable interactive probabilistic MDS model. The latent variables $\boldsymbol{\mu}_i, \boldsymbol{\mu}_j$ are Gaussian random variables with mean $\boldsymbol{\mu}_0$ and variance σ_0^2. The fixed position of points indicated by the user are also encoded in the prior of $\boldsymbol{\mu}$ using the specific positions $\boldsymbol{p}_{fix}$ and σ_{fix}^2. The variance parameters of each point σ_i, σ_j are hyperparameters of the model. The observed variable D_{ij} models the distance between the points sampled from two Gaussian distributions defined above. In a dataset of N instances, $N(N-1)/2$ unique pairs are considered.

is used in this framework, we have to identify what are the prior distributions needed to modify to encode the fixed-position constraints. More specifically, we need to identify the latent variables in the PMDS model for which we can modify their priors. As a result, we introduce an interactive PMDS (iPMDS) model, which can be used as an interactive version of MDS that can handle uncertainty in the user feedback. A precise form of the likelihood function with the modified prior in iPMDS is also constructed and compared to the objective function of the original metric MDS.

iPMDS: Model Construction

The PMDS model defined in Equation 5.4 is a standard way to explain how the distance D_{ij} relates to the latent variables $\boldsymbol{\mu}_i, \boldsymbol{\mu}_j$. However, in complex models, a graphical representation is usually used to present the relationship between variables. This representation is called *probabilistic graphical model* or *Bayesian*

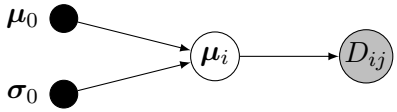
network (Jordan, 1998). The relationships represented in graphical models are the dependencies between variables and the independent assumptions. Examples for each kind of relationship will be given below. A *graphical model* for iPMDS is shown in Figure 5.6. In a graphical model, random variables are represented by cycles, and arrows indicate the *dependent* relationship.

From bottom to top of this figure, the first shaded node D_{ij} is an observed variable represent the pairwise distance. Assuming that the pairwise distance is symmetric, i.e., $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = d(\boldsymbol{x}_j, \boldsymbol{x}_i)$, there are thus $N(N-1)/2$ distinct pairs corresponding to D_{ij}. The *plate notation* represented by a rectangular around this variable indicates that this variable is repeated $N(N-1)/2$ times. Tracing to source of the arrows pointing to this observed variable, we see two *plates* of N repeated sets of variables. This is an explicit way to describe that a pairwise distance D_{ij} comes from two individual unknown points $\boldsymbol{z}_i, \boldsymbol{z}_j$. Each unknown point is represented by a latent variable shown in an unshaded node $\boldsymbol{\mu}_i$ or $\boldsymbol{\mu}_j$ and the corresponding variance parameters σ_i^2 or σ_j^2. It should be noted that the unknown points $\boldsymbol{z}_i, \boldsymbol{z}_j$ are not shown in the graphical model since these points are encapsulated in the variable D_{ij} according to the formulation in Equation 5.4. (For details, this equation describes that the standardized squared distance follows a noncentral chi-squared distribution, which in turn is a sum of squared normal distributions of the components in the unknown points $\boldsymbol{z}_i, \boldsymbol{z}_j$.)

The model's parameters are represented by filled black dots ● in the graphical model. They are not inferred (not learned) but are fixed in the model definition. The variances σ_i^2, σ_j^2 represent the uncertainty about the location of points. In this model, we opt to fix the variances ($\sigma_i = \sigma_j = \text{constant}, \forall i, j \in 1..N$) instead of modeling them as latent variables for a sake of simplicity. Moreover, fixing these parameters helps us not to be confused between the model's uncertainty about the location of each point and the users' uncertainty in their feedback.

The unknown points $\boldsymbol{z}_i$ are modeled by the location latent variables $\boldsymbol{\mu}_i$ and the variance parameters σ_i in an LD space. The traditional Hefner model only requires $\boldsymbol{z}_i$ to be a multivariate Gaussian in order to apply Formula 1 for constructing a noncentral chi-squared distribution for the pairwise distance. The location latent variable $\boldsymbol{\mu}_i$ in its turn is also modeled by another distribution,

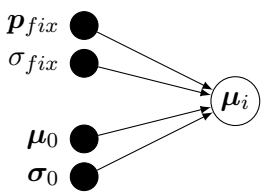
which is called a *prior distribution*. For example, in the graphical model in Figure 5.6, the relationship



encodes this hierarchy:

$$\boldsymbol{\mu}_i \sim \mathcal{N}(\boldsymbol{\mu}_0 = \mathbf{0}_r, \boldsymbol{\sigma}_0 = \mathbf{1}_r), \text{where } \boldsymbol{\mu}_i = [\mu_{i1}, \ldots, \mu_{ik}, \ldots, \mu_{ir}]^T$$
$$\frac{D_{ij}^2}{\sigma_{ij}^2} \sim \chi^2\left(r, \frac{d_{ij}^2}{\sigma_{ij}^2}\right), \text{where } d_{ij}^2 = \sum_{k=1}^{r}(\mu_{ik} - \mu_{jk})^2$$

Using this model allows us to integrate the fixed-position constraint directly into the prior distribution. Let us suppose that users indicate the positions for several points of interest. The ensemble of fixed positions is denoted by $\boldsymbol{p}_{fix}$. The users' uncertainty about the position of the fixed points is also captured in a parameter called σ_{fix}. Each fixed point can also have a variance, which captures the uncertainty in the users' feedback. A small value of variance indicates a low uncertainty in the users' feedback, i.e., users are sure about their fixed location. We assume that every fixed point has the same level of uncertainty for simplifying the model. In summary, the iPMDS model describes the prior distributions of the latent position variables as



$$\boldsymbol{\mu}_i \sim \begin{cases} \mathcal{N}(\boldsymbol{p}_i, \sigma_{fix}^2 \mathbf{1}) & \text{if the } i^{th} \text{ point is fixed,} \\ \mathcal{N}(\mathbf{0}, \mathbf{1}) & \text{otherwise.} \end{cases} \tag{5.9}$$

iPMDS: Model Inference

After defining the model, the next step is to do inference to estimate the unknown quantities represented by latent variables. Given the observed data that are the observed pairwise distances $\boldsymbol{D} = [D_{ij}]$ indexed by (i, j), we want to estimate the latent positions in $\boldsymbol{\mu} = [\boldsymbol{\mu}_i], i = 1..N$. This can be done by maximizing the log posterior $\log p(\boldsymbol{\mu} \mid \boldsymbol{D}) \propto \log\{p(\boldsymbol{D} \mid \boldsymbol{\mu})p(\boldsymbol{\mu})\} = \log p(\boldsymbol{D} \mid \boldsymbol{\mu}) + \log p(\boldsymbol{\mu})$. It

should be noted that the fixed-position constraints are encoded into the prior distributions $\boldsymbol{\mu}_i$ of the related points as in Equation 5.9.

Now let us show another usage of the graphical model in Figure 5.6. The plate notation is used not only to represent the variables that are repeated many times but also to introduce an *indenpendently identically distributed* assumption. For the prior distribution of $\boldsymbol{\mu}_i, \boldsymbol{\mu}_j$, the two plates of N variables indicates that each of N latent variables are independently distributed. In general, this assumption helps us to factorize the log prior as

$$\log p(\boldsymbol{\mu}) = \underbrace{\sum_i \log p(\boldsymbol{\mu}_i \mid \boldsymbol{\mu}_{fix}, \sigma^2_{fix})}_{\text{fixed points}} + \underbrace{\sum_i \log p(\boldsymbol{\mu}_i \mid \boldsymbol{\mu}_0, \sigma^2_0)}_{\text{other untouched points}} . \tag{5.10}$$

We can also rewrite this log prior in a compact form without indexing exactly the fixed points as

$$\log p(\boldsymbol{\mu}) = \sum_{i=1}^{N} \log p(\boldsymbol{\mu}_i \mid \boldsymbol{\mu}_0, \boldsymbol{\mu}_{fix}, \sigma^2_0, \sigma^2_{fix}). \tag{5.11}$$

Similarly, the plate of $N(N-1)/2$ observed variables D_{ij} indicates that all the pairwise distances are independently distributed, that allows us to factorize the log likelihood function as

$$\log p(\boldsymbol{D} \mid \boldsymbol{\mu}) = \sum_{1 \le i < j \le N} \log p(D_{ij} \mid \boldsymbol{\mu}_i, \boldsymbol{\mu}_j, \sigma^2_i, \sigma^2_j). \tag{5.12}$$

The final log posterior, which is our loss function, is expanded as

$$\begin{aligned} \mathcal{L}(\boldsymbol{\mu}) = & \sum_{1 \le i < j \le N} \log p(D_{ij} \mid \boldsymbol{\mu}_i, \boldsymbol{\mu}_j, \sigma^2_i, \sigma^2_j) \\ & + 2(N-1) \sum_{i=1}^{N} \log p(\boldsymbol{\mu}_i \mid \boldsymbol{\mu}_0, \boldsymbol{\mu}_{fix}, \sigma^2_0, \sigma^2_{fix}). \end{aligned} \tag{5.13}$$

Each of N location variable $\boldsymbol{\mu}_i$ appears in $N-1$ pairs connecting to other points $\boldsymbol{\mu}_j$. For each observed variable D_{ij}, there are thus two related location variables $\boldsymbol{\mu}_i, \boldsymbol{\mu}_j$. That gives the factor of $2(N-1)$ for the prior term in Equation 5.13.

In this loss function, the log likelihood is calculated from $N(N-1)$ pairs D_{ij}

while the log prior is calculated from $2N$ points $\boldsymbol{\mu}_i, \boldsymbol{\mu}_j$. Since these two terms are not well-calibrated, the optimization can be unstable and hard to converge. However, the formulation in Equation 5.13 is elegant in such a point that, if we divide both two terms by a constant $N(N-1)$, well-calibrated terms will appear. The first term becomes $\frac{1}{N(N-1)} \sum_{1 \leq i < j \leq N} \log p(D_{ij} \mid \boldsymbol{\mu}_i, \boldsymbol{\mu}_j, \sigma_i^2, \sigma_j^2)$, which is the average log likelihood (since there are $N(N-1)$ distinct pairs D_{ij}). The second term becomes $2\frac{1}{N}\frac{\cancel{(N-1)}}{\cancel{(N-1)}} \sum_{i=1}^{N} \log p(\boldsymbol{\mu}_i \mid \boldsymbol{\mu}_0, \boldsymbol{\mu}_{fix}, \sigma_0^2, \sigma_{fix}^2)$, which is two times the average log prior (of N data points). Therefore, using the average instead of the sum in practice helps us obtain stable results.

For a complete reference, the log likelihood in Equation 5.13 is derived from the PDF of a noncentral chi-square in Equation 5.7 in the case where $r = 2$ as

$$\begin{aligned} &\log p(D_{ij} \mid \boldsymbol{\mu}_i, \boldsymbol{\mu}_j, \sigma_i^2, \sigma_j^2) \\ &= \log\left(\frac{D_{ij}}{\sigma_{ij}^2}\right) - \frac{1}{2}\frac{(D_{ij} - d_{ij})^2}{\sigma_{ij}^2} + \log IE_0\left(\frac{D_{ij}\, d_{ij}}{\sigma_{ij}^2}\right), \end{aligned} \tag{5.14}$$

where $IE_0(.)$ is the exponentially-scaled modified Bessel function of *zero-degree*, which ensures the numerical stability. When $\sigma_{ij}^2 \to 0$, maximizing the log likelihood in Equation 5.14 is similar to minimizing the stress $(D_{ij} - d_{ij})^2$ of MDS. As such, MDS is a special case of our method iPMDS when σ tends to zero and the interaction is not used.

As observed in this equation, every term involves only one specific pair (i, j) and is evaluated as real value functions. Indeed, D_{ij} is the observed distance coming from the input data (a pairwise distance matrix), d_{ij} is a scalar variable, and σ_{ij}^2 is a scalar parameter. The function $IE_0(.)$ is thus a real function, which is implemented in terms of convergent series[4]. Therefore, the log likelihood in Equation 5.14 has an $\mathcal{O}(1)$ complexity. Similarly, the log prior of a 2D gaussian with a diagonal covariance matrix (for the latent variables $\boldsymbol{\mu}_i, \boldsymbol{\mu}_j$) also has an $\mathcal{O}(1)$ complexity. The complexity of the log posterior of the whole dataset of $N(N-1)/2$ pairs is thus $\mathcal{O}(N^2)$.

[4] https://docs.scipy.org/doc/scipy/reference/generated/scipy.special.ive.html

5.4 Concrete Example 2: PPCA Model and interactive PPCA

Similar to the iPMDS model, based on the probabilistic PCA model (PPCA) (Tipping and Bishop, 1999), we introduce an interactive version called interactive PPCA (iPPCA) (Vu and Frénay, 2019). In our proposed framework, we can use several probabilistic DR models, and the fixed-position constraints will be integrated into the prior distributions of the latent position variables. In this section, we introduce the PPCA model, a basic probabilistic DR model with the characteristics and the modeling process that are different from the PMDS model introduced before. However, the constraint integration remains the same under the definition of modified prior distribution in our framework.

5.4.1 Probabilistic Principal Components Analysis

For a quick introduction to the probabilistic PPCA model, we will first introduce how this model is useful, and why we need such kind of model. Then, we present a visual illustration of the generative process in PPCA in order to explain how it works. Lastly, a formal formulation using a graphical model is introduced for a complete model definition.

As introduced in Section 2.2.1, PCA is based on the eigendecomposition procedure to find its principal components, which are the dominant eigenvectors. On one hand, when the number of data points is larger than the number of features, we can construct the covariance matrix and apply the eigendecomposition procedure to it. On the other hand, when the number of features in the dataset is large, the covariance matrix is thus too large. An alternative solution is to decompose the centered data matrix using SVD. However, when both the number of features and data points are too large, the eigendecomposition solution is not suitable. In this case, we need an iterative solution to solve for only the top k largest eigenvalues (and eigenvectors) instead of solving for all of them. The probabilistic formulation of PCA uses the maximum likelihood principle with EM algorithm to solve this problem efficiently (Tipping and Bishop, 1999). Moreover, the original PCA does not work with missing data (missing feature values of each data point). PPCA treats all the unknown quantities as latent variables and thus can naturally deal with this kind of

missing data. PPCA is also useful when being combined in a probabilistic mixture model (Bishop, 1999; Tipping and Bishop, 1999) for *soft-clustering*, data compressing, and visualization.

The goal of conventional PCA is to find a projection matrix $\boldsymbol{W} \in \mathbb{R}^{p \times q}$ to map the data $\boldsymbol{X} \in \mathbb{R}^{N \times p}$ in an HD space to an embedding $\boldsymbol{Z} \in \mathbb{R}^{N \times q}$ in a subspace of lower dimensionality. The projection matrix $\boldsymbol{W}$ is optimized in such a way that the embedding preserves the variances in the data (Hotelling, 1933), or equivalently, minimizing the reconstruction error $||\boldsymbol{X} - \boldsymbol{Z}\boldsymbol{W}^T||^2$ (Pearson, 1901). Figure 5.7 shows an example for this viewpoint in a two-dimensional space. The conventional PCA projects the 2D points (shown in blue dots) onto the principal component that is the z axis. The projected points lie on this line, such as the red cross. This linear projection is simple and efficient, but it is not easy to understand the intuition behind the dominant eigenvectors that form the projection matrix. Let us consider another way of mapping between the high- and low-dimensional data.

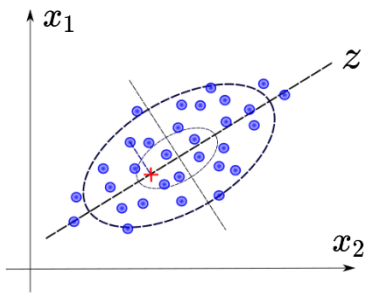


Figure 5.7: Example of simple 2D data and its principal component (a line) on the 1D space. We can project each 2D blue point onto this line to obtain an 1D representation such as the red cross on the z axis.

PPCA: A viewpoint through a generative process

We take the same above example of points in a 2D space and call the projected points as *latent positions* denoted by the variable z and the original points as *observed points*[5] denoted by the variable x. A probabilistic viewpoint of PPCA

[5]For a simple notation adapted for the example in a 2D space, we do not distinguish whether the random variables are univariate (for $z \in \mathbb{R}$) or multivariate (for $x \in \mathbb{R}^2$). They are both denoted by lowercase letters.

can be seen through these steps.

- The latent position z is drawn from a one-dimensional probability distribution, called the prior distribution $p(z)$.
- Our goal is to construct a distribution of the observed points x supposing that we know the latent positions z. This probability distribution is denoted as $p(x|z)$, used to measure how likely we obtain a particular x given a particular z.
- After having this model, the Bayes rule gives us a posterior distribution $p(z|x)$, which is used to find the latent positions of z after observing the input data points x.

The main idea of the PPCA model is the way it describes how the data is generated. A scalar value $\widetilde{z}$ is sampled from the prior distribution $p(z)$ and the corresponding point $\widetilde{x}$ in a two-dimensional space is then generated through a *random process*. Since we do not have enough information to map a point from an LD space to an HD space, we need to add some randomness into the generative process. By capturing the characteristics of this random process (by a mathematical model) and applying the Bayes' theorem, we can do the *reverse process* that maps the HD data into an LD subspace.

A step-by-step illustration for a generative process of this new viewpoint is shown in Figure 5.8. First, a scalar value of z is sample from a prior distributed $p(z)$ in the one-dimensional space, that gives us a sampled value $\widetilde{z} \sim p(z)$. Second, $\widetilde{z}$ is mapped to a two-dimensional space through a linear transformation with a projection vector $\vec{w}$ and a translation away from μ:

$$\widetilde{z} \rightarrow \mu + \widetilde{z}\vec{w}, \quad \text{with } \mu = \begin{bmatrix} \mu_0 \\ \mu_1 \end{bmatrix}, \vec{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}.$$

At this step, a scalar value $\widetilde{z}$ is transformed into a 2D point via linear mapping of vector $\vec{w}$. It should be noted that this vector will be learned. Now we can assume that this vector is the principal component of the data in a 2D space. That means $\vec{w}$ represents the direction along which the data are spread out the most. However, no information about other spreading directions of the data is available. For that reason, at the final step, an isotropic noise $\epsilon \sim \mathcal{N}(0, \sigma^2\mathbf{1})$ is

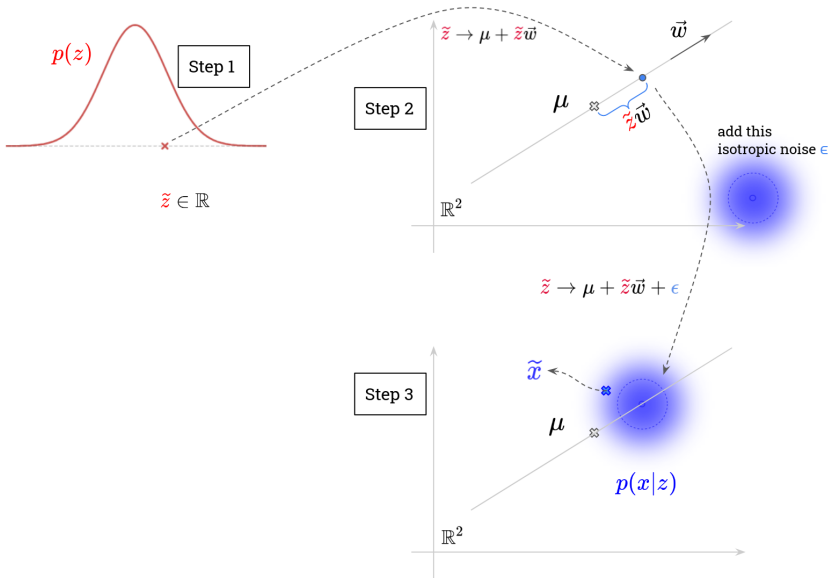


Figure 5.8: Generative process of PPCA.

added to the transformation:

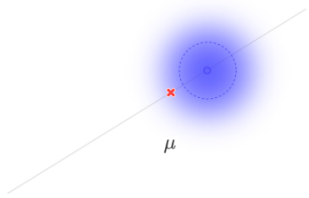
$$\tilde{z} \to \mu + \tilde{z}\vec{w} + \epsilon, \quad \text{where } \epsilon \sim \mathcal{N}\left(\begin{bmatrix}0\\0\end{bmatrix} \middle| \begin{bmatrix}\sigma^2 & 0\\0 & \sigma^2\end{bmatrix}\right).$$

The mapped point can thus be any point sampled from an isotropic Gaussian at the location $\mu + \tilde{z}\vec{w}$ with a variance $\sigma^2\mathbf{1}$ as shown as a blue cross in the last step in Figure 5.8. In summary, the presented process allows us to sample a point $\tilde{x}$ in a 2D space given a sample $\tilde{z}$ in a 1D space. That means we can formulate the distribution $p(x|z)$ to describe the above linear mapping. However, we have not seen how this generative process can construct the distribution of the whole dataset like the cloud of blue points presented in the example in Figure 5.7.

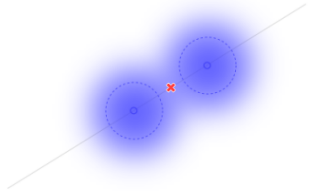
PPCA: From a generative process to a latent variable model

Let us see what can be obtained if the above process is repeated many times. The following series of plots on the right of the page show gradually a form of the density of the variable $x \in \mathbb{R}^2$ given the sample of $z \in \mathbb{R}$.

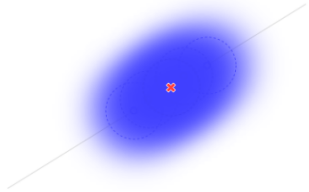
As a result of the presented process, a sample $\widetilde{z}$ is mapped to a point in 2D located at the position $\mu + \widetilde{z}\vec{w}$ with an additive Gaussian noise ϵ. The result in the last step in Figure 5.8 is shown on the right with all annotations removed to avoid clusters.



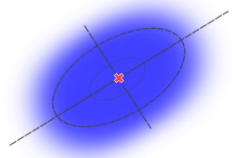
This generative process is repeated: another value of $\widetilde{z}$ is sampled and transformed to another location in 2D. The isotropic Gaussian at each location is blured (from high density at the center to low density outside, with the doted blue circle indicating *one standard deviation* σ) .



When the Gaussians at the mapped points (5 samples on the right) overlap, we can observe the density of the whole samples of $\widetilde{x}$. Notice that we start from the density of $\widetilde{x}$ given a particular $\widetilde{z}$. When we create this density for different samples $\widetilde{z}$, we will see the general form of the density of $\widetilde{x}$.



Now the process is repeated 10 times, an ellipse annotation is put on top of the density suggests that this density is also a Gaussian. This density is $p(x)$, which gives us a same picture as the density of a cloud of points in Figure 5.7.



As a final result of the generative process, a density distribution of the observed data $p(x)$ in the above figures is represented by a multivariate Gaussian located at $\boldsymbol{\mu}$ with a covariance matrix determined by the projection vector and the additive noise ϵ. This is another advantage of PPCA that can be used as a density estimation or an outlier detection method. Thanks to the elegant property of the linear Gaussian model, the posterior distribution $p(z|x)$ is also a Gaussian (Bishop, 2006, Sec. 2.3, Sec. 12.2). Hence, the maximum likelihood estimation (MLE) of PPCA has an analytical solution for the unknown quantities like the projection matrix and the variances. An iterative inference algorithm like Expectation-Maximization is also applied for solving the MLE problem for computational efficiency. However, in our proposed framework, we do not rely on the closed-form solution of each particular model but solve

the general maximum a posterior (MAP) by a gradient-based method (See Section 5.2). After constructing the PPCA model, we will make it interactive by integrating the users' feedback on the position of fixed points into this model.

5.4.2 Interactive PPCA Model

Our framework is designed to transform the users' feedback into fixed-position constraints and integrate them into a probabilistic DR model. iPPCA (Vu and Frénay, 2019) is an interactive method produced by this framework when the PPCA model is used. The most important element in this method is the encoding of constraints as an informative prior for the prior distribution of latent position variables in the PPCA model. The latent variable $\boldsymbol{z}_i \in \mathbb{R}^q$ with $i = 1..N$ is a representation in an LD space of the corresponding data point represented by the observed variable $\boldsymbol{x}_i \in \mathbb{R}^p$ in an HD space. Estimating sampled values for the embedding $\boldsymbol{Z}_{\mathrm{N\times q}} = [\boldsymbol{z}_i]$ in a 2D or 3D space ($q = 2$ or 3) gives us a visualization of the observed input data $\boldsymbol{X}_{\mathrm{N\times p}} = [\boldsymbol{x}_i]$. Without loss of generality, we assume that the observed data $\boldsymbol{X}$ are centered (i.e., subtracting the mean vector out of every data point). Since the latent position $\boldsymbol{z}_i$ is unknown, it can be modeled by a multivariate standard normal distribution with zero mean and unit variance. When users know in advance the positions for their point of interests, this kind of knowledge/feedback is encoded into the prior distribution of the corresponding latent variable in the iPPCA model as

$$\boldsymbol{z}_i \sim \begin{cases} \mathcal{N}(\boldsymbol{p}_i, \sigma^2_{fix}\mathbf{1}) & \text{if the point is indicated by users,} \\ \mathcal{N}(\mathbf{0}, \mathbf{1}) & \text{otherwise.} \end{cases} \tag{5.15}$$

Similarly to the iPMDS model introduced before, the prior distribution of the latent variable is controlled by four parameters: a location $\boldsymbol{\mu}_0 = \mathbf{0}_q$, a variance $\boldsymbol{\sigma}^2_0 = \mathbf{1}_q$, together with the fixed position indicated by users $\boldsymbol{\mu}_{fix} = \boldsymbol{p}_i$, and the uncertainty of the feedback σ^2_{fix}. Whether the latent position $\boldsymbol{z}_i$ is touch by user or not, the generative process for generating the corresponding observed variable $\boldsymbol{x}_i$ is kept the same: $\boldsymbol{x}_i \leftarrow \boldsymbol{W}\boldsymbol{z}_i + \boldsymbol{\epsilon}$[6]. The whole process is presented by the graphical model in Figure 5.9. For a complete model, details

[6]Notice that the input data are centered, the mean variable $\boldsymbol{\mu}$ is thus ignored in the generic mapping $\boldsymbol{x}_i \leftarrow \cancelto{\mathbf{0}}{\boldsymbol{\mu}} + \boldsymbol{W}\boldsymbol{z}_i + \boldsymbol{\epsilon}$.

for the projection matrix $\boldsymbol{W}$ and the noise $\boldsymbol{\epsilon}$ are given below.

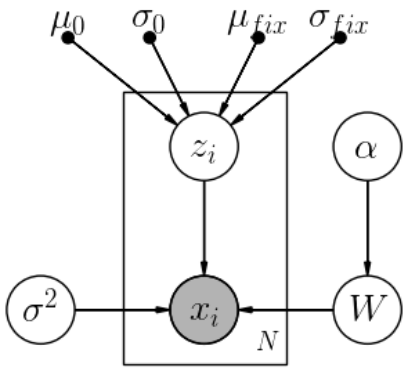


Figure 5.9: Graphical model for the proposed interactive PPCA method. Each of N data points $\boldsymbol{x}_i$ is represented by an observed variable in the shaded node. This observed variable is generated from the corresponding latent position variable $\boldsymbol{z}_i$ through the process $\boldsymbol{x}_i \leftarrow \boldsymbol{W}\boldsymbol{z}_i + \boldsymbol{\epsilon}$. Two other elements involving in this generative process are represented by the latent variables σ^2 (controlling the additive noise) and the projection matrix $\boldsymbol{W}$ (controlling the linear projection). The variances in each value in the projection matrix in their turns are controlled by another latent variable $\boldsymbol{\alpha}$. The latent position variable $\boldsymbol{z}_i$ follows a Gaussian distribution with default parameters $\boldsymbol{\mu}_0, \sigma_0$ if the ith point is not touched by the user. Otherwise, the fixed location parameter $\boldsymbol{\mu}_{fix}$ and the feedback uncertainty parameter σ_{fix} is used to control the prior distribution of the fixed point.

The isotropic noise is a multivariate Gaussian in the data space $\mathbb{R}^p$:

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{1}_p). \tag{5.16}$$

Notice that different samples in the generative process are added with the same noise $\boldsymbol{\epsilon}$. As shown in the previous example in Figure 5.8, this noise is controlled by a variance σ^2. Instead of fixing this noise variance, it can be modeled as a latent variable and can be learned. We choose a *LogNormal* distribution (with zero mean and unit variance) as prior for this variable to make sure that the value of σ^2 is always non-negative:

$$\sigma^2 \sim \text{LogNormal}(0, 1). \tag{5.17}$$

The unknown projection $\boldsymbol{W}$ is a $p \times q$ matrix where each scalar element

can be modeled by a Gaussian distribution. A convenient way to represent this matrix is to model each of p rows $\boldsymbol{w}_k \in \mathbb{R}^q, k = 1..p$ by a multivariate Gaussian with different variances for each of q latent dimension:

$$\boldsymbol{w}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\alpha}\mathbf{1}_q), \tag{5.18}$$

where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_l, \ldots, \alpha_q]$. This unknown quantity in fact can be considered as another variable that models the variance in each of q latent dimensions of the values in the projection matrix. In order to learn (to infer from data) these variances automatically, we can define a prior distribution for each of these variances:

$$\alpha_l \sim \text{InverseGamma}(1, 1). \tag{5.19}$$

Based on the Gaussian rule for a linear model, when both the prior $p(\boldsymbol{z}_i)$ and the noise $\boldsymbol{\epsilon}$ are Gaussians, the distribution of an observed variable $\boldsymbol{x}_i$ given all latent variables is thus a Gaussian:

$$\boldsymbol{x}_i \mid \underbrace{\boldsymbol{z}_i, \sigma^2, \boldsymbol{W}, \boldsymbol{\alpha}}_{\text{unknown quantities}} \sim \mathcal{N}(\boldsymbol{W}\boldsymbol{z}_i, \sigma^2\mathbf{1}_p). \tag{5.20}$$

After defining the prior distributions for all latent variables and the likelihood function, our framework can handle the inference step by doing MAP with a gradient-based optimization method. As a result, we can obtain the estimation of the latent positions $\boldsymbol{Z}$ with the quantified uncertainty σ^2 around each point, as well as the projection matrix $\boldsymbol{W}$ with quantified variances $\boldsymbol{\alpha}$. The derivation of the log posterior used for MAP estimation with the modified prior is the same as the iPMDS model in Section 5.3.2. This log posterior for an individual data point $\boldsymbol{x}_i$ is dominated by the log likelihood (log PDF of $\mathcal{N}(\boldsymbol{W}\boldsymbol{z}_i, \sigma^2\mathbf{1}_p)$) and the log prior of $\boldsymbol{W}$, which has an $\mathcal{O}(pq) = \mathcal{O}(p)$ complexity for $q \ll p$ thanks to the diagonal covariance matrix. The log posterior of the whole dataset is factorized by N independent data points and thus has an $\mathcal{O}(pN)$ complexity.

5.5 Experimental Results

The evaluation of interactive methods is subjective since the result depends on the interaction and the users' feedback. We demonstrate the usefulness

and several applications of our proposed framework via case studies. A similar approach of evaluation with case studies is used with other interactive DR methods like XGVis (Buja et al., 2001, 2008), V2PI-MDS (Endert et al., 2011; Leman et al., 2013), V2PI-PCA (Endert et al., 2011), and User-guided PPCA (House et al., 2015). We will define beforehand the goal and the task that the user has to perform for each case study. iPMDS and iPPCA are two concrete methods derived from our proposed framework. These two methods are evaluated with the same case studies. In each case study, the user can use our proposed interactive methods to move several points in order to manipulate the visualization. Since different users can move the points in different ways, our case studies use task-based scenarios to guide the users to move the points intentionally to achieve the goal.

In each different scenario, we assess if the interactive method can give the desired visualization and assess the visual quality as well as the interpretability of the visualization. Another quantitative measure such as the MDS stress is also used for the experiments with iPMDS. Three case studies are performed with various datasets of different types and different sizes. Besides the case studies, additional experiments to evaluate the proposed methods iPPCA and iPMDS are detailed in Section 9.1, including a quantification of uncertainty for each point with iPPCA, an interaction with iPMDS in the case of *incomplete data* (i.e., missing pairs in the pairwise distance matrix).

The fixed-position constraints are constructed from the fixed points indicated by users. Our framework proposes to model the users' uncertainty in their feedbacks via the hyperparameter $\boldsymbol{\sigma}^2_{fix}$, which is called the variance of fixed points. In all case studies, this hyperparameter is fixed to a small value of 10^{-3} for both iPMDS and iPPCA models to indicate that users are certain about their feedbacks. This small variance also keeps the fixed points close to the indicated position to help identify these points before and after the interaction easily. Therefore, one simple way to verify the correctness of our interactive model is to assess if the indicated points are placed close to the user's desired positions in the visualizations. The effect of this hyperparameter when users are not certain about their feedbacks will also be discussed later in Section 5.6.

iPPCA and iPMDS take different kinds of input. In our experiments, the same datasets (or subsets of the datasets) are used for both two methods with the same preprocessing step. With iPMDS, the Euclidean pairwise distances

are calculated and used as input data. Due to the similar results are obtained with these two methods, only the visualizations of iPMDS are presented in this section. Similar results of iPPCA are left at the end of this chapter for additional references.

5.5.1 Case Study 1: Interaction for Creating Transformation Effect

This experiment is performed on a subset of 250 points of the first five classes of the Digits dataset (Kaynak, 1995). Each data point is an 8×8 gray-scale image of a hand-written digit.

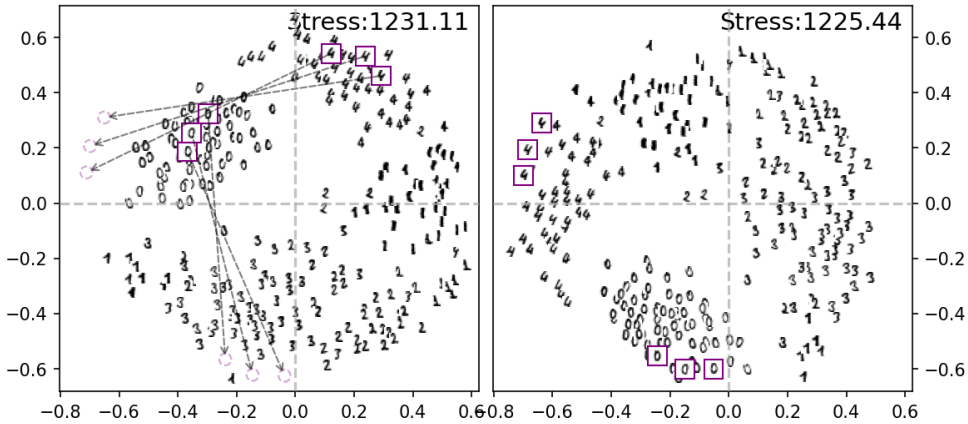
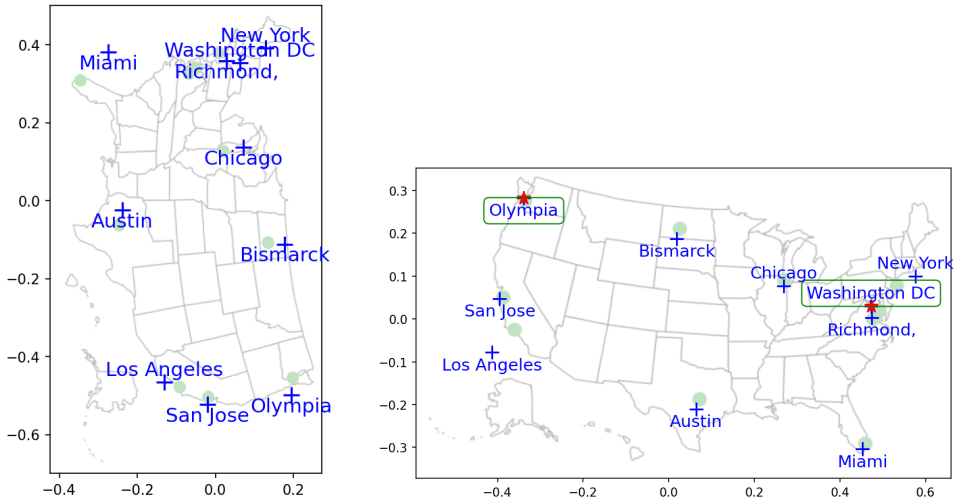


Figure 5.10: iPMDS with a rotation effect when interacting with several images of the Digits dataset.

The goal is to manipulate the visualization of this dataset in a simple way to make a rotation effect. First, users can observe the initial visualization (of iPMDS without any interaction) shown on the left of Figure 5.10. They can then move several images in this visualization to indicate their intention: several images of digit 4 are moved to the left edge and several images of digit 0 are moved to the bottom edge. According to the annotated arrows in this figure, we can expect that the whole visualization will be *'rotated'* anti-clockwise, following the direction guided by the moving points. iPMDS takes these fixed positions and finds a new configuration for all other points as shown on the left of Figure 5.10. In fact, iPMDS does not do a linear transformation to

rotate the whole visualization, but it learns the new positions for every point (including the points indicated by users) through an inference algorithm. The new embedding shows a rotation effect as expected.



(a) The 2D map found by the original metric MDS with stress = 0.003.

(b) The 2D map result of iPMDS with stress = 0.041. The position of *Olympia* and *Washington, D.C.* are indicated by the user.

Figure 5.11: Alignment of US cities on a map using MDS (a) and iPMDS (b). The US map (with the correct positions of 10 cities in green dots) is shown in background as ground truth.

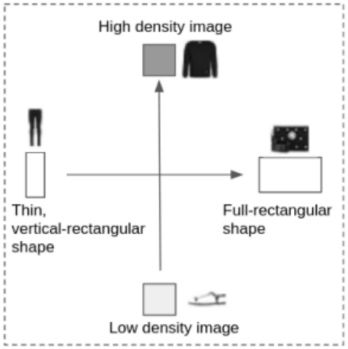
Another linear transformation effect can also be obtained via appropriate fixed anchor points. Let us retake the motivating example of the configuration of 10 cities in the US in Figure 5.1 at the beginning of this chapter. The distances between the 10 cities are calculated on the spherical Earth by the Haversine (great circle) distance, which gives 45 pairs to be used as input data. The original non-probabilistic metric MDS (with SMACOF algorithm) gives the solution (the best stress among 10 different runs) shown in Figure 5.11a. The map of the US with the correct position of the cities as green dots is plotted in the background as ground truth. The result of MDS is not useful and should be manually flipped and rotated to correspond to reality. This can be accomplished with iPMDS by fixing the position of 2 points corresponding to *Olympia* (on the West coast) and *Washington, D.C.* (on the East coast). Small errors can still be observed in the visualization, but the visualization is

now understandable. A similar case study performed on *weighted MDS* with six fixed points can be found in (Leman et al., 2011).

Since the visualizations produced by MDS, PMDS or PPCA can be in any arbitrary orientation (Kruskal, 1964a; Tipping and Bishop, 1999; Zinnes and MacKay, 1983), it is not easy to interpret the meaning of the axes. With appropriate indications from users, iPMDS and iPPCA can correct the orientation according to the guide of users. That means these methods can be used to align the visualizations to the axes of the coordinate system, and hence to explain the meaning of the axes. In the next case studies, we show how to use iPMDS to create understandable axes for both image and tabular datasets.

5.5.2 Case Study 2: Understandable Axes with Image Dataset

In this case study, we use a subset of 250 gray-scale 28×28 images from the Fashion-MNIST dataset (Xiao et al., 2017). Figure 5.12a shows an embedding of iPMDS, which reveals patterns of objects with different shapes (like shoes, trousers, bags, and T-shirt/pulls) or different zones of low/high-density images. However, it is not clear what is the meaning of the two coordinate axes. iPMDS allows users to implicitly propose the meaning of axes using examples. Based on the initial visualization of iPMDS without interaction in Figure 5.12(a), users can use images to describe the axes: three images of thin, long-shaped trousers are moved to the left; three images of full rectangular-shaped bags are moved to the right; three low-density images of sandals and shoes are moved to the bottom; and three high-density images of pulls are moved to the top. The first two changes indicate that the horizontal axis should represent the shape, while the two last changes indicate that the vertical axis should represent the pixel density of images. The schema beside summarizes the conceptual axes implied in the feedback. Figure 5.12(b) shows the iPMDS result where the global structure is similar to the initial visualization while reflecting the desired axes.



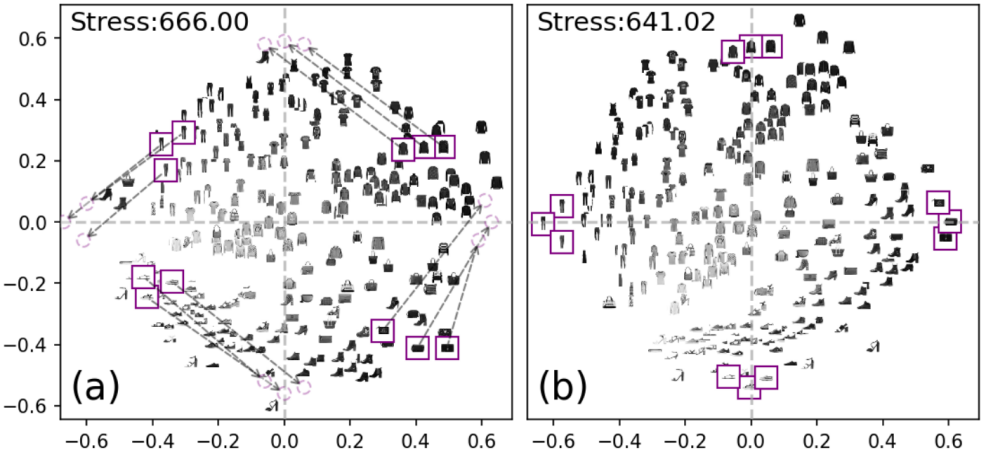


Figure 5.12: iPMDS with the subset of 250 images of the Fashion-MNIST dataset. (a): iPMDS result without interaction. The global structure can be explored in this visualization, but the axes do not have any meaning. The user can thus describe the desired axes by fixing several examples highlighted in purple squares. The arrows show the change toward the new positions. (b): New iPMDS visualization with the fixed points guided by the users shown in purple squares. The axes' meaning can be interpreted as follows. The x-axis represents *shape* with images of thin shape on the left and images of full rectangular shape on the right. The y-axis represents *pixel density* with dark, high-density images on the top and low-density images on the bottom.

5.5.3 Case Study 3: Interpretable Axes with Tabular Dataset

In the last case study, we use the Automobile tabular dataset consisting of 203 cars characterized by -26 features[7]. Among them, two characteristics are chosen to distinguish the cars in Figure 5.13(a): the number of doors (represented by colors) and the number of cylinders (represented by markers). Four different groups are easily revealed, however, the coordinate axes are not easy to understand. Using iPMDS, users can indicate different groups of cars with the interaction denoted in Figure 5.13(a): a car with four doors and four cylinders is placed in the first quadrant on the top right; a small car with two doors and two cylinders is placed in the second quadrant on the top left; a small sportive car with two doors and six cylinders is placed in the third quadrant on the bottom left; and a big wagon with four doors and eight cylinders is placed in the fourth quadrant on the bottom right.

[7]https://archive.ics.uci.edu/ml/datasets/automobile

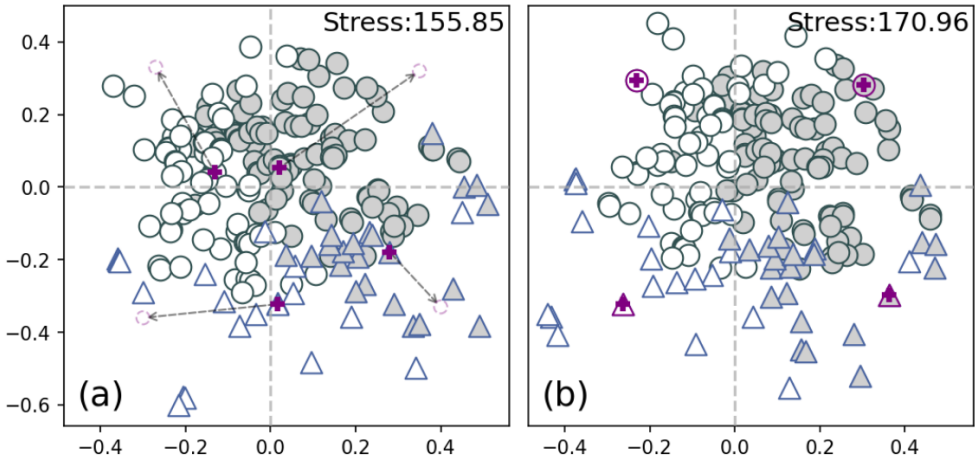
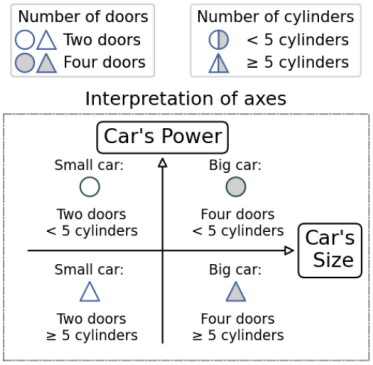


Figure 5.13: iPMDS with the Automobile dataset of 203 instances. (a): Initial embedding of iPMDS without interaction. Four groups of cars characterized by two features (number of doors and number of cylinders) can be distinguished. However, the boundary between these groups is not well aligned with the coordinate axes, making it hard to understand the axes. The positions of four cars from the four groups are fixed in the four quadrants. The arrows show the change towards new positions. (b): New visualization produced by iPMDS in which the boundary between groups aligns with the coordinate system. Interpretation of axes: The x-axis represents *car's size* with small cars of two doors on the left and larger cars of four doors on the right. The y-axis represents *car's power* with normal cars with few cylinders on the top and more powerful cars on the bottom.

The resulting visualization of iPMDS that takes the user constraints into account is shown in the schema beside. Axes can be interpreted as shown in Figure 5.13(c): small cars of two doors on the left and larger cars of four doors on the right. Thus, the x-axis represents the size of the car. Cars with two or four cylinders on the top and cars with more than four cylinders on the bottom, which makes the y-axis represent the power of the car. Therefore, the user has defined the axes implicitly by using only four examples placed in the four quadrants of the coordinate system.



Interpreting the meaning of the axes in the visualizations produced by non-linear DR methods is difficult. Creating visualizations with understandable axes is more difficult. Variances of popular linear DR methods such as PPCA and PMDS do not have explicit linear mapping due to the complex formulation in the generative process and the nonlinear objective functions. Even with linear methods like MDS, the visualization can be in arbitrary orientation. Users desire a powerful non-linear DR method but also want to interpret the visualization results. For that reason, an interactive method allowing users to define understandable axes can be useful in practice. Through the presented use cases, the proposed framework can help users manipulate the visualizations to interpret them the way they want.

5.6 Discussion

In this chapter, we propose a unified probabilistic framework to integrate fixed-position constraints into any probabilistic DR method. The chosen probabilistic approach is not only powerful for modeling HD data but also useful for representing users' constraints in the form of informative priors. We also contribute to unveiling classical PPCA and PMDS models in a simple language with visual explanations. Based on the proposed framework, we derive two concrete methods named interactive PMDS and interactive PPCA and demonstrate their applications through different case studies. This section discusses more details about the usage of our models, as well as their limitations.

5.6.1 Usage of the Proposed Models

The most important usage of our model is to correct a long-standing problem of the embedding produced by traditional PCA and MDS. Probabilistic PCA has a drawback called *ambiguous-rotation* phenomenon for which the visualization can be in any rotation around the origin. A *rotation-correction* using orthogonal components of the projection matrix is proposed, however, the final embedding is not unique (Tipping and Bishop, 1999). Similarly, the MDS visualization can also have any arbitrary orientation. In order to explore and understand the visual patterns in the MDS results, users can apply methods that find the best rotation for interpretation if external features are available (Bibal et al., 2018;

Marion et al., 2019). These methods differ from our approach in the point that we propose interactive probabilistic models to find, with the help of users, a visualization that is more understandable for them. Thanks to this approach, users can easily manipulate the visualization and create interpretable axes even with incomplete input data.

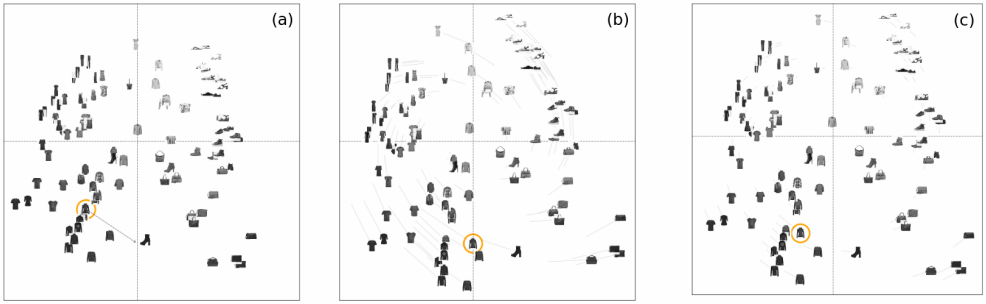


Figure 5.14: Effect of the uncertainty σ_{fix} of the users' feedback. (a) The user moves an image of a coat from a subset of the Fashion-MNIST dataset, which is marked by the orange circle. The results of iPPCA model are shown in (b) for a small value of uncertainty in the user's feedback $\sigma^2_{fix} = 1e^{-3}$ and in (c) for a large uncertainty $\sigma^2_{fix} = 0.2$. Small uncertainty indicates that the user is certain about the indicated position of the interacted point.

When using our models, we assume that users are always certain about their feedback. However, users can also express their uncertainty about their feedback, for example, when they are not sure about the position of fixed points. In this case, our model captures the uncertainty of every fixed point by the hyperparameter σ^2_{fix}. The example with a small subset of the Fashion-MNIST dataset in Figure 5.14 is produced by the iPPCA model with two different values of σ^2_{fix}. Supposing that the user moves an image of a coat marked by the orange circle toward the south-east direction as in Figure 5.14(a). A small uncertainty indicates that users are certain about their feedbacks and the users' constraints will regularize the model. Figure 5.14(b) shows the result after the interaction with $\sigma^2_{fix} = 1e^{-3}$. In contrast, with a larger uncertainty (e.g., $\sigma^2_{fix} = 0.2$ in Figure 5.14(c)), the effect of the constraints decreases. When the user is not sure about this feedback, the model learns mostly from data without being heavily regularized by the constraints.

The uncertainty about the inferred position of the latent variables can be obtained from our models[8], however, we do not show these quantities in the

visualization. To avoid confusion, only the uncertainty in the users' feedback is considered (while both kinds of uncertainty can be modeled at the same time in our model). It should be also noted that the same uncertainty is used for every fixed point in our model. When users interact with multiple points, the uncertainty in the feedback of each point can be captured by different hyperparameters. This could be useful for advanced users who understand the effect of the fixed points and want to have more freedom to test their hypotheses when moving the points.

The last point to notice in the usage of our models involves their hyperparameters. The proposed models are designed to be easy to use for end-users, who want to interact with the visualization without knowing the underlying mechanism. Even though the inference problem (of estimating the unknown quantities) in probabilistic models is not a trivial one, this step is totally transparent to users. We opt to transform the probabilistic inference into an optimization one that can be solved efficiently with modern gradient-based optimizers like Adam. The two concrete proposed models iPPCA and iPMDS have different natures but can be inferred in exactly the same way by maximizing the log posterior using gradient descent. Expert users can still choose different optimization algorithms or tune their hyperparameters such as the learning rate or the momentum. However, once the model is tuned (in the development phase), end-users can use it for interaction scenarios. Since the users' feedback are modeled together with the data, the *interactive part* is always a component in the model (see Figures 5.6 and 5.9). Therefore, the model tuned without interaction can work with users' interaction on any data point.

5.6.2 Limitations

We have found several limitations in our approach, which are the limitations of the proposed interactive framework and also of the concrete methods. We try to suggest solutions for these limitations that can be left for future work.

In theory, the proposed interactive framework can work with any latent

[8] A probabilistic approach allows to quantify the uncertainty about the estimated position of latent variables. In the iPPCA, these quantities are available through the variable σ^2 in the graphical model in Figure 5.9 - Section 5.4.2. For the iPMDS model, these quantities can be obtained if we model σ_i^2, σ_j^2 in the graphical model in Figure 5.6 - Section 5.3.2 as latent variables instead of model parametes.

variable DR model (a model that represents the unknown position of points in the LD space as latent variables). In practice, it is only proved to work with only simple latent variable models like PPCA and PMDS. We have not successfully integrated other potential models such as generative topographic mapping (GTM) (Bishop et al., 1998), gaussian process latent variable model (GP-LVM) (Lawrence, 2005), maximum entropy unfolding (MEU) (Lawrence, 2012) into this framework. GTM and PPCA are in the same family of the standard factor analysis model. GTM uses a nonlinear mapping and the EM algorithm for inference, which can be straightforward to integrate into our framework, but we have not done it yet. The two other models are more complex, both in the modeling and the inference step. In our framework, we have not implemented other efficient inference algorithms such as variational inference (required by GP-LVM) or sparse penalized MLE for the inverse of the covariance matrix (required by MEU). It is necessary to have at least one universal inference algorithm such as a variance of Markov chain Monte Carlo (MCMC) (Betancourt et al., 2017) or a variational inference (Blei et al., 2017) method to deal with complex models.

We also have a limitation in the workflow in Figure 5.2 where several steps are not carefully considered. First, building a probabilistic model is usually an iterative process of modeling, criticizing, and revising. We plan to focus more on the model criticism step to measure how well the model performs for a specific data exploration task (Blei, 2014). Second, the visualization assessment in an interactive context with different users is not well evaluated. Our experiment is conducted in supposing that all the user's feedbacks are sound and coherent, i.e., the indicated positions of the interacted points make sense. In fact, this assumption is not realistic, which reveals several shortcomings in our approach. We should study and experiment with other aspects related to the interacted points, such as how many points can we move freely to obtain a reliable result, or what are the feasible zones in which the points can be moved, and how the interactive user intent is modeled (Ruotsalo et al., 2014). Our models can be used as an interactive tool to help users to discover and understand patterns in their data. Different users may interact differently, and the visualization quality is left for subjective assessment by users.

The concrete models derived from our framework have also their own limitations. iPPCA is based on a linear mapping and its representation is very

limited. Even though the objective function is nonlinear, the mapping is just a linear projection corrupted by noise. We can change the linear mapping by a complex nonlinear function (Bishop et al., 1998). However, we are far away from powerful latent variable probabilistic models like VAE (Kingma and Welling, 2014). iPMDS itself also has a technical limitation. It only produces visualization in a 2-dimensional space. For visualizations in a 3-dimensional space, it requires a noncentral chi-squared distribution with 3 degrees of freedom and an approximation of the modified Bessel function of degree 0.5, which is slower and requires an extended work on the computation of gradient.

In this chapter, we promote the probabilistic approach with its advantage in handling missing data and uncertainty quantification. However, probabilistic methods also have disadvantages in both the modeling and computation aspects. First, probabilistic modeling is hard and is often done by trial-and-error since there is no mathematical model that perfectly describes the data. Moreover, the choice of distribution for each latent variable is not always evident. It should be chosen for computational efficiency, for an elegant formulation with a conjugate prior, for an easy approximation, etc. Second, each model requires different inference algorithms. Traditional graphical models use message passing algorithms, while general latent variable models can be inferred directly with MLE or EM algorithms. In most cases, we have to derive a closed-form solution for the estimation of unknown parameters/latent variables. However, probabilistic models are not always tractable, especially when the number of latent variables and the number of observed data points is large. In our unified framework, we have found an adequate representation for encoding fixed position constraints, implemented all the components of the framework by differentiable functions, and proposed to use a gradient-based optimizer that does not depend on the concrete model.

Chapter 6

Constraint-Preserving Score for Visualization Assessment

This chapter presents a new way of using pairwise constraints to evaluate the quality of a visualization based on the semantic information encoded in the user-defined similar/dissimilar constraints.

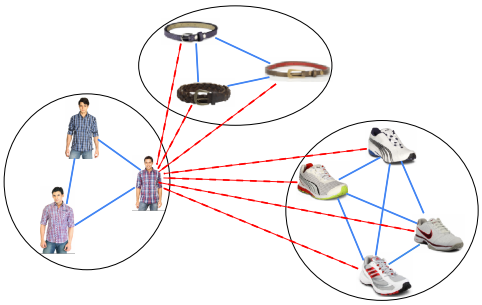
Contents

6.1 Quality Metrics for Visualization Assessment **142**
6.2 The Proposed Constraint-Preserving Score **147**
6.3 Experimental Results with f_{score} **151**
6.4 Empirical Characteristics of f_{score} **161**
6.5 Application of f_{score} for a Hyperparameter Tuning Problem . . . **167**
6.6 Discussion **173**

This chapter is based on our publication titled "Constraint Preserving Score for Automatic Hyperparameter Tuning of Dimensionality Reduction Methods for Visualization" (Vu, Bibal, and Frénay, 2021a).

Throughout this thesis, we try to combine the users' constraints with well-known visualization methods. The first kind of hierarchical constraint can be integrated into t-SNE or an SNE-based method. The second kind of fixed-position constraint is in a probabilistic framework for interactive visualization. In this last chapter for the technical contribution, we propose a novel idea using another kind of constraint, the *pairwise constraint*, to evaluate the quality of a visualization. Besides the traditional constraint integration approach, we introduce a new way of using constraints for assessing visualizations.

When observing a visualization, especially one of SNE-based methods like t-SNE, LargeVis, and UMAP, users often tend to search for groups of objects because they expect similar objects should be grouped together. These methods are designed to preserve neighborhood information in the data by $\overrightarrow{\textit{pulling}}$ similar objects close together while $\overleftrightarrow{\textit{pushing}}$ dissimilar objects far away. If we know in advance what are the pair of similar objects and dissimilar objects, we can evaluate how well these methods perform. This simple idea suggests that the relationship between similar/dissimilar objects can be used to measure the quality of a visualization. The relationship between similar/dissimilar objects is formulated in the form of *similar link* and *dissimilar link* constraints, which are collectively called *pairwise constraints*.



Here are examples of the pairwise constraints among instances of three different groups of sample images in a fashion product dataset. Similar links (plain blue) indicate images in the same groups. Dissimilar links (dashed red) indicate images of different groups. In order to assess a visualization, we propose to measure how these pairwise constraints are preserved in the visualization by a quantitative measure called *constraint-preserving score*.

Assessing the visualization quality is still an open question and each state-of-the-art visualization quality metric only captures one specific aspect of the visualization like neighborhood preservation. Section 6.1 will discuss in detail this problem. Our proposed constraint-preserving score can capture different aspects of the visualization according to the information encoded

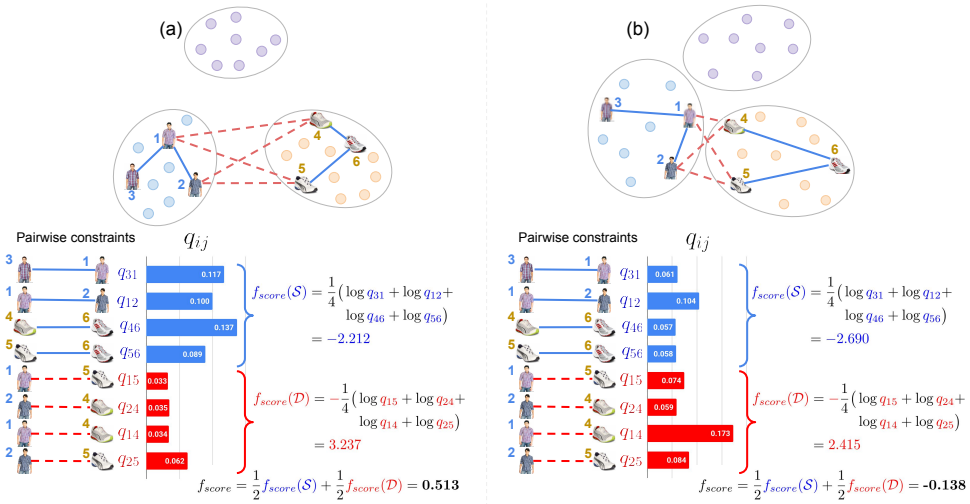


Figure 6.1: Illustration of how the proposed f_{score} assesses the visualizations. Two different visualizations (a) and (b) of the same dataset are shown on top with the same set of pairwise constraints including four similar links denoted by plain blue lines and four dissimilar links denoted by dotted red lines. For each pair (i, j) in the input constraints, the quantitative *strength* q_{ij} is calculated (by Equation 6.2) and is visualized by the bar charts. f_{score} for the similar/dissimilar links and the final f_{score} are calculated (using Equations 6.3, 6.4 and 6.5). More details are in Section 6.2. In the visualization (a), the selected images connected by similar link constraints are close and thus the q_{ij} values are large, while the images connected by dissimilar link constraints are distant, which makes the q_{ij} values small. The values of q_{ij} for the same set of input constraint for the visualization (b) are opposite since the similar images are placed far apart while the dissimilar ones are closer. f_{score} can measure this difference and give a high score for (a) and a much lower score for (b).

in the input constraints of users. In brief, this score measures how well the information encoded in input constraints is preserved in a visualization. An overview illustration of what are the quantities measured by our score is shown in Figure 6.1. Section 6.2 will explain how these quantities are measured. Experimental results and empirical characteristics of this score are presented in Sections 6.3 and 6.4. The proposed score can have a potential impact since it is very easy to use and works with any visualization method. Domain experts can express their knowledge in a simple form of similar or dissimilar groups of points. If needed, end-users can use labeled data (which are usually available for coloring groups in the visualization), even in a small amount. Thanks to this usage, this score can be used to automatically tune the hyperparameters of

visualization methods, which is presented in Section 6.5. Finally, we discuss several disadvantages and suggest solutions to overcome them in Section 6.6.

6.1 Quality Metrics for Visualization Assessment

Choosing the right evaluation metric for different machine learning models is not a trivial task. Even for a simple classification problem, the accuracy score is not always a good choice, in particular for unbalanced datasets. For a more difficult problem of object detection, not a single but 12 different metrics are currently used in practice for evaluating the performance of object detectors on the popular COCO dataset[1]. Specific problems require specific evaluation metrics, and new metrics are still proposed to characterize the performance of different algorithms. Evaluation metrics for supervised learning are well studied, however, fewer metrics are developed for unsupervised learning, particularly for DR methods. Evaluating a visualization method is difficult since we have to assess at the same time the visual appearance of the visualization, as well as the preservation of the structures in the original data.

From the viewpoint of end-users, they want to obtain the best visualization for their data by using a state-of-the-art DR method. Powerful visualization methods with excellent implementation are available, but their hyperparameters must be carefully tuned. Choosing good hyperparameter values is crucial since it predetermines the quality and usefulness of the visualization (McInnes et al., 2018b; Wattenberg et al., 2016). Typically, the most suitable visualization is chosen through trial-and-error, which is tedious and difficult for users. We break down the problem of automatically choosing a good visualization for end-users into two sub-problems. The first problem is to define a quality metric that can assess the visualization according to the users' criteria. The second problem is how to efficiently search through all possible visualizations corresponding to different combinations of hyperparameters of a DR method to find the best one.

[1] https://cocodataset.org/#detection-eval

Table 6.1: Properties of the five cluster-label-agnostic quality metrics to assess visualizations.

metric	range	description
CC	$[0, 1]$	Pearson correlation coefficient between pairwise distance vectors
NMS	$[0, +\infty)$	stress based on comparison of pairwise distance orders
CCA	$[0, +\infty)$	stress with emphasis put on LD
NLM	$[0, +\infty)$	stress with emphasis put on HD
$AUC[R_{NX}]$	$[-1, 1]$	how neighbors in HD are preserved in LD

6.1.1 How to Measure the Quality of a Visualization

In a general machine learning task, there are two types of metrics to measure the performance of an algorithm that can sometimes be confused[2]. The first type is the *optimization metric*, such as cross-entropy for a classification problem, mean squared error for a regression problem, or a value of a stress function for a DR problem. The second type is the *quality metric*, such as the accuracy for a classification problem, the R^2, known as *goodness-of-fit* for a regression task, or the $AUC[R_{NX}]$ score for a visualization. In this chapter, the two terms *(quality) metric* and *score* are used interchangeably[3].

Most quality metrics use class labels as ground truth information. There are fewer quality metrics for unsupervised methods that do not use class labels. The optimization metric such as the value of the objective function of DR methods can also be considered to measure the quality of the embedding (Bibal and Frénay, 2019). Several of them are summarizes in Table 6.1.

The *correlation coefficient* (CC) (Geng et al., 2005) computes the correlation between the pairwise distances in HD and LD. The well-known *Kruskal's non-metric stress* (NMS) (Kruskal, 1964a), often used as the objective function of non-metric multidimensional scaling, compares the pairwise distance orders in HD and LD. The *curvilinear component analysis stress* (CCA) (Demartines and Hérault, 1997) is a variant of Kruskal's stress with an emphasis on the

[2]Even though the term *performance* is also confused since we do not indicate that it is the performance of the optimization procedure or the performance of the output model (applied for unseen data in a supervised setting).

[3]The term *metric* indicates a quantitative measure, not a distance function in mathematics.

embedding distances. This metric evaluates the embedding quality by focusing on the correctness of close instances in LD. The *Sammon's non-linear mapping stress* (NLM) (Sammon, 1969) is similar to CCA but focuses on the closeness of instances in HD.

Finally, $AUC[R_{NX}]$, a quality metric that differs from the above scores, assesses the visualization by measuring how well neighborhoods in HD are preserved in LD (Lee and Verleysen, 2009). An average normalized intersection of the neighborhood sets in the two spaces is calculated for different neighborhood sizes k in a logarithmic scale. The area under this curve then gives the $AUC[R_{NX}]$ score that assesses the average DR quality on all scales (Lee and Verleysen, 2010).

6.1.2 How to Choose the Best Visualization

Choosing the best visualization is choosing the best combination of hyperparameters of a DR method. Powerful methods go together with a long list of hard-to-understand hyperparameters. For example, XGBoost[4] one of the most practically used machine learning methods and its twins, LightGBM [5], have hundreds of hyperparameters to tune. Luckily, widely used visualization methods like t-SNE and UMAP only expose about a dozen of hyperparameters, of which we divide into two orthogonal categories. The first category contains hyperparameters involving the optimization procedure such as learning rate, number of iterations, *early-stopping* criteria, etc. The second category contains the ones involving the quality of the visualization, including the neighborhood size (perplexity in t-SNE/LargeVis and n_neighbors in UMAP), the *tightness* between points (min_dist, only for UMAP[6]), the exaggeration factor (only for t-SNE[7]). These two sets of hyperparameters are rarely tuned together. The first set should be tuned in the development phase to make sure the optimization runs correctly, i.e., the loss function decreases constantly, the algorithm converges, the gradients do not explode due to a high learning rate. This set is pretty easy to tune by looking at the plot of the objective function and adjusting the corresponding hyperparameters. We assume that the first set is tuned correctly

[4] https://xgboost.readthedocs.io/en/latest/
[5] https://lightgbm.readthedocs.io/en/latest/
[6] https://umap-learn.readthedocs.io/en/latest/parameters.html
[7] https://opentsne.readthedocs.io/en/latest/parameters.html

and only focus on the perplexity for t-SNE/LargeVis and the combination of n_neighbors and min_dist for UMAP. These hyperparameters determine the structure in the visualization, e.g., local structures can be revealed with a small neighborhood size while a large one reveals more global structures. For users, they determine the *appearance* of the visualization (McInnes et al., 2018b; Poličar et al., 2019) and are the main topic of interest.

Related Work on Hyperparameter Tuning for DR Methods

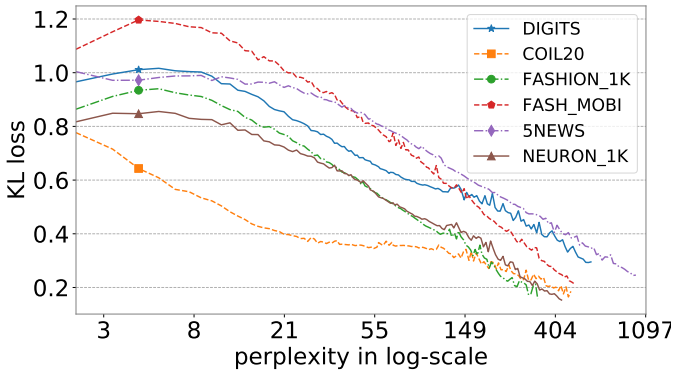


Figure 6.2: Evolution of the KL loss for several datasets, which tends to decrease systematically as the perplexity increases. Perplexities are chosen in logarithmic scale from $[2, N/3]$.

The suggested values for perplexity are between 5 and 50 (Maaten and Hinton, 2008). However, for large datasets, a much larger perplexity (of $N/100$) is also recommended (Kobak and Berens, 2019; Kobak and Linderman, 2021). Therefore, there is no evidence that the suggested perplexities are good for all datasets. The original t-SNE paper also proposes a simple method to select a good perplexity by looking at the Kullback-Leibler (KL) loss produced by several perplexities and choosing the lowest one. However, the KL loss tends to decrease when the perplexity increases (Cao and Wang, 2017), which is confirmed by our experiments, as shown in Figure 6.2. For this reason, using the KL loss for evaluating the embedding quality is not suitable since a very high perplexity would always be chosen.

Few papers in the literature attempt to derive the best hyperparameter

values for DR methods automatically. Strickert (Strickert, 2012) suggests using rank-based data to avoid perplexity calculation. Lee et al. (Lee et al., 2014) use a multi-scale approach by averaging all neighborhood sizes. Despite bypassing the perplexity selection problem, these two solutions do not solve the selection problem itself. Cao and Wang (Cao and Wang, 2017) try to tackle the problem by selecting the perplexity of t-SNE that minimizes a modified Bayesian information criteria (Schwarz, 1978)

$$\text{BIC} = 2\text{KL}(P||Q) + \frac{\text{perplexity}}{N}\log(N), \tag{6.1}$$

where $\text{KL}(P||Q)$ is the KL loss of t-SNE, and N is the number of instances. This score is based on the criterion for model selection (the Bayesian information criterion), which regularizes the KL loss of t-SNE by a *not-so-large* perplexity. A basic criterion for model selection is that among different models with similar performances, a simpler model is preferred. Applying this idea for t-SNE, the perplexity represents the model's complexity. Large perplexity requires more computation for calculating the neighbor graph and thus is considered as a factor to make the model more complex. BIC helps us not to choose the model with too small KL loss (best performance but too complex) but to choose a simpler model with not-so-large perplexity instead. However, this score is only applicable for t-SNE, and this approach requires an exhaustive linear search through all sampled values of perplexity.

Our Hyperparameter Tuning Strategy

In general, hyperparameters of DR methods can be tuned by trial-and-error or through a naive grid search. Better approaches exist, such as random search (Bergstra et al., 2011), which randomly samples combinations of hyperparameters. However, the parameter space in which the search takes place grows exponentially with respect to the number of hyperparameters.

We propose to use Bayesian optimization (BayOpt) (Močkus, 1975; Mockus et al., 1978) for solving this task. Even though BayOpt has successfully solved the problem of hyperparameters tuning for classification (Snoek et al., 2012) or experimental design/randomized experiments (Letham et al., 2019), to the best of our knowledge, until the time of publishing this work, BayOpt has not been

widely used for visualization yet. BayOpt looks at the evaluated visualizations, constructs a model to determine which combination of hyperparameters should we try next. Instead of trying many combinations, it only tried the potential one to discover the solution space or to find a better solution. The main idea of BayOpt, its advantage, and the combination of our proposed score within this method is presented in Section 6.5.1.

6.2 The Proposed Constraint-Preserving Score

Humans can often distinguish similar and dissimilar high-dimensional objects (e.g., comparing images by visual features such as the shape, colors, or objects therein). Our idea is to use the information given by pairwise links between objects to evaluate the quality of a visualization. Modern methods (t-SNE, LargeVis, and UMAP) preserve local structures in the dataset, i.e., instances that are similar in HD should be close in the embedding space. These methods are considered as successful when they reveal clear groups of similar instances. If examples of such group patterns are known in advance, they can be used to assess the quality of the visualization. We thus propose to measure how the given pairwise constraints are preserved in the visualization by a quantitative measure called *constraint-preserving score*. This section first explains how the pairwise constraints can be used for visualization assessment and then presents the derivation of the formula for quantifying the constraint preservation.

6.2.1 Pairwise Constraints for Visualization Assessment

The proposed constraint-preserving score is based on a set of predefined input pairwise constraints, including *similar link* and *dissimilar link* constraints. A similar link constraint connecting two similar objects indicates that they should be close. If in a visualization, these two objects are actually placed close together, the corresponding similar link constraint is said to be satisfied or to be preserved. The same idea is applied to dissimilar link constraints. We will detail later how the proposed score quantifies exactly the *amount of preservation* of each individual constraint. Therefore, a visualization can be considered as good when it preserves well the input constraints and can be considered as bad when it does not preserve the input constraints. The proposed constraint-preserving

score is a quantitative measure of how well the users' constraints are preserved in the visualization. This score is a function of not only the evaluated visualization but also the input users' constraints $\mathcal{C}$. We use the notation f_{score} to denote this score and to indicate that it is a function of input constraints: $f_{score}(\mathcal{C})$.

The input constraints determine the structure users want to evaluate in the visualization. f_{score} can measure the preservation of structures in the data since the information about the structures is totally encoded in the input pairwise constraints. Additionally, depending on the way the constraints are created, we can measure the preservation of different structures. Other quality metrics such as $AUC[R_{NX}]$ and the BIC-based score produce only one deterministic score value for a given visualization. f_{score} can give different results for a given visualization, depending on its input pairwise constraints. The most common structure users expect to see is the group structure in which similar objects (with the same labels) are placed close together to form groups. Besides, users can also collect different groups of similar objects according to their definition of similarity or according to the semantic of their data in order to create new custom pairwise constraints.

6.2.2 Derivation of the Constraint-Preserving Score

Given a set of user pairwise constraints, the proposed f_{score} measures how well the pairwise constraints are preserved in a particular embedding through two steps. First, the relationship between data points connected by each individual pair is quantified. This quantification for each type of similar/dissimilar link constraint is then transformed into a numerical measurement, and then finally combined to form the final score.

Constraint Measurement

We first define the *strength* of the input pairwise constraints in a given embedding. A similar link should have a high strength and a dissimilar link should have a low strength. The strength of a constraint can be measured as the inverse of the distance between two connected points. If a Student's t distribution is placed at the point $\mathbf{y}_i$ in the embedding, the strength of the constraint connecting $\mathbf{y}_i$

to another point $\mathbf{y}_j$ is defined as

$$q_{ij} = \frac{(1 + ||\mathbf{y}_i - \mathbf{y}_j||^2)^{-1}}{\sum_{k \neq l} (1 + ||\mathbf{y}_k - \mathbf{y}_l||^2)^{-1}}, \tag{6.2}$$

where the denominator is a normalization constant calculated from all pairs $\{(k, l)\}$ in the embedding. Since this quantification is normalized and is interpreted as a probability $\in [0, 1]$, the input embedding $\boldsymbol{Y}$ can be scaled without affecting the value of q_{ij}.

A similar formulation is used in t-SNE, LargeVis, and UMAP to model the neighborhood relationship in the embedding space. q_{ij} can be interpreted as the probability of $\mathbf{y}_i$ and $\mathbf{y}_j$ being neighbors in the embedding space. Therefore, for each similar link $(i, j) \in \mathcal{S}$, q_{ij} should be high. Inversely, q_{ij} is expected to be low for each dissimilar link $(i, j) \in \mathcal{D}$.

Constraint-Preserving Score

We propose to measure the preservation of all similar links $(i, j) \in \mathcal{S}$ in an embedding as the log-likelihood

$$f_{score}(\mathcal{S}) = \frac{1}{|\mathcal{S}|} \log \prod_{(i,j) \in \mathcal{S}} q_{ij} = \frac{1}{|\mathcal{S}|} \sum_{(i,j) \in \mathcal{S}} \log q_{ij}. \tag{6.3}$$

If all pairs of points connected by a similar link are close in the visualization, the log-likelihood $f_{score}(\mathcal{S})$ is high.

In contrast, the probability q_{ij} for each dissimilar link $(i, j) \in \mathcal{D}$ should be low. For all dissimilar links, we therefore propose to use the negative log-likelihood

$$f_{score}(\mathcal{D}) = -\frac{1}{|\mathcal{D}|} \log \prod_{(i,j) \in \mathcal{D}} q_{ij} = -\frac{1}{|\mathcal{D}|} \sum_{(i,j) \in \mathcal{D}} \log q_{ij}. \tag{6.4}$$

Another way to measure how well a dissimilar link (i, j) is preserved is to use $1 - q_{ij}$. Even though q_{ij} is interpreted as a probability of *being close*, $1 - q_{ij}$ is not an appropriate estimation for measuring the fact of *being distant*. Indeed, the value of q_{ij} in practice is very small due to the normalization constant in

the denominator in Equation 6.2. That makes $1 - q_{ij}$ close to one and makes the log-likelihood of all dissimilar links vanish[8].

As the scores for the similar and dissimilar constraints are normalized by the number of the corresponding constraints of each type, an equal contribution of these two scores is considered. The final constraint-preserving score is therefore the combination with equal contribution of both similar and dissimilar links

$$f_{score}(\mathcal{S}, \mathcal{D}) = \frac{1}{2} f_{score}(\mathcal{S}) + \frac{1}{2} f_{score}(\mathcal{D}). \tag{6.5}$$

$f_{score}(\mathcal{S}, \mathcal{D})$ is written as f_{score} for short. An embedding that retains as much as possible the pairwise constraint corresponds to a high f_{score}, which means that it has a good quality with respect to the encoded knowledge. We analyze here several properties of f_{score} that are considered as its advantages.

6.2.3 Advantages of f_{score}

First, f_{score} is simple and intuitive. Given a set of input pairwise constraints, we can visualize the strength and the score for each individual pair (as shown in the introductory Figure 6.1) according to Equations 6.3 and 6.4. In that way, it can be easy for users to identify which pairs that satisfy or violate the constraints. In contrast, $AUC[R_{NX}]$ measures the neighborhood preservation, which is not easy to visualize for end-users. Although for this score, we can plot the curve of $R_{NX}(K)$ values for different values of the neighborhood size K, however, analyzing this curve is not intuitive for end-users.

Second, f_{score} is computationally efficient and uses only the embedding in the LD space without access to the HD data. It has a computational complexity of $\mathcal{O}(N^2)$ since the quantification of strength for each individual constraint in Equation 6.2 requires a normalization over all N^2 pairs in the embedding $\boldsymbol{Y}$. The summation over all the input pairwise constraints can be efficiently vectorized via matrix slicing operations. In contrast, $AUC[R_{NX}]$ must access both the HD data and the visualization. This means that $AUC[R_{NX}]$ is not scalable for large datasets due to its complexity of $\mathcal{O}(p\ N^2\ log(N))$, where p is the number of dimensions of the HD data, which can be large. The BIC-based

[8] In a small/medium dataset of thousands of data points, the total number of pairs in the normalization constraint in the denominator of Equation 6.2 is a factor 10^6, which make the value of q_{ij} to be scaled down by a factor of 10^{-6}. Therefore, $1 - q_{ij} \approx 1$.

score, despite its simplicity, can only be used for t-SNE. For an embedding not generated by t-SNE, it requires to compute the KL loss of t-SNE with a complexity of $\mathcal{O}(p\ N^2)$.

Third, f_{score} is method-agnostic. The BIC-based score is designed and tested only for t-SNE. $AUC[R_{NX}]$ is also method-agnostic, however, the usage of ground truth information differs from f_{score}. Most machine learning evaluation metrics use class labels as ground truth information. Even though well-known evaluation metrics for unsupervised clustering methods (such as the entropy-based *V-Measure* score (Rosenberg and Hirschberg, 2007), or the normalized and adjusted mutual information scores (Vinh et al., 2010)) still use class labels. f_{score} does not use the class labels directly, but it can use the pairwise constraints generated from class labels if available. In that way, f_{score} can evaluate the visualization according to the semantic information encoded in the input constraints. $AUC[R_{NX}]$ introduces a pure unsupervised approach for visualization assessment, but it still demands HD data for calculating the sets of neighborhoods. It is designed to measure neighborhood preservation without considering the semantic information in the neighborhood. An extension of this score, called $AUC[G_{NN}]$ (de Bodt et al., 2019) filters out the set of neighbors to keep only the neighborhood of the same class and measures the gain in terms of KNN score. In this chapter, we only compare f_{score} with $AUC[R_{NX}]$ and the BIC-based score. The comparison to $AUC[G_{NN}]$ is left for future work[9].

6.3 Experimental Results with f_{score}

The above properties of f_{score} can be analyzed from its formulation in Equations 6.3, 6.4 and 6.5. In order to see how f_{score} works in practice with different kinds of data (text, image, or tabular data) and with the embeddings of different visualization methods, we do experiments with three modern and widely used visualization methods: t-SNE, LargeVis, and UMAP. t-SNE and LargeVis are both controlled by the perplexity hyperparameter. However, LargeVis is designed for large datasets, the impact of the perplexity is thus not significant when applied to medium-sized datasets used in our experiments. In contrast, t-SNE and UMAP are very sensitive to their hyperparameters. For that reason,

[9] At the moment we developed this work, we were not aware of $AUC[G_{NN}]$, which is a useful evaluation for unsupervised and (semi-)supervised DR methods in general.

in this section, we do experiments to show how f_{score} and other scores can capture the relationship between the hyperparameters and the visual quality of visualizations produced by t-SNE and UMAP. More extensive experiments will be performed in the next section to find out the characteristics of f_{score}.

6.3.1 Experimental Setup

This section briefly summarizes the experimented datasets, the process to create pairwise constraints, and the protocol to evaluate and compare the proposed f_{score} with other visualization quality metrics. In order to make sure that f_{score} can be useful in practice, in all our experiments, we evaluate f_{score} on six datasets of a wide variety, including image datasets of both simple grayscale images and color images, a text dataset, as well as a tabular dataset of real-world (processed) genetic data.

Experimental Datasets

DIGITS is a subset of 1797 handwritten digits of gray-scale 8x8 images (Kaynak, 1995). COIL20 contains 1494 gray-scale 32x32 images of 20 rotated objects (Nene et al., 1996). FASHION_1K contains 1000 gray-scale 28x28 images sampled from the Fashion-MNIST clothing dataset (Xiao et al., 2017). FASH_MOBI contains 1494 color images of the seven most numerous classes sampled from another real-world fashion product images dataset from Kaggle (Kaggle Open Datasets, 2019). The features are extracted with a pre-trained MobileNet (Howard et al., 2017), where the last fully connected layer is replaced by a global average pooling layer to obtain an output vector of 1280 dimensions (Lin et al., 2013). For these four image datasets, PCA is applied to keep 90% variance of the data. This speeds up the computation of pairwise distances and reduces the potential noise of outliers[10].

5NEWS contains 2957 text documents of 5 selected topics (*rec.autos, rec.sport.baseball, sci.space, sci.crypt* and *comp.sys.mac.hardware*) from the 20Newsgroups dataset. We use a traditional pipeline to process the text data. Documents are first converted to TF-IDF vectors that are then fed into a latent

[10]After applying PCA to keep 90% of variance that needs to correctly explain the original data, the number of dimensions for each dataset is 22 for DIGITS, 40 for COIL20, 72 for FASHION_1K and 268 for FASH_MOBI.

Dirichlet allocation model (Blei et al., 2003) to extract 15 hidden topics, which are the 15 features used by the DR methods.

The last real-world dataset is the NEURON_1K open dataset (10X Genomics, 2018) that contains 1301 brain cells from an E18 mouse. These cells have been processed and provided by 10X Genomics[11]. The processed data have 10 PCA features and 6 labels found by a graph-based clustering method.

Constraint Generation

The proposed constraint-preserving score requires a set of constraints in the form of similar and dissimilar links. As shown in Section 6.2, the pairwise constraints can be generated from groups of selected instances. Users can group the instances that they find similar to indicate that they should be connected by similar links. Similarly, instances in different groups indicate that they should be connected by dissimilar links. In order to objectively evaluate the proposed score, we use a standard setting in semi-supervised learning where only a small number of labels in the dataset are used. Pairwise constraints generated from labeled instances are used throughout our experiments as follows.

- First, for a dataset of C classes, a very small number k of labeled instances are randomly selected for each class.
- Then, a similar link is created for each possible pair of these k instances, leading to a set of similar links, containing $|\mathcal{S}| = C\ \dfrac{k(k-1)}{2}$ constraints[12].
- Finally, for each pair of classes, $\dfrac{k^2}{2}$ dissimilar links are created by considering all distinct pairs of instances that belong to two classes, leading to a set of dissimilar links, containing $|\mathcal{D}| = \dfrac{C(C-1)}{2}\ \dfrac{k^2}{2}$ constraints[13].

[11] 10X Genomics provides chromium single-cell gene expression solution and releases several public genetic datasets under the Creative Commons Attribution license. (https://support.10xgenomics.com/single-cell-gene-expression/datasets/3.0.0/neuron_1k_v3)

[12] There are $\dfrac{k(k-1)}{2}$ distinct similar links in each class, and there are C classes.

[13] There are $\dfrac{C(C-1)}{2}$ groups of two distinct classes. From each group of two distinct class, there are $\dfrac{k^2}{2}$ dissimilar links of distinct pairs connecting a point in one class to another point in the other class.

Computing Visualizations and Metrics

We compare f_{score} with the famous $AUC[R_{NX}]$ score (Lee and Verleysen, 2010) and the BIC-based score (Cao and Wang, 2017). In order to observe the evolution of f_{score} and other scores, we will calculate these scores for a large number of visualizations produced by t-SNE, LargeVis, and UMAP. A grid of hyperparameter values is created for each method. For t-SNE and LargeVis, a one-dimensional grid of perplexity values is sampled from a natural logarithmic scale from the range $[2,\ N/3]$. For UMAP, a two-dimensional grid is created with $n_neighbors \in [2,\ N/3]$ in natural logarithmic scale and $min_dist \in \{0.01, 0.025, 0.05, 0.075, 0.1, 0.25, 0.5, 0.75, 1.0\}$. For each combination of hyperparameters in the above grids, an embedding is calculated, and f_{score}, $AUC[R_{NX}]$ and the BIC-based score (if applicable) are computed.

6.3.2 Comparison of f_{score} and Other Qualities Metrics

f_{score} is compared with $AUC[R_{NX}]$ and the BIC-based score for evaluating t-SNE embeddings. f_{score} is also compared with $AUC[R_{NX}]$ for evaluating UMAP embeddings (the BIC-based score is not applicable for UMAP embeddings). f_{score} is used with $k = 10$ labeled instances for each class. More explanation of this choice is detailed in the next section. In brief, our experiments in Section 6.4 show that, for medium-size datasets with about a dozen of classes, with 10 or more labeled per class, we can obtain a stable and reliable f_{score}.

It should be noted that we do not try to compare f_{score} with other scores to find which score is the best. Each score can measure a particular aspect of a visualization. $AUC[R_{NX}]$ is built upon a solid theory of rank-based criteria that measures how well the neighborhood information is preserved. The BIC-based score is built upon the objective function of t-SNE combined with a criterion for model selection (the Bayesian information criterion), which regularizes the KL loss of t-SNE by a *not-so-large* perplexity. f_{score} measures how well the structures encoded in users' constraints are preserved. For that reason, choosing which metric(s) to use mainly depends on the need of users and the criteria we expect to preserve in the visualization. More than one metric can be helpful when these metrics measure different criteria. For example, f_{score} can be used as a complementary metric together with $AUC[R_{NX}]$ if users want to discover other structures in addition to the natural class structure in their data.

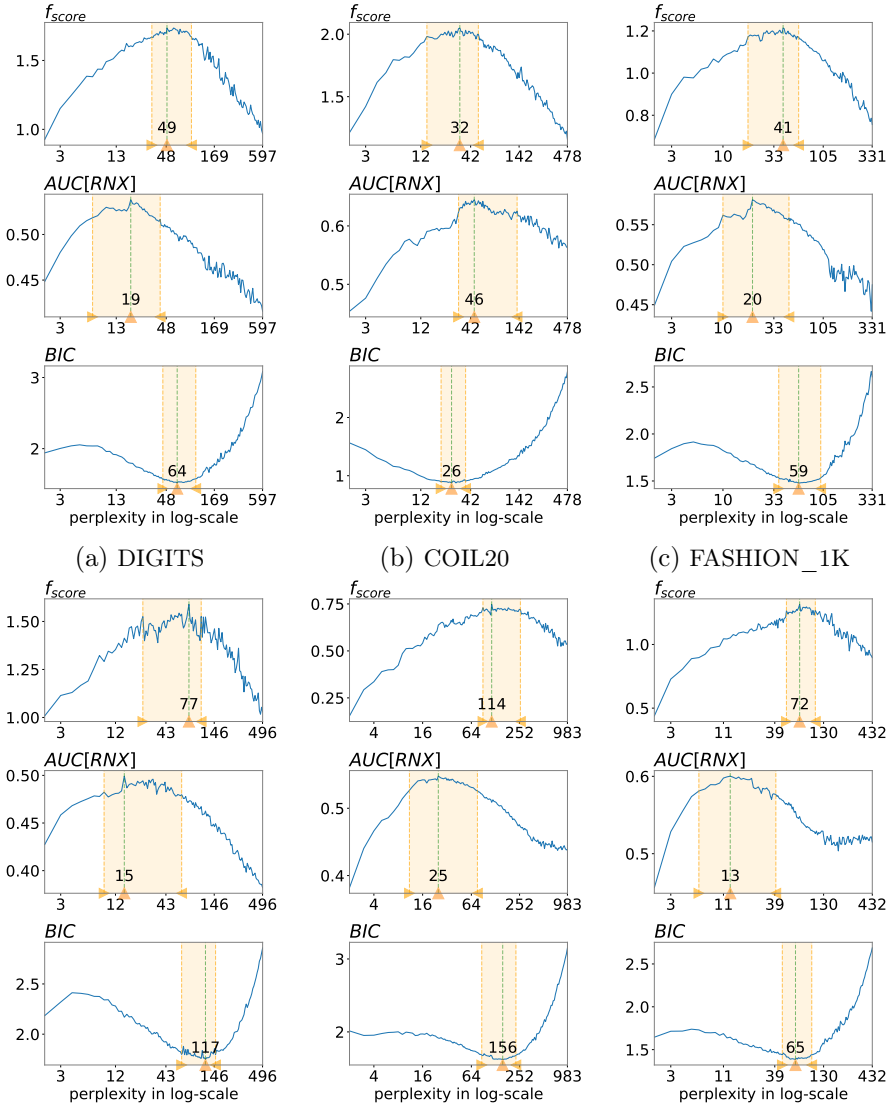
6.3.2.1 Comparison of f_{score} with $AUC[R_{NX}]$ and the BIC-based Score for t-SNE

Figure 6.3 shows that, for the six selected datasets, f_{score} agrees with $AUC[R_{NX}]$, the BIC-based score, or both of them. The agreement between these scores can be visually revealed through the overlap of the ranges of the top 5% scores (maximum values for f_{score} and $AUC[R_{NX}]$, minimum values for the BIC score).

In order to compare thoroughly the best solutions found by these scores, metamaps are used for visualizing the solution space of DR methods. Each point in the metamap is a t-SNE embedding corresponding to a perplexity value. Two points close to each other in the metamap correspond to perplexities that provide similar visualizations. The metamap can be constructed with any embedding method such as UMAP or t-SNE and is extremely useful in visual analytic tools like VisCoDer (Cutura et al., 2018) for discovering and comparing embeddings of different DR methods. In the case demonstrated in Figure 6.4, we have more than 100 visualizations for the NEURON_1K dataset. The metamaps are built using UMAP with large values of n_neighbors = 50 and min_dist = 0.5, which allow us to have a global view of all visualizations corresponding to different perplexities[14].

The four metamaps in this figure are colored by the values of perplexity, f_{score}, $AUC[R_{NX}]$, and the BIC-based score. The 5% of embeddings with the highest scores are highlighted. It can be seen that the three different scores can select visualization with different qualities. This is in line with Wattenberg et al. (2016), who state that we need more than one visualization to understand the hidden patterns in HD data. The visualizations in Figure 6.4 serve as a qualitative evaluation of the best visualizations found by the three scores. At the bottom of this figure, the same visualizations are shown without any information for supervision (i.e., no label for coloring the points). The contour in each plot shows the density estimation, which is calculated in the same way for all visualizations. In the visualizations promoted by f_{score} and the BIC-based score, several groups are correctly revealed. In contrast, in the visualization promoted by $AUC[R_{NX}]$, the whole embedding is considered as a single cluster, which makes it hard to recognize the different small groups.

[14]The hyperparameters of UMAP for constructing the metamap is manually selected with a single goal to obtain a global view of all created visualizations.



(a) DIGITS (b) COIL20 (c) FASHION_1K

(d) FASH_MOBI (e) 5NEWS (f) NEURON_1K

Figure 6.3: Comparison of f_{score}, $AUC[R_{NX}]$ and the BIC-based score for t-SNE embeddings. (b), (c): the ranges of the top 5% of maximum values (minimum values for the BIC score) overlap each other for the three scores. (d): f_{score} range mainly overlaps with $AUC[R_{NX}]$ score range. (a), (e) and (f): f_{score} ranges only overlap with the BIC-based score ranges.

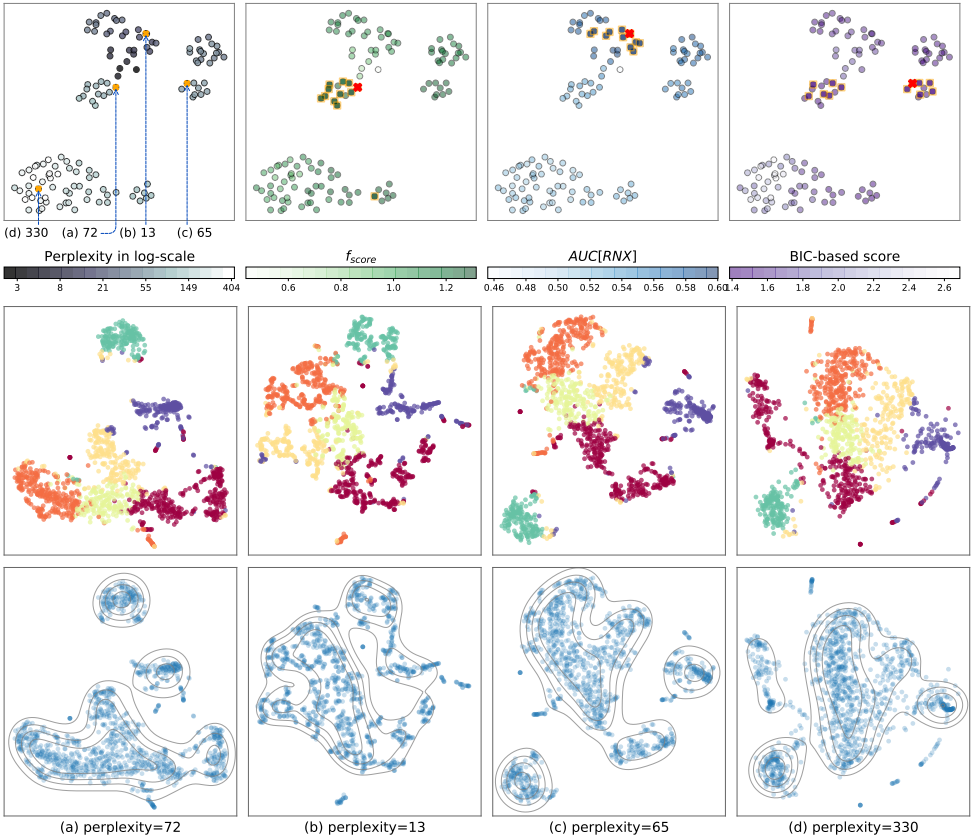


Figure 6.4: Metamaps and sample visualizations for NEURON_1K. The top 5% highest scores in the metamap according to each metric are highlighted on the top row. On the middle row, the visualizations are chosen using (a) f_{score}, (b) $AUC[R_{NX}]$, and (c) the BIC-based score. The last one (d) is not considered good by any of the scores. On the bottom row, the same visualizations are shown only with the contours calculated by a density estimation model, without any information for supervision. Visually, several groups are correctly revealed in (a) and (c) while the whole embedding in (b) is considered as a single cluster.

6.3.2.2 Comparison of f_{score} with $AUC[R_{NX}]$ for UMAP

Figure 6.5 shows the evolution of f_{score} and $AUC[R_{NX}]$ when the two hyperparameters n_neighbors and min_dist of UMAP are considered. For DIGITS, COIL20, and FASHION_1K, the evolution of f_{score} is clearer and smoother than the one of $AUC[R_{NX}]$. For NEURON_1K, the two scores discover dif-

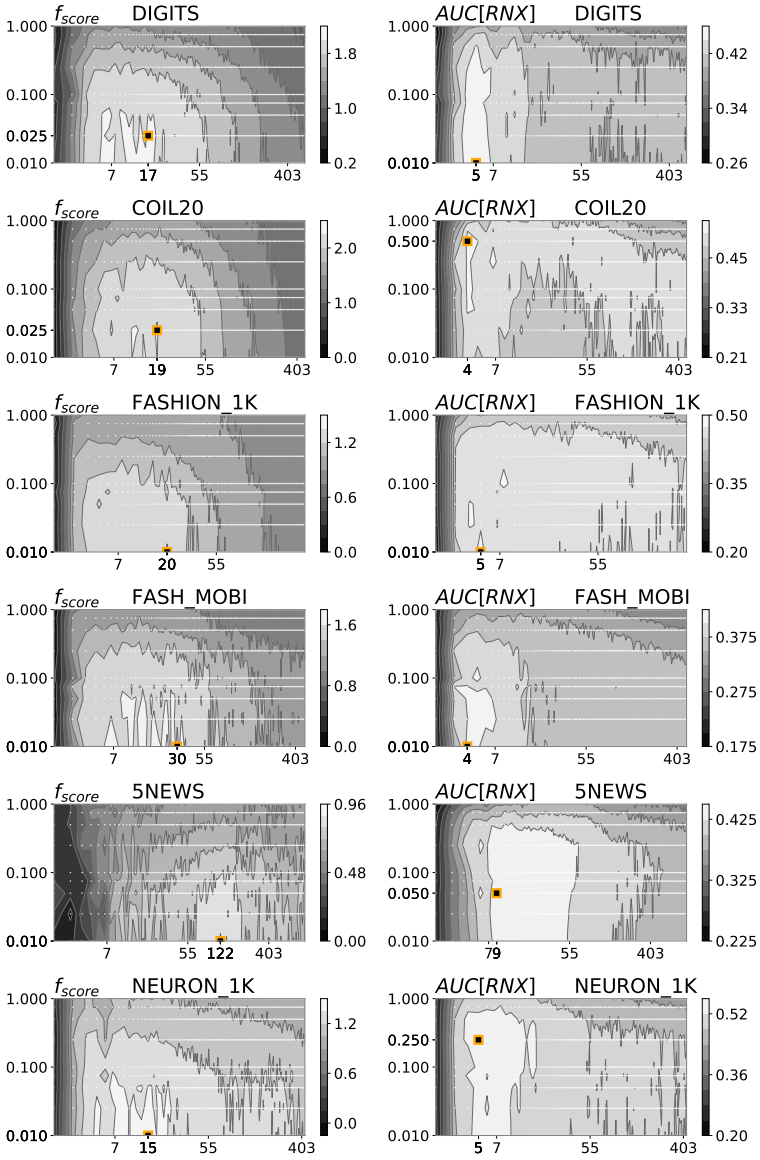


Figure 6.5: Comparison of f_{score} (on the left) and $AUC[R_{NX}]$ (on the right) for UMAP embeddings. The best combination of hyperparameters found by each score is located by the orange point in each dataset. In each plot, n_neighbors (on the horizontal axis) and min_dist (on the vertical axis) are shown in logarithmic scale. The light/dark region corresponds to the large/small values of the two scores.

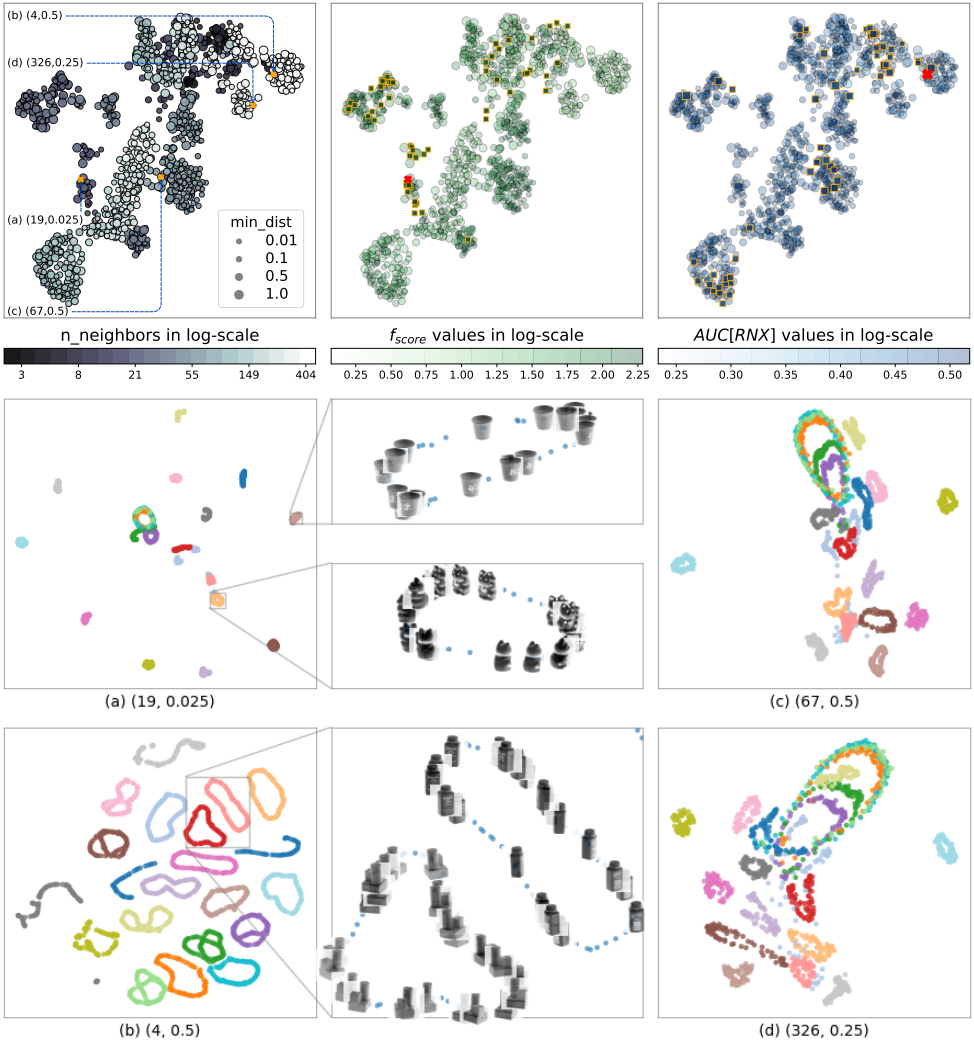


Figure 6.6: Metamaps and sample visualizations for COIL20. The top 5% highest scores in the metamap according to each metric are highlighted on the top row. On the bottom row, (a) is chosen by f_{score} and (b) is chosen by $AUC[R_{NX}]$. (c) is considered good by $AUC[R_{NX}]$ but not by f_{score}, and (d) is not considered good by any score. The detailed views when zooming in on several small groups in (a) are shown. The circle patterns are similar to the patterns in (b). However, the visualization in (a) reveals the global structure, while the one in (b) does not. When zooming in on several zones of the visualization in (b), objects in one group are closer to objects in other groups rather than to the ones in the same group.

ferent optimal regions. For FASH_MOBI and 5NEWS, $AUC[R_{NX}]$ reveals clearer regions of the best hyperparameters, but it mostly gives the same score for different min_dist while n_neighbors is fixed. In contrast, f_{score} discovers the influence of min_dist in conjunction with n_neighbors. The combination of these two hyperparameters is important for UMAP embeddings, since while n_neighbors controls local structures (the size of local neighborhoods), min_dist controls directly how tight the groups in the visualization are.

Figure 6.6 shows metamaps for UMAP embeddings and several selected visualizations for COIL20. In this case, we have more than 1000 visualizations of the COIL20 dataset corresponding to different combinations of n_neighbors and min_dist. The metamaps are built using UMAP with a large neighbor size (n_neighbors = 100, min_dist = 0.5) in order to obtain a global view of all visualizations. f_{score} considers the first visualization (a) as the best one. The next two visualizations are considered good by $AUC[R_{NX}]$, but not by f_{score}. In the second visualization (b), the groups clearly highlight the local structures but are not tight enough to reveal the global structures. In the third visualization (c), the groups are retracted and heavily overlap each other. This visualization has a high $AUC[R_{NX}]$ score since the neighborhood information is well preserved while the visualization is actually not clear. This same visualization is discouraged by f_{score}. The last visualization (d) belongs to the low score region in the metamap (with respect to both scores) with a too large n_neighbors and/or a too large min_dist.

As mentioned before, we do not conclude which score is better than the others since each score assesses the visualization by different aspects. Indeed, among all possible visualizations of a dataset, f_{score} and $AUC[R_{NX}]$ can encourage different visualizations. $AUC[R_{NX}]$ encourages the visualizations where the neighborhood is preserved. For instance, in the visualizations in Figure 6.6(b), local structures, like circle patterns, can clearly be identified. In contrast, f_{score} promotes visualizations where similar points are close to each other and dissimilar points are far from each other, according to the pairwise constraints. The visualization proposed by f_{score} can thus give a global view of the relative relation between small clusters in visualizations like the ones in Figure 6.6(a).

6.4 Empirical Characteristics of f_{score}

f_{score} is designed as a function of both evaluated visualizations and users' constraints. The properties of this score like the computational complexity can be analyzed theoretically from the mathematical formulation of the score. This section introduces the empirical characteristics of f_{score} that are found by analyzing the behavior of this score throughout experiments. Like other scores, we examine how f_{score} evolves with respect to the target hyperparameters of the target visualizations. Moreover, since f_{score} is also a function of users' constraints, it will be evaluated when the whole set of input constraints changes.

6.4.1 f_{score} as a Well-Behaved Function

As mentioned before, we generated a large number of visualizations with t-SNE, LargeVis, and UMAP with exhausted lists of theirs hyperparameters values. For t-SNE and LargeVis, f_{score} is evaluated as a function of perplexity ($f_{score} = f(perplexity)$). For UMAP, f_{score} is evaluated as a function of n_neighbors with the other hyperparameter min_dist is fixed to its recommended value ($f_{score} = f(n_neighbors,\ min_dist = 0.1)$). In order to correctly evaluate the evolution of f_{score} with respect to the neighborhood size of all three evaluated methods, the set of input constraints for each dataset should be fixed. For each dataset, $k = 10$ labeled instances per class are used for generating input pairwise constraints.

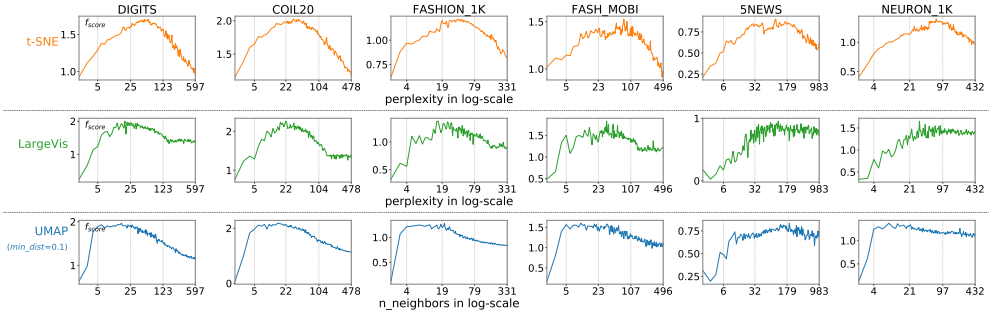


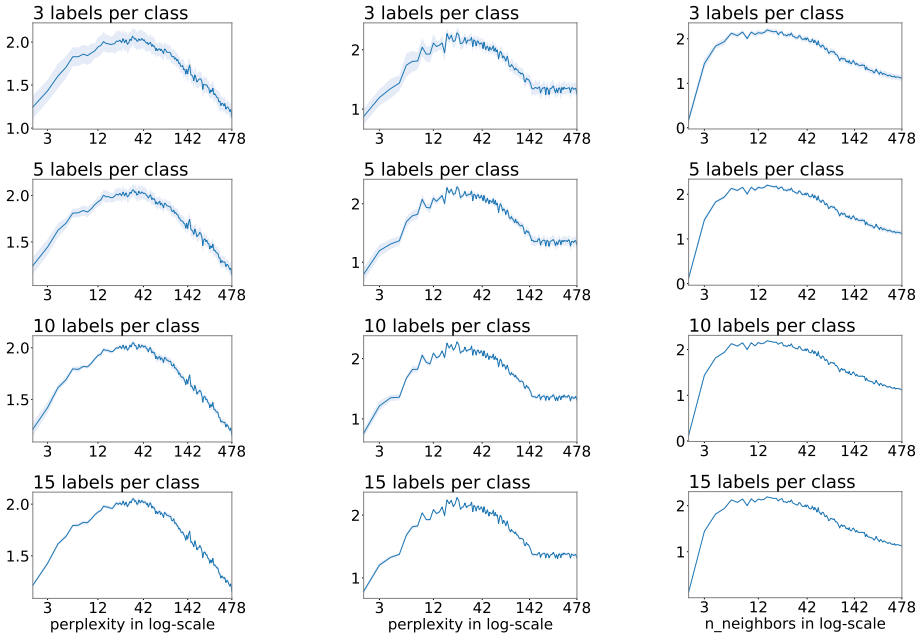
Figure 6.7: Evolution of f_{score} with respect to the hyperparameter of three DR methods for six datasets.

As shown in Figure 6.7, for each visualization method and for each dataset, f_{score} has the form of a convex-like function of perplexity or n_neighbors, i.e., a well-behaved function. We use the term *well-behaved* function to describe f_{score} because it is indeed not a random function, and we can observe a general rule, even vague, to describe the relationship between the hyperparameter value and the computed f_{score}. It increases as the number of neighbors (perplexity/n_neighbors) increases, then reaches its maximum value, and finally decreases when the number of neighbors becomes too large. This observation also holds true when evaluating f_{score} as a function of two parameters (n_neighbors and min_dist) for UMAP embeddings, as observed from previous experiments in Figure 6.5. As mentioned before, LargeVis is designed for large datasets and is not sensitive to the hyperparameter of neighbor size. Indeed, for a wide variety of experimented datasets of medium size, f_{score} does not change much for visualizations with the neighbor size larger than 100. In contrast, the impact of this hyperparameter for the visualizations of t-SNE and UMAP is significant and can be observed from the graphs of f_{score} in Figure 6.7.

We can also notice that f_{score} is not a smooth function. These graphs in fact only show a noisy estimation of possible values of f_{score} function for each neighbor size. The first reason is that the visualization corresponding to a scalar value of the neighborhood size is created by t-SNE, LargeVis, and UMAP with a random initialization. The second reason is that f_{score} is evaluated with a random set of pairwise constraints generated from a fixed number of labels per class. A smoother graph can be obtained if f_{score} is repeatedly evaluated many times on different visualizations (created with different random initializations) and on different sets of input constraints (with the same number of constraints).

6.4.2 Stability of f_{score}

In order to automatically create pairwise constraints that reflect the natural group structure in the dataset, we use a small number of labeled instances in each class and generate pairs between them. To investigate the number of constraints needed to obtain a reliable f_{score}, different values of the number of labeled instances per class are tested ($k = \{3, 5, 10, 15\}$). The sets of labeled instances are not accumulated, i.e., the set of 5 labels per class does not contain the set of 3 labels per class. For each value of k, f_{score} is repeatedly



(a) t-SNE (b) LargeVis (c) UMAP

Figure 6.8: Stability of f_{score} with the embeddings of (a) t-SNE, (b) LargeVis and (c) UMAP for the COIL20 dataset. The mean (blue line) and variance (the filled region around the line) are calculated for each perplexity/n_neighbors with a different number k of labeled instances per class (3, 5, 10, and 15).

evaluated 20 times, each time with different constraints generated from a set of randomly selected k instances per class. For the same value of k, different sets of generated constraints have different pairs, but these sets reflect the same structure of the data given by class labels. In other words, these sets have the same characteristics and only differ in the number of pairs.

We expect that, for a particular visualization, given different sets of input pairwise constraints with the same nature, f_{score} should give similar scores. Figure 6.8 shows the mean and variance of f_{score} for the embeddings of t-SNE, LargeVis and UMAP (with min_dist of 0.1) with the COIL20 dataset as an example. It can be seen from the figure that f_{score} functions as expected for the embeddings of all three methods. First, when the number k of labeled instances increases, f_{score}'s variance (shown by the filled shaded region around the mean in the plots) decreases. With $k = 10$ or larger, the variance is almost negligible,

showing that f_{score} is stable and reliable. This result is shown for COIL20, but also holds for the other datasets.

6.4.3 Flexibility of f_{score}

Through the previous experiment, f_{score} proved its stability when the number of input pairwise constraints changes while the sets of constraints have the same nature. In most cases, the constraints generated from class labels reflect naturally the class relationship between the instances. What if users want to assess other structures besides the class structure in the visualization? Traditional quality metrics can only assess the visualizations based on a fixed criterion. For instance, $AUC[R_{NX}]$ can only evaluate the neighborhood structure but not any other criterion. In contrast, f_{score} evaluates the visualizations based on its input constraints. These constraints reflect the structures that users want to see in the visualizations. This score is thus flexible, in the sense that the input constraints can be used to control how the visualization is assessed. In the following experiments with three real-world datasets, we demonstrate the flexibility of f_{score} by using this score to find the best visualization according to different criteria encoded in different sets of constraints.

For each dataset, two different sets of constraints are used. The first set is generated from $k = 10$ labeled instances from each class (called *natural classes* or *natural groups*) that reflects the natural class structure. The second set of constraints is created manually from custom groups (or *custom classes*) from the dataset. These custom groups are chosen based on the semantic of the data or by using another set of available labels for the dataset. In each custom group, $k = 10$ instances are randomly selected to generate pairwise constraints.

f_{score} will be used to find the best perplexity of t-SNE using two above sets of input constraints. Since we have computed many t-SNE visualizations with different values of perplexity in previous experiments, a simple linear search can give us the best visualization with the highest f_{score}. The next section presents another efficient way to find the best visualization assessed by f_{score}. For now, we assume that the task of finding the best visualization can be done efficiently.

With each dataset, the following tasks are performed, and the results are shown in four plots.

1. f_{score} is first used to find the best perplexity with the constraints of natural groups (f_{score} (*natural groups*)). The visualization corresponds to this perplexity is colored by the *natural classes* and shown on the top-left. The same visualization colored by the *custom classes* is shown on the bottom-left.

2. f_{score} is then used to find the best perplexity using the constraints of *custom groups* (f_{score} (*custom groups*)). The visualization corresponds to this perplexity is colored by the *natural classes* and shown on the top-right. The same visualization colored by the *custom classes* is shown on the bottom-right.

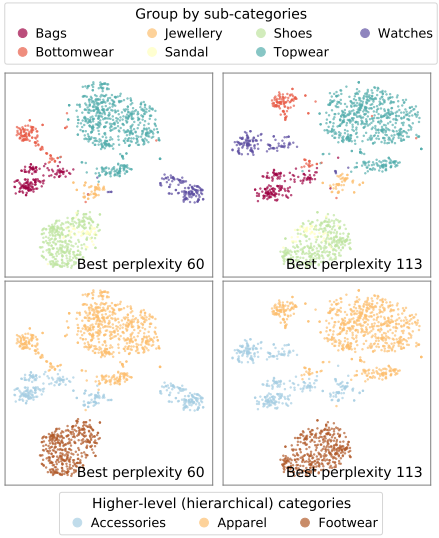
We will analyze these four plots for each dataset to see how f_{score} can be used in a flexible way to find the best visualization according to different sets of input constraints.

FASH_MOBI, a dataset of real-word color images

The first example considers the FASH_MOBI dataset with seven sub-categories: *Bags, Bottomwear, Jewellery, Sandal, Shoes, Topwear* and *Watches*.

The best visualization (perplexity = 60) shows seven detached sub-groups (the plot on the top-left). If users want to see more abstract and general groups, they can form higher-level groups such as

- *Accessories* as a group of {*Bag, Jewellery, Watches*},
- *Footwear* as a group of {*Sandal, Shoes*},
- *Apparel* as a group of {*Topwear, Bottomwear*}.



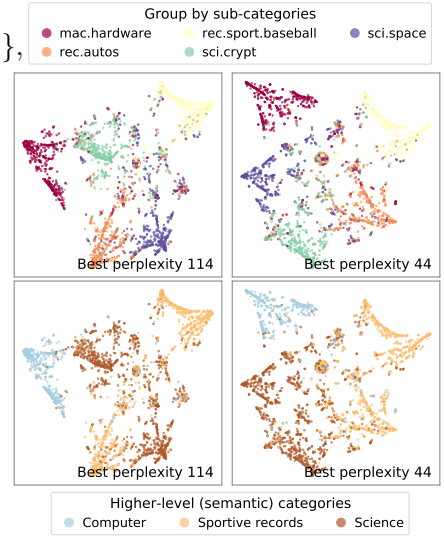
The previously chosen visualization did not reveal these three higher-level groups (the plot on the bottom-left). For

example, the *Watches* sub-group is placed far away from the *Jewellery* and *Bag* sub-groups, which do not highlight the structure of the *Accessories* group. Using the custom groups to compute f_{score} leads to a new best perplexity (113) that better reveals this structure (the plot on the bottom-right).

5NEWS, a subset of a text dataset

The second example focuses on semantic labels for the textual 5NEWS dataset. Three general topics can be created from the 5 original classes:

- sportive records group (*rec*) as a topic of {*rec.autos, rec.sport.baseball*},
- scientific group (*sci*) as a topic of {*sci.space, sci.crypt*},
- *comp.sys.mac.hardware* stays in its own group (*comp*).



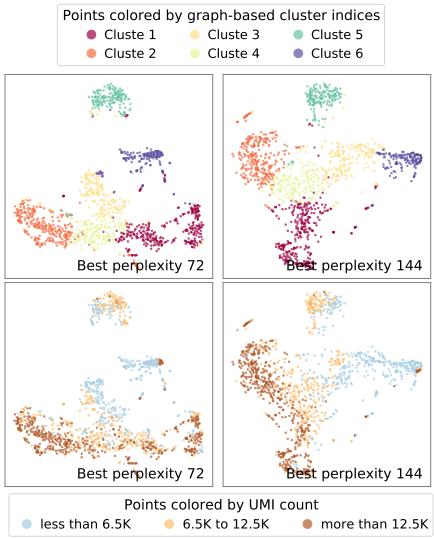
The problem of the visualization found with the constraints generated from the original class labels is that two sub-groups of the same topic can be placed far apart (the plot on the bottom-left). By using the new constraints generated from the three above semantic groups, f_{score} finds a better visualization in which elements in these semantic groups are placed close to each other (the plot on the bottom-right).

NEURON_1K, a dataset of processed gene expressions

The last example uses NEURON_1K. The original 1301 cells are grouped into 6 classes found by a graph-based clustering algorithm. These classes are characterized by the transcriptome profiles of individual cells (presented in the RNA sequences).

However, another important aspect to characterize individual cells is the count of the absolute number of molecules: the unique molecular identifier (UMI) (Kivioja et al., 2011). Therefore, the cells can be grouped into three new groups:

- the ones with less than 6.5K molecules,
- the ones having from 6.5K to 12.5K molecules,
- the ones with more than 12.5K molecules.



The plot on the bottom-left (found using the constraints of cluster indices) does not reveal the structure of *UMI count*. In contrast, the one on the bottom-right shows a trend of decrease of *UMI count* along the horizontal axis.

In summary, we have shown that f_{score} is designed to be used as a simple and intuitive quality score with several properties such as computational efficiency and method agnostic. Through experiments, we have also found that f_{score} is a well-behaved function, stable and flexible. This score can thus be used as an efficient measure for the task of tuning hyperparameters of DR methods, which is normally very costly due to the cost of computing the embedding and the cost of calculating the quality score.

6.5 Application of f_{score} for a Hyperparameter Tuning Problem

f_{score}, as other state-of-the-art quality metrics, can be used to measure the quality of a visualization and thus can be used to choose the best visualization corresponding to the best hyperparameter(s) of a method. The scope of this chapter is narrowed down to the hyperparameters that directly affect the appearance of the visualizations. This section presents how to apply f_{score} for automatically tuning these hyperparameters of t-SNE and UMAP.

Since the computation of an embedding with t-SNE and UMAP is expensive, generating all embeddings to choose the best one is absolutely not an efficient solution. We expect to have (i) a cheap and reliable metric, as well as (ii) an efficient optimization procedure to search through the solution space of all embeddings to find the best one with a minimum number of *evaluations*. An evaluation in this context is an actual calculation to obtain an embedding with a particular (set) of hyperparameter(s) and a calculation of its score. f_{score} is efficient and method agnostic. What is missing here is a more efficient optimization procedure than the naive linear or random search. We propose to combine f_{score} with *Bayesian Optimization*, the most used method for optimizing expensive-to-evaluate functions (Brochu et al., 2010).

6.5.1 Bayesian Optimization for Hyperparameter Tuning

Bayesian optimization (BayOpt) is a strategy for finding the extremum (minimum or maximum) of an objective function f with as few evaluations as possible (Močkus, 1975). The objective function can be any complex non-convex black-box function that does not have a closed-form expression, or that does not have an accessible derivative. Directly finding the extremum of this kind of function is therefore impossible. However, the function values, possibly noisy, can be observed for some sampled input values.

Our goal is to find the best perplexity for t-SNE and the best combination of n_neighbors and min_dist for UMAP while actually running t-SNE/UMAP as few as possible. This task could be trivial if we know the distribution of f_{score} for different (combination of) hyperparameter values, however, this kind of information is not available. It should be noted that we do not have any visualization generated beforehand like in the setting of the previous experiments. That means we start from scratch, with a given dataset and a given set of input constraints, our goal is to tune the hyperparameters for t-SNE or UMAP using f_{score} and BayOpt. We have to decide which combination of hyperparameters should be tried in order to find the best one. No relationship between f_{score} and the hyperparameters is available, but we have to model this relationship using a very limited resource (the number of evaluations we take). BayOpt can construct a statistical model describing this relationship between the tuned hyperparameters and the target function, in our case, the f_{score} function.

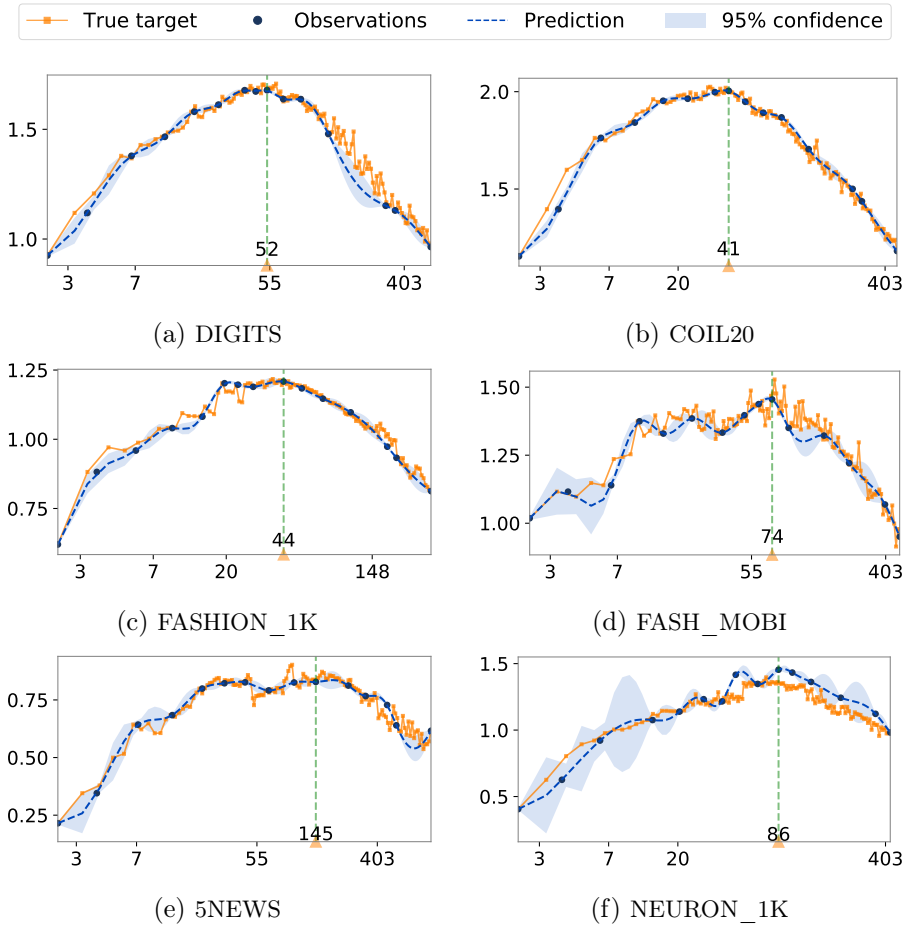
The goal of BayOpt here is not to approximate this unknown f_{score} function for all hyperparameter values, but instead to estimate its maximum from a set of observed input samples of hyperparameters and f_{score} values for the corresponding visualizations. Based on past observations, BayOpt will predict the most promising hyperparameters to evaluate. It is a hard problem since, at each step, we have to decide to create new visualization with hyperparameters close to the one with a high score (exploiting the potential solution space) or with different hyperparameters to discover/explore new zones in the solution space. In order to compromise this trade-off between exploration and exploitation, several strategies exist to guide the optimization process to discover the parameter space: *maximum probability of improvement*, *expected improvement*, and *lower or upper confidence bound* (Brochu et al., 2010).

6.5.2 Using f_{score} in Bayesian Optimization

Based on the nature of f_{score} discovered in the last section, the exploration strategy is chosen to explore the largest parameter space possible. The expected improvement (EI) acquisition function is thus a good choice for the surrogate function of BayOpt, as it maximizes the expected improvement over the current best parameters and has proven its efficiency in practice (Snoek et al., 2012). The parameter ξ of BayOpt controls the trade-off between global search (exploration) and local optimization (exploitation). Here, ξ is set to a large value (0.25) to stimulate exploration, which works well for all experimented datasets without any effort to tune this parameter. The reason for this combination is that f_{score} diverses enough through the space of hyperparameters (see Sections 6.3.2 and 6.4), an exploration strategy can help to make sure that every feasible zone in the solution space will be discovered. In contrast, if other scores like $AUC[R_{NX}]$ are used, we should try an exploitation strategy to focus only on a particular feasible zone of high score[15]. In addition, there is a small variance in f_{score}, BayOpt also takes it into account by adding small values to the diagonal of the kernel function of the underlying Gaussian process model.

[15] An exploitation strategy should be used for $AUC[R_{NX}]$ since this score function does not diverse well enough, and gives very similar scores for a wide range of hyperparameter values (see Figure 6.5). That makes large *flat-like* regions in the solution space, and we need to drill down onto these regions to find the best location.

6.5.2.1 Tuning One Hyperparameter for t-SNE



(a) DIGITS (b) COIL20

(c) FASHION_1K (d) FASH_MOBI

(e) 5NEWS (f) NEURON_1K

Figure 6.9: Tuning t-SNE's perplexity for six datasets using BayOpt. f_{score} is evaluated only for the embeddings of 15 selected perplexities shown by the dark blue points. The dotted blue line presents the predicted f_{score} for all other perplexities. The filled blue region represents the uncertainty of the prediction. The green vertical line indicates the best predicted perplexity. The orange lines are the true values of f_{score}, only used as references to see how well the BayOpt prediction approximates the true targets.

Figure 6.9 demonstrates how BayOpt works for tuning t-SNE perplexity for all six selected datasets. The *true target* is the score values for each perplexity and is used only as a reference to compare with the estimate score values predicted by BayOpt. Remarkably, f_{score} needs to be evaluated for

only 15 selected perplexities. These perplexity values are selected by BayOpt iteratively, starting with five random perplexities. The pairs of perplexity and the corresponding f_{score} are used to update the BayOpt model at each iteration. The next predicted perplexity to evaluate is the most promising perplexity value that does not decrease f_{score}. It should be noted that BayOpt does not explicitly approximate the score function, but it tries to find the maximum value instead. BayOpt does not only find the best hyperparameter values but also indicates the region in which it is not certain about its prediction, which is usually the region of too high or too low perplexity values.

6.5.2.2 Tuning Two Hyperparameters for UMAP

Tuning hyperparameters for UMAP is a more difficult task since its two-dimensional hyperparameter grid is larger than the one-dimensional grid of t-SNE. Instead of evaluating thousands of combinations of values for two hyperparameters, BayOpt converges after only 50 iterations for all six experimental datasets. Figure 6.10 shows how BayOpt finds the region with the best combinations for the six datasets. The uncertainty of BayOpt prediction is not shown in this plot. In comparison with the full grid used for f_{score} in Figure 6.5, BayOpt approximates the region of the highest score more efficiently with a very limited number of evaluations.

In practice, BayOpt is used to tune multiple hyperparameters. Contour plots of every pair of hyperparameters are used to investigate the region with the best combinations. One advantage of the BayOpt approach is that it does not only maximize the target score function but also gives predicted scores for all hyperparameter combinations. Indeed, in each plot in Figure 6.10, only 50 points are exactly evaluated. The contour is calculated upon the predicted value of the underlying Gaussian process model for all other points. Without spending too much resource to obtain a full grid, the estimated score given by BayOpt is reliable enough to point out the best hyperparameters.

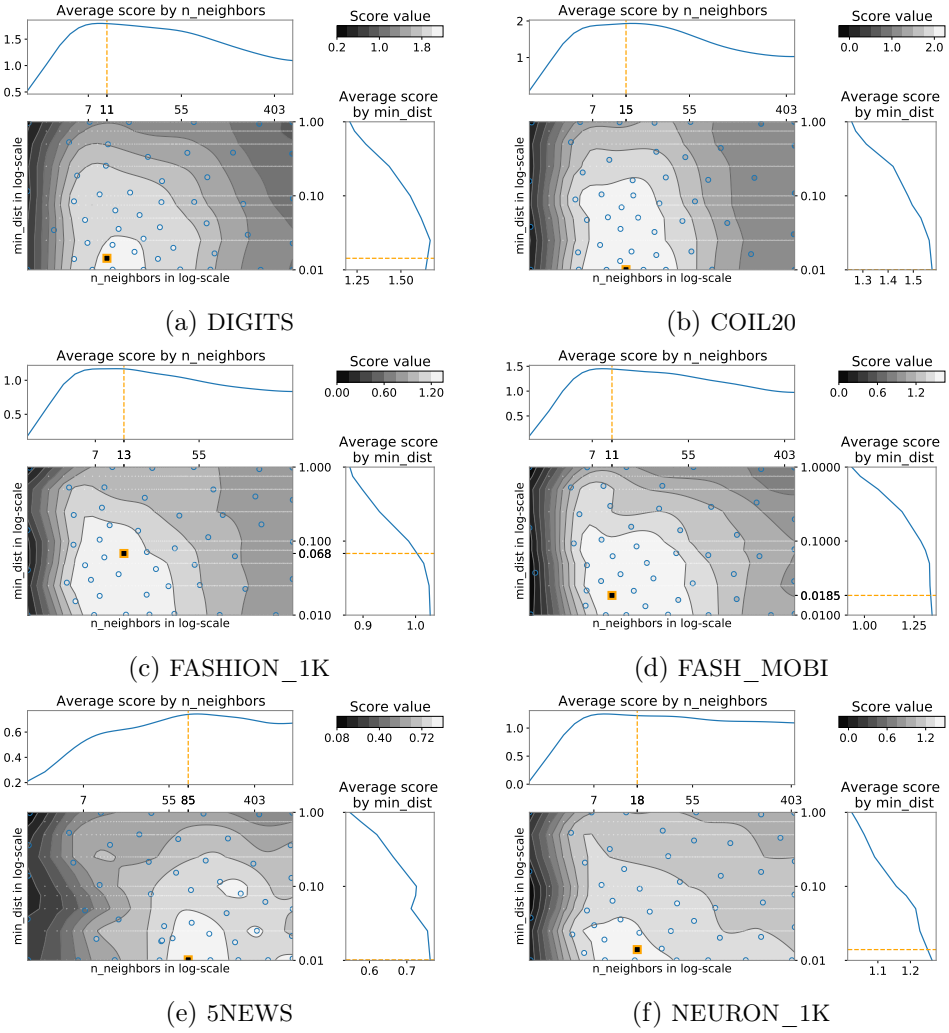


Figure 6.10: Tuning two hyperparameters of UMAP using BayOpt. In each plot, 50 points (combinations of n_neighbors and min_dist) are evaluated and shown by the blue dots. The contour plots are constructed from the predicted f_{score} for all other points in the grid. The light/dark region corresponds to the large/small values of f_{score}. The orange points indicate the best predicted hyperparameters.

6.6 Discussion

This chapter introduces another aspect of combining users' constraints with DR methods. Instead of adapting DR methods by the constraints, pairwise constraints between similar and dissimilar objects are used to assess visualizations of SNE-based methods. A new constraint-based score is introduced to measure the quality of visualizations by evaluating how well the semantic information encoded in input pairwise constraints is preserved in the visualization. f_{score} does not require calculating neighborhood information in the HD space or an expensive objective function of a non-linear DR method. Furthermore, it is complementary to other quality metrics, while being flexible (as the score can change with respect to the user's input constraints) and cheaper to compute. Based on this score, we propose to use Bayesian optimization to efficiently find the best hyperparameters instead of traditional search-based methods. The proposed workflow facilitates the use of DR methods by making the choice of difficult-to-understand hyperparameters easier and helps users to discover different visualizations with various perspectives on the structure of data. f_{score} can have a potential impact since it is very easy to use and works with any visualization method. Domain experts can express their knowledge in a simple form of similar or dissimilar groups of points. If needed, end-users can use labeled data (which are usually available for coloring groups in the visualization), even in a small amount.

Constraint-based clustering selection (COBS) (Van Craenendonck and Blockeel, 2017) is a closely related work to our approach. This work proposes a simple way to select a clustering method and its hyperparameters for small artificial tabular datasets. 931 clustering solutions are generated for each dataset, including 180 clusterings using K-means, 351 using spectral clustering, and 400 using DBSCAN. Two sets of similar and dissimilar link constraints are generated from labels. COBS counts the number of constraints that are preserved (points connected by similar pairs stay in the same cluster, points connected by dissimilar pairs are in different clusters) and rank these counts to find the best clustering. Our approach does not require greedily enumerating all possible solutions and quantifies the strength of each pair instead of simply counting the satisfying pairs.

Limitations

f_{score} is introduced with several good properties and empirical characteristics that help users to facilitate the choice of hyperparameters for DR methods. However, it also has several limitations, in both the usage of this score and also the evaluation that we perform in this chapter.

Limitation of f_{score}

f_{score} is a function of input pairwise constraints, which makes this score flexible. However, this is also inconvenient since f_{score} depends on the constraints, and the quality of the measure also depends on the quality of the input constraints. When the pairwise constraints are not available, f_{score} is useless. Several solutions exist to address this issue. Users can observe their data or a subset of the data and manually construct the pairs. Users can also identify several small sets of similar objects in their opinion to generate similar/dissimilar pairs. It can be easy for image datasets, but it is more difficult for other kinds of datasets that are not easy to interpret visually. Our interactive tools for collecting users' constraints will be discussed in Chapter 7. It should be noted that we do not need the ground truth class labels, but only need groups of similar objects to form the pairwise constraints.

Limitation in the Evaluations

Visualization experts can ask why don't we use the metrics in the information visualization domain to assess the embeddings. For example, several *scagnostics measures* (Dang and Wilkinson, 2014; Wilkinson et al., 2005) can be used for assessing scatter plots. However, based on the study "The art of using t-SNE for single-cell transcriptomics" (Kobak and Berens, 2019) and the similar work for UMAP (Becht et al., 2019), we learn that the chosen quality metrics ($AUC[R_{NX}]$ or our proposed score) are more suitable for assessing embeddings of medium/large datasets. A brilliant work of our college Morariu et al. (Under review) tackles this aspect by proposing a new approach for assessing visualization with a combination of human judgments, interpretability measures, and accuracy measures. Another missing in our evaluation and in this thesis is a lack of user-base experiments. This point will be discussed in the next chapter.

Chapter 7

Discussion

This section is devoted to discussing our approach of constraint integration for DR methods. Other issues related to constraint acquisition, visualization assessment, and interactive tools for user interaction are also discussed.

7.1 Discussion on Our Approach

We tackle the constraint integration in DR to address the mismatch between the users' needs and the visualization results. Several reviewed methods also promote injecting users' subjective constraints to enrich base DR methods. However, one should be aware of an opposite direction that removes the subjectivity from the embedding of DR methods. Conditional t-SNE (Kang et al., 2020) is designed to remove "known/prior" information out of the t-SNE embedding. The revealed structure in the visualization can sometimes be obvious and not interesting since it does not bring much new information about the data. For instance, a visualization of the MNIST dataset containing 10 classes of hand-written digits can show 10 separate clusters. This embedding correctly represents the input data but does not bring new views or insights. In this case, the embeddings that capture complementary unknown structures (not the obvious class structure) to provide new insights about the HD data may be desired. From this perspective, we can consider a new direction of using users' constraints to guide DR methods to find new and *surprising* insights from data.

Another interesting perspective that we found through this thesis is a uni-

fying perspective. In our survey in Chapter 3, we reviewed a wide variety of discrete methods and tried to generalize them in several general formulations. We suggest integrating constraints into a generalized method that unifies a family of DR methods. This unifying perspective has been proposed for different DR methods. Linear methods like PCA, MDS, LDA, CCA, and LPP are generalized under a constrained matrix optimization framework (Cunningham and Ghahramani, 2015). Nonlinear and spectral methods like Laplacian Eigenmaps, LLE, and Kernel PCA are generalized in the maximum entropy unfolding (MEU) framework (Lawrence, 2012). Distance-preserving (MDS-based) and neighborhood-preserving (SNE-based) DR methods are also generalized under the minimum-distortion embedding (MDE) formulation (Agrawal et al., 2021). Local neighbor embeddings methods like Laplacian Eigenmaps, t-SNE, UMAP are also generalized by a recent *attraction-repulsion spectrum* formulation (Böhm et al., 2020). By studying these unifying methods, we can better understand the relationship between common DR methods, their missing characteristics, and thus can find out how constraints can fit into these methods.

7.2 Discussion on Our Limitations

The limitations of our proposed methods have been discussed in the corresponding chapters. Here we discuss three limitations of our approach, including the assumption when designing constrained methods, our evaluation, and applications of our proposed methods.

Our first limitation relates to the assumption about input constraints. It is generally assumed that a sufficient set of high-quality and consistent constraints is provided. The input constraints are assumed to be available in a usable form or can be easily collected from users. However, this assumption does not always hold true for real interactive scenarios. We discuss more this issue in Section 7.3 and also make the effort to build interactive tools for constraint acquisition for our interactive methods.

Our second limitation is a lack of user evaluation. Through this thesis, interactive DR methods and a user-based score are proposed, but none of them is fully evaluated with real users. The author of the proposed methods does the interactive experiments/scenarios by himself. However, the interaction aspect is also considered, for which interactive prototypes are built for each of our

methods (see Section 7.5). Bibal, Dumas, and Frénay (2019) suggest user-based experiment guidelines for measuring interpretability in ML. Following these guidelines, we have to define beforehand who are the users of our method, what is the goal of the evaluation, and what are the metrics for evaluation. We could study these guidelines to adapt them for evaluating constrained DR methods in future work. We are aware of this limitation and also try to address the user aspect in our work. We identify targeted users for each of our methods, use task-based scenarios to guide users to use these methods to achieve a goal, and combine both qualitative and quantitative evaluations to assess the embeddings. More discussion about the visualization assessment with users' constraints can be found in Section 7.4.

The third limitation in this thesis is a lack of useful applications with real-world data. We claimed that users' feedback is extremely useful for discovering patterns in exploratory data analysis. However, we do not have many examples where real users can apply our methods for their data to get new insights. Several real-world datasets (the Fashion product image dataset and a pre-processed single-cell gene expression dataset) are used in our work, but more experiments with datasets from different domains could be more useful.

7.3 Discussion on Constraint Acquisition

In constrained DR methods, the focus is put on the parametric representation of the constraints and the algorithm. We want to discuss and draw attention to the constraint acquisition step.

The first source of constraints comes from feedbacks or domain knowledge of the users. Sacha et al. (2017b) review different interactive scenarios for collecting users' feedback in visual analytic. Constraints can also be generated from supervision information such as class labels. Constraints can arise in the problem setting as well. For example, if the features of a dataset can be grouped by their nature (such as demographic and economic features for a dataset of country indicators), they can be used as constraints in multi-view methods.

When constraints have to be manually collected from users, it can be difficult to gather a large set of them. However, they can be enriched by constraint *propagation and pruning* techniques (Cevikalp et al., 2008; Davidson, 2012; Guo

et al., 2016; Li et al., 2008). Similarly, if the constraints are partial class labels in a semi-supervised setting, traditional label propagation methods can be applied, such as a KNN-based approach (Zhang and Zhou, 2007), graph-based class discovery (Nie et al., 2009) or random walk algorithms to assign virtual labels for unlabeled data (Nie et al., 2011). Several techniques have recently been proposed to label large datasets with very little initial labeled data called weak supervision (Ratner et al., 2017, 2019).

Another efficient solution to enrich the constraint set is to use an *active learning* approach. Active learning algorithms can help to identify the data points that are going to be annotated according to different prioritization strategies (Settles, 2009). Active constraint selection has been successfully used to select representative pairwise constraints to improve the performance of semi-supervised clustering methods (Mallapragada et al., 2008; Xiong et al., 2013). This approach is also applied to DR for selecting informative triplets (Tamuz et al., 2011) or creating pairwise constraints with high diversity (Wang et al., 2017). However, there are very few methods in this topic, which can be a potential direction for future works.

7.4 Discussion on Visualization Assessment

Qualitative measures like *scagnostics measures* (Wilkinson et al., 2005) have been discussed before (in Section 6.6). We discuss briefly in this section quantitative measures involving users' preferences, particularly constrained DR methods.

When constrained DR methods are used for extracting features for a subsequent classification or clustering task, the quality of the embeddings is usually assessed by the performance of the corresponding targeted tasks. Evaluating the quality of constrained DR methods for visualization is a more difficult task. For these methods, we usually focus on measuring the quality of the visualization. The rank-based score (measuring how well the neighborhoods in the HD space are preserved in the LD space) is the state-of-the-art quality metric for SNE-based methods (Lee and Verleysen, 2009, 2010). Some other quality metrics like the distance consistency measure (DSC) (Sips et al., 2009) have proven to correspond to what users like in visualizations (Sedlmair and Aupetit, 2015). If multiple aspects of the visualization should be taken into

account, one can combine several of these quality metrics (Bibal and Frénay, 2016, 2019; Morariu et al., Under review).

However, with both types of constrained DR methods (for feature extraction and visualization), it is also important to consider assessing the preservation of constraints. Few constrained DR methods follow this aspect. One simple approach to assess constraint preservation is to count the number of unsatisfied constraints to select the best clustering result (Van Craenendonck and Blockeel, 2017). Our proposed f_{score} can also quantitatively measure the preservation of pairwise constraints in the visualization (Vu et al., 2021a). Another way to evaluate constraint preservation is through user experiments. The idea is to gather experts of the domain and ask them how well they grasp information in a visualization of a constrained method, in comparison to the one of an unconstrained method. This kind of evaluation can provide a good approximation of the quality of a constrained visualization but is difficult to set up. Other strategies can be used to indirectly assess the quality of the constrained visualization. For instance, an analysis of the low-dimensional space distortions (Aupetit, 2007; Nonato and Aupetit, 2018) can be performed to check if the constrained visualization is more faithful than an unconstrained one.

7.5 Discussion on Interactive Tools

For each method presented in this thesis, a simple prototype is created to collect users' constraints. Figure 7.1 shows a web-base prototype for the iPPCA and iPMDS methods in Chapter 5. The version of this tool on the left targets end-users, who only want to observe the visualization result for their datasets and interact with them. Users can simply select one or multiple objects in the visualization, move them to the desired positions. The system takes this feedback and replaces the old visualization with a new one. A more complicated version of this tool on the right targets expert users. More options to control hyperparameters and examine the optimization process of iPMDS and iPPCA are available, as well as a view with *position tracing* to help visualize the change of the embedding after users' interaction.

HC*t*-SNE method in Chapter 4 requires input constraints of a desired hierarchical structure. We have not built an interactive graphical interface for this method. However, expert users can still interact with our method via an

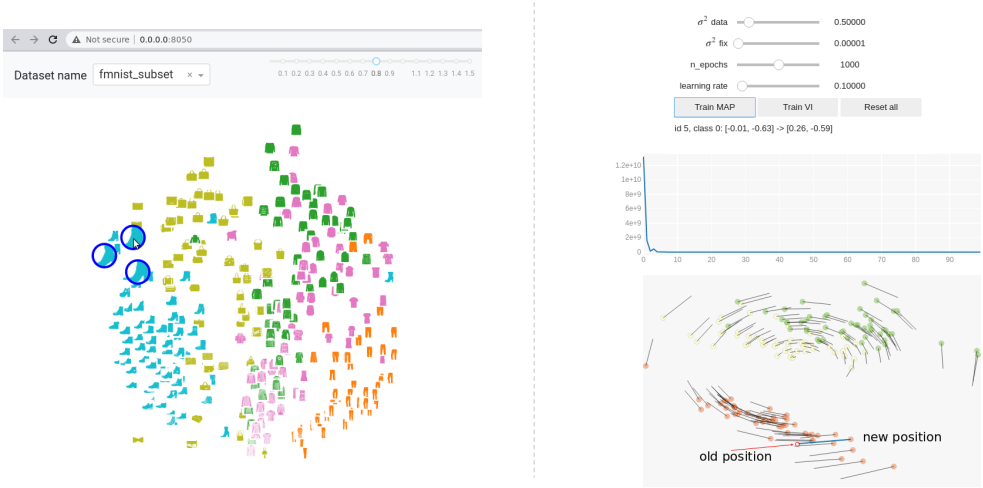


Figure 7.1: A web-based prototype of an interactive tool that allows users to select and move points in the visualization using the proposed iPPCA and iPMDS methods in Chapter 5. On the left is a simple version of this tool that targets end-users who only want to interact with the visualization. On the right is a more complicated version that targets expert users who want to observe the learning process, as well as fully control hyperparameters of the proposed methods.

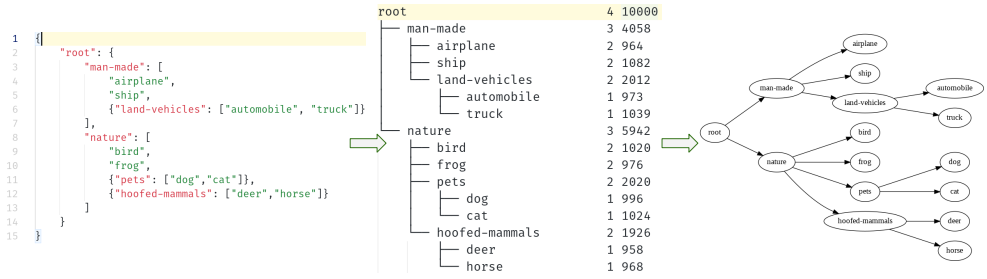


Figure 7.2: A fully automated workflow for generating hierarchical constraints for HCt-SNE in Chapter 4. End-users or expert users can enter their expected structure a text file with friendly JSON format.

automatic pipeline shown in Figure 7.2. Users input their hierarchy in a specific text file in JSON format. When they save this file, the system constructs the corresponding tree and call HCt-SNE automatically. The result visualization is produced in a separate file.

The final tool for f_{score} method in Chapter 6 is shown in Figure 7.3. This tool targets end-users, who want to measure the quality of a visualization using f_{score}.

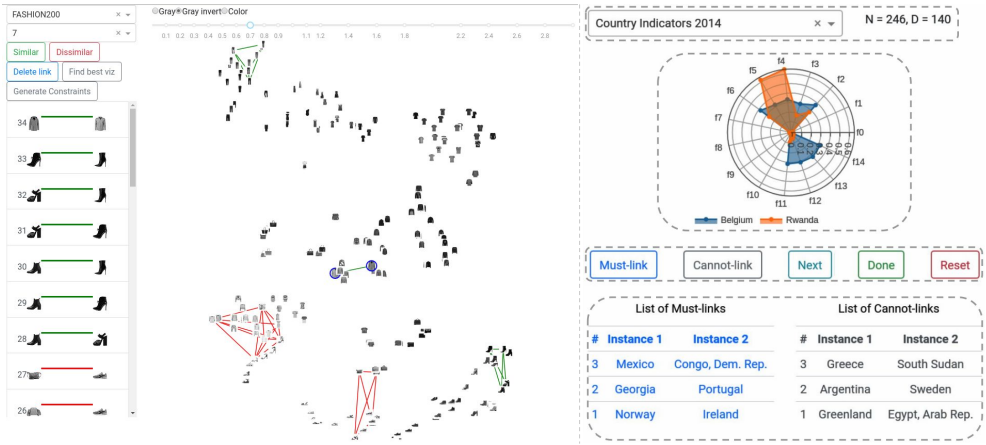


Figure 7.3: A web-based prototype of an interactive tool that allows users to create pairwise constraints used for f_{score} in Chapter 6. This tool can also visualize the calculated score for each pair. The version on the left works with image datasets while the version on the right is for tabular datasets.

Users can first select points or groups of points in the visualization and indicate that they are similar or dissimilar. A list of corresponding similar/dissimilar constraints is generated and shown on the left panel. The version of this tool on the left only works with image datasets. A modified version for tabular data (on the right) allows users to compare visually the features of any two randomly picked data points via a radar chart. After having a set of pairwise constraints, each individual pair can be highlighted on the visualization with its corresponding score (optional).

These tools presented in this section are only used in the development of the proposed methods. They are not designed and tested with real users, which is our limitation. However, these tools can make the constraint collection process easier and help analyze the output results visually.

In summary, we have presented novel constrained DR methods for visualization and a set of interactive tools for demonstrating different usages of our methods. From a viewpoint of end-users, when they face a visualization without knowing the underlying DR methods, how can they know which type of constraints and what constrained methods they can use? Let us assume that the users can access all types of constraints. Choosing the right constrained method for a specific dataset is totally based on the goal of users. For example,

if the orientation of the visualization is not correct or the meaning of the axes is not clear, they can use our proposed methods in Chapter 5 with fixed-position constraints to realign the visualization or to create understandable axes. If they have any idea about the semantic information inherited from the dataset, they can explicitly describe this kind of information using hierarchical constraints with HCt-SNE method in Chapter 4. When a set of similar/dissimilar pairs are available, users can use them to evaluate the quality of the visualization with f_{score} in Chapter 6.

Chapter 8

Conclusion and Future Work

This section summarizes the main results of the thesis guided by the research questions defined in the introduction and suggests a future research direction according to our perspective. This thesis aims to combine users' feedback and DR methods for visualization. From a machine learning viewpoint, we consider users' feedback as constraints. Every single result in this thesis is kept coherent thanks to our most important research question on the representation of the constraints. We focus on transforming users' constraints into adequate terms that can be used to enrich base DR methods or to assess the visualization.

Main Results of the Thesis

First, a comprehensive survey on the constraint integration for DR is conducted. With a focus on different kinds of constraints used in DR methods, we propose a categorization that covers both interactive visualization methods and traditional constrained DR methods. Less experienced readers can follow this categorization to find suitable constrained DR methods that work with their particular type of constraints. Two other perspectives are also suggested for more experienced readers. From a perspective of characteristics of constrained methods, readers can choose appropriate methods that preserve desired criteria in their data. From a *problem-solution* perspective, this survey helps to identify frequently encountered issues of common DR methods and potential solutions thanks to the additional constraints.

Second, a long-standing problem of t-SNE is tackled. t-SNE preserves neighborhoods but is not designed to preserve the global structure. Local structures in the form of separated groups can be revealed in t-SNE embeddings while the relative distances between these groups may mean nothing. Our idea is to use the semantic relationship between these groups to enhance the global structure. This kind of information can be represented by a hierarchical tree and can be injected directly into the visualization via our proposed method HCt-SNE. The representation of hierarchical constraints as a differentiable regularization term can also be applied to other SNE-based methods.

Third, we fill a gap in the literature of constrained DR with a novel unified probabilistic DR framework. Our idea is based on an analogy between users' prior knowledge and a prior distribution of a probabilistic model. When users interact with a visualization, they can move points to desired positions. This kind of feedback is considered as *fixed-position* constraints and used to construct informative prior distributions in our probabilistic framework. As a result, two derived methods called iPPCA and iPMDS can be used in an interactive context to allow users to manipulate the visualization. With users' feedback, these methods can create understandable axes for a visualization.

Fourth and finally, we present a novel way to use constraints for visualization assessment via the proposed constraint-preserving score. Visualizations are made for humans and are usually evaluated qualitatively and subjectively. State-of-the-art quality metrics can be costly to compute or not efficient enough to capture different aspects of a visualization. We let users encode their needs about the visualization in the form of pairwise constraints between selected similar/dissimilar points. By measuring how well the users' pairwise constraints are preserved in the visualization, we can evaluate the quality of a visualization according to the opinion of users. This score is computationally efficient and is empirically proved to be a well-behaved function, flexible, and reliable.

Short-term Future Work

As discussed in the previous chapter, the biggest shortcoming of this thesis is the lack of a user-based experiment. Improving in this direction can help us to improve the usage of our methods for interactive applications. For short-term future work, we can integrate pairwise constraints into widely used visualization methods like PCA and t-SNE. PCA and t-SNE are not designed to preserve distances explicitly, using pairwise constraints can enrich these methods by forcing them to preserve distances between similar/dissimilar points. For example, PCA preserves global information such as the variance in the data, however, separated groups are rarely seen in its embedding. Using pairwise constraints, we may enforce similar and dissimilar pairs to enhance group patterns in the visualization. Towards this idea, we have a preliminary result of integrating pairwise constraints into the Gram matrix used for PCA and the affinity matrix in t-SNE (Appendix 9.5).

Long-term Future Work towards a General Constrained DR Framework

We also need to look further to find a potential perspective for future research. In the same topic of combining constraints and DR methods, we suggest a general constrained DR framework in which we can combine several kinds of constraints with different base DR methods in the same unified framework.

The suggested constraint DR framework is inspired from (i) the minimum-distortion embedding (MDE) formulation (Agrawal et al., 2021) and (ii) the contrastive learning paradigm (Chen et al., 2020; Hjelm et al., 2019; Oord et al., 2018). The MDE formulation helps to unify common DR methods in a simple and general formulation. The contrastive paradigm helps to create an efficient constraint representation.

On the one hand, the MDE framework includes a wide variety of DR methods, from classical PCA or MDS to modern neighbor embedding methods like t-SNE and UMAP. The criteria for measuring the faithfulness of the embedding in MDE is measured by a distortion function, which can be interpreted as a distance-preserving criterion in MDS or neighborhood-preserving criterion in

SNE-based methods. In that way, MDE can find an embedding that minimizes the overall distortion while respecting the optional constraints in the embedding.

On the other hand, contrastive learning (also called self-supervised learning) methods learn representations by *contrasting* positive and negative data points. The triplet loss used in our HCt-SNE method is a kind of contrastive loss. For any data point, contrastive methods aim to learn a representation function in such a way that similar points stay close to each other, while dissimilar ones are put far apart.

We found a strong relationship between these two paradigms and suggest this combination. The idea of fusing the MDE formulation with the contrastive learning paradigm is that we can learn a representation in such a way that the global distortion of the whole dataset is minimized, while the local similar/dissimilar points attract/repulse each other. The research problems are still based on our introduced research questions of constraint representation and constraint optimization. The former focuses on defining by a contrastive loss and the latter focuses on combining a contrastive loss with a distortion function and solving the new objective function with, probably, a gradient-based method.

Chapter 9

Appendices

9.1 iPDR Additional Results with iPPCA

We have introduced in Chapter 5 two concrete methods iPMDS and iPPCA derived from the proposed framework in Section 5.2. The experimental results through case studies with iPMDS are presented in Section 5.5. In this section, we present similar results of iPPCA with the same case studies. Additional results with the iPMDS model in an incomplete data setting are also presented.

9.1.1 Three case studies with iPPCA

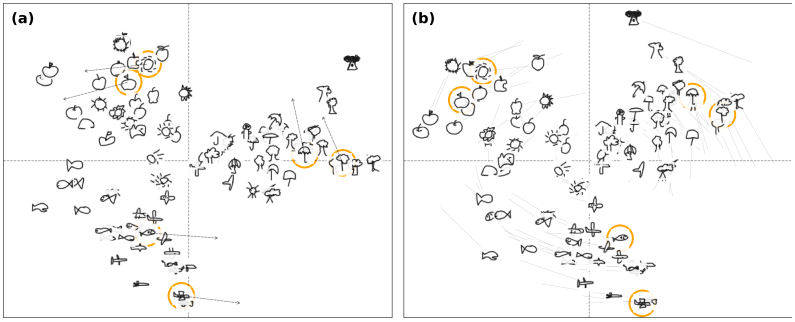


Figure 9.1: Interaction with the QuickDraw dataset shows the rotation effect.

In the first case study, 90 images of 6 classes (*airplane, apple, fish, sun, tree,*

umbrella) of the QuickDraw[1] dataset are randomly picked. Fig. 9.1 (a) shows three distinct groups with markers for the interacted points: two *horizontal shape* images (,) are pulled to the right, two *vertical shape* images (,) are pulled to the top and two *round shape* images (,) are pulled to the left.

The second case study is for creating understandable axes for an image dataset with iPPCA. Fig. 9.2 (a) shows the embedding of 100 images of clothes randomly selected from the Fashion dataset. A long dress is moved down to the left. A black coat is moved to the bottom. A rectangular bag at the bottom-right corner is pulled up. The sandals and the sneakers are pulled up to the left towards the top. As a result of these user-steering constraints, in Fig. 9.2 (b), the horizontal axis presents shape (with vertical-rectangular shapes on the left and full-rectangular shapes on the right), while the vertical axis presents color density of the clothes (with the lighter color on the top and the darker color on the bottom).

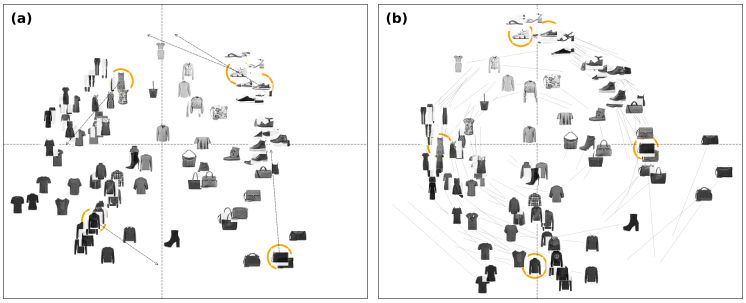


Figure 9.2: Interaction with Fashion-MNIST dataset to create axes of shape and density.

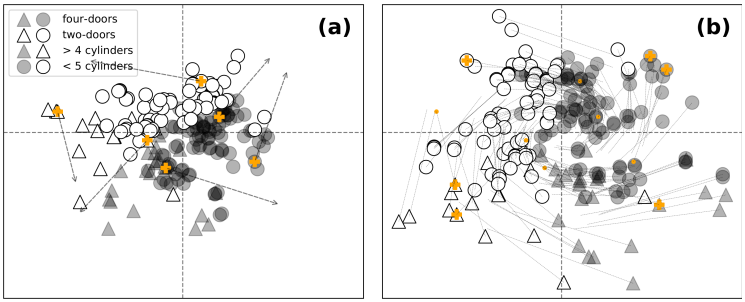


Figure 9.3: Interaction with Automobile dataset to create axes of car power and size.

[1] https://quickdraw.withgoogle.com/data

The last case study is for creating understandable axes for the Automobile tabular dataset with iPPCA. The user can move a 2-doors-4-cylinders car (○) to the top-left corner, two 2-doors-6-cylinders cars (△) to the bottom-left corner, a 4-doors-6-cylinders car (▲) to the bottom-right corner and two 4-doors-4-cylinders cars (●) to top-right corner. In the new embedding shown in Fig. 9.3 (b), we can explain the two axes induced by the user constraints. The vertical axis represents the car's power with the more-than-4-cylinders cars above and the less-than-5-cylinders cars below. The horizontal axis represents the size of the car with the small 2-doors cars on the left and the larger 4-doors cars on the right.

9.1.2 Interaction with iPMDS with incomplete input data

The probabilistic approach considers missing data as unknown quantities and models them together with other latent variables in the model. For that reason, probabilistic DR methods can handle different kinds of missing or noisy data. In the case of iPMDS, the missing data setting is also called *incomplete data* (Zinnes and MacKay, 1983). In traditional machine learning methods, missing data are encountered when the values of features in the input data points are missing. In contrast, we consider the (symmetric) matrix of pairwise distances as input for iPMDS. Missing data are encountered when pairs of distances in this matrix are missing. A visual explanation of missing pairs or incomplete data for iPMDS is shown in Figure 9.4. iPMDS takes an $N \times N$ matrix of pairwise distances as input. If the distances are symmetric (i.e., $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = d(\boldsymbol{x}_j, \boldsymbol{x}_i)$), we only need to measure $N(N-1)/2$ distinct pairs. However, when there are less than $N(N-1)/2$ pairs are available, we say that the input data is *incomplete*.

In order to demonstrate how iPMDS works with incomplete data, we first evaluate iPMDS without interaction in different settings where $p\%$ of the input pairwise distances are missing. We then show that iPMDS can work in the interactive mode to process users' constraints in an incomplete data setting.

iPMDS without interaction with incomplete data

The first experiment is performed on a subset of 250 points of the first five classes of the Digits dataset. From the complete data consisting of 31.125

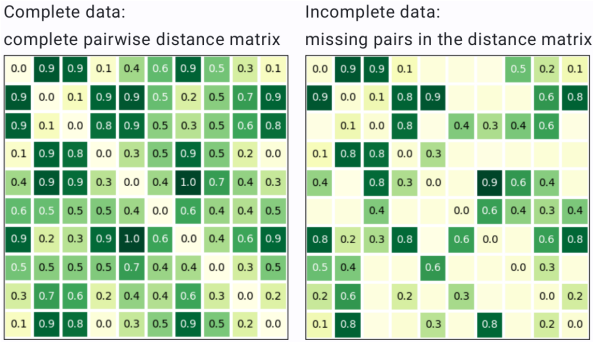


Figure 9.4: Complete and incomplete data in the iPMDS model. The incomplete data arises when there are missing pairs in the input pairwise distance matrix.

unique pairs, $p\%$ of them are randomly removed to simulate incomplete data.

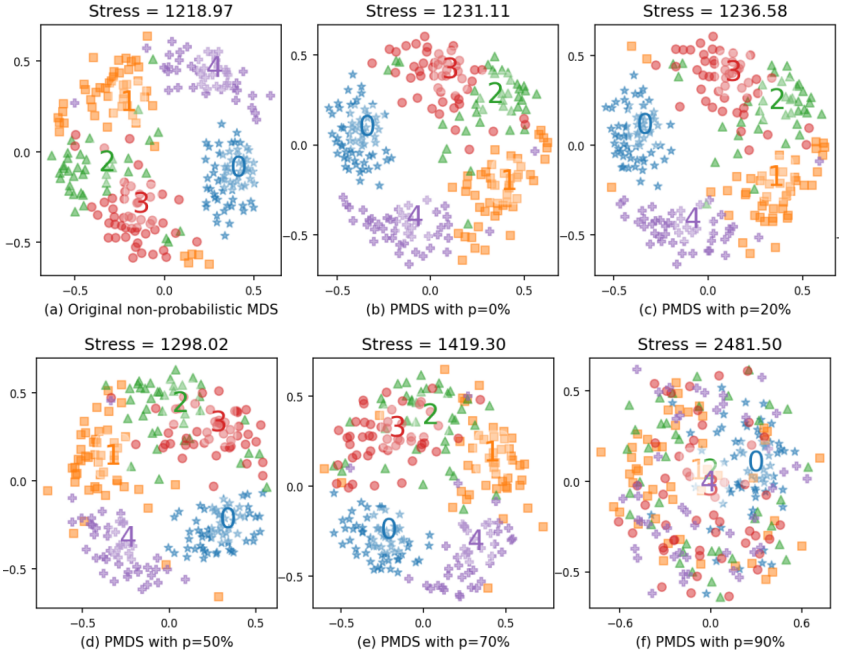


Figure 9.5: iPMDS visualizations with different setting of incomplete data for a subset of Digits dataset of 250 instances (31.125 unique pairs). (a): Visualization of the original non-probabilistic metric MDS. (b): Visualization of iPMDS with complete data. (c) - (f): Visualizations of iPMDS in incomplete data setting with 20%, 50%, 70% and 90% missing pairs, respectively. iPMDS is run without interaction.

Figure 9.5 shows a series of visualizations of the original non-probabilistic MDS (a), the results of iPMDS without interaction without any missing pair (b), and with missing pairs (c) - (f). With the incomplete data up to 70% of missing pairs, the visualization of iPMDS is still readable. The error in the visualizations increases when the percentage of missing pairs increases as expected. For a complete evolution of the quantitative error of the distance preservation (measured by the MDS stress), we run iPMDS (without interaction) with an increasing percentage of missing pairs p from 0% to 95%. For each value of p, iPMDS is run 20 times with different random initialization. The mean and standard deviation of the stress values are reported in Figure 9.6.

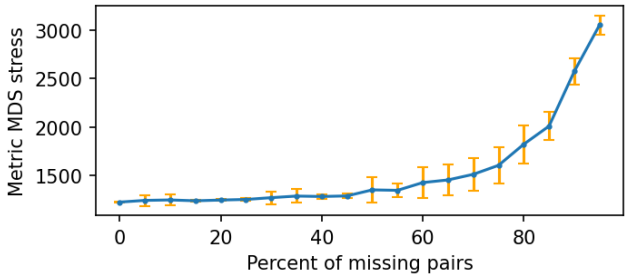


Figure 9.6: Evolution of the iPMDS stress for a subset of the Digits dataset with 20 increasing values of the percentage of missing pairs $p \in [0, 95]$. For each value of p, iPMDS is run 20 times, the mean value of stress score is shown in blue and the standard deviations are shown with orange bars.

The experiment with incomplete data is performed with a subset of the Digits dataset. When less than 50% of the pairwise distances are missing, iPMDS can still produce good visualizations with low stress (low error). This advantage in an incomplete data setting holds true for other datasets.

iPMDS in interactive mode with incomplete data

Moreover, we can still perform an interactive experiment with incomplete data with iPMDS. The same scenario as the first use case in Section 5.5 is applied in this second experiment for the same subset of 250 images of the Digits datasets with 30% of missing pairs. The same kind of interaction is performed to move several images of digit 4 and digit 0 to the opposite directions. As expected, the whole visualization is flipped as shown in Figure 9.7.

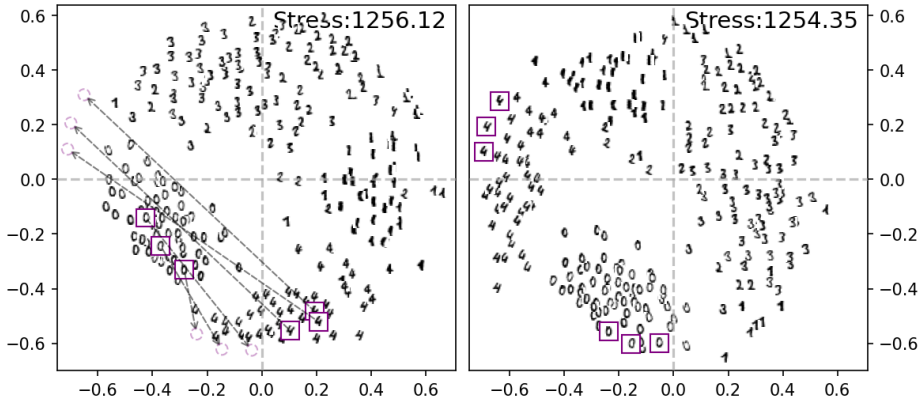


Figure 9.7: iPMDS with 30% missing pairs (only 21,787 pairs are available).

9.2 Quality Metrics

This section summarizes the formula of the quality metrics introduced in the thesis. Let d^x_{ij} and d^y_{ij} be, respectively, the distance between instances i and j in HD and LD. Let d^x and d^y be the distances matrices for all pairs of points in HD and LD.

1. The Correlation Coefficient is defined as:

$$\text{CC} = \text{pearson_correlation}(d^x, d^y) = \frac{\text{Cov}(d^x, d^y)}{\sigma(d^x)\sigma(d^y)}$$

2. For measuring the distance order in NMS, an isotonic transformation d^{iso} is performed on d^x. The Kruskal's stress is then computed using this transformation:

$$\text{NMS} = \sqrt{\frac{\sum_{ij}(d^{iso}_{ij} - d^y_{ij})^2}{\sum_{ij} d^y_{ij}}}$$

3. The Curvilinear Component Analysis Stress function is defined as:

$$\text{CCA} = \sum_{ij}(d^x_{ij} - d^y_{ij})^2 F_\lambda(d^y_{ij}),$$

in which $F_\lambda(d^y_{ij})$ is a decreasing-weighting function of d^y_{ij}. Examples of

weighting functions include the step function or $1 - sigmoid(d_{ij}^y)$. The stress function of Sammon's Nonlinear mapping is:

$$\text{NLM} = \frac{1}{\sum_{ij} d_{ij}^x} \sum_{ij} \frac{(d_{ij}^x - d_{ij}^y)^2}{d_{ij}^x}$$

4. Rank-based quality metrics $AUC[R_{NX}]$ (Lee and Verleysen, 2009, 2010) and $AUC[G_{NN}]$ (de Bodt et al., 2019).

   The notations used in the formulas of these two metrics are defined as follows. ν_i^k and ρ_i^k denote the set of k nearest neighbors of the data point i in the HD and LD spaces. $\hat{\nu}_i^k$ and $\hat{\rho}_i^k$ denote the sets of k nearest neighbors that have the same label as the data point i in the HD and LD spaces. c_i denotes the class label of the data point i. N denotes the number of instances in the dataset. The steps for calculating the two metrics are detailed in Table 9.1.

Neighborhood preserving	KNN gain
	$\hat{\rho}_i^k = \lvert\{j \in \rho_i^k : c_j = c_i\}\rvert,$ $\hat{\nu}_i^k = \lvert\{j \in \nu_i^k : c_j = c_i\}\rvert$
$Q_{NX}(k) = \frac{1}{Nk} \sum_{i=1}^{n} \lvert\nu_i^k \cap \rho_i^k\rvert$	$G_{NN}(k) = \frac{1}{Nk} \sum_{i=1}^{n} (\hat{\rho}_i^k - \hat{\nu}_i^k)$
$R_{NX}(k) = \frac{(N-1)Q_{NX}(k) - k}{N-1-k}$	
$AUC[R_{NX}] = \left(\sum_{k=1}^{N-2} \frac{R_{NX}(k)}{k}\right) / \left(\sum_{k=1}^{N-2} \frac{1}{k}\right)$	$AUC[G_{NN}] = \left(\sum_{k=1}^{N-2} \frac{G_{NN}(k)}{k}\right) / \left(\sum_{k=1}^{N-2} \frac{1}{k}\right)$

Table 9.1: Comparing the steps for calculating two metrics $AUC[R_{NX}]$ (on the left) and $AUC[G_{NN}]$ (on the right).

9.3 Analysis of AUC Curves for Embeddings of SNE-based Methods

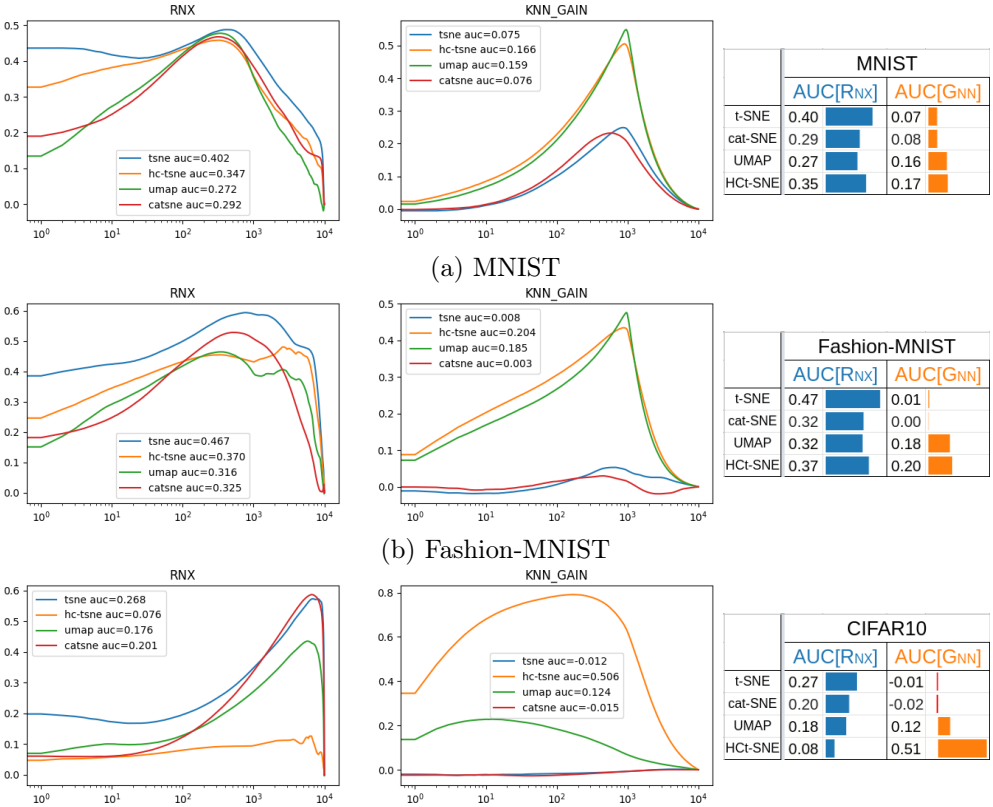


Figure 9.8: The curves of $AUC[R_{NX}]$ (on the left) and $AUC[G_{NN}]$ (on the right) for subsets of 10K images of the three datasets used in Chapter 4: (a) MNIST, (b) Fashion-MNIST, and (c) CIFAR10. Four SNE-based methods are experimented: original unsupervised t-SNE, supervised class-aware t-SNE, supervised UMAP, and our proposed HCt-SNE.

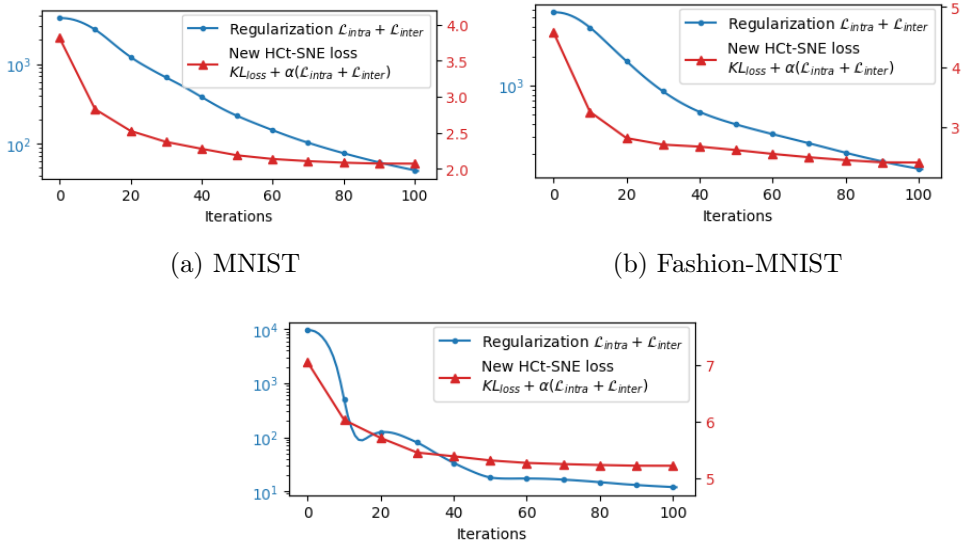
$AUC[R_{NX}]$ (Lee and Verleysen, 2009, 2010) and its variance $AUC[G_{NN}]$ (de Bodt et al., 2019) are important metrics for evaluating the quality of SNE-based methods. The detailed formulae of these scores are introduced in Appendix 9.2. In this section, we analyze the full curve of $AUC[R_{NX}]$ and $AUC[G_{NN}]$ scores for four experimented methods in Chapter 4 (original

unsupervised t-SNE, class-aware t-SNE, supervised UMAP, and our proposed HCt-SNE). Subsets of 10K images of MNIST, Fashion-MNIST, and CIFAR10 are used. The curves and the summarized scores are shown in Figure 9.8. From these curves, we can find out, with different SNE-based methods, how well the neighborhood is preserved when the neighborhood size varies.

In these plots, the neighborhood size is shown in a logarithm base 10 scale. First, there is a change of the curve when the neighborhood size goes around 10^3. Since each used dataset has 10K images of 10 balanced classes, each class in each dataset has roughly 10^3 data points. For the MNIST and Fashion-MNIST datasets, the classes are well separated, and thus, around the neighborhood size of 10^3, we have maximum gain in terms of both R_{NX} and G_{NN}. Second, with all three datasets, t-SNE (the blue curve of R_{NX}) better preserves the neighborhood information with both small and large neighborhood sizes. Third, for CIFAR10, the R_{NX} curve of HCt-SNE does not make distinctions between small and large neighborhood size since this method is regularized by a (supervised) hierarchical structure, which is far different from the neighborhood information. And finally, HCt-SNE and supervised UMAP have better gain in terms of G_{NN}, particular for CIFAR10, while the original t-SNE has almost no gain with both small and large neighborhood size.

9.4 Choosing the Regularization Coefficient α in HCt-SNE

As presented in Chapter 4, the regularization coefficient α is the most important hyperparameter we need to tune in HCt-SNE. In our experiment, we use an old t-SNE embedding as initialization for a quick convergence (after 100 iterations as shown in the figures below). With a medium-size dataset (a subset of 10K images), a learning rate $\eta = N/12$ is used as suggested by Kobak and Berens (2019). Recalled from Algorithm 1, the regularization coefficient α determines the balance between the unsupervised objective (t-SNE KL loss) and the *supervised* objective (regularization term built upon constraints/supervised



(a) MNIST

(b) Fashion-MNIST

(c) CIFAR10

Figure 9.9: The overall loss of HCt-SNE is a combination of the original KL loss of t-SNE and the regularization term, balanced by the coefficient α. The overall loss is shown in red with the scale on the right axis. The regularization is shown in blue in log scale on the left axis.

information from users):

$$\mathcal{L}_{HCt-SNE} = \mathcal{L}_{t-SNE} + \alpha \underbrace{\left(\mathcal{L}_{intra} + \mathcal{L}_{inter}\right)}_{\text{regularization terms}}. \tag{9.1}$$

In order to choose an acceptable value for this hyperparameter, we need to observe the values of both the regularization and the combination of this regularization with the unsupervised objective KL loss. Figure 9.9 shows the overall loss function of HCt-SNE and the regularization term for three datasets with the following chosen regularization coefficients α: 7.5×10^{-4} for MNIST and Fashion-MNIST, and 5×10^{-3} for CIFAR10. Notice that these values of α for each dataset are chosen manually by a trial-and-error process. We still need a suitable metric to measure the compromise between the two objectives in HCt-SNE in order to tune this hyperparameter automatically.

9.5 Pairwise Constraints with DR Methods

In this thesis, we have proposed constrained methods using fixed position constraints on individual points and hierarchical constraints on semantic groups. Another type of constraint, pairwise constraint, is widely used in semi-supervised classification and constrained clustering methods. As presented in Chapter 3, this kind of constraint is also used for DR, particularly to enhance the covariance matrix in PCA/LDA-based methods and to enhance the neighbor graph for graph-based methods. In these existing methods, the pairs are used to modify the feature weights of the data via weighted distances between similar/dissimilar points. For short-term future work, we propose to use these pairwise constraints to modify the weight of each pair. Since preserving pairwise constraints can be interpreted as, in a narrow sense, preserving distances, this kind of constraint can be integrated into DR methods that do not have an explicit distance-preservation criterion such as t-SNE (or PCA).

9.5.1 Pairwise Constraints to Enhance Gram Matrix for PCA

Based on the usage of the Gram matrix $\boldsymbol{X}\boldsymbol{X}^T$ to derive PCA components instead of the covariance matrix, our idea is to modify pairs of instances connected by the pairwise constraints to enhance this Gram matrix.

All the principal components can be derived from the data matrix $\boldsymbol{X}_{\mathrm{N}\times\mathrm{p}}$ of N instances and p features using SVD as

$$\begin{aligned} \boldsymbol{X}_{\mathrm{N}\times\mathrm{p}} &= \boldsymbol{U}_{\mathrm{N}\times\mathrm{N}}\boldsymbol{S}_{\mathrm{N}\times\mathrm{p}}\boldsymbol{V}^T_{\mathrm{p}\times\mathrm{p}}, \\ \boldsymbol{Z}_{\mathrm{N}\times\mathrm{k}} &= \boldsymbol{U}[:,:k]_{\mathrm{N}\times\mathrm{k}}\boldsymbol{S}[:k,:k]_{\mathrm{k}\times\mathrm{k}}, \end{aligned} \tag{9.2}$$

where $\boldsymbol{U}$ is a unitary matrix and $\boldsymbol{S}$ is the diagonal matrix of singular values. From this formulation, the column of $\boldsymbol{V}$ are principal directions and the columns of $\boldsymbol{U}\boldsymbol{S}$ are principal components. The two matrix $\boldsymbol{U}, \boldsymbol{S}$ can also be obtained from the Gram matrix $\boldsymbol{G} = \frac{1}{N-1}\boldsymbol{X}\boldsymbol{X}^T$, where they are the eigenvectors and eigenvalues of the eigen decomposition of $\boldsymbol{G}$, respectively. Based on observation, the Gram matrix is modified as $\boldsymbol{G}'_{ij} = \omega_{ij}\boldsymbol{G}_{ij}$, where

$$\begin{cases} \omega_{ij} > 1 \text{ if } (i,j) \text{ is a must-link/similar pair,} \\ \omega_{ij} < 1 \text{ if } (i,j) \text{ is a cannot-link/dissimilar pair.} \end{cases} \tag{9.3}$$

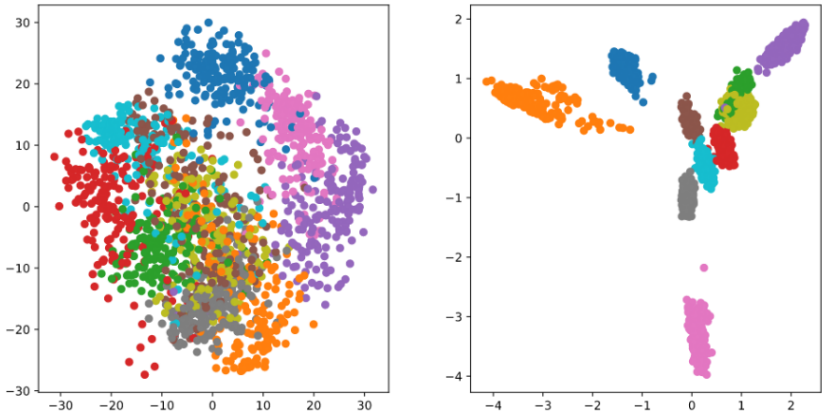


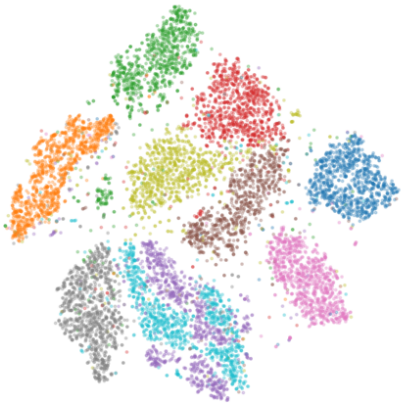
Figure 9.10: Demonstration for the idea of integrating pairwise constraints into the Gram matrix for PCA for the Digit dataset. On the left is the result of the original PCA. On the right is the result of a modified version of PCA using Gram matrix enhanced by pairwise constraints.

A preliminary result of this idea is presented in Figure 9.10 for the Digits dataset (1797 handwritten digits of gray-scale 8x8 images). Extensive experiments and comparisons with other methods, as well as a study on the effect of the number of pairs should be conducted in future work. We also need to find out a reasonable explanation for the axes in the visualization (and an interpretable interpolation of the data points along the axes).

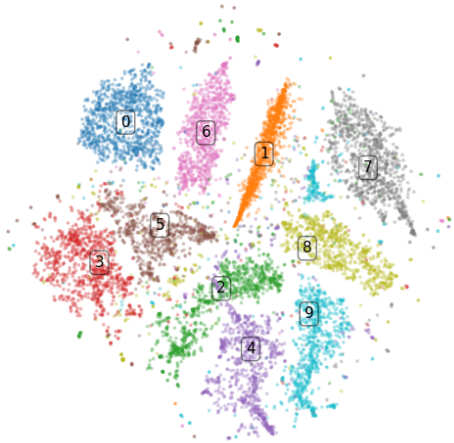
9.5.2 Pairwise Constraints to Enhance the Affinities in t-SNE

Inspired by the idea of class-aware t-SNE (*cat*-SNE) (de Bodt et al., 2019), we propose to use pairwise constraints to modify the affinity matrix $\boldsymbol{P}$ in t-SNE. *cat*-SNE uses class labels to produce a better version of this affinity matrix $\boldsymbol{P}$ by filtering out the points in a neighborhood but do not belong to the same class. We also modify this matrix $\boldsymbol{P}$, simply by enhancing a pair $p_i j$ by a weight ω_{ij}, such as the one proposed in equation 9.3. A preliminary result of this idea is presented in Figure 9.11 for a subset of 10K images of the MNIST dataset.

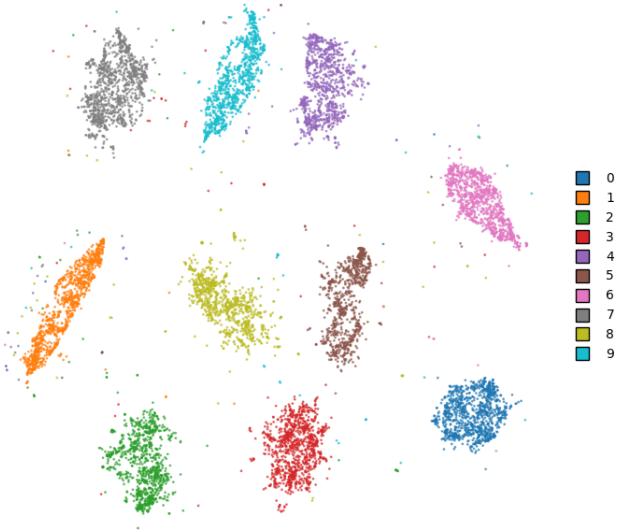
In summary, these preliminary results only show a small suggestion about the effect of pairwise constraints to modify the weights of pairs in order to enhance the representation of HD data (via a Gram matrix or an affinity matrix).



(a) t-SNE



(b) Class-aware t-SNE



(c) t-SNE with an enhanced affinity matrix $\boldsymbol{P}$

Figure 9.11: Preliminary result of t-SNE visualization with an enhanced affinity matrix $\boldsymbol{P}$ for a subset of 10K images of the MNIST dataset (c), in comparison with the original t-SNE visualization (a) and the class-aware t-SNE (b).

We can achieve an improvement in the separation of groups in the visualization, however, more work should be done to create a robust method that can work with a very limited number of pairs.

Bibliography

Sameer Agarwal, Josh Wills, Lawrence Cayton, Gert Lanckriet, David Kriegman, and Serge Belongie. Generalized non-metric multidimensional scaling. In *Artificial Intelligence and Statistics*, pages 11–18, 2007. 37

Charu C Aggarwal, Alexander Hinneburg, and Daniel A Keim. On the surprising behavior of distance metrics in high dimensional space. In *International conference on database theory*, pages 420–434. Springer, 2001. 14

Akshay Agrawal, Alnur Ali, and Stephen Boyd. Minimum-distortion embedding. *arXiv preprint arXiv:2103.02559*, 2021. 24, 176, 185

Shotaro Akaho. A kernel method for canonical correlation analysis. *arXiv preprint cs/0609071*, 2006. 49, 53, 55

E. Amid and M. K. Warmuth. TriMap: Large-scale Dimensionality Reduction Using Triplets. *arXiv preprint arXiv:1910.00204*, 2019. 37, 38, 65

Sanjeev Arora, Hrishikesh Khandeparkar, Mikhail Khodak, Orestis Plevrakis, and Nikunj Saunshi. A theoretical analysis of contrastive unsupervised representation learning. In *Proceedings of the 36th International Conference on Machine Learning*, pages 5628–5637, Long Beach, CA, 2019. 38, 91, 92

Michaël Aupetit. Visualizing distortions and recovering topology in continuous projection techniques. *Neurocomputing*, 70(7-9):1304–1330, 2007. 179

Maria-Florina F Balcan, Yingyu Liang, Vandana Kanchanapally, and David Woodruff. Improved distributed principal component analysis. In *Advances in Neural Information Processing Systems*, volume 27, 2014. 13

Aharon Bar-Hillel, Tomer Hertz, Noam Shental, and Daphna Weinshall. Learning a mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research*, 6 (Jun):937–965, 2005. 32, 47, 53, 55

Etienne Becht, Leland McInnes, John Healy, Charles-Antoine Dutertre, Immanuel WH Kwok, Lai Guan Ng, Florent Ginhoux, and Evan W Newell. Dimensionality reduction for visualizing single-cell data using umap. *Nature biotechnology*, 37(1):38–44, 2019. 174

Peter N. Belhumeur, Joao P Hespanha, and David J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on pattern analysis and machine intelligence*, 19(7):711–720, 1997. 47

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003. 33, 35

Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7(11), 2006. 42, 45, 46

Anna C Belkina et al. Automated optimized parameters for t-distributed stochastic neighbor embedding improve visualization and analysis of large datasets. *Nature communications*, 10(5415):1–12, 2019. 83

Aurélien Bellet, Amaury Habrard, and Marc Sebban. A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv:1306.6709*, 2013. 39

Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966. 14

James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Proc. NIPS*, pages 2546–2554, Grenada, Spain, Dec. 2011. 146

Michael Betancourt, Simon Byrne, Sam Livingstone, and Mark Girolami. The geometric foundations of hamiltonian monte carlo. *Bernoulli*, 23(4A):2257–2298, 2017. 137

Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? In *International conference on database theory*, pages 217–235. Springer, 1999. 14

Adrien Bibal and Benoit Frénay. Learning interpretability for visualizations using adapted cox models through a user experiment. In *NIPS Workshop on Interpretable Machine Learning in Complex Systems*, 2016. 179

Adrien Bibal and Benoit Frénay. Measuring quality and interpretability of dimensionality reduction visualizations. In *SafeML ICLR Workshop*, 2019. 143, 179

Adrien Bibal, Rebecca Marion, and Benoit Frenay. Finding the most interpretable mds rotation for sparse linear models based on external features. In *Proc. ESANN*, pages 537–542, 2018. 48, 52, 55, 134

Adrien Bibal, Bruno Dumas, and Benoit Frénay. User-based experiment guidelines for measuring interpretability in machine learning. In *EGC Workshop on Advances in Interpretable Machine Learning and Artificial Intelligence*, 2019. 177

Adrien Bibal, Rebecca Marion, Rainer von Sachs, and Benoit Frénay. BIOT: Explaining multidimensional mds embeddings using the best interpretable orthogonal transformation. *Neurocomputing*, 453:109–118, 2021. 48, 52, 55

Christopher M Bishop. Bayesian PCA. In *Proc. Advances in Neural Information Processing Systems*, pages 382–388, 1999. 100, 121

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, 2006. ISBN 0387310738. 14, 124

Christopher M Bishop. Model-based machine learning. *Philosophical Transactions of the Royal Society A*, 371, 2013. 99

Christopher M Bishop, Markus Svensén, and Christopher KI Williams. GTM: The generative topographic mapping. *Neural computation*, 10(1):215–234, 1998. 28, 100, 137, 138

David M Blei. Build, compute, critique, repeat: Data analysis with latent variable models. *Annual Review of Statistics and its Application*, 1:203–232, 2014. 137

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003. 153

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017. 137

Jan Niklas Böhm, Philipp Berens, and Dmitry Kobak. A unifying perspective on neighbor embeddings along the attraction-repulsion spectrum. *arXiv preprint arXiv:2007.08902*, 2020. 176

Hanen Borchani, Gherardo Varando, Concha Bielza, and Pedro Larranaga. A survey on multi-output regression. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(5):216–233, 2015. 50

Ingwer Borg and Patrick J. F. Groenen. *Modern Multidimensional Scaling*. Springer Science & Business Media, 2005a. 14

Ingwer Borg and Patrick JF Groenen. *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media, 2005b. 13, 107

Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint*, 2010. 168, 169

Andreas Buja, Deborah F Swayne, Michael Littman, Nathaniel Dean, and Heike Hofmann. XGvis: Interactive data visualization with multidimensional scaling. *Journal of Computational and Graphical Statistics*, pages 1061–8600, 2001. 27, 52, 54, 128

Andreas Buja, Deborah F Swayne, Michael L Littman, Nathaniel Dean, Heike Hofmann, and Lisha Chen. Data visualization with multidimensional scaling. *Journal of Computational and Graphical Statistics*, 17(2):444–472, 2008. 27, 31, 52, 54, 128

D. Cai, X. He, and J. Han. Semi-supervised discriminant analysis. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–7, Oct 2007a. 44, 47, 53, 55

Deng Cai, X. He, Kun Zhou, Jiawei Han, and H. Bao. Locality sensitive discriminant analysis. In *IJCAI*, 2007b. 44, 53, 55

Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer, 2013. 93

Yanshuai Cao and Luyu Wang. Automatic selection of t-SNE perplexity. In *ICML AutoML Workshop*, pages 1–7, Sydney, Australia, Oct. 2017. 145, 146, 154

J Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika*, 35(3): 283–319, 1970. 30

Marco Cavallo and Çağatay Demiralp. Exploring dimensionality reductions with forward and backward projections. In *Procs. KDD Workshop on Interactive Data Exploration and Analytics (IDEA)*, Aug 2017. 40, 52, 54

Hakan Cevikalp, Jakob Verbeek, Frédéric Jurie, and Alexander Klaser. Semi-supervised dimensionality reduction using pairwise equivalence constraints. In *Proc. VISAPP*, pages 489–496, Funchal, Portugal, Jan. 2008. 32, 34, 35, 53, 54, 177

Guoqing Chao, Yuan Luo, and Weiping Ding. Recent advances in supervised dimension reduction: A survey. *Machine Learning and Knowledge Extraction*, 1(1):341–358, 2019. 23, 40

Jianhui Chen, Jieping Ye, and Q. Li. Integrating global and local structures: A least squares framework for dimensionality reduction. *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007. 45, 46, 53, 55

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proc. ICML*, pages 1597–1607, Vienna, Austria, 2020. 38, 65, 185

J. Choo, Hanseung Lee, Jaeyeon Kihm, and Haesun Park. iVisClassifier: An interactive visual analytics system for classification based on supervised dimension reduction. *2010 IEEE Symposium on Visual Analytics Science and Technology*, pages 27–34, 2010. 39, 53, 54

John P Cunningham and Zoubin Ghahramani. Linear dimensionality reduction: Survey, insights, and generalizations. *The Journal of Machine Learning Research*, 16(1):2859–2900, 2015. 176

Rene Cutura, Stefan Holzer, Michaël Aupetit, and Michael Sedlmair. VisCoDeR: A tool for visually comparing dimensionality reduction algorithms. In *Proceedings of European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pages 105–110, Bruges, Belgium, Apr. 2018. 155

Tuan Nhon Dang and Leland Wilkinson. Scagexplorer: Exploring scatterplots by their scagnostics. In *2014 IEEE Pacific visualization symposium*, pages 73–80. IEEE, 2014. 174

Ian Davidson. Knowledge driven dimension reduction for clustering. In *Proc. IJCAI*, pages 1034–1039, California, USA, Jul. 2009. 33, 34, 53, 54

Ian Davidson. Two approaches to understanding when constraints help clustering. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1312–1320. ACM, 2012. 177

Ian Davidson and Sugato Basu. A survey of clustering with instance level constraints. In *Proc. ACM TKDD*, pages 1–41, 2007. 23, 32

Cyril de Bodt, Dounia Mulders, Michel Verleysen, and John A Lee. Perplexity-free t-sne and twice student tt-sne. In *Proc. ESANN*, 2018. 20

Cyril de Bodt, D. Mulders, D. López-Sánchez, Michel Verleysen, and John A. Lee. Class-aware t-SNE: cat-SNE. In *Proceedings of European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pages 409–414, 2019. 20, 65, 81, 83, 151, 193, 194, 198

Cyril de Bodt, D. Mulders, Michel Verleysen, and John A Lee. Fast Multiscale Neighbor Embedding. *IEEE Trans. Neural Netw. Learn. Syst.*, pages 1–15, 2020. doi: 10.1109/TNNLS.2020.3042807. 20

Vin De Silva and Joshua B Tenenbaum. Sparse multidimensional scaling using landmark points. Technical report, Technical report, Stanford University, 2004. 31

Marc Peter Deisenroth, A Aldo Faisal, and Cheng Soon Ong. *Mathematics for Machine Learning*. Cambridge University Press, 2020. 13, 40

Valentin Delchevalerie, Alexandre Mayer, Adrien Bibal, and Benoît Frénay. Accelerating t-sne using fast fourier transforms and the particle-mesh algorithm from physics. In *Proc. IJCNN*, 2021. 19

Pierre Demartines and Jeanny Hérault. Curvilinear component analysis: A self-organizing neural network for nonlinear mapping of data sets. *IEEE Transactions on Neural Networks*, 8(1):148–154, 1997. 143

Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012. 14

Alex Endert, Chao Han, Dipayan Maiti, Leanna House, and Chris North. Observation-level interaction with statistical models for visual analytics. In *Visual Analytics Science and Technology (VAST), 2011 IEEE Conference on*, pages 121–130. IEEE, 2011. 26, 28, 29, 52, 54, 128

Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049, 2016. 10

Bernd Fischer, Volker Roth, and Joachim M Buhmann. Clustering with the connectivity kernel. In *Advances in Neural Information Processing Systems*, pages 89–96, 2004. 34

Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936. 41

Dean P Foster, Sham M Kakade, and Tong Zhang. Multi-view dimensionality reduction via canonical correlation analysis. *Technical Report TR-2008-4*, 2008. 49

Damien François. *High-dimensional data analysis: optimal metrics and feature selection*. PhD thesis, Université catholique de Louvain, 2007. 14

Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001. 98

Jerome H Friedman. Regularized discriminant analysis. *Journal of the American statistical association*, 84(405):165–175, 1989. 47

Keinosuke Fukunaga. *Introduction to statistical pattern recognition*. Elsevier, 2013. 41, 43

Weifeng Ge. Deep metric learning with hierarchical triplet loss. In *Proceedings of the European Conference on Computer Vision*, pages 269–285, Munich, Germany, 2018. 38, 65

Ian Gemp, Brian McWilliams, Claire Vernade, and Thore Graepel. EigenGame: PCA as a nash equilibrium. In *Proc. International Conference on Learning Representations (ICLR)*, page 1, 2021. 12, 13

Xin Geng, De-Chuan Zhan, and Zhi-Hua Zhou. Supervised nonlinear dimensionality reduction for visualization and classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 35(6):1098–1107, 2005. 47, 143

Paolo Giordani and Henk AL Kiers. Principal component analysis with boundary constraints. *Journal of Chemometrics*, 21(12):547–556, Oct. 2007. 39, 52, 54

Jacob Goldberger, Geoffrey E Hinton, Sam T Roweis, and Russ R Salakhutdinov. Neighbourhood components analysis. In *Proc. NIPS*, pages 513–520, 2005. 65

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org. 90

Valerio Grossi, Andrea Romei, and Franco Turini. Survey on using constraints in data mining. *Data Mining and Knowledge Discovery*, 31:424–464, 2017. 23

Baolin Guo, Chenping Hou, F. Nie, and Dong yun Yi. Semi-supervised multi-label dimensionality reduction. *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 919–924, 2016. 50, 53, 55, 177

Yaqian Guo, Trevor Hastie, and Robert Tibshirani. Regularized linear discriminant analysis and its application in microarrays. *Biostatistics*, 8(1):86–100, 2007. 47

Chao Han, Leanna House, Scotland C. Leman, and Ramin Homayouni. Expert-guided generative topographical modeling with visual to parametric interaction. In *PloS one*, 2016. 28

Xiaofei He and Partha Niyogi. Locality preserving projections. In *Advances in neural information processing systems*, pages 153–160, 2004. 33, 35, 43

Robert A. Hefner. *Extensions of the law of comparative judgment to discriminable and multidimensional stimuli*. PhD thesis, University of Michigan, 1958. URL https://www.proquest.com/openview/4bb7b79427ee17aa3a7c03ce9df5ddb7. 28, 107, 108, 109, 110

Geoffrey E Hinton and Sam T Roweis. Stochastic neighbor embedding. In *Proc. NIPS*, pages 857–864, 2003. 16

Geoffrey E Hinton, Christopher KI Williams, Michael D Revow, et al. Adaptive elastic models for hand-printed character recognition. In *NIPS*, volume 4, pages 512–519, 1991. 100

Stephen C Hirtle. Representational structures for cognitive space: Trees, ordered trees and semi-lattices. In *International Conference on Spatial Information Theory*, pages 327–340. Springer, 1995. 64

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *Proc. ICLR*, 2019. 185

Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. *The annals of statistics*, 36(3):1171–1220, 2008. 100

Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417, 1933. 10, 13, 121

Harold Hotelling. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377, 1936. 48

Leanna House, Scotland Leman, and Chao Han. Bayesian visual analytics: BaVA. *Statistical Analysis and Data Mining*, 8:1–13, 2015. 26, 28, 31, 52, 54, 128

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 152

HaoShuang Hu, Da-Zheng Feng, and Qing-Yan Chen. A novel dimensionality reduction method: Similarity order preserving discriminant analysis. *Signal Processing*, 182:107933, 2021. 36

Rui Huang, Guopeng Zhang, and Junli Chen. Semi-supervised discriminant isomap with application to visualization, image retrieval and classification. *International Journal of Machine Learning and Cybernetics*, 10(6):1269–1278, 2019. 36, 53, 54

Dong Hyun Jeong, Caroline Ziemkiewicz, Brian Fisher, William Ribarsky, and Remco Chang. iPCA: An interactive system for pca-based visual analytics. *Computer Graphics Forum*, 28(3):767–774, 2009. 39, 52, 54

Dong Hyun Jeong, Soo-Yeon Ji, Evan A Suma, Byunggu Yu, and Remco Chang. Designing a collaborative visual analytics system to support users' continuous analytical processes. *Human-centric Computing and Information Sciences*, 5(1):5, 2015. 39, 52, 54

Yangqing Jia, Feiping Nie, and Changshui Zhang. Trace ratio problem revisited. *IEEE Transactions on Neural Networks*, 20(4):729–735, 2009. 43

Michael Irwin Jordan. *Learning in graphical models*, volume 89. Springer Science & Business Media, 1998. 116

Bo Kang, Darío García García, Jefrey Lijffijt, Raúl Santos-Rodríguez, and Tijl De Bie. Conditional t-SNE: more informative t-sne embeddings. *Machine Learning*, pages 1–36, 2020. 175

C Kaynak. Methods of combining multiple classifiers and their applications to handwritten digit recognition. Master's thesis, Bogazici University, 1995. 129, 152

J. R. Kettenring. Canonical analysis of several sets of variables. *Biometrika*, 58(3):433–451, 1971. 49, 53, 55

Mohammad Emtiyaz Khan and Håvard Rue. The bayesian learning rule. *arXiv preprint arXiv:2107.04562*, 2021. 98

Hannah Kim, Jaegul Choo, Changhyun Lee, Hanseung Lee, Chandan K Reddy, and Haesun Park. PIVE: Per-iteration visualization environment for real-time interactions with dimension reduction and clustering. In *AAAI*, pages 1001–1009, 2017. 26, 27, 31, 54

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. ICLR*, pages 577–584, 2015. 104

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *Proc. ICLR*, 2014. arXiv preprint arXiv:1312.6114. 138

Teemu Kivioja, Anna Vähärautio, Kasper Karlsson, Martin Bonke, Sten Linnarsson, and Jussi Taipale. Counting absolute number of molecules using unique molecular identifiers. *Nature Proceedings*, pages 1–18, 2011. 167

Dmitry Kobak and Philipp Berens. The art of using t-SNE for single-cell transcriptomics. *Nature communications*, 10(5416):1–14, 2019. 59, 79, 82, 83, 145, 174, 195

Dmitry Kobak and George C Linderman. Initialization is critical for preserving global data structure in both t-sne and umap. *Nature biotechnology*, 39(2):156–157, 2021. 59, 145

Alex Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, University of Toronto, 2009. 58, 81

Joseph B Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964a. 13, 107, 131, 143

Joseph B Kruskal. Nonmetric multidimensional scaling: a numerical method. *Psychometrika*, 29(2):115–129, 1964b. 13

Brian Kulis et al. Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4):287–364, 2012. 39, 91

LD Landau and EM Lifshitz. Statistical physics, v. 5. *Course of Theoretical Physics*, 23, 1980. 19

Neil Lawrence. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *Journal of Machine Learning Research*, 6(Nov):1783–1816, 2005. 100, 137

Neil Lawrence. A unifying probabilistic perspective for spectral dimensionality reduction: Insights and new models. *The Journal of Machine Learning Research*, 13(1):1609–1638, 2012. 137, 176

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. In Gökhan BakIr, Thomas Hofmann, Bernhard Schölkopf, Alexander J Smola, and Ben Taskar, editors, *Predicting Structured Data*, pages 191–246. MIT Press, July 2007. 91

Yann LeCun, Corinna Cortes, and CJ Burges. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist, 2010. 81

John A Lee and Michel Verleysen. *Nonlinear Dimensionality Reduction*. Springer Science & Business Media, 2007. 12, 45

John A Lee and Michel Verleysen. Quality assessment of dimensionality reduction: Rank-based criteria. *Neurocomputing*, 72(7-9):1431–1443, 2009. 82, 144, 178, 193, 194

John A Lee and Michel Verleysen. Scale-independent quality criteria for dimensionality reduction. *Pattern Recognition Letters*, 31(14):2248–2257, 2010. 82, 144, 154, 178, 193, 194

John A Lee, Emilie Renard, Guillaume Bernard, Pierre Dupont, and Michel Verleysen. Type 1 and 2 mixtures of kullback–leibler divergences as cost functions in dimensionality reduction based on similarity preservation. *Neurocomputing*, 112:92–108, 2013. 91

John A Lee, Diego Hernán Peluffo-Ordóñez, and Michel Verleysen. Multiscale stochastic neighbor embedding: Towards parameter-free dimensionality reduction. In *Proceedings of European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pages 177–182, Bruges, Belgium, Apr 2014. 20, 146

John A Lee, Diego H Peluffo-Ordóñez, and Michel Verleysen. Multi-scale similarities in stochastic neighbour embedding: Reducing dimensionality while preserving both local and global structure. *Neurocomputing*, 169:246–261, 2015. 20, 91

C. Leman and A. Endert. A bi-directional visualization pipeline that enables visual to parametric interation (v2pi). 2010. 26, 30, 31, 53, 54

S. Leman, L. House, D. Maiti, A. Endert, and Chris North. Visual to parametric interaction (v2pi). *PLoS ONE*, 8, 2013. 29, 52, 54, 128

Scotland C Leman, Leanna House, Dipayan Maiti, Alex Endert, and Chris North. A bi-directional visualization pipeline that enables visual to parametric interation (v2pi). *PLOS One*, 2011. 131

Benjamin Letham, Brian Karrer, Guilherme Ottoni, Eytan Bakshy, et al. Constrained bayesian optimization with noisy experiments. *Bayesian Analysis*, 14(2):495–519, 2019. 146

Jake Lever, Martin Krzywinski, and Naomi Altman. Points of significance: Principal component analysis. *Nature methods*, 14(7):641–643, 2017. 12

Zhenguo Li, Jianzhuang Liu, and Xiaoou Tang. Pairwise constraint propagation by semidefinite programming for semi-supervised classification. In *Proceedings of the 25th international conference on Machine learning*, pages 576–583. ACM, 2008. 178

Yingyu Liang, Maria-Florina Balcan, and Vandana Kanchanapally. Distributed pca and k-means clustering. In *The Big Learning Workshop at NIPS*, volume 2013. Citeseer, 2013. 13

Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013. 152

George C Linderman, Manas Rachh, Jeremy G Hoskins, Stefan Steinerberger, and Yuval Kluger. Fast interpolation-based t-SNE for improved visualization of single-cell RNA-seq data. *Nature Methods*, 16(3):243–245, 2019. 19, 59, 81

Zhiguo Long, Hua Meng, and Michael Sioutis. Semi-supervised dimensionality reduction by linear compression and stretching. *IEEE Access*, 8:27308–27317, 2020. 35

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008. 16, 58, 61, 145

David B. MacKay. Alternative probabilistic scaling models for spatial data. *Geographical Analysis*, 15:173–186, 1983. 100, 108

David B MacKay. Probabilistic multidimensional scaling: An anisotropic model for distance judgments. *Journal of Mathematical Psychology*, 33:187–205, 1989. 108

David B. MacKay and Joseph L. Zinnes. A probabilistic model for the multidimensional scaling of proximity and preference data. *Marketing Science*, 5:325–344, 1986. ISSN 0732-2399. 28, 100, 107, 108, 114

David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003. 99

Pavan Kumar Mallapragada, Rong Jin, and Anil K Jain. Active query selection for semi-supervised clustering. In *2008 19Th international conference on pattern recognition*, pages 1–4. IEEE, 2008. 178

Rebecca Marion, Adrien Bibal, and Benoit Frénay. BIR: A method for selecting the best interpretable multidimensional scaling rotation using external variables. *Neurocomputing*, 342:83–96, 2019. 48, 52, 55, 135

Leland McInnes, John Healy, and James Melville. UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018a. 16, 65, 81

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. UMAP: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861, 2018b. 142, 145

Meng Meng, Jia Wei, Jiabing Wang, Qianli Ma, and Xuan Wang. Adaptive semi-supervised dimensionality reduction based on pairwise constraints weighting and graph optimizing. *Int. J. Machine Learning & Cybernetics*, 8:793–805, 2017. 35

Karl Øyvind Mikalsen, C. Soguero-Ruíz, F. M. Bianchi, and R. Jenssen. Noisy multi-label semi-supervised dimensionality reduction. *Pattern Recognit.*, 90:257–270, 2019. 50

J. Močkus. On bayesian methods for seeking the extremum. In G. I. Marchuk, editor, *Optimization Techniques IFIP Technical Conference Novosibirsk, July 1–7, 1974*, pages 400–404, 1975. 146, 168

Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. The application of bayesian methods for seeking the extremum. *Towards Global Optimization*, 2:117–128, 1978. 146

Vladimir Molchanov and Lars Linsen. Interactive design of multidimensional data projection layout. 2014. 40, 52, 54

Cristina Morariu, Adrien Bibal, Rene Cutura, Benoît Frénay, and Michael Sedlmair. DumbleDR: Predicting user preferences of dimensionality reduction embedding quality. Under review. 174, 179

Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2021. URL probml.ai. 98

Ashwin Narayan, Bonnie Berger, and Hyunghoon Cho. Density-preserving data visualization unveils dynamic patterns of single-cell transcriptomic variability. *Nature Biotechnology*, 2021. 59, 64

Sameer A. Nene, Shree K. Nayar, and Hiroshi Murase. Columbia object image library (COIL-20). Technical report, 1996. 11, 152

F. Nie, Shiming Xiang, Yun Liu, and C. Zhang. A general graph-based semi-supervised learning with novel class discovery. *Neural Computing and Applications*, 19:549–555, 2009. 178

F. Nie, Dong Xu, X. Li, and Shiming Xiang. Semisupervised dimensionality reduction and classification through virtual label regression. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 41:675–685, 2011. 178

Feiping Nie, Dong Xu, Ivor Wai-Hung Tsang, and Changshui Zhang. Flexible manifold embedding: A framework for semi-supervised and unsupervised dimension reduction. *IEEE Transactions on Image Processing*, 19(7):1921–1932, 2010. 45, 46, 53, 55

Luis Gustavo Nonato and Michaël Aupetit. Multidimensional projection for visual analytics: Linking techniques with distortions, tasks, and layout enrichment. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 25(8):2650–2673, 2018. 179

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. 2018. 185

Karl Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901. 10, 12, 121

Nicola Pezzotti, Thomas Höllt, Boudewijn PF Lelieveldt, Elmar Eisemann, and Anna Vilanova. Hierarchical stochastic neighbor embedding. *Computer Graphics Forum*, 35:21–30, 2016. 65

Pavlin G. Poličar, Martin Stražar, and Blaž Zupan. openTSNE: a modular python library for t-SNE dimensionality reduction and embedding. *bioRxiv*, 2019. 82, 145

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003. 101

Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, volume 11, page 269. NIH Public Access, 2017. 178

Alexander Ratner, Braden Hancock, Jared Dunnmon, Frederic Sala, Shreyash Pandey, and Christopher Ré. Training complex models with multi-task weak supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4763–4771, 2019. 178

Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 410–420, 2007. 151

Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000. 26, 33, 35

Tuukka Ruotsalo, Giulio Jacucci, Petri Myllymäki, and Samuel Kaski. Interactive intent modeling: Information discovery beyond search. *Commun. ACM*, 58(1):86–92, December 2014. ISSN 0001-0782. 137

Dominik Sacha, Michael Sedlmair, Leishi Zhang, John A Lee, Jaakko Peltonen, Daniel Weiskopf, Stephen C North, and Daniel A Keim. What you see is what you can change: Human-centered machine learning by interactive visualization. *Neurocomputing*, 268: 164–175, 2017a. 24

Dominik Sacha, Leishi Zhang, Michael Sedlmair, John Aldo Lee, Jaakko Peltonen, Daniel Weiskopf, Stephen C. North, and Daniel A. Keim. Visual interaction with dimensionality reduction: A structured literature analysis. *IEEE Transactions on Visualization and Computer Graphics*, 23:241–250, 2017b. 22, 96, 177

John W Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, 18(5):401–409, 1969. 144

Rakesh Kumar Sanodiya, Sriparna Saha, and Jimson Mathew. Semi-supervised orthogonal discriminant analysis with relative distance: integration with a moo approach. *Soft Computing*, 24(3):1599–1618, 2020. 35, 43

Ashutosh Saxena, Abhinav Gupta, and Amitabha Mukerjee. Non-linear dimensionality reduction by locally linear isomaps. In *Neural Information Processing*, pages 1038–1043. Springer, 2004. 35

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proc. CVPR*, pages 815–823, Boston, USA, Jun. 2015. 38, 65, 92

Gideon Schwarz. Estimating the Dimension of a Model. *The Annals of Statistics*, 6(2): 461–464, 1978. 146

Michael Sedlmair and Michaël Aupetit. Data-driven evaluation of visual quality measures. In *Computer Graphics Forum*, volume 34, pages 201–210, 2015. 178

Burr Settles. Active learning literature survey. 2009. 178

Guowan Shao, Fanmao Liu, and Chunjiang Peng. Semi-supervised uncertain linear discriminant analysis. In *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*, pages 149–160. Springer, 2020. 47

Xin Shu, Y. Gao, and H. Lu. Efficient linear discriminant analysis with locality preserving for face recognition. *Pattern Recognit.*, 45:1892–1898, 2012. 45, 53, 55

Vikas Sindhwani, Partha Niyogi, Mikhail Belkin, and Sathiya Keerthi. Linear manifold regularization for large scale semi-supervised learning. In *Proc. of the 22nd ICML Workshop on Learning with Partially Classified Training Data*, volume 28, 2005. 45, 46, 53, 55

Mike Sips, Boris Neubert, John P Lewis, and Pat Hanrahan. Selecting good views of high-dimensional data using class consistency. In *Computer Graphics Forum*, volume 28, pages 831–838, 2009. 178

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Proc. NIPS*, pages 2951–2959, Nevada, USA, Dec. 2012. 146, 169

Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in neural information processing systems*, pages 1857–1865, 2016. 92

Y. Song, F. Nie, and C. Zhang. Semi-supervised sub-manifold discriminant analysis. *Pattern Recognit. Lett.*, 29:1806–1813, 2008a. 44, 47, 53, 55

Yangqiu Song, Feiping Nie, Changshui Zhang, and Shiming Xiang. A unified framework for semi-supervised dimensionality reduction. *Pattern recognition*, 41(9):2789–2799, 2008b. 45, 46

David Spiegelhalter. *The art of statistics: Learning from data*. Penguin UK, 2019. 59, 103

Marc Strickert. No perplexity in stochastic neighbor embedding. In *Workshop New Challenges in Neural Computation*, pages 68–115, Graz, Austria, Aug 2012. 146

Masashi Sugiyama. Local fisher discriminant analysis for supervised dimensionality reduction. In *ICML*, 2006. 42, 53, 55

Masashi Sugiyama, Tsuyoshi Idé, Shinichi Nakajima, and Jun Sese. Semi-supervised local fisher discriminant analysis for dimensionality reduction. *Machine Learning*, 78:35–61, Jan. 2008. 43, 52, 53, 54

Tingkai Sun and S. Chen. Locality preserving cca with applications to data visualization and pose estimation. *Image Vis. Comput.*, 25:531–543, 2007. 49, 53, 55

Yaxin Sun, Qing Ye, Rong Zhu, and Guihua Wen. Cognitive gravity model based semi-supervised dimension reduction. *Neural Processing Letters*, pages 1–24, 2017. 44, 53, 55

Omer Tamuz, Ce Liu, Serge Belongie, Ohad Shamir, and Adam Tauman Kalai. Adaptively learning the crowd kernel. *arXiv preprint arXiv:1105.1033*, 2011. 37, 38, 178

Jian Tang, Jingzhou Liu, Ming Zhang, and Qiaozhu Mei. Visualizing large-scale and high-dimensional data. In *Proc. WWW*, pages 287–297, Montréal, Canada, Apr. 2016. 16

Wei Tang and Shi Zhong. Pairwise constraints-guided dimensionality reduction. *Computational Methods of Feature Selection*, pages 295–312, 2007. 33, 34, 36

Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000. 26, 30

10X Genomics. 1k brain cells from an e18 mouse, 2018. URL https://support.10xgenomics.com/single-cell-gene-expression/datasets/3.0.0/neuron_1k_v3. 153

Kaggle Open Datasets. Fashion product images dataset, 2019. URL https://www.kaggle.com/paramaggarwal/fashion-product-images-dataset. 152

Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999. 13, 28, 100, 120, 121, 131, 134

Bin Tong, Weifeng Jia, Yanli Ji, and Einoshin Suzuki. Linear semi-supervised dimensionality reduction with pairwise constraint for multiple subclasses. *IEICE TRANSACTIONS on Information and Systems*, 95(3):812–820, 2012. 36

Warren S Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17(4): 401–419, 1952. 13, 30, 107

Toon Van Craenendonck and Hendrik Blockeel. Constraint-based clustering selection. *Machine Learning*, 106(9):1497–1521, 2017. 173, 179

Laurens van der Maaten. Accelerating t-SNE using tree-based algorithms. *Journal of Machine Learning Research*, 15:3221–3245, 2014a. 19, 81

Laurens van der Maaten. Accelerating t-SNE using tree-based algorithms. *Journal of Machine Learning Research*, 15:3221–3245, 2014b. 81

Laurens van der Maaten and Kilian Weinberger. Stochastic triplet embedding. In *Machine Learning for Signal Processing (MLSP), 2012 IEEE International Workshop on*, pages 1–6. IEEE, 2012. 37, 38, 53, 54, 65

Laurens van der Maaten, Eric Postma, and Jaap van den Herik. Dimensionality reduction: a comparative. *Journal of Machine Learning Research*, 10:66–71, 2009. 12

Vincent van Unen, Thomas Höllt, Nicola Pezzotti, Na Li, Marcel JT Reinders, Elmar Eisemann, Frits Koning, Anna Vilanova, and Boudewijn PF Lelieveldt. Visual analysis of mass cytometry data by hierarchical stochastic neighbour embedding reveals rare cell types. *Nature communications*, 8(1740):1–10, 2017. 65

Jarkko Venna, Jaakko Peltonen, Kristian Nybo, Helena Aidos, and Samuel Kaski. Information retrieval perspective to nonlinear dimensionality reduction for data visualization. *Journal of Machine Learning Research*, 11(Feb.):451–490, 2010. 91

Michel Verleysen. Machine learning with limited size datasets. https://ijcci.scitevents.org/Documents/Previous_Invited_Speakers/2020/IJCCI_2020_KS_4_Presentation.pdf, 2020. [Keynote Lecture at IJCCI2020]. 14, 15

Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *The Journal of Machine Learning Research*, 11:2837–2854, 2010. 151

Viet Minh Vu and Benoit Frénay. User-steering interpretable visualization with probabilistic principal components analysis. In *Proc. ESANN*, pages 349–354, 2019. 29, 31, 52, 54, 95, 97, 120, 125

Viet Minh Vu, Adrien Bibal, and Benoit Frénay. Constraint preserving score for automatic hyperparameter tuning of dimensionality reduction methods for visualization. *IEEE Transactions on Artificial Intelligence*, 2:269–282, 2021a. 36, 139, 179

Viet Minh Vu, Adrien Bibal, and Benoit Frénay. HCt-SNE: Hierarchical constraints with t-SNE. In *Proc. IJCNN*, 2021b. 37, 38, 53, 54, 57, 58, 64, 72

Viet Minh Vu, Adrien Bibal, and Benoit Frénay. iPMDS: Interactive probabilistic multidimensional scaling. In *Proc. IJCNN*, 2021c. 29, 31, 52, 54, 95, 97, 107

Viet Minh Vu, Adrien Bibal, and Benoit Frénay. Integrating constraints into dimensionality reduction methods: a survey. *submitted to Neurocomputing*, 2021d. 21

De Wang, Feiping Nie, and Heng Huang. Unsupervised feature selection via unified trace ratio formulation and k-means clustering (track). In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 306–321. Springer, 2014. 43

H. Wang, C. Ding, and Heng Huang. Multi-label linear discriminant analysis. In *ECCV*, 2010. 50, 53, 55

Huan Wang, Shuicheng Yan, Dong Xu, Xiaoou Tang, and Thomas Huang. Trace ratio vs. ratio trace for dimensionality reduction. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007. 43

Na Wang, Xia Li, and Piao Chen. Global and local scatter based semi-supervised dimensionality reduction with active constraints selection in ensemble subspaces. *Pattern Analysis and Applications*, 20(3):733–747, 2017. 178

Zheng Wang, Y. Song, and C. Zhang. Transferred dimensionality reduction. In *ECML/PKDD*, 2008. 51

Martin Wattenberg, Fernanda Viégas, and Ian Johnson. How to use t-SNE effectively. *Distill*, 1(10):e2, 2016. 19, 20, 58, 60, 61, 62, 90, 142, 155

Michael Wilber, Iljung S Kwak, David Kriegman, and Serge Belongie. Learning concept embeddings with combined human-machine expertise. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 981–989, 2015. 37, 38, 53, 54

Leland Wilkinson, Anushka Anand, and Robert Grossman. Graph-theoretic scagnostics. In *Information Visualization, IEEE Symposium on*, pages 21–21. IEEE Computer Society, 2005. 174, 178

A. G. Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. *NeuRIP2020*, abs/2002.08791, 2020. 98

John Winn and Christopher M. Bishop. *Model-based Machine Learning*. 2018. URL http://mbmlbook.com. early access. 99

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv:1708.07747*, 2017. 81, 131, 152

Sicheng Xiong, Javad Azimi, and Xiaoli Z Fern. Active learning of constraints for semi-supervised clustering. *IEEE Transactions on Knowledge and Data Engineering*, 26(1): 43–54, 2013. 178

S. Yan, Dong Xu, B. Zhang, H. Zhang, Qiang Yang, and Stephen Lin. Graph embedding and extensions: A general framework for dimensionality reduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:40–51, 2007. 34

Shuicheng Yan and Huan Wang. Semi-supervised learning by sparse representation. In *SDM*, 2009. 34, 35

Su Yan, Sofien Bouaziz, Dongwon Lee, and Jesse Barlow. Semi-supervised dimensionality reduction for analyzing high-dimensional data with constraints. *Neurocomputing*, 76(1): 114–124, 2012. 35

Liu Yang and Rong Jin. Distance metric learning: A comprehensive survey. *Michigan State Universiy*, 2(2):4, 2006. 39

Xin Yang, Haoying Fu, Hongyuan Zha, and Jesse Barlow. Semi-supervised nonlinear dimensionality reduction. In *Proceedings of the 23rd International Conference on Machine learning*, pages 1065–1072. ACM, 2006. 26, 27, 31, 53, 54

Jieping Ye and Qi Li. A two-stage linear discriminant analysis via qr-decomposition. *IEEE transactions on pattern analysis and machine intelligence*, 27:929–41, 07 2005. doi: 10.1109/TPAMI.2005.110. 47

Guoxian YU, Hong PENG, Jia WEI, and Qianli MA. Robust locality preserving projections with pairwise constraints. 2010. 34, 53, 54

Changqing Zhang, Huazhu Fu, Q. Hu, P. Zhu, and Xiaochun Cao. Flexible multi-view dimensionality co-reduction. *IEEE Transactions on Image Processing*, 26:648–659, 2017. 50, 53, 55

Daoqiang Zhang, Zhi-Hua Zhou, and Songcan Chen. Semi-supervised dimensionality reduction. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 629–634. SIAM, 2007a. 34, 36

M. Zhang and Z. Zhou. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognit.*, 40:2038–2048, 2007. 178

M. Zhang and Z. Zhou. A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26:1819–1837, 2014. 50

W. Zhang, X. Xue, Zichen Sun, Y. Guo, and H. Lu. Optimal dimensionality of metric space for classification. In *ICML '07*, 2007b. 46

Yin Zhang and Z. Zhou. Multilabel dimensionality reduction via dependence maximization. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (2008)*, 2008. 50

Zhao Zhang, Tommy WS Chow, and Mingbo Zhao. Trace ratio optimization-based semi-supervised nonlinear dimensionality reduction for marginal manifold visualization. *IEEE Transactions on Knowledge and Data Engineering*, 25(5):1148–1161, 2013. 43

Zhenyue Zhang and Hongyuan Zha. Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *SIAM journal on scientific computing*, 26(1): 313–338, 2004. 26

Yuansheng Zhou and Tatyana O Sharpee. Using global t-SNE to preserve inter-cluster data structure. *bioRxiv*, 2018. 65

Joseph L. Zinnes and Richard A. Griggs. Probabilistic, multidimensional unfolding analysis. *Psychometrika*, 39:327–350, 1974. 100, 108

Joseph L. Zinnes and David B. MacKay. Probabilistic multidimensional scaling: Complete and incomplete data. *Psychometrika*, 48:27–48, March 1983. ISSN 1860-0980. doi: 10.1007/BF02314675. 28, 100, 108, 114, 131, 189