Dissimilarity-based learning for complex data

Bassam Mokbel

Dissertation

vorgelegt zur Erlangung des Grades
Doktor der Naturwissenschaften (Dr. rer. nat.)

Disputation am 22. Mai 2015

Universität Bielefeld, Technische Fakultät

First Examiner: Prof. Dr. Barbara Hammer, Bielefeld University, Germany
Second Examiner: Prof. Dr. Alessandro Sperduti, University of Padova, Italy

Submitted in January 2015, defended in May 2015, published in January 2016.

Printed on non-aging paper according to ISO 9706.

Bielefeld University – Faculty of Technology
P.O. Box 10 01 31
D-33501 Bielefeld, Germany

Bassam Mokbel
Theoretical Computer Science research group
CITEC – Cognitive Interaction Technology Center of Excellence
Inspiration 1, D-33619 Bielefeld, Germany
http://www.cit-ec.de/tcs
bmokbel@techfak.uni-bielefeld.de

Abstract

Rapid advances of information technology have entailed an ever increasing amount of digital data, which raises the demand for powerful data mining and machine learning tools. Due to modern methods for gathering, preprocessing, and storing information, the collected data become more and more complex: a simple vectorial representation, and comparison in terms of the Euclidean distance is often no longer appropriate to capture relevant aspects in the data. Instead, problem-adapted similarity or dissimilarity measures refer directly to the given encoding scheme, allowing to treat information constituents in a relational manner.

This thesis addresses several challenges of complex data sets and their representation in the context of machine learning. The goal is to investigate possible remedies, and propose corresponding improvements of established methods, accompanied by examples from various application domains. The main scientific contributions are the following:
(I) Many well-established machine learning techniques are restricted to vectorial input data only. Therefore, we propose the extension of two popular prototype-based clustering and classification algorithms to non-negative symmetric dissimilarity matrices.
(II) Some dissimilarity measures incorporate a fine-grained parameterization, which allows to configure the comparison scheme with respect to the given data and the problem at hand. However, finding adequate parameters can be hard or even impossible for human users, due to the intricate effects of parameter changes and the lack of detailed prior knowledge. Therefore, we propose to integrate a metric learning scheme into a dissimilarity-based classifier, which can automatically adapt the parameters of a sequence alignment measure according to the given classification task.
(III) A valuable instrument to make complex data sets accessible are dimensionality reduction techniques, which can provide an approximate low-dimensional embedding of the given data set, and, as a special case, a planar map to visualize the data's neighborhood structure. To assess the reliability of such an embedding, we propose the extension of a well-known quality measure to enable a fine-grained, tractable quantitative analysis, which can be integrated into a visualization. This tool can also help to compare different dissimilarity measures (and parameter settings), if ground truth is not available.
(IV) All techniques are demonstrated on real-world examples from a variety of application domains, including bioinformatics, motion capturing, music, and education.

Acknowledgments

I would like to thank my colleagues, friends, and family for the ongoing support during the creation of this thesis.

In particular, I am thankful to my loving girlfriend Kaja, who was incredibly understanding and supportive throughout these years.

Finally, I thank my supervisors Prof. Barbara Hammer and Prof. Alessandro Sperduti. I am ever grateful for Barbara's excellent guidance and lasting encouragement.

"Colorless green ideas sleep furiously."
– Noam Chomsky

Contents

1. Introduction **1**

1.1. Motivation 1
1.2. Data and representation 3
1.2.1. Workflow pipeline for machine learning applications 3
1.2.2. Challenges of complex data 7
1.2.3. Feature-based representation 9
1.2.4. Dissimilarity-based representation 12
1.2.5. Other types of data representation 15
1.3. Thesis overview 16
1.3.1. Scientific contributions 16
1.3.2. Structural overview 17
1.3.3. Publications and funding in the context of this thesis 19

2. Tools for supervised and unsupervised learning with dissimilarity data **23**

2.1. Motivation 23
2.1.1. Scientific contributions and structure of the chapter 25
2.2. Relational learning vector quantization 26
2.2.1. Introduction 26
2.2.2. Generalized learning vector quantization 27
2.2.3. Pseudo-Euclidean embedding of dissimilarity data 28
2.2.4. GLVQ for dissimilarity data 30
2.2.5. Reducing computational demand via Nyström approximation . . . 31
2.2.6. Interpretability of relational prototypes 32
2.2.7. Experiments 32
2.2.8. Concluding remarks 36
2.3. Relational generative topographic mapping 37
2.3.1. Introduction 37
2.3.2. Generative topographic mapping (GTM) 38
2.3.3. Relational GTM 39
2.3.4. Experiments 40
2.3.5. Concluding remarks 43

3. Adaptive metrics for complex data **45**
3.1. Motivation . . . 45
3.1.1. Scientific contributions and structure of the chapter . . . 47
3.2. Vector-based metric learning in LVQ . . . 48
3.2.1. Motion tracking data . . . 50
3.2.2. Proof-of-concept example . . . 53
3.3. Sequence alignment as a parameterized dissimilarity measure . . . 56
3.4. Learning scoring parameters from labeled data . . . 58
3.5. Practical implementation . . . 61
3.5.1. Algorithm overview . . . 61
3.5.2. Meta-parameters . . . 63
3.5.3. Proof-of-concept with artificial data . . . 64
3.5.4. RGLVQ error function surface . . . 66
3.5.5. Influence of crispness on the alignment . . . 68
3.6. Experiments with real-world data . . . 70
3.6.1. Experimental procedure . . . 70
3.6.2. Copenhagen Chromosomes . . . 71
3.6.3. Intelligent tutoring systems for Java programming . . . 74
3.6.4. Reducing computational demand . . . 77
3.7. Discussion . . . 81

4. Unsupervised suitability assessment for data representations **83**
4.1. Motivation . . . 83
4.1.1. Scientific contributions and structure of the chapter . . . 84
4.2. Low-dimensional Euclidean embeddings . . . 85
4.3. Quantitative quality assessment . . . 90
4.3.1. Principles of quality assessment for DR . . . 90
4.3.2. Evaluating DR based on the co-ranking matrix . . . 94
4.3.3. Point-wise quality measure . . . 95
4.3.4. Parameterization of the quality measure . . . 97
4.3.5. Experiments with real-world data . . . 106
4.3.6. Discussion . . . 117
4.4. Comparing dissimilarity-based representations . . . 118
4.4.1. Introduction . . . 118
4.4.2. How to compare dissimilarity measures? . . . 119
4.4.3. Comparison of metrics for the Euclidean vector space . . . 120
4.4.4. Comparison of non-Euclidean settings . . . 121
4.4.5. Discussion . . . 126

5. Conclusion **129**

A. Additional information **133**
A.1. Derivative of soft alignment 133
A.2. Information about the Chromosomes data set 137
A.3. Information about the Sorting data set 138

B. Publications in the context of this thesis **139**

Bibliography **143**

List of Tables

2.1. Classification accuracy of RGLVQ for benchmark data 34
2.2. Classification accuracy of RGLVQ with approximation techniques 35
2.3. Classification accuracy of RGTM for benchmark data 41
2.4. RGTM meta-parameters for visualization experiments 41

3.1. Accuracies of RGLVQ with metric adaptation for Chromosomes data . . . 73
3.2. Accuracies of RGLVQ with metric adaptation for Sorting data 77
3.3. Runtimes for random soft alignment derivatives 79
3.4. Softmin thresholding in RGLVQ metric learning for Chromosomes data . 79
3.5. Nyström approximation in RGLVQ metric learning for Chromosomes data 80

A.1. Symbol occurrence statistics for the Chromosomes data 137
A.2. Alphabet description for the Sorting data 138

List of Figures

1.1. Typical processing pipeline in machine learning 3

2.1. Visualization of RGLVQ exemplars for e-book data 37
2.2. RGTM and RSOM visualization of classical music data set 43
2.3. RGTM visualizations for different labeling thresholds 44

3.1. Schematic of Kinect motion capturing skeleton 51
3.2. Flag semaphore alphabet 52
3.3. GRLVQ relevance profile for flag pose data 53
3.4. GMLVQ relevance matrix for walking sequence data 55
3.5. RGLVQ classification accuracy with metric adaptation for toy data 65
3.6. Scoring matrices learned by RGLVQ with metric adaptation for toy data . 66
3.7. Cost surface for metric adaptation in RGLVQ for simplified Gap data . . 67
3.8. Cost surfaces for metric adaptation in RGLVQ for Replacement data . . . 68
3.9. Dynamic programming matrices for soft alignment under different crispness 69
3.10. Results of RGLVQ with metric adaptation for Chromosomes data 72
3.11. Results of RGLVQ with metric adaptation for Sorting data 76

4.1. Co-ranking matrix structure 95
4.2. Quality for t-SNE embedding of the Swiss Roll data 98
4.3. Artificial example for the parameterization of the quality measure 99
4.4. Schematics of the parameterization in the co-ranking matrix 100
4.5. Artificial benchmark data sets with three clusters 102
4.6. Quality curves for the artificial data with three clusters 104
4.7. Quality curves with different parameterization for artificial data 105
4.8. Pointwise quality for t-SNE embedding of the Swiss Roll data 106
4.9. Pointwise quality for embeddings of the Runner data 108
4.10. Quality curves for embeddings of the Runner data 109
4.11. Pointwise quality for PCA embedding of the Coil data 110
4.12. Quality curves for embeddings of the Coil data 110
4.13. Pointwise quality for t-SNE embedding of the COIL data 111
4.14. Quality curves for embeddings of the MNIST data 112

4.15. t-SNE embedding of the MNIST data 113
4.16. Pointwise quality for t-SNE embedding of the MNIST data 114
4.17. Runtime for random evaluations with the co-ranking matrix 115
4.18. Effects of subsampling on the quality measure 116
4.19. Artificial data with uniform and clustered distributions 121
4.20. Comparison of different L-norms for uniformly distributed data 122
4.21. Comparison of different L-norms for clustered data 122
4.22. Overall comparison via quality curves for artificial data 123
4.23. Comparison of different dissimilarity measures for App description texts . 124
4.24. Overall comparison of different dissimilarity measures for Java programs . 125
4.25. Pointwise comparison of different dissimilarity measures for Java programs 127

Chapter 1.

Introduction

Chapter overview *This chapter introduces the research topics presented in this work, and establishes some basic terminology as well as mathematical formalization. Our notion of complex data is presented, along with its particular challenges for machine learning and the important role of data representations in this context. The chapter closes with an overview of the key contributions and structure of the thesis.*

1.1. Motivation

Due to rapid advances of information technology in recent decades, very large amounts of digital data have become available in nearly every discipline and application field today. The possibilities to collect data grow significantly, with an increasing availability of high-resolution sensor technology, massive storage capacities, as well as pervasive networking and computing options. Given the overwhelming size and detail of modern data collections, it can become hard for human users to access the underlying information and extract knowledge, even for experts in the field. Therefore, research in *machine learning* and related areas[1] aims to provide techniques, which facilitate or automate the interpretation of given input data.

While their underlying methodology can take many shapes, the techniques usually address a general problem structure, which applies to a multitude of application domains. Typical problem types in machine learning are, for example, *regression*, *clustering*, *classification*, and *dimensionality reduction*; see [35, 16] for a comprehensive introduction. Accordingly, a generic interface for the input data is necessary, since knowledge and information can be represented in different ways in every application scenario. Hence, all given information must be arranged in a standardized input format, which we refer to as *data representation*. However, the treatment according to a generic problem framework

[1] The fields of data analysis, computational intelligence, neural computation, pattern recognition, information retrieval, statistical modeling, and machine learning all share the common principle to automatically abstract from given input data, and thereby infer knowledge.

with a structurally restricted representation poses a challenge: some types of data are difficult to handle, due to their inherent complexity. In this thesis, we identify particular aspects that characterize such *complex data*, and investigate their consequences for machine learning. We propose techniques to tackle these problems by facilitating and extending existing methods, and evaluate their properties in several real-world application examples.

In the following Section 1.2, we will describe different aspects that characterize data in the context of machine learning, and discuss the crucial role of data representations. After introducing the typical workflow in a machine learning scenario, we focus on two prevalent forms of data representation, the *feature-based* and the *dissimilarity-based* representation. Their strenghts and weaknesses with regard to complex data are debated, and exemplified in a simple application with a text database. Section 1.3 provides an overview of the thesis' structure, and presents the main scientific contributions in this work, linking to the core topics of the remaining chapters.

Mathematical notation The list below describes the general style of mathematical notation in this thesis:

Sets are capital letters of various fonts, e.g.	$X,\ \mathcal{A},\ B$
Data are roman letters, e.g.	$x, y \in X$
Data vectors are denoted by boldface roman letters, e.g.	$\mathbf{x}, \mathbf{y} \in \mathbb{R}^D$
Other vectors or items are greek and roman letters, e.g.	$\alpha \in \mathbb{R}^N,\quad b \in B$
Set elements are enumerated by superscripts, e.g.	$\{\mathbf{x}^1, \ldots, \mathbf{x}^N\} = X$
Typical **enumerators** are i, j, k, l, p, q, s, t, as in:	$\mathbf{x}^i, \mathbf{x}^j \in X$
Raw data items or **sequences** are overlined letters, e.g.	$\bar{a}^i, \bar{a}^j \in \mathcal{A}$
Entries in a vector are the corresponding italic letter, enumerated by subscripts per dimension:	$\mathbf{x}^i = \left(x_1^i, \ldots, x_D^i\right) \in \mathbb{R}^D$
Exponentials of enumerated items are parenthesized:	$(x_k^i)^2 = x_k^i \cdot x_k^i$
Matrices are boldface capital letters (if possible), e.g.	$\mathbf{X},\ \mathbf{D},\ \boldsymbol{\Omega}$
The **transpose** is signified by the $^\intercal$ symbol:	$\mathbf{x}^\intercal,\ \mathbf{X}^\intercal,\ (\mathbf{x}^i)^\intercal$
Entries may also be addressed by a **bracket notation**,	
... the k-th **element** in vector $\mathbf{x}$:	$\left[\mathbf{x}^i\right]_k = x_k^i$
... the element at **row** i, **column** k of matrix $\mathbf{X}$:	$[\mathbf{X}]_{(i,k)} = x_k^i$
... **all** elements in **row** i, or respectively **column** i:	$[\mathbf{X}]_{(i,\cdot)} = \mathbf{x}^i = [\mathbf{X}^\intercal]_{(\cdot,i)}$
... sometimes we will use a **shorthand**:	$\mathbf{X}_{ik} = [\mathbf{X}]_{(i,k)}$

1.2. Data and representation

1.2.1. Workflow pipeline for machine learning applications

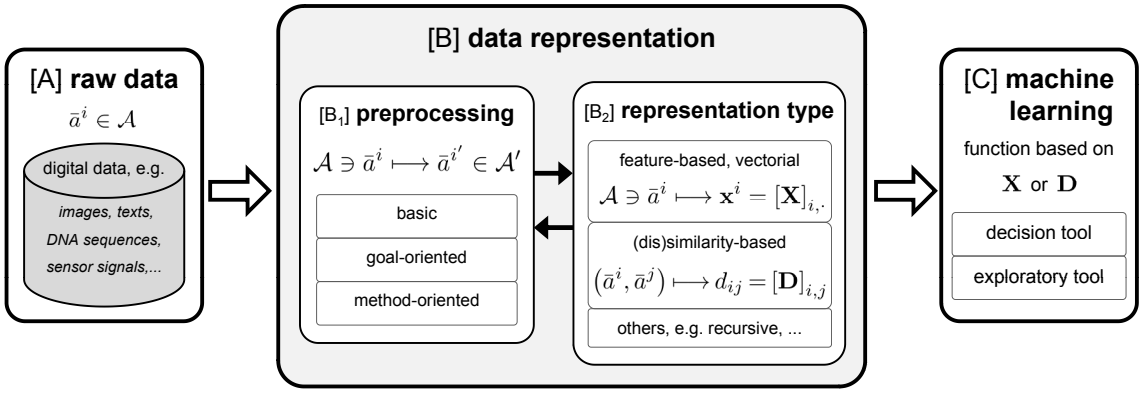


Figure 1.1.: The typical *processing pipeline* for a machine learning application: A collection of raw data is transferred to a generic data representation, e.g. based on features or pairwise (dis)similarities, often via several preprocessing steps. Thereafter, a machine learning method can be applied to create a model of the given data, typically with the possibility to extend the model to unseen data of the same form. This model can serve as an automatic decision scheme, or an exploratory tool for a human expert of the respective domain.

So far, we described the benefits of machine learning techniques, and their ability to process digital data in a generic framework. In this thesis, we will recognize *machine learning* as a means to create a model of a data collection. To realize this goal in a technical system, the workflow is typically organized as a chain of processing stages. A general, abstract overview of such a *processing pipeline* is illustrated in Figure 1.1.

The following example demonstrates a typical realization of this workflow in practice: A database of scanned handwritten digits (0-9) is given in the form of grayscale images with 16-by-16 pixels, corresponding to the *"raw data"* in block [A]. With several preprocessing techniques (from the established image recognition literature), the pictures are rotated and centered according to the visible cipher, see block [B_1] *"preprocessing"*. Each image is then turned into a real-valued vector by concatenating the rows of pixel values, see block [B_2] *"representation type"*. Finally, in block [C] *"machine learning"*, the resulting set of vectors (together with expert annotations of the correct digit in the respective image) is used to train a classifier model, e.g. a *support vector machine* (SVM), which is able to recognize some percentage, e.g. 96%, of the images correctly.

The specific implementation of all steps in the workflow depends heavily on the application goal, and it involves crucial design decisions. This ranges from the appropriate representation of information to the selection of a particular machine learning technique,

as well as meta-parameters. Generally, the choice of a representation is highly non-trivial, especially with regard to complex data and their structural characteristics. In practice, the challenges of workflow design are typically faced with human ingenuity and informed decisions based on expertise in the particular application domain. Ultimately, domain-specific knowledge and hand-crafted adaptation by human experts remain a neccessity in most practical scenarios. In recent literature, this fundamental problem received more and more attention, and methods to circumvent domain-specific, hand-tailored design are addressed in the context of novel research branches, such as *autonomous learning*[2] [34]. This thesis marks one step towards the autonomous, task-driven learning of representations for complex data. In the following, we will take a closer look at the single stages in the described workflow.

[A] **Raw information** In the beginning of the pipeline, we have digital information available in some predefined format. We will refer to this as *raw data* or *raw information*. For example, data could be digital images (in a photo database), texts (product descriptions in an online store), DNA sequences (in a biological directory of bacteria), sensor signals (recorded by a robotics platform), or many other kinds of digital information from various application scenarios. The given format needs to be clearly interpretable, and it must allow to distinguish single entities, i.e. *instances*, of data. For example, if the data came from sensors in an industrial machine, one could consider the readings of all sensors at a distinct point in time as a single instance. Typically, a finite collection of instances forms a *data set*, for example a record of historical sensor readings.

[B] **Data representation** In order to use machine learning techniques, the raw information needs to be represented in an appropriate form. This *data representation* provides a single, generic interface to machine learning methods in the subsequent pipeline stage. We will discuss some established types of representation in the following sections, focusing on the two prevalent forms: *feature-based* and *dissimilarity-based* representations. They have different strengths and weaknesses in their ability to reproduce the original raw information. Additionally, machine learning methods are often limited to a specific form of input data. We will therefore examine the representation's characteristic properties as well as limitations in Sections 1.2.3 and 1.2.4, and exemplify the consequences in a practical application scenario.

[B_1] *Preprocessing* Transferring raw information to a generic data representation is a crucial step which usually involves hand-tailored solutions according to expert knowledge in the application field. Several preprocessing steps may be performed during this transition, in order to prepare the data representation in favor of a given machine learning problem. We will roughly distinguish three types of preprocessing operations, with

[2] `http://www.autonomous-learning.org`

regard to their purpose (although they cannot be seen as fully separable in practice):

(I) *Basic preprocessing* steps are applied to make the data representation robust in terms of technical and numerical aspects. This may include simple normalization steps, like a *whitening* transformation [35]. One particular intention is the reduction of *noise*, which is a common phenomenon in real-world data. For example, electronic sensors often yield small numerical fluctuations in their readings, due to technical or physical limitations and uncertainty. Based on expert knowledge about the given application, one could therefore remove small changes in the data representation via a simple thresholding or averaging scheme, without dismissing critical information.

(II) *Goal-oriented preprocessing* aims to prepare the data representation in such a way, that important information w.r.t. the given application is strongly expressed, while unwanted and misleading information is reduced or entirely removed. A typical example is to account for *invariances* in the data: given a machine learning problem, it may be beneficial to treat some pairs of data instances as equal, although their representation differs. Therefore, the preprocessing steps should yield equal representations for the respective pairs. In this way, the resulting representation becomes invariant w.r.t. certain changes in the raw data, and it ignores some information which is irrelevant for the given problem.

(III) *Method-oriented preprocessing* is based on the fact that machine learning methods are often bound to a specific type of data representation, for theoretical or technical reasons. Some methods may require a certain numerical domain for the input values, or assume a specific distribution therein. If an algorithm requires input values that constitute a notion of proximity between data instances, it would need either similarity, or dissimilarity values (distances being a special case thereof). In many practical applications, problem-specific (dis)similarity measures are available for this purpose. Depending on the required input format for the machine learning method (similarities or dissimilarities), we can perform a transformation in either direction, i.e. converting similarities to dissimilarities, and vice versa. The mathematical background, as well as standard procedures, can be found in the literature, see [106], for example. We will address this particular topic in more detail for dissimilarity-based data representations, in Section 1.2.4.

[B_2] *Representation type* Generally, preprocessing is used to prepare the data representation, but also to alter an existing one. Several consecutive operations may be necessary to arrive at the intended representation, with the desired properties. Therefore, even when a generic format is accomplished (like a set of feature vectors), additional preprocessing might be performed, as indicated by the forward and backward arrows between [B_1] and [B_2] in Figure 1.1. Common types of data representations are described in more

detail in Sections 1.2.3 and 1.2.4. While subsequent machine learning techniques mainly process the information contained in the given representation, they may incorporate additional knowledge. This could be expert knowledge about an assumed data distribution, or expected output values, which are known to be correct for certain inputs. A very common case of additional knowledge are available *class labels*: discrete categories to which given data instances are attributed, and which constitute referential evidence in the application scenario. Since not all machine learning techniques require this extra information, we do not include it explicitly in our description of data representations, or in our schematic of the processing pipeline. Instead, it is assumed to be provided by an expert user who controls the overall machine learning workflow.

[C] **Machine learning** After a data representation is fixed, it can be used as input for an adequate machine learning algorithm. This algorithm creates a model of the available input data, during a *training* procedure. The model can be realized in various forms. Typically, we have a mapping of every input to an output value or vector, however, this mapping might not be accessible in an explicit functional form. Therefore, extending the trained model to unseen input data can sometimes require additional effort. A large variety of methods is readily available for different purposes. The exact goal depends on the application scenario. In the following, we will refer to this as the *learning goal* or *learning problem*. In this thesis we will focus on two general purposes of machine learning:

- *As an exploratory tool, to provide assistance for field experts:*
  This can make large collections of digital data more accessible, for example by clustering the data set, or by creating a visualization based on a low-dimensional embedding. The learning process is usually *unsupervised*, meaning that known output values (e.g. class-labels) are not included in the training. However, available labels can provide helpful indication, when domain experts want to inspect and explore the data model.

- *As a discrete decision tool, to automatically categorize input data:*
  Based on given data samples, a classification model can be trained, and is thereafter able to assign new, unseen data to the predefined classes, respectively. In this case, available class labels are incorporated in the training, also known as *supervised* learning. (Since we are only interested in discrete decisions, we will not consider machine learning techniques for regression in this thesis.)

Although there are specialized techniques for either one purpose, some methods address both objectives with one learning model.

1.2.2. Challenges of complex data

In this section, we will briefly characterize complex data and the specific challenges for machine learning.

Example – literature database To exemplify the nature of complex data, let us assume a practical scenario: a hypothetical collection of electronic books from the English fictional literature, where we can access the full text of every book in a standardized digital format. Suppose that the raw data of book number i in the collection $\mathcal{A}$ is given as a sequence $\bar{a}^i \in \mathcal{A}$, with entries $\bar{a}^i = \left(a_1^i, \ldots, a_I^i, \ldots, a_{|\bar{a}|}^i\right) \in \Sigma_{\mathsf{char}}^{|\bar{a}|}$ comprised of alphabetic characters and punctuation symbols in the alphabet Σ_{char}. (This example is inspired by the *Project Gutenberg*[3] online database, which will be addressed in experiments in later chapters.) Let us assume that our learning goal is a clustering of all books, as an exploratory tool for field experts: unsupervised clustering can help experts to gain a structured overview of the entire database, in order to identify thematic groups and styles, for example, to distinguish genres like science fiction, historical novels, romantic literature, etc.

One aspect of complexity is the sheer length of sequences: books are likely to contain more than 500,000 characters, with an average of 5 letters per word in the English language. Very long sequences are common in some application fields, such as bioinformatics, where DNA or RNA strings are processed, or in industrial applications, where discrete time series are addressed. In this particular example, the sequence lengths can be reduced effectively, by shifting the symbolic domain to a higher level of abstraction: instead of characters as the basic symbolic alphabet, we can consider words. Given the raw texts, we can easily extract a respective sequence of words by scanning the text for delimiters, such as white spaces and punctuation marks. Thereby, the set of all appearing unique terms in the entire book collection yields a symbolic alphabet Σ_{term}, on which these word sequences are based. However, the reduction to roughly one fifth of the original length is then traded for a considerably larger alphabet: while $|\Sigma_{\mathsf{char}}|$ would be less than 100 symbols, the number of unique words $|\Sigma_{\mathsf{term}}|$ would likely surpass 20,000, even if different grammatical forms of words are not distinguished[4].

Other complex characteristics, which are common in real-world data, are structural relations. Texts written in a natural language exhibit intricate structural properties. The most obvious is the sequential succession of characters (and words): a *sequence* is characterized by the fact that symbols are ordered and thus have a relation to their respective predecessor or successor. Additionally, we can observe the hierarchical forming of symbolic entities: a certain sequence of letters forms a term, and a certain group of terms forms a phrase or a sentence. This hierarchy follows *syntactic* and grammatical

[3] `http://www.gutenberg.org`

[4] See `http://www.mine-control.com/zack/guttenberg/` for some statistical observations on the Project Gutenberg database.

rules of the language. If we consider symbolic entities in this hierarchy (like terms or phrases), there are additional *semantic* relationships among them, which are tightly linked to the established syntactic structures. The main reason is that the semantic meaning of terms is highly contextual. For example, when an adjective is used in a predicative manner, it is related to a noun (or pro-noun) and thereby modifies the noun's meaning. These relationships lead to further structural formation.

Aspects of complex data The example demonstrates several aspects of complexity in real-world data. On an abstract level, we can summarize the following challenges which characterize complex data sets:

- *Curse of dimensionality:* In complex data, many different types of information can be present. This heterogeneity leads to many degrees of freedom, so that a large number of features is required for its description. In terms of vectors, this corresponds to a very high dimensionality. At the same time, data are often sparse, due to the sheer amount of possible feature combinations in addition to the fact that only few descriptors are usually present for every given data point. This causes many machine learning algorithms to suffer from the *curse of dimensionality*, a well-known effect regarding high-dimensional spaces, see e.g. [15].

- *Big data:* The term "big data" describes an emerging research topic, which addresses limitations of current machine learning techniques regarding very large data sets, see [63, 140]. These issues overlap significantly with the challenges of complex data, where a high number of instances and/or high dimensionality is common. Due to aspects of compositionality (see below), it may be hard or even impossible to treat data instances separate from their context in the data set, or to isolate individual constituents of the data. A fast random access to the entire data set may thus be required, which implies that the data representation must fit entirely into the working memory of the machine.

- *Compositionality:* The encoding of complex data often involves syntactic structures, with pervasive relationship rules, and inherent contextual dependencies. Therefore, it is hard to isolate basic constituents of the data from their context. At the same time, identifying distinct relationships between syntactic elements can be difficult. As a result, there may be infinite possibilities to form structural entities, e.g. by grouping possibly related constituents. The basic problems of compositionality in complex data have been addressed in the literature, see [49].

To make complex data sets accessible for established machine learning methods, the data representation plays an important role. In the following, we will explain the feature-based, and the dissimilarity-based representation in more detail, and discuss in how far complex aspects can be captured with these data description schemes.

1.2.3. Feature-based representation

One common way of representing a collection of data is based on *features*, i.e. a fixed set of quantitative attributes or criteria, for which a corresponding numeric value can be provided for every data instance. Assuming a fixed ordering of features, each data instance (also referred to as *sample*) is therefore described by a finite tuple of values, and can thus be expressed as a *feature vector* $\mathbf{x} = (x_1, \ldots, x_D) \in \mathbb{R}^D$ in the D-dimensional Euclidean space. We assume that a finite number of N instances is given in a data set, so we have a set of vectors: $\mathbf{x}^i \in X \subset \mathbb{R}^D,\ i \in \{1, \ldots, |X| = N\}$. The individual vectors are usually assembled to form the rows of a matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$, where $[\mathbf{X}]_{i,k} = \left[\mathbf{x}^i\right]_k = x^i_k,\ i \in \{1, \ldots, N\}, k \in \{1, \ldots, D\}$. This matrix $\mathbf{X}$ is often referred to as the *data table*[5]. The choice and formal definition of quantitative features is typically hand-tailored to a specific application or learning goal. According to this specification, attributes are collected and prepared for each data instance. This process is called *feature extraction* in the literature, and may involve various operations to recover, analyze, or evaluate properties from raw digital information.

Since data are thereby defined as points in a vector space, we can directly refer to the *Euclidean distance*[6] between vectors. Given data $\mathbf{x}^i$ and $\mathbf{x}^j$, it is defined as

$$\|\mathbf{x}^i - \mathbf{x}^j\| = \sqrt{\sum_{k=1}^{D} \left(x^i_k - x^j_k\right)^2}\ .$$

A feature-based representation is generally applicable to many different kinds of information, as long as a fixed set of features can be extracted from the raw data. Consequently, there is a large variety of established machine learning algorithms available, which require inputs in the form of feature vectors, see [35, 73, 114]. However, the simplicity and generality of the format involves certain caveats. As stated earlier, it is not clear a priori which representation is well-suited for a given problem. This extends to the choices of defining and extracting features in a constructive manner. In practice, these design decisions are typically based on expertise in the particular application domain.

We assume that every feature has a meaning in the context of the given application, hence the semantic interpretation of vector dimensions is straightforward. However, it is also possible to obtain data vectors where this is not the case. For example, vectors may result from a low-dimensional embedding of high-dimensional data, e.g. by referring to the popular *principal component analysis* (PCA) [35]. Thereby, the semantic interpretation of vector dimensions becomes more complex, or may not exist at all. This describes a generalization of the feature-based data representation, which we refer to as a *vectorial* representation.

[5] By convention, data $\mathbf{x}$ are row vectors (and rows of $\mathbf{X}$) in this thesis. For the sake of coherence with some referenced literature, we will explicitly point out when data are treated as column vectors.

[6] Machine learning algorithms typically address the derivative of the distance function, wherefore it is common to consider squared Euclidean distances $\|\mathbf{x}^i - \mathbf{x}^j\|^2$, for the sake of simplicity.

Example – feature representation of literature collection In our example from Section 1.2.2, a collection of e-books is given in the form of plain text sequences. For these raw data, we can define a set of features in a straightforward fashion, by referring to the statistics of words occurring in each text. To measure word occurrences, let us assume a simple dictionary for now, which consists of the most common 1000 nouns in the English language, where $k \in \{1, \ldots, D = 1000\}$ refers to each individual noun. Then, the book $\bar{a}^i$ can be represented as a vector $\mathbf{x}^i \in \mathbb{R}^D$, in which every entry x_k^i states, how often the k-th term of the dictionary occurs in this book.

Given the goal to distinguish books according to major topics, the data representation should capture the overall thematic content of a book. Therefore, we can enhance the described feature extraction by including basic preprocessing steps, well-known in text analysis and natural language processing. For example, the *Porter stemming* algorithm [108] reduces all terms in the texts (and in the dictionary) to their word stem, so that different grammatical forms and inflections are no longer distinguished. Thereby, the data representation becomes invariant to different grammatical usages of the same basic terms. Another common preprocessing step is to scale the term frequencies inversely according to their total number of occurrences in the entire book collection. This decreases the numerical impact of very common (and usually less-discriminative) words in relation to rare terms in the corpus.

Representing complex data with vectors According to the previously described challenges of complex data, we will examine the limitations of a vectorial data representation.

Curse of dimensionality When dealing with complex data, it is typical that a high number of features is necessary to sufficiently describe a data instance. In our example, the number of features D is determined by the size of the considered dictionary, and it directly affects at what level of detail the statistics of term occurrences are realized. With a large corpus of texts containing diverse themes, the vocabulary can be very different among the books. To cover this variety, and potentially account for unseen data, which is not in the collection so far, the dictionary must have a reasonable size. This leads to high-dimensional, but only sparsely populated vectors. Usually, it is not clear a priori, which features are relevant. An abundance of irrelevant information can cause the distribution and structure of the data to be unnecessarily complex and not favorable for the given problem. Dimensionality reduction techniques can provide help in this case, and we will discuss this topic in detail in Chapter 4.

Big data The memory demand of the data table $\mathbf{X}$ has a complexity of $\mathcal{O}(N \cdot D)$, meaning that a very high number of instances (e.g. $N > 200,000$) in conjunction with a very large quantity of features (e.g. $D > 30,000$) can lead to a data table, which does not

fit into the working memory of a common computer[7]. However, the dimensionality of the vectorial representation is fixed priorly, so that the memory demand increases only linearly with additional data. Moreover, one can rely on operations in Euclidean vector spaces to circumvent memory issues, such as vector quantization, feature selection, and vectorial matrix decompositions. Hence, vectorial representations are a good solution to tackle the computational demand of big data.

Compositionality The possibilities to incorporate aspects of compositionality in a feature-based data representation are inherently limited. Given the simple tabular structure, the data description is highly restricted, as compared to intricate encoding schemes that are possible in the original raw information. In our example, each book consists of a sequence of terms $\bar{a}^i \in \mathcal{A}$, with entries $\bar{a}^i = \left(a^i_1, \ldots, a^i_I, \ldots, a^i_{|\bar{a}^i|}\right) \in \Sigma^{|\bar{a}^i|}_{\mathsf{term}}$. A feature representation based on the occurrence statistics of single terms (i.e. symbols in Σ_{term}) fully neglects the symbol's sequential order, among other structures, such as syntax, grammar, sentence separation, etc. The only possibility is to define features that take relations into account and convey them quantitatively. For example, to incorporate the sequential order to a certain degree, we can refer to the occurrence statistics of subsequences, so-called *n-grams* of symbols. n-grams are tuples in Σ^n_{term}, typically with a length of $1 < n < 10$, in practice. One obvious drawback is that the alphabet size increases exponentially by $|\Sigma_{\mathsf{term}}|^n$. Consequently, if the frequencies of all possible[8] n-grams are recorded as features, the memory complexity becomes $\mathcal{O}\left(N \cdot D^n\right)$. Hence, even for relatively small D, the problems of high dimensionality apply.

Implicit structural features There exists a very elegant trick to get around the computational burden posed by n-grams, provided subsequent machine learning algorithms rely on pairwise dot products only: the n-gram representation can be computed only implicitly, relying on the popular kernel trick [118]. This way, structure kernels arise, such as the spectrum kernel or string kernel, allowing for an efficient computation which is polynomial in n, by relying on dynamic programming or suffix trees [90, 84]. Similar ideas have also been proposed for more general data structures, such as trees or graphs, although there exist principled problems which prohibit an efficient computation of expressive kernels for complex graph structures [41, 98]. We will not investigate structure kernels in detail in this thesis, rather we take the more general view of a dissimilarity-based representation of data, as described in the following section.

[7] A double-precision floating point number typically uses 64 bits of memory, i.e. 8 bytes. Therefore, an array of $200,000 \cdot 30,000$ values would take ≈ 48 gigabytes.

[8] In natural language text, many n-grams of words could be excluded, because they are semantically incorrect and would never appear in any text.

1.2.4. Dissimilarity-based representation

In contrast to extracting features for every single instance of raw data, we can consider pairs of instances, and evaluate how much they have in common, or, conversely, how different they are. Treating pairs of data instead of single items constitutes a fundamentally different approach. We will distinguish three general ways to obtain a (dis)similarity-based data representation from raw information:

(I) *The given information naturally exhibits pairwise proximity values:*
In some cases, the available data directly yields pairwise relations in a quantitative manner, such as the connectivity strengths in social networks, travel distances between geographic locations, or opinion-based surveys about item similarities.

(II) *An algorithm evaluates the proximity between given pairs of raw data:*
Using specific algorithms, we can numerically assesses some notion of *proximity*[9] between raw data items. Many different techniques are available for this purpose, often specialized for a certain data format, and with a specific application field in mind, e.g. the *dynamic time warping* measure to compare audio signals [101].

(III) *Pairwise distances are calculated based on an existing vectorial representation:*
Whenever a vectorial representation of the data is available, as described in the previous section, any mathematical distance expression for vectors can be utilized. Apart from the common Euclidean distance, one can use (parameterized) variants, such as L_p-metrics, general quadratic forms, or divergence measures.

We will focus on the options (II) and (III), where an algorithm or a mathematical function – in the following called *(dis)similarity measure* – is applied to obtain pairwise proximity values. Formally, we can define a (dis)similarity measure as a function $d : \mathcal{A} \times \mathcal{A} \to \mathbb{R}$, which assigns a non-negative scalar to every given pair $\left(\bar{a}^i, \bar{a}^j\right) \in \mathcal{A}^2$ of raw (preprocessed) data, as

$$\left(\bar{a}^i, \bar{a}^j\right) \longmapsto d(\bar{a}^i, \bar{a}^j) = [\mathbf{D}]_{i,j}$$

where $\mathbf{D}$ is the *(dis)similarity matrix* holding all pairwise results. We will occasionally use the shorthand $d_{ij} = d(\bar{a}^i, \bar{a}^j)$, when the reference to concrete data is not necessary, or it is clear from the context.

It can be show, that a similarity matrix can always be transferred to a corresponding dissimilarity matrix, without loosing information. The inverse is not true. In terms of representing data, dissimilarities are therefore a more general format than similarities. Hence, we will refer to *dissimilarity measures* and *dissimilarity-based* data representations exclusively, in the following, whereby similarities are implicitly included in our discussion.

[9] Throughout this thesis, we will use *proximity* as a general term: depending on the given context, it can mean either the similarity, or the dissimilarity between the respective entities.

Relationship between vectorial and dissimilarity-based data representations Option (III) in our list states a simple observation: if vectorial data $\mathbf{x}^i \in \mathbb{R}^D$ are given in the Euclidean space $\mathbb{R}^D$, we can directly obtain a dissimilarity-based representation. Referring to the canonical metric based on the vector norm, we can obtain the Euclidean distance $d_{ij} = \|\mathbf{x}^i - \mathbf{x}^j\|$ for all pairs (i, j). In contrast, when a pairwise dissimilarity measure is given, with values $d(\bar{a}^i, \bar{a}^j)$, then an accurate *Euclidean embedding* might not exist. That is, there may not exist a mapping function $\psi : \mathcal{A} \to \mathbb{R}^{D'}$ to points in a real vector space $\mathbb{R}^{D'}$, so that their distances accurately reflect the given dissimilarities, i.e. ψ satisfies

$$d(\bar{a}^i, \bar{a}^j) = \|\psi(\bar{a}^i) - \psi(\bar{a}^j)\| \quad \text{for all} \quad i, j \in \{1, \ldots, |\mathcal{A}|\}\,.$$

Therefore, a *dissimilarity-based representation* is a more general format, compared to a *feature-based* or *vectorial representation*. This gain relies on the fact that a dissimilarity function is less restrictive than a metric distance (of which the Euclidean distance is a special case).

So far, we required only a non-negative output in our definition of a dissimilarity function. In the remainder of this thesis, we will typically assume the following properties for a dissimilarity-based data representation: *non-negativity*, *symmetry*, and the *identity of indiscernibles* (also known as *reflexivity*). That is, for all i, j it holds:

- $d_{ij} \geq 0$ (*non-negativity*),
- $d_{ij} = d_{ji} \iff \mathbf{D} = \mathbf{D}^\intercal$ (*symmetry*),
- $d_{ii} = 0 \iff \mathbf{D}$ has zero diagonal (*identity of indiscernibles*).

We may refer to this type of dissimilarities as *relational* data, in the following. These requirements are common in practice, for several reasons: Non-negativity and zero self-dissimilarities are intuitive conditions for any natural notion of dissimilarity between items. In addition, many machine learning algorithms rely on symmetry to maintain certain mathematical properties, for example a symmetrical invariance within the underlying objective function, such as the quantization error [76]. In the definition of practical dissimilarity measures, the three stated conditions are often inherently satisfied. A metric distance is additionally required to obey the *triangle inequality*. By dropping this restriction, dissimilarity-based representations become more flexible and general. However, the three stated conditions still ensure that there exists an accurate embedding of the given dissimilarities to vectorial distances in a *pseudo-Euclidean space*. This refers to an indefinite inner product space, which provides a less restrictive concept of pairwise distances, as compared to the classical Euclidean space, and it will be addressed in Chapters 2 and 3.

Examples of dissimilarity measures A simple and yet powerful dissimilarity measure for symbolic sequences is the *normalized compression distance* (NCD) [87]. The NCD is

an approximation of a theoretical, universal distance measure for symbolic strings, the so-called "normalized information distance", which is uncomputable. This theoretical distance relies on the uncomputable Kolmogorov complexity of a string [88], which can, however, be approximated by a real-world compression algorithm. If $\bar{a}$ is a symbolic string, let $\mathcal{C}(\bar{a})$ be the length of a byte sequence returned by a lossless compression algorithm, from which the exact input sequence can be reconstructed. For two sequences $\bar{a}^i, \bar{a}^j$, and some compatible real-world compression function $\mathcal{C}$, the NCD is defined as

$$d_{\mathsf{NCD}}(\bar{a}^i, \bar{a}^j) = \frac{\mathcal{C}\left(\bar{a}^i \bullet \bar{a}^j\right) - \min\left\{\mathcal{C}\left(\bar{a}^i\right), \mathcal{C}\left(\bar{a}^j\right)\right\}}{\max\left\{\mathcal{C}\left(\bar{a}^i\right), \mathcal{C}\left(\bar{a}^j\right)\right\}}$$

where $(\bar{a}^i \bullet \bar{a}^j)$ is the concatenation of strings. The working principle is based on the fact that the compressor utilizes recurring patterns in the string, in order to encode it more efficiently and achieve a reduction in size. If $\bar{a}^i$ and $\bar{a}^j$ exhibit shared patterns, then these similarities will facilitate the compression of their concatenation $\mathcal{C}\left(\bar{a}^i \bullet \bar{a}^j\right)$. See [87] for an elaborate formal introduction of the NCD measure. The normalized information distance, due to its theoretical definition, can be seen as a metric in the precise mathematical sense, and thus it also fulfills the three conditions stated above [87]. However, its practical counterpart, the NCD, often yields numerical deviations, due to its inherent approximation based on a real-world compressor. To create a dissimilarity-based representation using the NCD, we therefore assume simple numerical corrections to ensure that $\mathbf{D}$ is non-negative, symmetrical, and has a zero diagonal.

Other examples for popular (dis)similarity measures are

- the *Hamming distance* from information theory [55]
- *alignment* functions, popular in bioinformatics applications [122]
- the *earth-mover's distance*, used in image retrieval and pattern recognition [113]
- the *Jaccard index* or *Tanimoto coefficient* from statistics [62]

Example – dissimilarity representation of literature collection Let us return to our previous example of English books. Given any dissimilarity measure d for books $\bar{a}^i \in \mathcal{A}$, we can create a dissimilarity matrix $\mathbf{D}$ with entries $[\mathbf{D}]_{i,j} = d(\bar{a}^i, \bar{a}^j)$. According to option (III), on page 12, we can obtain dissimilarities by referring to an existing vectorial representation. In this case, we can use the features of term frequencies, established in the previous section, and denote the Euclidean distance between corresponding feature vectors $\mathbf{x}^i, \mathbf{x}^j$ as $d_{\mathsf{tf}}(\bar{a}^i, \bar{a}^j) = \|\mathbf{x}^i - \mathbf{x}^j\|$. In contrast, we can use the NCD measure $d_{\mathsf{NCD}}(\bar{a}^i, \bar{a}^j)$, referring to option (II), on page 12. However, d_{tf} dismisses the sequential structure of terms, so a stylistic expression in the ordering of the words is not captured by the data representation. Instead, d_{NCD} is based on a compression algorithm, which typically uses a sliding-window-technique to find recurring patterns in a given sequence.

Therefore, the NCD dissimilarity will reflect differences in the sequential patterns of the given strings, and is not invariant to a reordering of words.

Representing complex data via dissimilarities As in the previous section, we will debate in how far this type of representation is affected by the challenges of complex data.

Curse of dimensionality The dimensionality of a dissimilarity-based representation can be determined by referring to the *pseudo-Euclidean* embedding of given dissimilarities, see Chapter 2 and [106]. An upper bound for its dimensionality is given by the number of instances in the data set, i.e. the dimensionality scales with N. Therefore, the underlying space is always limited to the dimensionality necessary to cover the known data, and we avoid potential overhead to represent yet unseen information in future data. This means that problems regarding the curse of dimensionality are avoided, unless a very large number of instances is addressed.

Big data The memory complexity of a dissimilarity matrix is $\mathcal{O}(N^2)$, since all pairs of instances are addressed. If a subsequent machine learning method relies on a fast access to the entire matrix, the number of instances must be restricted to fit the data into the working memory, e.g. up to $N \approx 40,000$ for modern desktop computers. This inherent quadratic memory complexity poses a major limitation when addressing big data sets with dissimilarity-based representations. However, possible remedies have been proposed in the literature, such as low-rank matrix approximation techniques like the *Nyström approximation* [137], which we will utilize in Chapters 2 and 3.

Compositionality A significant advantage of a dissimilarity-based representation is that structural aspects of the data can be incorporated, if they are addressed by the mechanics of the dissimilarity function. The context and structure of data constituents can be compared by algorithmic means. For example, alignment measures are designed specifically to compare strings by aligning similar sequential patterns. In order to address more diverse aspects of compositionality, it is possible to combine the results from different dissimilarity functions, or even nested comparison algorithms.

1.2.5. Other types of data representation

Another very popular way to represent complex data is in terms of non-vectorial structures which explicitly encode the correlation of data constituents. Examples for such structures are sequences of possibly unlimited length, tree structures or graph structures. Naturally, such representations require machine learning technology which is capable of dealing with such complex structures instead of mere vectorial data, kernels, or dissimilarities only. Successful approaches cover recursive networks, graph networks, or statistical relational models [53, 52, 8, 99, 115]; however, we will not consider these complex machine learning models in this thesis. Rather, we will focus on ways how

to better handle complex data if represented in vectorial form or in terms of pairwise dissimilarities only.

1.3. Thesis overview

1.3.1. Scientific contributions

In the previous sections, we have identified challenges of complex data in the context of machine learning. The next chapters introduce techniques to address some of the raised issues, focusing on the general problem of representing complex data sets and making them accessible for machine learning methods. The following are the main scientific contributions of this thesis:

(A) Interface of established classification models to dissimilarity data

Section 1.2.4 explains how a dissimilarity-based data representation is better suited to incorporate aspects of compositionality from the given raw information, as compared to feature vectors. However, many well-established machine learning techniques are restricted to vectorial input data only. Therefore, we propose the extension of two popular prototype-based clustering and classification methods to non-negative symmetric dissimilarities (so-called *relational data*):

(i) the *relational learning vector quantization* (relational LVQ), – an intuitive supervised classification scheme, based on Hebbian learning principles;

(ii) the *relational generative topographic mapping* (relational GTM) – an unsupervised learning method, useful for data inspection and visualization.

Both techniques offer an easy access and interpretation of the resulting model, which constitutes a particular convenience when addressing complex data sets.

(B) Task-driven learning of dissimilarity-based data representations

So far, we discussed different generic formats to represent data, which serve as a single interface to machine learning methods. The representation typically encodes available information without taking their relevance for the learning goal into account. Therefore, several so-called *metric learning* techniques have been proposed to automatically adapt the data representation with respect to a specific task, via its underlying metric, see [11] for an overview. While this is well-established for vector quantization based on feature representations, see [119, 121, 43], we propose to transfer this principle to adapt parameterized dissimilarity measures in conjunction with the classifier training itself. This establishes a first step towards the autonomous, task-driven learning of representations for complex data.

(C) Expert tools to assess the suitability of data representations

Machine learning methods for dimensionality reduction provide a low-dimensional vectorial embedding of a given data set, while preserving the original (high-dimensional) neighborhood structure as well as possible. They are a valuable instrument to make complex data accessible, for example by visualizing the data set in two or three dimensions. Since dimensionality reduction usually implies information loss, we propose a fine-grained extension to a quality assessment technique, which indicates the reliability of the embedding for every given data instance. Additionally, we transfer this principle to enable an unsupervised comparison between different dissimilarity-based data representations arising from the same raw information. Both techniques can be integrated directly in embedded visualizations, like 2D or 3D scatter plots of the data. This provides tools to investigate the suitability of a data representation visually and quantitatively w.r.t. a certain application.

(D) Diverse practical application examples

This thesis proposes techniques to facilitate and extend existing machine learning algorithms, with regard to complex data and their representation. To demonstrate the capability of each method, we address corresponding real-world problems from a variety of application domains; some of them were investigated in collaboration with field experts. The underlying problems and experimental results are presented in the corresponding chapters, covering the following topics:

- *intelligent tutoring systems* (ITSs), which provide computer-based assistance and feedback for students, e.g. to learn programming skills;
- sequential data from several *bioinformatics* databases;
- *motion capturing* for human pose detection, as well as motion sequences
- symbolic sequences in the context of *music information retrieval*, as well as *text mining* data sets from e-books and smartphone applications.

1.3.2. Structural overview

The remainder of this thesis is structured as follows:

Chapter 2 "Tools for supervised and unsupervised learning with dissimilarity data" (p. 23)
In this chapter, contribution **(A)** is presented, where two well-established clustering and classification models (LVQ and GTM) are extended by an interface for dissimilarity data. We first introduce LVQ, and propose its extension to *relational LVQ*. Its classification performance is evaluated on benchmark sets of dissimilarity data, before briefly highlighting the interpretable classifier model on an example from the literature database Project Gutenberg.

Thereafter, GTM is described, with its counterpart for dissimilarity data, *relational GTM*. Again, its capabilities as a classifier are demonstrated for benchmark data sets, and the inherent possibility to visualize the data model is exemplified for a set of symbolic sequences derived from a classical music database.

Chapter 3 "Adaptive metrics for complex data" (p. 45)

This chapter covers contribution **(B)**, where we propose the learning of metric parameters to adjust a dissimilarity-based data representation in favor of a given classification task. We first review an established metric learning scheme, based on feature vectors for LVQ. Thereafter, we transfer this principle to relational LVQ, where parameters of an alignment dissimilarity measure for sequences are adapted to improve the classification accuracy. Metric learning is illustrated and evaluated on several real-world examples, using motion capturing data, educational data from the ITS domain, and sequences from bioinformatics.

Chapter 4 "Unsupervised suitability assessment for data representations" (p. 83)

In this chapter, we cover the contribution **(C)**, which addresses a quantitative analysis to assess the suitability of data representations. First, we briefly review established dimensionality reduction techniques, which provide a visual overview of the neighborhood structures in a given data set. We discuss a quality measure to evaluate how accurately these neighborhoods are represented, before extending the existing method to allow for more fine-grained inspection and control. Thereafter, the principle is transferred to an unsupervised comparison of dissimilarity-based data representations. These techniques are demonstrated on several benchmark data sets, including motion capturing frames, as well as a text database of smartphone applications.

Chapter 5 "Conclusion" (p. 129)

The last chapter summarizes this work and points out future research perspectives.

1.3.3. Publications and funding in the context of this thesis

The following articles have been published in the context of this thesis:
(More detailed references are provided in the Appendix Section B, on page 139.)

Journal articles

[J15] B. Mokbel, B. Paassen, F.-M. Schleif, and B. Hammer. Metric learning for sequences in relational LVQ. *Neurocomputing*, (accepted/in press), 2015.

[J14] S. Gross, B. Mokbel, B. Paassen, B. Hammer, and N. Pinkwart. Example-based feedback provision using structured solution spaces. *Int. J. of Learning Technology*, 9(3):248–280, 2014.

[J13] B. Mokbel, W. Lueks, A. Gisbrecht, and B. Hammer. Visualizing the quality of dimensionality reduction. *Neurocomputing*, 112:109–123, 2013.

[J12] A. Gisbrecht, B. Mokbel, F.-M. Schleif, X. Zhu, and B. Hammer. Linear time relational prototype based learning. *Int. J. of Neural Systems*, 22(5), 2012.

[J11] A. Gisbrecht, B. Mokbel, and B. Hammer. Relational generative topographic mapping. *Neurocomputing*, 74(9):1359–1371, April 2011.

Conference articles

[C14c] B. Mokbel, B. Paassen, and B. Hammer. Efficient adaptation of structure metrics in prototype-based classification. In *ICANN 2014*, pages 571–578, 2014.

[C14b] B. Mokbel, B. Paassen, and B. Hammer. Adaptive distance measures for sequential data.[10] In *ESANN 2014*, pages 265–270, 2014.

[C14a] S. Gross, B. Mokbel, B. Hammer, and N. Pinkwart. How to select an example? A comparison of selection strategies in example-based learning. In *ITS 2014*, pages 340–347, 2014.

[C13d] A. Gisbrecht, B. Hammer, B. Mokbel, and A. Sczyrba. Nonlinear dimensionality reduction for cluster identification in metagenomic samples. In *IV 2013*, pages 174–179, 2013.

[C13c] S. Gross, B. Mokbel, B. Hammer, and N. Pinkwart. Towards providing feedback to students in absence of formalized domain models. In *AIED 2013*, pages 644–648, 2013.

[C13b] B. Mokbel, S. Gross, B. Paassen, N. Pinkwart, and B. Hammer. Domain-independent proximity measures in intelligent tutoring systems. In *EDM 2013*, pages 334–335, 2013.

[C13a] S. Gross, B. Mokbel, B. Hammer, and N. Pinkwart. Towards a domain-independent ITS middleware architecture. In *ICALT 2013*, pages 408 – 409, 2013.

[C12g] S. Gross, B. Mokbel, B. Hammer, and N. Pinkwart. Feedback provision strategies in intelligent tutoring systems based on clustered solution spaces. In *DeLFI 2012*, pages 27–38, 2012.

[10] Winner of the *best student paper* award at ESANN 2014.

[C12f] B. Mokbel, S. Gross, M. Lux, N. Pinkwart, and B. Hammer. How to quantitatively compare data dissimilarities for unsupervised machine learning? In *ANNPR 2012*, volume 7477 of *LNCS*, pages 1–13, 2012.

[C12e] F.-M. Schleif, B. Mokbel, A. Gisbrecht, L. Theunissen, V. Dürr, and B. Hammer. Learning relevant time points for time-series data in the life sciences. In *ICANN 2012*, volume 7553 of *LNCS*, pages 531–539, 2012.

[C12d] A. Gisbrecht, B. Mokbel, and B. Hammer. Linear basis-function t-SNE for fast nonlinear dimensionality reduction. In *IJCNN 2012*, pages 1–8, 2012.

[C12c] B. Mokbel, W. Lueks, A. Gisbrecht, M. Biehl, and B. Hammer. Visualizing the quality of dimensionality reduction. In *ESANN 2012*, pages 179–184, 2012.

[C12b] A. Gisbrecht, W. Lueks, B. Mokbel, and B. Hammer. Out-of-sample kernel extensions for nonparametric dimensionality reduction. In *ESANN 2012*, pages 531–536, 2012.

[C12a] B. Hammer, B. Mokbel, F.-M. Schleif, and X. Zhu. White box classification of dissimilarity data. In *HAIS 2012*, volume 7208 of *LNCS*, pages 309–321, 2012.

[C11c] B. Hammer, B. Mokbel, F.-M. Schleif, and X. Zhu. Prototype-based classification of dissimilarity data. In *IDA 2011*, pages 185–197, 2011.

[C11b] B. Hammer, M. Biehl, K. Bunte, and B. Mokbel. A general framework for dimensionality reduction for large data sets. In *WSOM 2011*, pages 277–287, 2011.

[C11a] B. Hammer, A. Gisbrecht, A. Hasenfuss, B. Mokbel, F.-M. Schleif, and X. Zhu. Topographic mapping of dissimilarity data. In *WSOM 2011*, pages 1–15, 2011.

[C10d] B. Mokbel, A. Gisbrecht, and B. Hammer. On the effect of clustering on quality assessment measures for dimensionality reduction. In *NIPS Workshop on Challenges in Data Visualization*, 2010.

[C10c] A. Gisbrecht, B. Mokbel, and B. Hammer. The Nyström approximation for relational generative topographic mapping. In *NIPS Workshop on Challenges in Data Visualization*, 2010.

[C10b] A. Gisbrecht, B. Mokbel, A. Hasenfuss, and B. Hammer. Visualizing dissimilarity data using generative topographic mapping. In *KI 2010*, pages 227–237, 2010.

[C10a] A. Gisbrecht, B. Mokbel, and B. Hammer. Relational generative topographic map. In *ESANN 2010*, pages 277–282, 2010.

[C09a] B. Mokbel, A. Hasenfuss, and B. Hammer. Graph-based representation of symbolic musical data. In *GbRPR 2009*, volume 5534 of *LNCS*, pages 42–51, 2009.

Non-refereed publications

[TR12] B. Mokbel, M. Heinz, and G. Zentgraf. Analyzing motion data by clustering with metric adaptation. In *Proc. of ICOLE 2011*, number MLR-01-2012 in Machine Learning Reports, pages 70–79, 2012. ISSN: 1865-3960.

[TR11] W. Lueks, B. Mokbel, M. Biehl, and B. Hammer. How to evaluate dimensionality reduction? - improving the co-ranking matrix. *CoRR*, abs/1110.3917, 2011.

Funding acknowledgments

The following institutions and associated grants are gratefully acknowledged:

- The *Cognitive Interaction Technology Center of Excellence* ('CITEC', EXC 277), funded by the *German Science Foundation* (DFG)
- The project *Learning Feedback in Intelligent Tutoring Systems* (FIT) in the Priority Programme 1527 *"Autonomous Learning"*, funded by the *German Science Foundation* (DFG) under grant number HA 2719/6-1.
- A travel scholarship by the *German Academic Exchange Service* (DAAD).

Chapter 2.

Tools for supervised and unsupervised learning with dissimilarity data

Chapter overview *This chapter presents the extension of two well-established clustering and classification models (LVQ for supervised, and GTM for unsupervised learning) by an interface for dissimilarity data. We arrive at "relational LVQ" and "relational GTM", in which the benefits of dissimilarity-based data representations can be utilized, while an intuitive access to the classifier model remains possible. Due to an inherently large memory complexity, we introduce and evaluate approximation techniques for relational methods, in the particular case of relational LVQ.*

Parts of this chapter are based on:

[J12] A. Gisbrecht, B. Mokbel, F.-M. Schleif, X. Zhu, and B. Hammer. Linear time relational prototype based learning. *Int. J. of Neural Systems*, 22(5), 2012.

[C12a] B. Hammer, B. Mokbel, F.-M. Schleif, and X. Zhu. White box classification of dissimilarity data. In *HAIS 2012*, volume 7208 of *LNCS*, pages 309–321, 2012.

[J11] A. Gisbrecht, B. Mokbel, and B. Hammer. Relational generative topographic mapping. *Neurocomputing*, 74(9):1359–1371, April 2011.

[C10a] A. Gisbrecht, B. Mokbel, and B. Hammer. Relational generative topographic map. In *ESANN 2010*, pages 277–282, 2010.

2.1. Motivation

In the previous Chapter, we have characterized complex data, and the problem that simple feature vectors may not be appropriate to capture the underlying information. Instead, the data can be represented via pairwise proximity values: problem-adapted similarity or dissimilarity measures address the raw data instances in pairs, and thereby refer directly to the given encoding scheme, allowing to treat information constituents in a relational manner. Such measures are widely used in many application fields: In bioinformatics tools, the comparison between raw data instances (e.g. symbolic sequences, mass spectra, or metabolic networks) is driven by complex alignment techniques, back-

ground information, or general principles of information theory, see [107, 94, 61]. Another example is the popular *earth-mover's distance* [113] used in image processing, which takes accumulated pixel transformations into account to compare the given images. Multimedia applications process audio, video, and motion sequences via specialized alignment algorithms, such as the well-known *dynamic time warping*, see [101].

Often, the resulting dissimilarity-based data representations cannot be transferred directly to an equivalent vectorial description of the data: if the proximity measure does not fulfill the properties of a metric, an accurate Euclidean embedding of the data does not exist. In addition, even if a truthful Euclidean embedding exists, its dimensionality can be as large as the number of data instances N, and the worst case complexity of its computation is $\mathcal{O}(N^3)$. Hence, it is infeasible for many practical applications. This constitutes a limitation for several classical machine learning and data mining tools in their original form. For example, various clustering and classification algorithms have been proposed for vectorial input data only, such as the popular *learning vector quantization* (LVQ) [73, 114] for supervised learning, which relies on data vectors; as well as, the *self-organizing map* (SOM) [73], *neural gas* (NG) [97], and the *generative topographic mapping* (GTM) [17] for unsupervised learning.

Therefore, many kernel-based classifiers, as well as "kernelized" extensions of established methods have been proposed in the last decades, which offer the possibility to address pairwise similarities as input data, given by inner products in a (potentially unknown) *kernel space*[1]. The most well-known and well-investigated example is the *support vector machine* (SVM) [30], which first introduced the kernel trick. Extensions of classical methods are, e.g., *kernel NG* [109, 118] and *kernel SOM* [93, 96, 19]; for an overview, please refer to [37]. However, kernel techniques are limited to positive semi-definite similarity matrices, as we will elaborate in Subsection 2.2.3.

On the other hand, the learning tasks become more and more complex, so that the specific objectives and the relevant information are not clear a priori. This leads to increasingly interactive systems, which allow humans to shape the objectives according to human insights and expert knowledge at hand and to extract the relevant information on demand [67]. This principle requires intuitive interfaces to the machine learning technology which enable humans to interpret the way in which decisions are taken by the system. Hence these requirements lead to the necessity that machine learning techniques provide information, which can directly be displayed to the human observer.

Although techniques like SVM or *Gaussian processes* [110] provide efficient state-of-the-art algorithms with excellent classification ability, it is often not easy to manually inspect the way in which decisions are taken. Hence, it is hard to visualize its decisions to domain experts in such a way that the results can be interpreted and valuable knowledge can be inferred based thereon. The same argument, although to a lesser degree, is valid for alternatives such as the *relevance vector machine* [127] or sparse models, which,

[1] In the literature, the kernel space is also called *feature space*.

though representing decisions in terms of sparse vectors or class representatives, typically still rely on complex nonlinear combinations of several terms [127, 26].

Dissimilarity- or similarity-based machine learning techniques such as nearest neighbor classifiers rely on distances of given data to known labeled data points. Hence it is usually very easy to visualize their decision: the closest data point or a small set of closest points can account for the decision, and this set can directly be inspected by experts in the same way as any data instance. Because of this simplicity, (dis)similarity techniques enjoy a large popularity in application domains, whereby the methods range from simple *k-nearest neighbor* classifiers (k-NN) [35] to advanced techniques, such as *affinity propagation* (AP) [39], which represents a clustering in terms of typical exemplars.

(Dis)similarity-based techniques can be distinguished by different criteria:

- The number of data used to represent the classifier, ranging from dense models, such as k-NN, to sparse representations, like prototype-based methods. To arrive at easily interpretable models, a sparse representation in terms of few data points is necessary.

- The degree of supervision, ranging from unsupervised clustering techniques, like AP, to supervised learning methods, taking class labels into account.

- The complexity of the dissimilarity measure which the methods can deal with, ranging from vectorial techniques restricted to Euclidean spaces, adaptive techniques which learn the underlying metric [119], up to tools which can deal with arbitrary (dis)similarities [50]. Typically, Euclidean techniques are well-suited for simple classification scenarios, but fail if high dimensionality or complex structures are encountered.

2.1.1. Scientific contributions and structure of the chapter

In this chapter, we propose two additions to the range of available clustering and classification methods for dissimilarity data, which offer particularly interesting features regarding complex data sets. – Both methods are based on sparse prototypes that cover the data, and allow for an easy inspection of the classifier model:

Relational LVQ In Section 2.2, we present an extension to LVQ: a supervised technique, which is based on intuitive Hebbian learning principles and offers model transparency via class exemplars in the data space.

Relational GTM In Section 2.3, we briefly introduce an extension to GTM: an unsupervised method, which includes the possibility to visualize class structures in a low-dimensional map, similar to SOM, but with inherent regularization options.

The key ingredient is taken from existing approaches in the domain of unsupervised learning [50, 106]: if prototypes are represented implicitly as linear combinations of data in a so-called *pseudo-Euclidean* embedding or, more generally, a *Krein space* (see [106, p.77]), all necessary distances between data and prototypes can be computed without an explicit reference to a vectorial representation. This principle holds for relational data, i.e. a non-negative, symmetric dissimilarity matrix with zero diagonal.

2.2. Relational learning vector quantization

2.2.1. Introduction

Learning vector quantization (LVQ) constitutes one of the few methods to infer a sparse representation in terms of prototypes from a given data set in a supervised way [73]. Hence, it offers a good starting point as an intuitive classification technique, in which decisions can directly be inspected by humans. Although original LVQ has been introduced on rather heuristic grounds [73], recent developments in this context provide a solid mathematical derivation of its generalization ability and learning dynamics: explicit large-margin generalization bounds of LVQ classifiers are available [31, 119]; further, the dynamics of LVQ-type algorithms can be derived from cost functions which model the classification accuracy referring to the hypothesis margin or a statistical model, for example [119, 121]. Interestingly, already the dynamics of simple LVQ, as proposed by Kohonen, provably leads to surprisingly good generalization characteristics when investigated in the framework of the theory of online learning [14].

When dealing with modern application scenarios, one of the largest drawbacks of LVQ type classifiers is their dependency on the Euclidean metric. Because of this, LVQ is not suited for complex or heterogeneous data sets where input dimensions have different relevance or where high dimensionality leads to accumulated noise disrupting the classification. This problem can partially be avoided by vector-based metric learning approaches, see e.g. [119], which turn LVQ classifiers into state-of-the-art techniques, e.g. for applications involving humanoid robotics, computer vision, or medical diagnostics, see [33, 69, 6]. We will investigate this approach later, in Chapter 3.

However, if data are inherently non-Euclidean, these techniques cannot be applied. In this section, we propose an extension of *generalized LVQ* (GLVQ) [114, 119] to general dissimilarity data; GLVQ being a popular LVQ-type algorithm derived from a cost function which is related to the hypothesis margin. This way, the technique becomes directly applicable for data sets, which are characterized in terms of relational data only. Interestingly, the classification performance is comparable to the state-of-the-art, but GLVQ additionally offers an intuitive interface in terms of prototypes [28].

Due to its dependency on the dissimilarity matrix, relational GLVQ has squared complexity, and the computation of the dissimilarities often constitutes the bottleneck in applications. By integrating approximation techniques [137], the effort can be reduced

to linear time and memory complexity. We demonstrate the feasibility of this approach with the popular *SwissProt* protein database [18].

2.2.2. Generalized learning vector quantization

We assume a set of given data $x^i \in X,\ i = 1, \ldots, N$. In the case of classical LVQ, we assume that data are vectors $x^i = \mathbf{x}^i \in \mathbb{R}^D$. Prototypes $\mathbf{w}^j \in \mathbb{R}^D,\ j = 1, \ldots, M$ decompose the data into *receptive fields*

$$R(\mathbf{w}^j) = \left\{ \mathbf{x}^i \;\middle|\; d(\mathbf{x}^i, \mathbf{w}^j) \leq d(\mathbf{x}^i, \mathbf{w}^k) \text{ for all } k = 1, \ldots, M \right\},$$

where d is the squared Euclidean distance $d(\mathbf{x}^i, \mathbf{w}^j) = \|\mathbf{x}^i - \mathbf{w}^j\|^2$. In broad terms, the goal of vector quantization techniques is to find prototypes, which represent a given data set as accurately as possible, so that the representatives 'cover' the data. In supervised learning, all data $\mathbf{x}^i$ are equipped with class labels $c(\mathbf{x}^i) \in \{1, \ldots, L\}$, where L is the total number of classes. Similarly, every prototype carries a priorly fixed label $c(\mathbf{w}^j)$.

A data point is classified according to the class of its closest prototype. This assignment can be evaluated via the *classification accuracy* $\sum_{j=1}^{M} \sum_{\mathbf{x}^i \in R(\mathbf{w}^j)} \hat{\delta}\big(c(\mathbf{x}^i) = c(\mathbf{w}^j)\big)/N$ with the Kronecker delta function $\hat{\delta}$. As an explicit objective for optimization, this function is not a good choice, due to vanishing gradients and discontinuities. Therefore, LVQ relies on a reasonable heuristic by performing Hebbian updates of the prototypes, given a data point, see [73]. Recent alternatives derive similar update rules from explicit objective functions, which are related to the classification accuracy, but display better numerical properties such that efficient optimization is possible [119, 114, 121].

Given a data point $\mathbf{x}^i$, we will use $\mathbf{w}^+(\mathbf{x}^i)$ to refer to the prototype which is closest to $\mathbf{x}^i$ and has a matching label $c\big(\mathbf{w}^+(\mathbf{x}^i)\big) = c(\mathbf{x}^i)$. The squared distance between them is $d\big(\mathbf{x}^i, \mathbf{w}^+(\mathbf{x}^i)\big)$, and will be denoted as $d^+(\mathbf{x}^i)$, as a shorthand. Accordingly, $\mathbf{w}^-(\mathbf{x}^i)$ refers to the closest prototype with a different label $c\big(\mathbf{w}^-(\mathbf{x}^i)\big) \neq c(\mathbf{x}^i)$, and the corresponding distance is denoted by $d^-(\mathbf{x}^i)$.

Generalized LVQ (GLVQ) [114] relies on a cost function, which can be related to the generalization ability of the classifier [119]. GLVQ minimizes the error term

$$E_{\mathrm{GLVQ}} = \sum_{i}^{N} E^i_{\mathrm{GLVQ}} = \sum_{i}^{N} \Phi\left(\frac{d^+(\mathbf{x}^i) - d^-(\mathbf{x}^i)}{d^+(\mathbf{x}^i) + d^-(\mathbf{x}^i)}\right), \tag{2.1}$$

where Φ is a differentiable monotonic function, such as the hyperbolic tangent or the sigmoid. Since a data point $\mathbf{x}^i$ is classified correctly, if and only if $d^-(\mathbf{x}^i)$ is larger than $d^+(\mathbf{x}^i)$, this cost function constitutes a reasonable choice. It has been shown that the difference $d^+(\mathbf{x}^i) - d^-(\mathbf{x}^i)$ can be related to the so-called hypothesis margin of LVQ classifiers, a quantity which directly regulates the generalization ability of the resulting classifier [119]. For numerical reasons, this numerator is normalized to the interval $[-1, 1]$ to prevent divergence of the prototypes.

For the learning algorithm, update rules can be derived thereof, by means of standard gradient techniques. After presenting the data sample $\mathbf{x}^i$, its closest correct and wrong prototype, respectively, are adapted according to the rules:

$$\begin{aligned}\Delta\mathbf{w}^+(\mathbf{x}^i) \sim & -\Phi'\big(\mu(\mathbf{x}^i)\big)\cdot\mu^+(\mathbf{x}^i)\cdot\nabla_{\mathbf{w}^+(\mathbf{x}^i)}d^+(\mathbf{x}^i)\\ \Delta\mathbf{w}^-(\mathbf{x}^i) \sim & \;\;\Phi'\big(\mu(\mathbf{x}^i)\big)\cdot\mu^-(\mathbf{x}^i)\cdot\nabla_{\mathbf{w}^-(\mathbf{x}^i)}d^-(\mathbf{x}^i)\end{aligned} \tag{2.2}$$

where

$$\mu(\mathbf{x}^i)=\frac{d^+(\mathbf{x}^i)-d^-(\mathbf{x}^i)}{d^+(\mathbf{x}^i)+d^-(\mathbf{x}^i)}\ ,$$

$$\mu^+(\mathbf{x}^i)=\frac{2\cdot d^-(\mathbf{x}^i)}{\big(d^+(\mathbf{x}^i)+d^-(\mathbf{x}^i)\big)^2}\ ,\quad \mu^-(\mathbf{x}^i)=\frac{2\cdot d^+(\mathbf{x}^i)}{\big(d^+(\mathbf{x}^i)+d^-(\mathbf{x}^i)\big)^2}\ .$$

For the squared Euclidean distance, the derivative yields $\nabla_{\mathbf{w}^j}d(\mathbf{x}^i,\mathbf{w}^j)=-2(\mathbf{x}^i-\mathbf{w}^j)$, leading to Hebbian update rules of the prototypes according to the class information. GLVQ constitutes one particularly efficient method to adapt the prototypes according to a given labeled data set. Alternatives can be derived based on a labeled Gaussian mixture model, see e.g. [121]. Since the latter can be highly sensitive to model meta-parameters [14], we focus on GLVQ.

2.2.3. Pseudo-Euclidean embedding of dissimilarity data

In this section, we recall and transfer theoretical insights from [106, 50, 28] for our cause. We assume that data x^i are represented by non-negative pairwise dissimilarities $d_{ij}=d(x^i,x^j)$, and $\mathbf{D}\in\mathbb{R}^{N\times N}$ refers to the corresponding dissimilarity matrix. We also assume symmetry $d_{ij}=d_{ji}$ and a zero diagonal $d_{ii}=0$, and thereby refer to *relational data*, as explained in Chapter 1. d_{ij} may not be Euclidean distances, i.e. for pairs (x^i,x^j) it is not guaranteed that Euclidean vectors $(\mathbf{x}^i,\mathbf{x}^j)$ can be found with $d_{ij}=\|\mathbf{x}^i-\mathbf{x}^j\|$.

However, for every such dissimilarity matrix $\mathbf{D}$, there exists an embedding in a so-called *pseudo-Euclidean space*, which is a vector space with an indefinite inner product, see [106]. The squared pairwise distances in this pseudo-Euclidean embedding yield exactly the dissimilarities in $\mathbf{D}$. Accordingly, an associated similarity matrix $\mathbf{S}$ exists, which contains the pairwise inner products calculated in this space. Even though the explicit embedding may not be known to us, we can obtain the inner products from the given dissimilarities, by so-called *double centering*, as explained in [128, p.258] and [106]:

$$\mathbf{S}=-\mathbf{U}\mathbf{D}\mathbf{U}/2\quad\text{where}\quad\mathbf{U}=(\mathbf{I}-\mathbf{1}\mathbf{1}^\intercal/N), \tag{2.3}$$

with the identity matrix denoted as $\mathbf{I}$ and an N-dimensional column vector of ones $\mathbf{1}$.

We can explicitly determine the pseudo-Euclidean embedding via an eigenvector decomposition of $\mathbf{S}$: Let p be the number of positive eigenvalues, and q be the number of negative eigenvalues. Then, a symmetric bilinear form is induced by $\langle\mathbf{x}^i,\mathbf{x}^j\rangle_{p,q}=(\mathbf{x}^i)^\intercal\mathbf{I}_{p,q}\mathbf{x}^j$, where $\mathbf{I}_{p,q}$ is a diagonal matrix with p entries 1, and q entries -1. Taking

the eigenvectors of $\mathbf{S}$ multiplied by the square root of their absolute eigenvalues, we can obtain vectors $\mathbf{x}^i$ in pseudo-Euclidean space, so that $d_{ij} = \langle \mathbf{x}^i - \mathbf{x}^j, \mathbf{x}^i - \mathbf{x}^j \rangle_{p,q}$ holds for every pair of data, see [106].

Hence, we can obtain facts about the embedding, by observing the eigenspectrum of $\mathbf{S}$: If and only if $\mathbf{S}$ is positive semidefinite (*psd* in the following), then the inner products originate from a Euclidean vector space, which is a special case of the pseudo-Euclidean space. Correspondingly, the entries in $\mathbf{D}$ are squared Euclidean distances. However, if some eigenvalues are negative, a Euclidean space is not sufficient to represent the given (dis)similarities. In general, we can summarize these characteristics of $\mathbf{S}$ by a *signature*: the tuple $(p, q, N - p - q)$, in which $N - p - q$ are the remaining zero eigenvalues.

Positive semi-definite similarity matrices are also referred to as *kernel* matrices or *Gram* matrices, and many machine learning methods require this type of similarity data, such as the popular *support vector machine* (SVM) [30]. If $\mathbf{S}$ is not *psd*, a correction of the matrix can be employed, at the risk of some information loss. The following techniques are common: the spectrum of the matrix $\mathbf{S}$ is changed via operations such as *clip* (negative eigenvalues are set to 0), *flip* (absolute values are taken), or *shift* (a summand is added to all eigenvalues). For a detailed explanation, please refer to [106, 28]. Interestingly, some operations, such as *shift*, do not affect the location of local optima in some important cost functions, like the quantization error, see [76]. However, the transformation can severely affect the performance of optimization algorithms, see [50]. As an alternative, data points can be constructed as vectors, in which the elements are given by the similarities to all other data. Then, standard distance measures or kernel functions are applied to pairs of these vectors, as if they were classic feature vectors. In the following, this correction technique is referred to as *similarities as features*. In [28], an extensive comparison of these preprocessing methods was conducted for a variety of benchmarks with SVM.

All of the named operations which involve an eigenvalue decomposition have a computational complexity of $\mathcal{O}(N^3)$, for example the explicit embedding in the pseudo-Euclidean space, as well as checking and correcting the signature to ensure a *psd* kernel. In the following sections, we will utilize the fact that a pseudo-Euclidean embedding must exist for relational data. However, by addressing it only implicitly, we will circumvent the computational burden to create the explicit embedding, as proposed in [50].

Conversely, if the data are given by symmetric similarities in a matrix $\mathbf{S}$, they can be seen as inner products between vectors in an (unknown) pseudo-Euclidean embedding. We can determine the corresponding squared distances, as

$$[\mathbf{D}]_{(i,j)} = [\mathbf{S}]_{(i,i)} + [\mathbf{S}]_{(j,j)} - 2\,[\mathbf{S}]_{(i,j)} \ . \tag{2.4}$$

This conversion requires $\mathcal{O}(N^2)$ time. Hence, it is generally possible to create relational data (i.e. dissimilarities) from such similarities, and we will omit a separate discussion of equivalent similarity-based data representations in the remainder of this thesis.

2.2.4. GLVQ for dissimilarity data

Vector operations in learning algorithms can be directly transferred to the pseudo-Euclidean space, i.e. we can define prototypes as linear combinations of data in this space. Hence, we can perform techniques such as GLVQ explicitly in pseudo-Euclidean space since it relies on vector operations only. One problem of this explicit transfer is the computational complexity of the embedding, which is $\mathcal{O}(D^3)$, and, further, the fact that out-of-sample extensions to new data points characterized by pairwise dissimilarities are not immediate. Because of this fact, we are interested in efficient techniques which implicitly refer to this embedding only. As a side effect, such algorithms are invariant to coordinate transforms in pseudo-Euclidean space. The key assumption is to restrict prototype positions to linear combinations of data points of the form

$$\mathbf{w}^j = \sum_i^N \alpha_i^j \mathbf{x}^i \text{ with } \sum_i^N \alpha_i^j = 1\,.$$

Hence, each vector $\alpha_j = \left(\alpha_1^j, \ldots, \alpha_N^j\right),\ j \in \{1, \ldots, M\}$ holds the coefficients describing the respective prototype $\mathbf{w}^j$ implicitly, as shown in [50]. Since prototypes are located at representative points in the data space, this is reasonable. According to [50], dissimilarities can then be computed implicitly, by means of

$$\begin{aligned} d(\mathbf{x}^i, \mathbf{w}^j) &= \sum_k^N \alpha_k^j d_{ik} - \frac{1}{2}\sum_{kl}^N \alpha_k^j \alpha_l^j d_{kl} \\ &= \left[\mathbf{D}\cdot\alpha^j\right]_i - \frac{1}{2}\cdot(\alpha^j)^\intercal \mathbf{D}\alpha^j\,. \end{aligned} \tag{2.5}$$

This observation constitutes the key to transfer GLVQ to relational data. Prototype $\mathbf{w}^j$ is represented implicitly via the coefficient vector α^j and distances are computed by means of these coefficients. The corresponding cost function of relational GLVQ (RGLVQ) becomes:

$$E_{\mathrm{RGLVQ}} = \sum_i^N E_{\mathrm{RGLVQ}}^i = \sum_i^N \Phi\left(\frac{[\mathbf{D}\alpha^+]_i - \frac{1}{2}\cdot(\alpha^+)^\intercal\mathbf{D}\alpha^+ - [\mathbf{D}\alpha^-]_i^+ - \frac{1}{2}\cdot(\alpha^-)^\intercal\mathbf{D}\alpha^-}{[\mathbf{D}\alpha^+]_i - \frac{1}{2}\cdot(\alpha^+)^\intercal\mathbf{D}\alpha^+ + [\mathbf{D}\alpha^-]_i - \frac{1}{2}\cdot(\alpha^-)^\intercal\mathbf{D}\alpha^-}\right),$$

where, as before, the closest correct and wrong prototype are referred to, now in terms of the corresponding coefficients α^+ and α^-, respectively. A simple stochastic gradient descent leads to adaptation rules for the coefficients α^+ and α^- in relational GLVQ: component k in the respective coefficient vector is adapted as

$$\begin{aligned} \Delta\alpha_k^+ &\sim -\Phi'(\mu(\mathbf{x}^i))\cdot\mu^+(\mathbf{x}^i)\cdot\frac{\partial\left([\mathbf{D}\alpha^+]_i - \frac{1}{2}\cdot(\alpha^+)^\intercal\mathbf{D}\alpha^+\right)}{\partial\alpha_k^+} \\ \Delta\alpha_k^- &\sim \Phi'(\mu(\mathbf{x}^i))\cdot\mu^-(\mathbf{x}^i)\cdot\frac{\partial\left([\mathbf{D}\alpha^-]_i - \frac{1}{2}\cdot(\alpha^-)^\intercal\mathbf{D}\alpha^-\right)}{\partial\alpha_k^-} \end{aligned}$$

where $\mu(\mathbf{x}^i)$, $\mu^+(\mathbf{x}^i)$, and $\mu^-(\mathbf{x}^i)$ are as in Equation 2.2. For a prototype $\mathbf{w}^j$, the partial derivative yields

$$\frac{\partial \left([\mathbf{D}\alpha^j]_i - \frac{1}{2} \cdot (\alpha^j)^\intercal \mathbf{D}\alpha^j \right)}{\partial \alpha_k^j} = d_{ik} - \sum_l^N d_{kl}\alpha_l^j \, .$$

Naturally, alternative gradient techniques can be used. After every adaptation step, normalization takes place to guarantee $\sum_i^N \alpha_i^j = 1$. We also restrict the possible prototype positions to the convex hull of the data by enforcing all $\alpha_i^j \geq 0$ in every iteration. This way, a learning algorithm which adapts prototypes in a supervised manner, similar to GLVQ, is given for general dissimilarity data, whereby prototypes are implicitly embedded in pseudo-Euclidean space. The prototypes are initialized as random vectors corresponding to random values α_i^j which sum to one. It is possible to take class information into account by setting all α_i^j to zero which do not correspond to the class of the prototype. Out-of-sample extension of the classification to new data is possible, based on the following observation, see [50]: given a novel data point x, which is characterized by its pairwise dissimilarities $\mathbf{D}_x$ to all the data used for training, the dissimilarity of x to a prototype $\mathbf{w}^j$ is $d(x, \mathbf{w}^j) = \mathbf{D}_x^\intercal \cdot \alpha^j - \frac{1}{2} \cdot (\alpha^j)^\intercal \mathbf{D}\alpha^j$.

2.2.5. Reducing computational demand via Nyström approximation

RGLVQ (just like SVM) depends on the full dissimilarity matrix and thus displays quadratic computational and memory complexity in N. Depending on the chosen dissimilarity measure, the main computational bottleneck is given by the calculation of the dissimilarity matrix itself. Alignment of biological sequences, for example, is quadratic in the sequence length (linear, if approximations such as FASTA are used), so that a computation of the full dissimilarities for about 11,000 data points (the size of the *SwissProt* data set as considered below) would already lead to a computation time of more than eight days (with 4 processor cores at 2.5 GHz, alignment done by the Smith-Waterman algorithm [122]) and a storage requirement of about 500 Megabyte, assuming double precision.

The Nyström approximation, as introduced by Williams and Seeger in [137], allows for an efficient approximation of a kernel matrix by a low-rank matrix. This method can be directly transferred to dissimilarity data, see [116]. The basic principle is to pick a set of representative landmarks $\mathcal{V} \subset X$, $|\mathcal{V}| = V$, and consider the rectangular sub-matrix $\mathbf{D}_{\mathcal{V},X}$ of dissimilarities between landmarks and all data instances (e.g. sequences). This matrix is of linear size, assuming that V is fixed. The full matrix can be approximated in an optimal way, in the form $\mathbf{D}^\nu = \mathbf{D}_{\mathcal{V},X}^\intercal \mathbf{D}_{\mathcal{V},\mathcal{V}}^{-1} \mathbf{D}_{\mathcal{V},X} \approx \mathbf{D}$ where $\mathbf{D}_{\mathcal{V},\mathcal{V}}$ is the square sub-matrix of $\mathbf{D}$, and $\mathbf{D}_{\mathcal{V},\mathcal{V}}^{-1}$ refers to its pseudo-inverse. While calculating the full pairwise dissimilarity matrix $\mathbf{D}$ takes $\mathcal{O}(N^2)$ time, the complexity to produce its approximated counterpart $\mathbf{D}^\nu$ is dominated by the calculation of $\mathbf{D}_{\mathcal{V},X}$ in $\mathcal{O}(V \cdot N)$ steps, and the pseudo-inverse $\mathbf{D}_{\mathcal{V},\mathcal{V}}^{-1}$, which is $\mathcal{O}(V^3)$. This results in an overall complexity of $\mathcal{O}(V^2 \cdot N)$,

which becomes profitable when N is increasing while V is assumed to be constant. The resulting approximation is exact, if V corresponds to the rank of matrix $\mathbf{D}$.

Note that the Nyström approximation can be directly integrated into the distance computation of relational GLVQ, in such a way that the overall training complexity is linear instead of quadratic. We refer to results obtained by a Nyström approximation by the superscript RGLVQ$^{\nu}$. We use 10% landmarks by default, i.e. $V = \lfloor 0.1 \cdot N \rfloor$.

2.2.6. Interpretability of relational prototypes

Relational GLVQ extends GLVQ to general dissimilarity data. Unlike Euclidean GLVQ, it represents prototypes indirectly by means of coefficient vectors, which are not directly interpretable since they correspond to positions in pseudo-Euclidean space. However, because of their representative character, we can approximate these pseudo-Euclidean points by their respective closest *exemplars*, i.e. raw data instances originally contained in the training set. Unlike prototypes, these exemplars can be directly inspected. We refer to such an approximation as `k`-approximation, if a prototype is substituted by its `k` closest exemplars. We will see in experiments that the resulting classification accuracy is still rather good for small values $\mathtt{k} \in \{1, \ldots, 5\}$. We refer to results, which were obtained using a `k`-approximation, by the subscript RGLVQ$_{\mathtt{k}}$.

2.2.7. Experiments

We evaluate relational GLVQ for several benchmark data sets characterized by pairwise dissimilarities. These data sets have been used extensively in [28] to evaluate SVM classifiers for general (dis)similarity data. Since SVM requires a *psd* kernel matrix, appropriate preprocessing has been done in [28] in 5 variants: *flip*, *clip*, *shift*, and *similarities as features* in conjunction with the linear and Gaussian kernel, respectively. In addition, we consider a few benchmarks from the biomedical domain. The data sets are as follows:

Amazon47 consists of 204 data points from 47 classes, representing books and their similarity based on customer preferences. The similarity matrix $\mathbf{S}$ was symmetrized and transferred by means of $\mathbf{D} = \exp(-\mathbf{S})$, see [76].

Aural Sonar consists of 100 signals with two classes (target of interest/clutter), representing sonar signals with dissimilarity measures according to an ad hoc classification of humans.

Cat Cortex consists of 65 data points from 5 classes. The data originate from anatomic studies of cats' brains. The dissimilarity matrix displays the connection strength between 65 cortical areas. A preprocessed version as presented in [47] was used.

Chromosomes constitutes a benchmark data set from the *Copenhagen Chromosomes* database of cytogenetics [91]. A set of 4,200 human chromosomes from 21 classes

(the autosomal chromosomes) are represented by grey-valued images. These are transferred to strings measuring the thickness of their silhouettes. These strings are compared using edit distance with insertion/deletion costs 4.5 [122, 103].

Face Recognition consists of 945 samples with 139 classes, representing faces of people, compared by the cosine similarity.

Patrol consists of 241 data points from 8 classes, corresponding to seven patrol units (and non-existing persons, respectively). Similarities are based on clusters named by people.

Proteins is a data set described in [100], consisting of 213 globin proteins, which are compared based on their evolutionary distance. The samples originate from different protein families: hemoglobin-α, hemoglobin-β, myoglobin, etc. Here, we distinguish 4 classes: HA, HB, MY, GG/GP.

SwissProt consists of 10,988 samples of protein sequences in 32 classes, and is a subset from the well-known *SwissProt* database [18]. The considered subset refers to the release 37, mimicking the setting as proposed in [74]. The full database consists of 77,977 protein sequences. The 32 most common classes such as Globin, Cytochrome a, Cytochrome b, Tubulin, Protein kinase st, etc. provided by the Prosite labeling [42] where taken, leading to 10,988 sequences. We calculate a similarity matrix based on a 10% Nyström approximation. These sequences are compared using exact Smith-Waterman alignment. This database is the standard source for identifying and analyzing protein measurements such that an automated sparse classification technique would be very desirable. A detailed analysis of the prototypes of the different protein sequences opens the way towards an inspection of typical biochemical characteristics of the represented data.

Vibrio consists of 1,100 samples of vibrio bacteria populations characterized by mass spectra. The spectra contain approx. 42,000 mass positions. The full data set consists of 49 classes of vibrio-sub-species. The mass spectra are preprocessed with a standard workflow using the *BioTyper* software [94]. Typically, mass spectra display strong correlations between neighboring entries, due to the dependency of subsequent masses. Therefore, problem-adapted similarities, as described in [94], are beneficial. In our case, similarities are calculated using a specific similarity measure provided by the BioTyper software. The Vibrio similarity matrix $\mathbf{S}$ has a maximum score of 3. The corresponding dissimilarity matrix is obtained as $\mathbf{D} = 3 - \mathbf{S}$.

Voting contains 435 samples in 2 classes, representing categorical data compared based on the value difference metric.

	RGLVQ	**AP**	**SVM**	Signature	# Prototypes
Aural Sonar	88.4 (1.6)	68.5 (4.0)	87.0-85.8*	(61,38,1)	10
Amazon47	81.0 (1.4)	75.9 (0.9)	82.4-19.7*	(192,1,11)	94
Cat Cortex	93.0 (1.0)	80.4 (2.9)	95.0-72.0	(41,23,1)	12
Chromosomes	92.7 (0.2)	89.5 (0.6)	95.1-92.2	(1951,2206,43)	63
Face Rec.	96.4 (0.2)	95.1 (0.3)	96.1-95.7*	(45,0,900)	139
Patrol	84.1 (1.4)	58.1 (1.6)	88.0-61.3*	(54,66,121)	24
Proteins	92.4 (1.9)	77.1 (1.0)	98.8-97.6	(169,38,6)	20
SwissProt	81.6 (0.1)	82.6 (0.3)	82.1-78.0	(2028,2,8958)	64
Vibrio	100 (0.0)	99.0 (0.0)	100	(499,502,99)	49
Voting	94.6 (0.5)	93.5 (0.5)	95.1-94.5*	(16,1,418)	20

Table 2.1.: Mean classification accuracies (and standard deviations) of prototype-based classification with relational GLVQ, in comparison to SVM (an SMO implementation) with *psd* corrections, and to AP with posterior labeling, for several dissimilarity data sets. The accuracies are obtained in a ten-fold cross-validation with ten repeats (only two-fold for SwissProt). SVM results, marked with *, are taken from [28]. The number of prototypes used for RGLVQ and AP, as well as the signature of the corresponding dissimilarity matrix are included. For SVM, the respective best and worst result using the different preprocessing mechanisms (*flip*, *clip*, *shift*, and *similarities as features* with linear and Gaussian kernel) are reported.

In case the given dissimilarities were not numerically symmetric, we symmetrized the matrix $\mathbf{D}$ by using $\widetilde{\mathbf{D}} = (\mathbf{D} + \mathbf{D}^\intercal)/2$. Diagonal values were set to zero, ignoring any self-dissimilarities: $\left[\widetilde{\mathbf{D}}\right]_{(i,i)} = 0,\ \forall i \in \{1, \ldots, N\}$.

As pointed out in [28], these matrices cover a diverse range of different characteristics, so that they constitute a well-suited test bench to evaluate the performance of algorithms for similarities/dissimilarities. In addition, benchmarks from the biomedical domain have been added, which constitute interesting applications per se. Many data sets are non-Euclidean, the signatures[2] can be found in Table 2.1. For every data set, we optimized the number of prototypes in a repeated cross-validation, see Table 2.1. The evaluation of the results is done by means of the classification accuracy obtained on the test set in a ten-fold cross-validation with ten repeats (two-fold cross-validation for SwissProt). Classes are reasonably balanced in the data sets, the largest observed difference in class sizes being 26% of the total number of data points. For this reason, and to maintain comparability with [28], we consider the classification accuracy to be an appropriate evaluation measure. For comparison, we report the results of a SVM after appropriate preprocessing of the dissimilarity matrix to guarantee a *psd* kernel [28]. In addition, we report the results of AP [39], a powerful unsupervised exemplar-based technique, which optimizes the quantization error for arbitrary similarity matrices, based on a message-passing algorithm for a corresponding factor graph representation of the cost function.

[2]For the signatures, we considered an eigenvalue to be numerically zero, if its absolute was $\leq 10^{-4}$.

	RGLVQ	**RGLVQ$_1$**	**RGLVQ$_3$**	**RGLVQ$^\nu$**	**RGLVQ$_1^\nu$**	**RGLVQ$_3^\nu$**
Aural Sonar	88.4 (1.6)	78.7 (2.7)	86.4 (2.7)	86.4 (0.8)	79.7 (2.6)	84.3 (2.6)
Amazon47	81.0 (1.4)	67.5 (1.4)	77.2 (1.0)	81.4 (1.1)	66.2 (2.6)	77.7 (1.2)
Cat Cortex	93.0 (1.0)	81.8 (3.5)	89.6 (2.9)	92.2 (2.3)	79.8 (5.5)	89.5 (2.8)
Chromosomes	92.7 (0.2)	90.2 (0.0)	91.2 (0.2)	78.2 (0.4)	84.4 (0.4)	86.3 (0.2)
Face Rec.	96.4 (0.2)	96.8 (0.2)	96.8 (0.1)	96.4 (0.2)	96.6 (0.3)	96.7 (0.2)
Patrol	84.1 (1.4)	51.0 (2.0)	69.0 (2.5)	85.6 (1.5)	52.7 (2.3)	72.0 (3.7)
Proteins	92.4 (1.9)	69.6 (1.7)	79.4 (2.9)	55.8 (2.8)	64.1 (2.1)	54.9 (1.1)
Vibrio	100 (0.0)	99.0 (0.1)	99.0 (0)	99.2 (0.1)	99.9 (0.0)	100 (0.0)
Voting	94.6 (0.5)	93.7 (0.5)	94.7 (0.6)	90.5 (0.3)	89.5 (0.9)	89.6 (0.9)

Table 2.2.: Mean classification accuracies (and standard deviations) of relational GLVQ, obtained in a repeated ten-fold cross-validation. We compare the training with the full dissimilarity matrix to the Nyström approximation technique, as well as the use of full prototype coefficients α^j in comparison to the k-approximation technique.

In this case, the classification is obtained by posterior labeling. For RGLVQ, we train the technique with the full dissimilarity matrix, and compare the result to the sparse models, obtained via k-approximation with $\mathtt{k} \in \{1, 3\}$ and a Nyström approximation of the dissimilarity matrix using 10% of the training data. The mean classification accuracies are reported in Table 2.2 and Table 2.1.

Interestingly, in all cases but one (the almost Euclidean data set *Proteins*), results are comparable to SVM taking the respective best preprocessing, as reported in [28]. Unlike SVM, relational GLVQ makes this preprocessing superfluous. In contrast, SVM may require preprocessing to guarantee a *psd* kernel matrix. Further, different preprocessing can lead to very diverse accuracy as shown in Table 2.1, no single preprocessing being universally suited for all data sets. Thus, these results seem to substantiate the finding of [76], that the correction of a non-*psd* Gram matrix can influence the classification accuracy. Further, an improvement of the classification accuracy as compared to the state-of-the-art unsupervised prototype-based technique AP (using the same number of prototypes) can be observed, which is statistically significant in all cases (according to a two-sided t-test with a 5% significance level). This shows the benefits of including supervision in the training objective, if classification is the goal.

Unlike for SVM, which is based on support vectors in the data set, solutions are represented as typical prototypes. Similar to AP, these prototypes can be approximated by k nearest exemplars, representing the classification explicitly in terms of few data points instead of prototypes. As can be seen from Table 2.2, in only two cases the 3-approximation leads to a loss in accuracy of more than 5%. Interestingly, a 3-approximation of a prototype-based classifier for the SwissProt benchmark even leads to an increase of the accuracy from 81.6% to 84.0%.

The Nyström approximation offers a linear time and space approximation of relational GLVQ performed on the full matrix. The changes in accuracy due to this approximation

are documented in Table 2.2 for all except the SwissProt data set – since the computation of the full dissimilarity matrix for the Swissprot data set would require more than 8 days on a standard PC, we used a Nyström approximation right from the beginning for SwissProt. The quality of the approximation depends on the rank of the dissimilarity matrix. Thus, the results differ a lot depending on the characteristics of the eigenvalue spectrum for the data. Interestingly, it seems possible in more than half of the cases to substitute full relational GLVQ by this linear complexity approximation without much loss of accuracy. Recently, an efficient test has been proposed, which allows to judge the suitability of a Nyström approximation prior to training, based on the subsampling only, see [60].

As a further demonstration, we show the result of RGLVQ, trained to classify a small set of e-books, taken from the *Project Gutenberg* literature database[3]. The pairwise dissimilarities of 84 books, from 4 different authors, were calculated by the *normalized compression distance* (NCD) [87]. One prototype per class is used with a 3-approximation for visual inspection. In Fig. 2.1, books and representative exemplars found by RGLVQ_3 are displayed in a 2D embedding, obtained by a dimensionality reduction technique (*t-distributed stochastic neighbor embedding* (t-SNE) [131], see Chapter 4). While SVM leads to a classification accuracy of more than 95% (like RGLVQ), it picks almost all data points as support vectors, i.e. no direct interpretation is possible. In case of RGLVQ_3, we can see how the data structure is mostly covered by the exemplars in the visualization, some of them being well-known works of the respective author, like Jane Austen's "Emma" and Jules Verne's "A Journey to the Centre of the Earth".

2.2.8. Concluding remarks

This section presented an extension of generalized learning vector quantization to non-Euclidean data sets, characterized by relational data. By referring to an implicit embedding of data in a pseudo-Euclidean space, we have a theoretical foundation to represent prototypes in the data space. A corresponding extension of the cost function of GLVQ was proposed, and a very powerful learning algorithm can be derived. In most cases, it achieves a classification performance comparable to SVM, but without the necessity for preprocessing to ensure a *psd* kernel matrix. It yields the possibility to interpret the classification model in terms of the prototypes, and their corresponding exemplars in a k-approximation. As a first step to an efficient linear approximation, the Nyström technique has been tested, leading to promising results in a number of benchmarks, particularly making the technology feasible for interesting large data collections, such as the SwissProt database.

[3] `http://www.gutenberg.org`

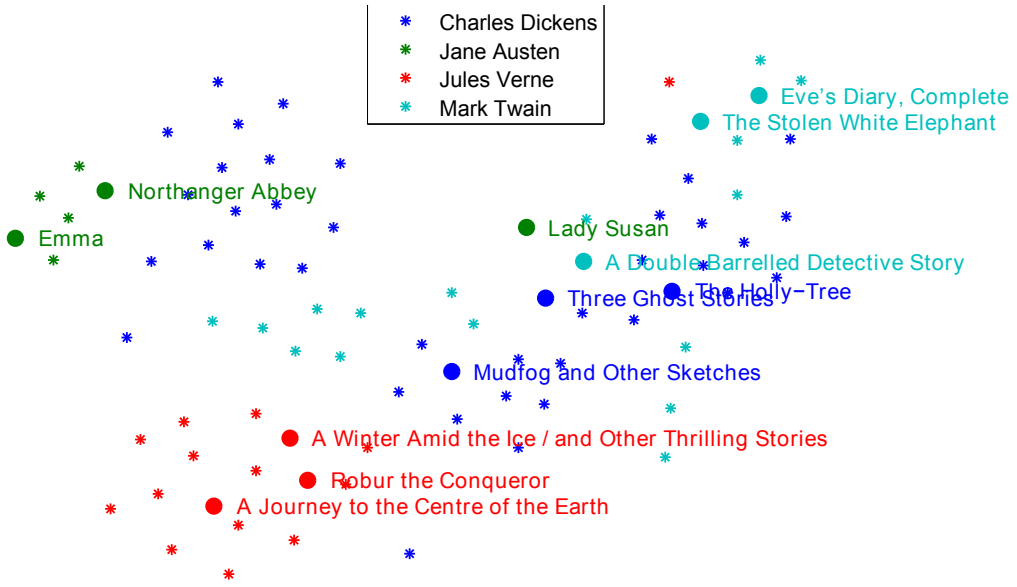


Figure 2.1.: Visualization of e-book data set and typical exemplars found by RGLVQ_3.

2.3. Relational generative topographic mapping

2.3.1. Introduction

Classical data mining tools such as the *self-organizing map* (SOM) [73], or its statistical counterpart, the *generative topographic mapping* (GTM) [17], provide a sparse representation of high-dimensional data by means of latent points arranged in a low-dimensional neighborhood structure, which is useful for visualization. GTM has been proposed as a probabilistic model to represent high-dimensional data by a sparse lattice of points in a latent space, such that visualization, compression, and data inspection become possible. However, SOM and GTM have been introduced for Euclidean vectors only. Several extensions of SOM to the more general setting of (dis)similarity data, have been proposed, including *median SOM* which restricts prototype locations to data points [74], *online SOM* and *batch SOM* using a kernelization of the classical approach [19, 139], and methods which rely on *deterministic annealing* techniques borrowed from statistical physics [45]. For GTM, a complex noise model, as proposed in [126], allows the extension of the method to discrete structures, like sequences.

In [50], the *relational SOM* has been proposed: an extension of SOM for dissimilarity data, relying on the same principle used for relational LVQ earlier in this chapter. In this section, we transfer this idea to GTM. We show that an EM algorithm can be derived to obtain the parameters of the model by maximizing the data log-likelihood. The performance of this method – *relational GTM* – is demonstrated on several benchmark sets. Since the basic principle is analog to the one for relational LVQ discussed earlier, we will keep our presentation and experimental evaluation in this section rather concise.

2.3.2. Generative topographic mapping (GTM)

The GTM [17] yields a generative probabilistic model for vectorial data $\mathbf{x} \in \mathbb{R}^D$. The model is a mixture of Gaussians, where the centers are induced by a regular grid of latent points $\mathbf{u}$ in a latent space. The latent points are mapped to prototypical target vectors $\mathbf{u} \mapsto \mathbf{w} = f(\mathbf{u}, \mathbf{H})$ in the data space, via a function f. This function is parameterized by $\mathbf{H}$. A typical choice for f is a generalized linear regression model

$$f : \mathbf{u} \longmapsto \Phi(\mathbf{u}) \cdot \mathbf{H} \,,$$

with base functions Φ, e.g. equally spaced Gaussians with variance σ^{-1}. Every latent point $\mathbf{u}$ induces the Gaussian distribution

$$p(\mathbf{x}|\mathbf{u}, \mathbf{H}, \beta) = \left(\frac{\beta}{2\pi}\right)^{D/2} \exp\left(-\frac{\beta}{2} \left\|\mathbf{x} - f(\mathbf{u}, \mathbf{H})\right\|^2\right) \tag{2.6}$$

of variance β^{-1}, generating a mixture of M modes

$$p(\mathbf{x}|\mathbf{H}, \beta) = \sum_{k=1}^{M} p(\mathbf{u}^k) p(\mathbf{x}|\mathbf{u}^k, \mathbf{H}, \beta) \tag{2.7}$$

in which $p(\mathbf{u}^k)$ is typically chosen uniformly, as $p(\mathbf{u}^k) = 1/M$. GTM training optimizes the data log-likelihood

$$\ln\left(\prod_{i=1}^{N}\left(\sum_{k=1}^{M} p(\mathbf{u}^k) p(\mathbf{x}^i|\mathbf{u}^k, \mathbf{H}, \beta)\right)\right) \tag{2.8}$$

with respect to $\mathbf{H}$ and β. This can be done by means of an *Expectation Maximization* (EM) approach. The generative mixture component $\mathbf{u}^k$ for a data point $\mathbf{x}^i$ is treated as a latent variable. If we choose a uniform distribution of the latent points, which is peaked with $p(\mathbf{u}^k) = 1/M$ at their grid positions, and a generalized linear regression model, EM training can be performed. EM computes the responsibilities

$$R_{ki}(\mathbf{H}, \beta) = p(\mathbf{u}^k|\mathbf{x}^i, \mathbf{H}, \beta) = \frac{p(\mathbf{x}^i|\mathbf{u}^k, \mathbf{H}, \beta) p(\mathbf{u}^k)}{\sum_{k'} p(\mathbf{x}^i|\mathbf{u}^{k'}, \mathbf{H}, \beta) p(\mathbf{u}^{k'})} \tag{2.9}$$

of the k-th component, for point number n, in alternation with the model parameters $\mathbf{H}$ and β. The parameters $\mathbf{H}$ are given by

$$\mathbf{\Phi}^\intercal \mathbf{G}_{\text{old}} \mathbf{\Phi} \mathbf{H}_{\text{new}}^\intercal = \mathbf{\Phi}^\intercal \mathbf{R}_{\text{old}} \mathbf{X} \,, \tag{2.10}$$

in which $\mathbf{R}$ are the responsibilities, $\mathbf{\Phi}$ is the matrix of base functions evaluated at points $\mathbf{u}^k$, and $\mathbf{X}$ is the data matrix. $\mathbf{G}$ is a diagonal matrix holding the accumulated responsibilities $[\mathbf{G}]_{(n,n)} = \sum_i R_{ki}(\mathbf{H}, \beta)$. To compute the variance β^{-1}, we solve

$$\frac{1}{\beta_{\text{new}}} = \frac{1}{ND} \sum_{k,i} R_{ki}(\mathbf{H}_{\text{old}}, \beta_{\text{old}}) \left\|\mathbf{\Phi}(\mathbf{u}^k)\mathbf{H}_{\text{new}} - \mathbf{x}^i\right\|^2 . \tag{2.11}$$

2.3.3. Relational GTM

As for relational LVQ in the previous section, we now assume the case that data x^i are no longer represented by single vectors $\mathbf{x}^i$, but indirectly, in terms of pairwise dissimilarities $d_{ij} = d(x^i, x^j)$. Like before, we refer to relational data, and can thus assume that a pseudo-Euclidean embedding with corresponding data vectors $\mathbf{x}^i$ exists, but is unknown to us. Hence, prototypical targets $\mathbf{w}$ in the data space cannot be determined explicitly as vectors, and a direct computation of the probability from Equation (2.6) is no longer possible. Therefore, we will employ the same technique as previously in Subsection 2.2.4, and replace the explicit distance between prototypes $\mathbf{w} = f(\mathbf{u}, \mathbf{H})$ and data vectors $\mathbf{x}$. We, again, assume an implicit representation of each prototype by a linear combination of the data, as proposed in [50]:

$$\mathbf{w}^k = \sum_{i=1}^{N} \alpha_i^k \mathbf{x}^i \text{ where } \sum_{i=1}^{N} \alpha_i^k = 1\,. \tag{2.12}$$

Hence, a prototype $\mathbf{w}^k$ is represented by the coefficient vector α^k. This allows to compute distances between the (unknown) data vectors and the prototypes, as follows:

$$\left\|\mathbf{x}^i - \mathbf{w}^k\right\|^2 = [\mathbf{D}\alpha^k]_i - \frac{1}{2} \cdot (\alpha^k)^\intercal \mathbf{D}\alpha^k \tag{2.13}$$

In [50], this observation was utilized to derive a relational variant of SOM. The same principle allows us to generalize GTM to relational data, given by a dissimilarity matrix $\mathbf{D}$. We restrict prototype vectors $\mathbf{w}^k$ to linear combinations of data points, as in (2.12). Hence, we can directly treat the mapping of latent points to prototype points as a mapping from the latent space to the coefficients:

$$f : \mathbf{u}^k \longmapsto \alpha^k = \Phi(\mathbf{u}^k) \cdot \mathbf{H} \tag{2.14}$$

where Φ refers to base functions, e.g. equally spaced Gaussians, with variance σ^{-1} in the latent space. We want to point out that the coefficients are not restricted to non-negative values in this case, i.e. $\alpha_i^k \in \mathbb{R}$. Therefore, the (unknown) target vectors may lie outside the convex hull of the data points in the pseudo-Euclidean embedding. To achieve a representative topological map, this seems like a reasonable assumption, if a smooth mapping from the latent space to the data space is intended.

To apply (2.13), we set the restriction

$$\sum_i [\Phi(\mathbf{u}^k) \cdot \mathbf{H}]_i = 1 \tag{2.15}$$

Based on (2.6), the likelihood function (2.8) can be computed without an explicit reference to the prototypes $\mathbf{w}^k$, since the distance is given by (2.13). As for GTM, we can employ an EM optimization scheme to arrive at solutions for the parameters β and $\mathbf{H}$. Again, the mode $\mathbf{u}^k$ responsible for data point $\mathbf{x}^i$ serves as a latent variable. EM

training computes the responsibilities (2.9) using the distance (2.13), in alternation with the parameters $\mathbf{H}$ and β, which are obtained by optimizing the expectation

$$\sum_{k,i} R_{ki}(\mathbf{H}_{\text{old}}, \beta_{\text{old}}) \ln p(\mathbf{x}^i | \mathbf{u}^k, \mathbf{H}_{\text{new}}, \beta_{\text{new}}) \tag{2.16}$$

with respect to $\mathbf{H}$ and β, under the constraint (2.15). This constrained optimization problem can be solved with the method of Lagrange multipliers, which is detailed in [J11]. The result is analog to the equations (2.10) and (2.11), with two minor differences:

- In (2.10), the data matrix $\mathbf{X}$ is replaced by the identity matrix $\mathbf{I}$. This results from the fact that data points $\mathbf{x}^i$ are represented in the α-space of linear combinations of data by a vector of zeros, with one entry $[\alpha']_i = 1$.
- In (2.11) the squared Euclidean distance is given by the implicit distance computation from (2.13).

We refer to this method as *relational GTM* (RGTM). The initialization uses a two-dimensional embedding of the dissimilarities as Euclidean vectors, obtained via the dimensionality reduction technique *multidimensional scaling* (MDS), see [80] as well as Chapter 4. The details of the initialization method are provided in [J11].

2.3.4. Experiments

First, we test RGTM on several benchmark dissimilarity data sets as introduced in the previous Section 2.2, which have also been used in [28, 50]: *Cat cortex* (65 data points and 4 classes), *Patrol* (241 points, 8 classes), *Voting* (435 samples, 2 classes), *Protein*[4] (226 points, 5 classes), *Aural sonar* (100 points, 11 classes). For every data set, a symmetric dissimilarity matrix with zero diagonal is given, which originates from a problem-adapted dissimilarity measure based on raw data instances, as explained in Section 2.2.7. The data sets were preprocessed in the same way as for RGLVQ.

Since these data sets are labeled, it is possible to evaluate the result via the classification accuracy obtained by posterior labeling. Thereby, posterior labeling of RGTM takes place based on the majority label of the accumulated responsibility of a latent point for data points carrying this label. We report the results of a cross-validation (CV) with ten repeats, where we use 2-fold CV for the *Cat cortex* data and *Aural sonar* data and 10-fold CV for the other data sets to maintain comparability with the results from [50]. To apply cross-validation, out-of-sample extensions of the assignments can be computed in the same manner as for RGLVQ, see Subsection 2.2.4. In all cases, we use 100 latent points and 4 base functions given by Gaussians. This global parameter setting was optimized with regard to all data sets.

[4] This set slightly differs from the setting in Section 2.2.7. We used 5 classes, as proposed in [47], with a rather unbalanced class distribution: HA (31.86%), HB (31.86%), MY (17.26%), GG/GP (13.27%), and others (5.75%)

	RNG	DA	**RGTM**
Cat cortex	69.8% (7.6)	**80.3%** (8.3)	76.5% (6.3)
Proteins	91.9% (1.6)	90.7% (0.8)	**93.6%** (0.4)
Aural sonar	83.4% (1.4)	**85.6%** (2.6)	83.7% (2.6)
Patrol	66.5% (2.4)	52.1% (5.1)	**66.6%** (4.6)
Voting	95.0% (0.4)	**95.1%** (0.5)	93.8% (0.6)

Table 2.3.: Mean classification accuracies (and corresponding standard deviations) obtained by a repeated cross validation on the described benchmark data sets.

The initial β, which determines the bandwidth of the base functions, has only a slight effect on the algorithm, if it stays in a reasonable interval. Here, the number of base functions is chosen as small as possible to preserve the topology of the data. Changing the number of latent points generally changes only the sampling of the data but the shape of the map stays the same. With a smaller number, the algorithm is faster and sparsity of the representation is increased; with a larger number, the algorithm is slower but more details in the data relations can be discovered. For an example of this scaling effect, please refer to [J11].

The respective classification accuracies obtained on the test set are listed in Table 2.3. For comparison, we report the classification accuracy of *deterministic annealing* (DA) and *relational neural gas* (RNG) as presented in [50]. We can observe that RGTM is always competitive to these two alternatives and is even better for three of the five classification tasks. Hence, RGTM offers a feasible alternative to DA and RNG as a classifier.

Method	**Parameter**	**Setting**
RGTM	number of latent points	900 (30-by-30 grid)
RGTM	number of base functions	4 (2-by-2 grid)
RGTM	number of training epochs	30
RSOM	number of neurons	900 (30-by-30 grid)
RSOM	number of training epochs	500
RSOM	initial neighborhood range	$N/2$

Table 2.4.: Meta-parameter settings used for the visualization experiments.

Visualization of classical music data set In the following, the visualization features of RGTM are briefly demonstrated. For a more thorough discussion, please refer to the article [J11]. We show the topographic mapping in case of a dissimilarity data set, which is derived from a classical music archive. The individual pieces of music are

originally given in the form of symbolic sequences in the MIDI file format[5], describing the progression of musical notes for all playing instruments in parallel. The information content is therefore comparable to written music sheets. We used a dissimilarity measure presented in [C09a], which imposes a tree structure on the polyphonic note progressions over time, in order to separate parallel melodic sequences, e.g. to split lead melodies from the accompanying chord progressions. For each musical piece, the separated melodic lines are concatenated to a single symbolic sequence, using a relative encoding for the progression of pitch changes and rhythmic expression. Overall, this yields a partial invariance to pitch translation (transposition) and time scaling, for individual parallel melodies. All sequence pairs are finally compared with the *normalized compression distance* (NCD), see [87], resulting in a relational data representation.

The data set consists of 1068 sonatas by 5 composers from two consecutive eras of western classical music: the period of the Viennese Classic (by Ludwig van *Beethoven*, Wolfgang Amadeus *Mozart*, and Joseph *Haydn*, around 1730-1820 AD), and the Baroque era (by Domenico *Scarlatti* and Johann Sebastian *Bach*, around 1600-1760 AD). The musical pieces were taken from the online MIDI database *"Kunst der Fuge"*[6]. Class labels are assigned according to the composer, however, we will omit a quantitative evaluation of the classification results, since there is no ground truth available for this kind of data set. Still, the visualization features of RGTM can be demonstrated in comparison to the existing RSOM, as shown in Figure 2.2. We can see that RGTM displays the class structure more distinctly, i.e. clusters for composers are more clearly visible.

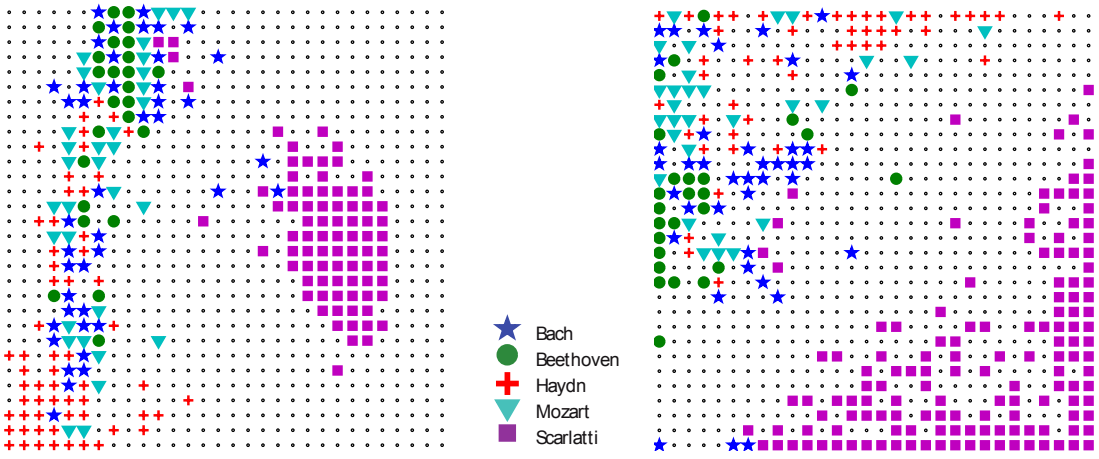
In this case, RGTM and RSOM were trained with the parameters listed in Table 2.4. As before, we used the majority vote for posterior labeling. The variance of the base functions, σ^{-1}, was set in a way that fits the distance between neighboring base function centers. In the RSOM, the *neighborhood range* defines how much the update process of one neuron influences the neighboring neurons in the RSOM grid, for details, see [50]. Its initial value r_0 was set to half the number of data points. The range is annealed exponentially to 0.01 during training, by calculating the range for the current epoch as $r_c = r_0 \cdot (0.01/r_0)^{(\mathtt{e}_c/\mathtt{e})}$, where $\mathtt{e}$ refers to the total number of epochs, and $\mathtt{e}_c$ is the current epoch count. The chosen settings were optimized for each method according to visual appearance only.

After the training of RGTM, different labeling strategies can be employed, which we will exemplify for this data set. Labeling by majority vote – as previously applied – means that a prototype in the grid is assigned the label of the majority of data points in its receptive field. This is displayed in Figure 2.2 for both, the RGTM and RSOM. Alternatively, one can label the RGTM prototypes as follows: a latent point in the grid is assigned the class label, which is carried by the data points that have the highest

[5] http://midi.org/

[6] http://www.kunstderfuge.com

accumulated responsibility for this latent point. Note that every latent point will have at least some small responsibility with respect to any data point. Therefore, all prototypes in the map would get assigned a class label, and no unlabeled points (dead units) would appear. To control this behavior for visualization tasks, it is useful to set a threshold for the value of responsibility, below which the latent point remains without any label, i.e., a class label is only assigned to a latent point, if the responsibility of at least one data point for this latent point exceeds the threshold. Thus, adjusting this threshold controls how many dead units will appear in the map eventually. The resulting maps are shown in Figure 2.3 for two different threshold values. These visualizations emphasize the overall class distribution, as opposed to the map with majority vote labeling in Figure 2.2, where local spatial and structural relationships are more accurately represented.

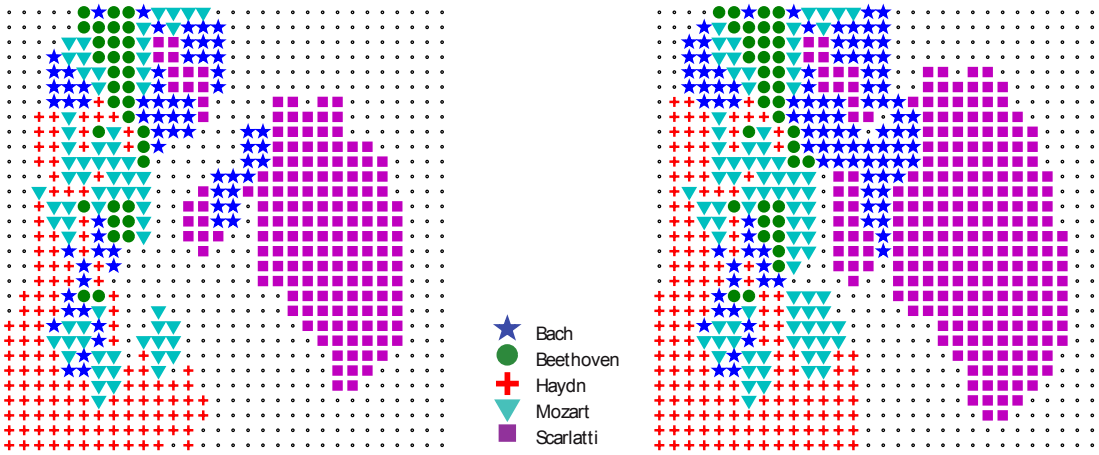


(a) RGTM **(b)** Classes **(c)** RSOM

Figure 2.2.: RGTM (left) and RSOM (right) visualization of a data set of classical sonatas by Beethoven (102), Haydn (172), Mozart (147), Bach (92), and Scarlatti (555). The prototypes (latent points) in the grid are marked using posterior labeling by the majority vote principle. The RGTM grid shows a noticeable separation of the musical pieces by composer, where mostly the comprehensive work of Bach marks a blend between the Viennese Classic and Baroque era. The arrangement seems meaningful since Bach's work is considered influential for both musical eras. Also the distinct style of Scarlatti is represented. In the grid on the right, generated with RSOM, the separation of the composers is less distinct.

2.3.5. Concluding remarks

In this section, we have described how GTM can be extended to address dissimilarity-based data representations. The experiments demonstrated that the classification performance is comparable to alternatives, such as deterministic annealing and relational



(a) RGTM - responsibility threshold 10^{-3}

(b) Classes

(c) RGTM - responsibility threshold 10^{-8}

Figure 2.3.: Two visualizations of the sonatas data set using the posterior labeling by responsibilities with different threshold values. On the left, where the threshold is higher (10^{-3}), there are more unlabeled prototypes than in the map on the right, where the threshold is set lower (10^{-8}). For comparison see Fig. 2.2 (left), where the posterior labeling was done by majority vote.

neural gas. However, as a particular benefit, RGTM yields a sparse representation of the data in terms of latent points in a regular grid (a latent space). Given that the latent grid is a low-dimensional structure, we can utilize it to visualize class structures of the data in a topographic map, similar to SOM. Additionally, it is possible to regularize the mapping of latent points to their targets in the data space (the prototypes) appropriately. However, since each prototype is described by a vector of coefficients w.r.t. all data, we arrive at a $\mathcal{O}(N^2)$ time and memory complexity, like in the case of relational LVQ. As a possible remedy, the Nyström approximation can be applied for RGTM as well, in analogy to Section 2.2.5. We will omit a further investigation of this subject here; please refer to [C10c] for an evaluation of the Nyström method in the context of RGTM.

Chapter 3.

Adaptive metrics for complex data

Chapter overview *In this chapter, we discuss techniques to adapt data dissimilarities in a way that facilitates classification with LVQ. This is achieved by learning the underlying metric parameters during classifier training, according to given class labels. Established metric learning schemes for vectorial LVQ are briefly reviewed and demonstrated. To address more complex dissimilarity-based data representations, we propose the transfer of this idea to relational LVQ, using an alignment measure for symbolic sequences.*

Parts of this chapter are based on:

[J15] B. Mokbel, B. Paassen, F.-M. Schleif, and B. Hammer. Metric learning for sequences in relational LVQ. *Neurocomputing*, (accepted/in press), 2015.

[C14c] B. Mokbel, B. Paassen, and B. Hammer. Efficient adaptation of structure metrics in prototype-based classification. In *ICANN 2014*, pages 571–578, 2014.

[C14b] B. Mokbel, B. Paassen, and B. Hammer. Adaptive distance measures for sequential data. In *ESANN 2014*, pages 265–270, 2014.

[TR12] B. Mokbel, M. Heinz, and G. Zentgraf. Analyzing motion data by clustering with metric adaptation. In *Proc. of ICOLE 2011*, number MLR-01-2012 in Machine Learning Reports, pages 70–79, 2012. ISSN: 1865-3960.

3.1. Motivation

All similarity- or dissimilarity-based classification and clustering techniques crucially depend on the underlying metric or proximity measure to address the data. Hence, these techniques fail if the choice of the metric or its parameterization are not suited for the given task. This observation motivated research about metric adaptation strategies based on given training data: today, several highly efficient metric learners are readily available for the vectorial setting, and the area constitutes a well-established field of research, see e.g. the excellent overview articles [11, 75].

For vectorial data representations, metric learning generally aims at an automatic adaptation of the Euclidean distance towards a more general (possibly local) quadratic form, based on auxiliary information. Most strategies act solely upon the metric and

are not interlinked with the subsequent classification or clustering method. This has the advantage that efficient, usually convex optimization schemes can be derived. However, no such technique currently offers an adaptation which is efficient with respect to data size and dimensionality, which can deal with local metrics, and which can be accompanied by guarantees of learning theory.

By linking metric adaptation to the subsequent classification tool, the property of a convex cost function is lost, depending on the considered classifier. However, metric learning can be integrated efficiently into the classification scheme, and results from learning theory can be derived by referring to the resulting function class. This has been demonstrated in the context of learning vector quantization (LVQ), where metric learning opened the way towards efficient state-of-the-art results in various areas, including biomedical data analysis, robotic vision, and spectral analysis [6, 33, 69, 13, 72, 9]. In Chapter 2, we already pointed out several benefits of LVQ-based classifiers. One of the striking properties is the intuitive definition of the classifier models in terms of prototypical representatives. They enjoy a wide popularity in application domains, particularly if human inspection and interaction are necessary, or life-long model adaptation is considered [117, 73, 71]. Modern LVQ schemes are accompanied by mathematical guarantees about their convergence behavior and generalization ability [119, 121]. Metric adaptation techniques in LVQ do not not only enhance the representational power of the classifier, but also facilitate interpretability by means of an attention focus regarding the input features and possible direct data visualization in case of low-rank matrices [119, 24]. We will briefly demonstrate the capabilities of these techniques later in this chapter, using real-world data sets from a motion tracking camera.

As we have pointed out earlier, most classical LVQ approaches can process vectorial data only, limiting the suitability of these methods regarding complex data structures, such as sequences, trees or graph structures, for which a direct vectorial representation is often not available. In Chapter 2, we presented *relational LVQ* as an extension to address dissimilarity data, in which an implicit pseudo-Euclidean embedding opens the possibility of smooth prototype updates, even for discrete data structures. These techniques yield competitive results to modern kernel classifiers, see [51]. However, relational LVQ shares the sensitivity of LVQ with respect to a correct metric parameterization. For structure metrics, such as sequence alignment, metric parameters correspond to the choice of the underlying scoring matrix in case of symbolic sequences over a discrete alphabet, or the choice of relevance weights for the sequence entries in case of sequences of numeric vectors. Note that there exist ad hoc techniques how to pick a suitable scoring function e.g. in the biomedical domain: prime examples are given by the PAM or BLOSUM matrices often used for aligning DNA sequences, which rely on simple evolutionary models and corresponding data sets [57, 122]. It is, however, not clear in how far these scoring matrices are suitable for a given classification task. Thus, the question arises, how to extend metric learning strategies to the case of structure metrics.

It has been pointed out in a recent survey [11] that structure metric learning constitutes a novel, challenging area of research with high relevance, and only a few promising approaches exist, particularly in the context of sequence alignment. Sequence alignment plays a major role in the biomedical domain, for processing time series data, or for string comparisons. Its optimum computation is usually based on dynamic programming or even more efficient approximations thereof. The question of how to infer an optimal scoring matrix from aligned sequences has been investigated under the umbrella term of 'inverse alignment'. Several promising approaches have been proposed in this context. While the resulting techniques can be accompanied by theoretical guarantees in simple settings, more complex approaches often rely on heuristics, see e.g. [48, 125, 12]. A popular platform which combines various adaptation methods for scoring functions is offered by SEDiL, for example [20].

In our scenario, however, we are dealing with the different question of how to infer structure metric parameters, given a classification task. Hence, optimal alignments are not known, rather data are separated into given classes, and metric parameters should be adapted such that sequences within one class are considered similar by the alignment. Eventually, this question aims at the identification of structural invariances for the given classification task at hand: which structural substitutions do not deteriorate the classification result? In this chapter, we will investigate in how far structure metric learning can be introduced into relational LVQ in a similar way as for its vectorial counterparts. For this purpose, we approximate discrete alignment by a differentiable function, and show that metric learning is possible based on the relational LVQ cost function and gradient mechanisms.

3.1.1. Scientific contributions and structure of the chapter

This chapter presents the following key contributions:

- A novel approach for metric learning is proposed, driven by the cost function of the *relational LVQ* classification technique, in order to adapt parameters of a dissimilarity measure for structured data, in particular symbolic sequences. Metric adaptation is performed in conjunction with the classifier's own optimization procedure, providing a seamless integration.

- The proposed learning scheme is realized and demonstrated in particular for sequence alignment, where the complex choice of the underlying scoring parameters is inferred from the data. Practical experiments show how metric adaptation does not only facilitate class-discrimination, but also increases the interpretability of the classifier model.

- Several approximation techniques are investigated, in order to compensate for the inherent high computational cost of the metric learning algorithm.

The remainder of the chapter is structured as follows: First, in Section 3.2, we will briefly introduce metric learning techniques in LVQ for vectorial data representations. Its advantages regarding the interpretability of the classifier model, will be demonstrated in a small application example for human pose detection for motion capturing data. Thereafter, we will focus on parameterized dissimilarity measures, instead of vectorial data descriptions. Considering sequence alignment as a particularly interesting case, we will explain its objective and efficient computation via dynamic programming, in Section 3.3. By approximating the alignment with a smooth function, derivatives become well-defined, and metric adaptation can be integrated into the relational LVQ update rules. In this context, we introduce efficient approximations that warrant the feasibility of the algorithm. In Section 3.5, we demonstrate the behavior of our method in simple mock-up scenarios, where ground truth for the metric parameters is available, and the resulting cost surfaces can be inspected directly. Afterwards, in Section 3.6, we investigate the efficiency and effectiveness of the technique in two real-world examples, one dealing with discrete sequences from bioinformatics, where the scoring matrix is adapted, the other originating from the domain of educational tutoring systems, where metric parameters correspond to the relevance of multi-dimensional sequence entries. Finally, we discuss additional approximations to tackle large data sets: on the one hand, alignment paths with small contribution can be ignored; on the other hand, general-purpose approximations, such as the Nyström technique, can be integrated easily into the workflow to reduce the number of necessary distance calculations. We briefly underline the validity of these techniques in one of our example scenarios, before closing with a brief summary of open questions in Section 3.7. We will occasionally refer to additional information in the Appendix.

3.2. Vector-based metric learning in LVQ

We recall, from Chapter 2, that the learning behavior and classification scheme of GLVQ relies on a distance measure d:

- To classify a data point, it is assigned the label of its closest prototype, w.r.t. d.
- To train the classifier, the cost function in Eq. 2.1 (page 27) is minimized, wherein each summand evaluates the difference $d(\mathbf{x}^i, \mathbf{w}^+) - d(\mathbf{x}^i, \mathbf{w}^-)$ for a data $\mathbf{x}^i$.

In the classical GLVQ algorithm, by Sato and Yamada [114], d is the squared Euclidean distance $d(\mathbf{x}^i, \mathbf{w}^j) = \|\mathbf{x}^i - \mathbf{w}^j\|^2$. However, it has been shown, that this distance function can be exchanged by generalized Euclidean metrics [119], or less conventional choices like divergence measures [65]. In [119], the authors proposed to use a general quadratic form:

$$d_{\mathbf{\Lambda}}(\mathbf{x}^i, \mathbf{w}^j) = (\mathbf{x}^i - \mathbf{w}^j)\mathbf{\Lambda}(\mathbf{x}^i - \mathbf{w}^j)^\intercal \quad \text{with} \quad \mathbf{\Lambda} = \mathbf{\Omega}^\intercal\mathbf{\Omega} \; . \tag{3.1}$$

This distance $d_{\mathbf{\Lambda}}$ is parameterized by a matrix $\mathbf{\Lambda}$, employing the comprehensible notion that $\mathbf{\Omega}$ defines a linear mapping, by which the data and the prototypes are transformed:

$$d_{\mathbf{\Lambda}}(\mathbf{x}^i, \mathbf{w}^j) = \Big((\mathbf{x}^i - \mathbf{w}^j)\mathbf{\Omega}^\intercal\Big)^2 = \Big(\mathbf{x}^i\mathbf{\Omega}^\intercal - \mathbf{w}^j\mathbf{\Omega}^\intercal\Big)^2 .$$

Therefore, the result is simply given by the standard squared Euclidean distance for the transformed vectors, after the mapping. The condition $\mathbf{\Lambda} = \mathbf{\Omega}^\intercal\mathbf{\Omega}$ ensures symmetry in $\mathbf{\Lambda}$, resulting in an overall symmetric distance function.

As in other supervised metric learning schemes, the key idea in [54] and [119] is to adapt metric parameters $\mathbf{\Lambda}$ in such a way, that the distance function facilitates classification for the given data. This goal is realized by training $\mathbf{\Lambda}$ in conjunction with the GLVQ online learning scheme: The prototype updates remain identical to the classic GLVQ algorithm, although using the parameterized distance. After each update step, matrix $\mathbf{\Lambda}$ is also adapted in a stochastic gradient descent to optimize the GLVQ cost function. Hence, every iteration of the training algorithm includes two separate updates: (i) the prototype update with a fixed metric according to the current $\mathbf{\Lambda}$, and (ii) the metric adaptation with fixed prototypes. A derivative of the cost function E_{GLVQ} w.r.t. $\mathbf{\Omega}$ yields the metric update rules (the full gradient terms can be found in [119]). Two particular variants of GLVQ have been proposed, which employ the inherent metric adaptation scheme for the distance $d_{\mathbf{\Lambda}}$:

GRLVQ: *Generalized Relevance Learning Vector Quantization*, see [54].
In this variant, $\mathbf{\Lambda}$ is restricted to a diagonal matrix, so that each entry $\big[\mathbf{\Lambda}\big]_{(k,k)}$ scales the contribution of the corresponding k-th dimension in the data and prototypes, i.e. $[\mathbf{x}^i]_k, [\mathbf{w}^j]_k$. While this does not utilize all degrees of freedom in the parameters, it allows for a very straightforward interpretation: if data are provided in a feature-based representation, $\big[\mathbf{\Lambda}\big]_{(k,k)}$ can be seen as the *relevance* of a certain feature for class discrimination in the classifier model. Therefore, we will refer to the diagonal of $\mathbf{\Lambda}$ as the *relevance profile*.

GMLVQ: *Generalized Matrix Relevance Learning Vector Quantization*, see [119].
In case of GMLVQ, $\mathbf{\Lambda}$ is a full matrix. In addition to the scaling of single dimensions via the diagonal entries, this also allows to regulate the emphasis of every pairwise correlation between data dimensions. An off-diagonal entry $\big[\mathbf{\Lambda}\big]_{(k,l)}, k \neq l$ states, how relevant the correlation of the feature pair (k, l) is for class discrimination in the model. Obviously, this concept is more powerful to alter the metric, however, its interpretation can be less intuitive in practical applications. We will refer to $\mathbf{\Lambda}$ as the *relevance matrix*.

In summary, the combination of GLVQ and metric learning offers interesting benefits: On the one hand, the resulting prototype vectors can be interpreted directly (in terms of the feature-based data representation), and yield intuitive access to the classification model. On the other hand, trained metric parameters can reveal the semantic influence of feature

dimensions for the classification problem, and yield additional insight for experts in the field. Please refer to [119] for an elaborate discussion about metric learning in GLVQ, and to [6] for a particularly successful practical application in the biomedical domain. In the following, we will present a simple real-world example, as a proof-of-concept to demonstrate the utility of metric learning in GLVQ.

3.2.1. Motion tracking data

The 3-dimensional tracking of human and animal body movement is important in various areas of science. Researchers, for example in the fields of biology, medicine, robotics, or sports, investigate such data to reveal patterns and complex interaction rules in natural motion [101, 104, 32]. Since the precision and the availability of motion tracking technology is increasing, intelligent analysis methods become necessary to assist researchers in identifying relevant information in large amounts of data. Although the raw data usually consists of multiple 3-dimensional vectors, the data precision and characteristics vary depending on the kind of tracking system. Today, many kinds of systems are available, ranging from large expensive motion capturing setups involving several distributed cameras and delivering very robust data at a high spatial resolution, to less sophisticated, small, cheap, and mobile solutions using only a single camera. Hence, there is a variety of options available for researchers to gather motion data. However, regarding the automatic analysis of this complex data, there is no general recipe in order to extract high-level information. Tools for clustering and visualization (see overviews in e.g. [80], and [35, ch. 10]) are widely applicable and can make the data accessible for experts in order to gain motor-functional insights from complex motion scenarios. In this context, metric learning algorithms [54, 119, 43] offer useful features. On the one hand, the prototype-based clustering technique can be used to categorize motion patterns, yielding a classifier for later recorded data, while the resulting prototypes may reveal typical poses or patterns, since they can be interpreted directly. On the other hand, with the addition of metric learning, the most relevant joint angles or spatial correlations can be identified automatically.

Our following experiments use a small data set of human poses, and a motion sequence, recorded with the single-camera tracking system *Kinect*[1] from Microsoft. We will also refer to other motion tracking data sets later in this thesis.

Representation In general, tracking data is often given as a sequence of multiple 3-dimensional vectors over a certain number of time steps, in the following referred to as *frames*. Each vector represents the positions of certain points on the target body in a steady coordinate system defined by the tracking device, in the following referred to as the *world coordinate system*. In partially rigid bodies of animals or humans, the movement is constrained by the underlying skeleton and the capabilities of the joints.

[1] `http://research.microsoft.com/en-us/um/redmond/projects/kinectsdk/`

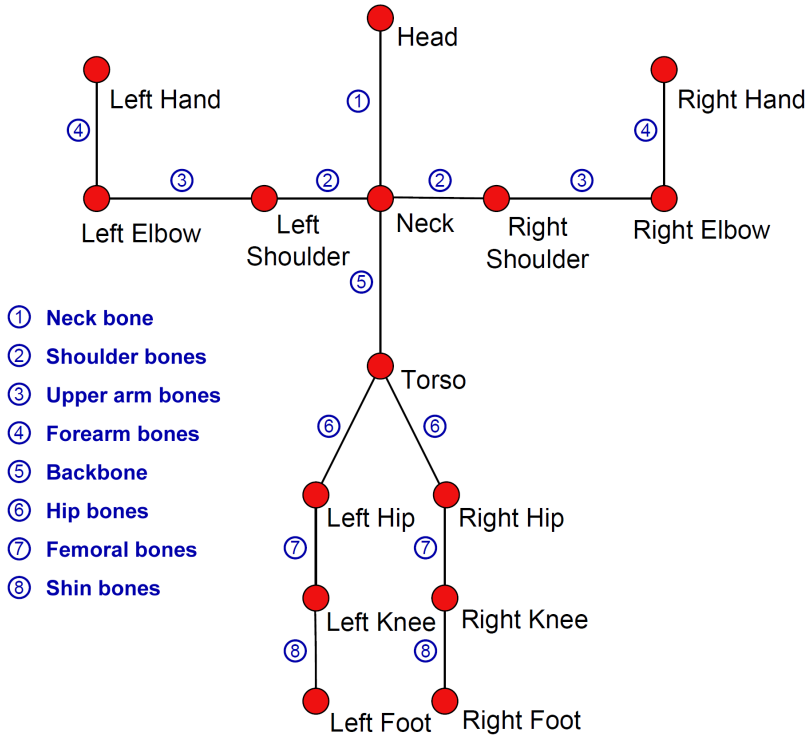


Figure 3.1.: The joints and bones of the skeleton provided by OpenNI & NiTE (Version 1.3.0) when using a Kinect camera (view from the back).

Rigid parts, called *segments* or *bones* are connected by flexible joints characterized by their *degrees of freedom* (DoF) and their *motion range*. Therefore, it is sufficient to track only a few points (*markers*) on the body, and model its skeletal properties based on prior knowledge about the tracking target, instead of tracking a high-resolution point cloud, for example. Usually, for every frame, the locations of the joints are calculated from the marker positions, but some markerless tracking devices yield joint positions directly, like in our technical setup.

Kinect, the single-camera system which we use, provides an RGB image and a depth view of the scene. To access the preprocessed information of joint positions, we used the software OpenNI[2], and the middleware NiTE[3] which infers a human skeleton structure by depth and texture cues only, without the need for special physical tracking markers, like reflective dots on the target body. NiTE & OpenNI provide 3D coordinates for every joint of this simplified human skeleton, see Figure 3.1. Because of the system's technical limitations, there are significant simplifications as compared to a natural skeleton: only the most important joints are considered, and some bones remain in a fixed orientation relative to each other. From the given joint positions expressed in world coordinates,

[2] `http://www.openni.ru` and `http://github.com/OpenNI/OpenNI2`
[3] `http://www.openni.ru/files/nite/index.html`

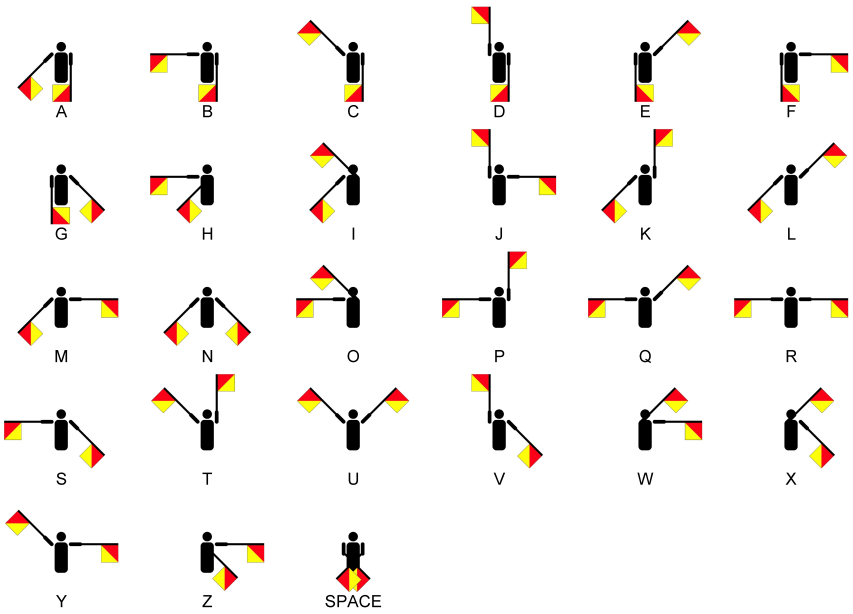


Figure 3.2.: Flag semaphore signals. The 'Space' signal was not used in our recorded data set. (Picture from Wikimedia Commons[4].)

we derived a more abstract representation, based on *joint angles*. This is important, since the data representation should reflect the skeleton's ability to move joints (mostly) independently from the other joints. In case of the Kinect software, we achieved this preprocessing step by utilizing particular restrictions and rigidness in the skeleton. The details are specified in a technical report, see [TR12].

To arrive at the joint angle representation, we employed the concept of a *kinematic chain*: A skeleton can be interpreted as a graph structure, usually an acyclic directed graph (i.e. tree), with one joint serving as the root node. This (strictly hierarchical) structure is called an *open* kinematic chain and yields the basis for representing motion data in many technical domains, see e.g. [101] for a thorough description. In our case, the root is the neck joint, and all edges are directed away from the neck. Centered at every joint, we established a local coordinate system, which is independent of the bones in lower hierarchical levels. Therefore, the orientation of a bone which connects to the next joint in the chain can be described in the local system. The use of spherical coordinates is typical for this purpose, i.e. angles, where ϕ defines the rotation around the Z-axis and θ is the elevation from the X-Y-plane. Our naming scheme for the joint angles is the following: Every bone has a parent and a child node in the chain. While the orientation of the bone is expressed in the local coordinate system of the parent joint, we name the angle according to the child, i.e. the bone's end. For example, "`R Elbow Theta`" is the elevation of the right upper arm bone in the right shoulder's local

[4]Fig. 3.2 image source: `http://commons.wikimedia.org/wiki/File:Semaphore_Signals_A-Z.jpg`

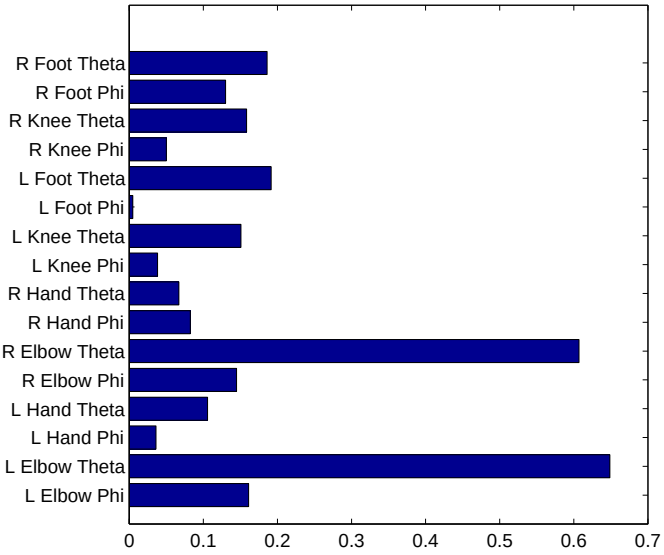


Figure 3.3.: The bar plot shows the diagonal of the matrix $\mathbf{\Lambda}$, i.e. the relevance profile of the GRLVQ classifier model, for the flag semaphore data set.

coordinate system.

This data representation yields desired invariances: A translational and rotational invariance of the entire body w.r.t. the world coordinate system leads to poses being represented equally, if the tracked person uses the same body posture, but is standing in a different location or orientation within the camera's field of vision. Further, joint angles for a certain bone are (mostly) not affected by changes in the other joints of the kinematic chain. In order to train GRLVQ and GMLVQ models in the following experiments, we use 16 of these local angles to describe the input data in a feature-based representation. The angles[5] are listed in Figure 3.3.

3.2.2. Proof-of-concept example

We used two small data sets, where we have strong assumptions about the relevance of features for class discrimination, and can thus compare our expectations with the outcome of the metric learning process.

Flag semaphore data The first data set consists of 26 static poses, which can be distinguished by the shoulder angles only. The poses are taken from the *flag semaphore*, a code to communicate at a distance by means of visual signals, common in the maritime world prior to the Morse code. The signaling person would usually hold flags, rods, or

[5]Due to technical restrictions in the camera system and software, several of the torso joints remain rigid. Therefore, our calculation of joint angles is only valid for the limbs, as specified in [TR12]. In the experiments, we used only the movable limb joints, see Fig. 3.3.

paddles in their hands for better visibility, although their presence or absence does not affect the code. Certain constellations of arm orientations represent the letters in the alphabet from A to Z, see Figure 3.2. These 26 poses (we omitted any special signals like the 'Space' symbol) have been recorded from 4 different test subjects, resulting in 104 total data points with 4 samples per class. We trained the GLVQ algorithm (without metric adaptation), and the GRLVQ (with a diagonal matrix $\mathbf{\Lambda}$) with one prototype per class on the entire data set. While GLVQ achieved 90% training accuracy, GRLVQ resulted in 100% training accuracy. Due to the small number of available samples, we tested the generalization ability of the GRLVQ model only in terms of an online recognition in front of the camera, with positive results: all signal poses were classified correctly, and fluent switches between neighboring poses were possible. Due to the mentioned invariances in the data representation, the classification model was not affected by global orientation changes of the body[6].

The relevance profile, shown in Figure 3.3, clearly singles out the angles of the left and right elbow elevation as the most important for class separation, which matches our expected response of the algorithm. Since the relevance learning scheme finds merely some possible configuration to separate the classes, the other angles are also contributing partially. The emphasis on the two important shoulder angles leads to an increased robustness of the classification. For example, the orientation of the legs becomes irrelevant for the signal recognition.

Walking-sequence data The second data set consists of four short sequences, showing two different walking styles, each recorded from two different persons. The two walking behaviors represent our classes in the data, which are partly antagonistic w.r.t. joint angle progressions: The first is a normal straight human walk, where the left arm and right leg move in one direction at the same time (e.g. forward), while the right arm and left leg move in the opposite direction (e.g. backwards). The other walking style uses the opposite (unnatural) combination, where the left arm and left leg move in the same direction at a time, and the right limbs in the opposite direction at the same time. All sequences together consist of 265 frames, which were recorded at a rate of 30 Hertz, they show about 3 strides of each walking style per person. We used each frame as an individual data sample, without any handling or preprocessing of the time-series aspects in the data. The class labels per frame correspond to the walking style.

GLVQ (without metric adaptation), and GMLVQ (with an adaptive matrix $\mathbf{\Lambda}$) were trained with 5 prototypes per class in 50 epochs, on a random sample consisting of 90% of all data instances, leaving the rest for testing the model. With GLVQ, we achieved a classification accuracy of 88% on the training set and 75% on the test set. The GMLVQ model yields an improved accuracy, with 91% on the training set and 92% on the test

[6] A video of the online recognition is available on the web, which demonstrates the robustness and the invariances of our classification model: `http://mokbel.de/phd/Flag_Recognition.avi`

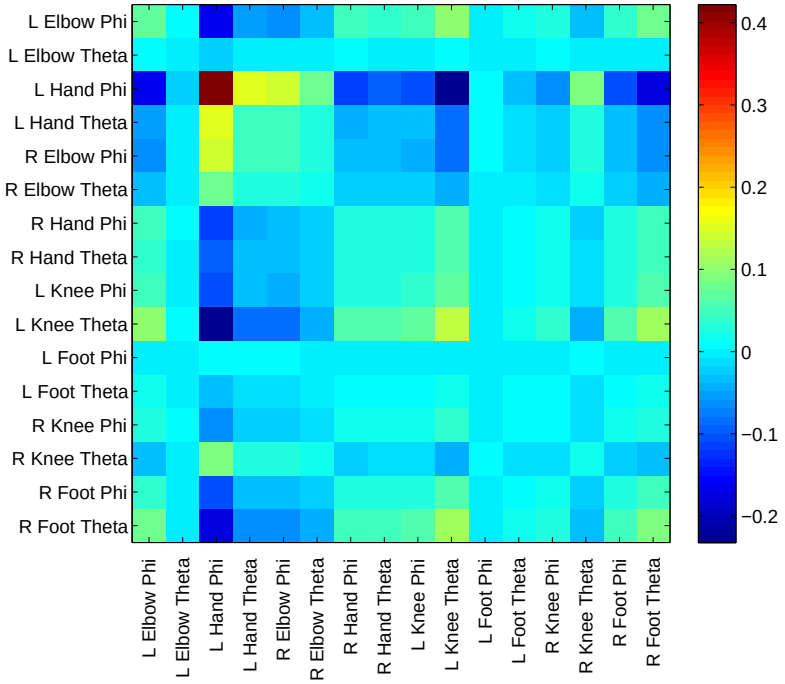


Figure 3.4.: The matrix $\mathbf{\Lambda}$ of the GMLVQ classifier model trained on the walking-sequence data set. Correlations of the left and right limb angles have high absolute values, and are thus utilized more in the problem-adapted metric of the trained classifier.

set. In addition to the enhanced classification results, GMLVQ offers additional insights with the adapted metric: the trained parameters $\mathbf{\Lambda}$ are shown in Figure 3.4. Note, that the sign of the values in the matrix are not really meaningful and interpretable as the respective positive or negative correlations of the joint angles in either one of the classes. Instead, only the absolute values in the matrix say, how much the pairwise correlation of these dimensions was utilized for the class separation found by the classifier model, which might translate to semantic meaning regarding the classes. As expected in this case, a pattern of correlations of the left and right limb angles is clearly visible, with strongly expressed correlations between the left and right knee and elbow angles, for instance.

From these examples, we can see how metric learning in GLVQ offers an interesting perspective as a powerful classifier, but also as a data analysis tool for real-world applications. However, the concept relies on vectorial data representations. Since Chapter 2 described the relational GLVQ method to address dissimilarity-based data representations, we will now investigate the possibility to transfer metric learning to dissimilarity measures in relational GLVQ, in the remainder of this chapter.

3.3. Sequence alignment as a parameterized dissimilarity measure

In the previous section, we have discussed how the distance measure is a crucial component in the LVQ classifier; not only in terms of the classification accuracy (since LVQ is based on a winner-takes-all scheme), but also in terms of a more intuitive understanding of the classifier model. In this section, we want to transfer this to *relational GLVQ* (RGLVQ), which was introduced in Chapter 2. In case of vector-based GLVQ, a parameterized distance can be adapted automatically in conjunction with the learning process. We are interested in possibilities to extend RGLVQ in a similar fashion, aiming at the twofold goal: to improve the accuracy and generalization ability of the resulting prototype model, and to enhance its interpretability by learning explicit structural invariances in terms of metric parameters. In the following, we will consider one particularly relevant type of structured data and corresponding metric, namely sequential data and sequence alignment. Note that the proposed rationale can be extended to alternative structure metrics, as long as they are differentiable with respect to metric parameters.

Assume an alphabet Σ is given, which can be discrete or continuous. We denote sequences with entries $a_I \in \Sigma$ as $\bar{a} = (a_1, \ldots, a_I, \ldots, a_{|\bar{a}|})$. Thereby, their length $|\bar{a}|$ can vary. The set of all sequences is denoted as $\mathcal{A} = \Sigma^*$. We assume that a symmetric dissimilarity measure $d_\lambda : \Sigma \times \Sigma \to \mathbb{R}$, with zero self-dissimilarities, is given to quantify the dissimilarity between single elements of the alphabet. This measure involves parameters λ which we would like to adapt by means of metric learning. Common choices of the dissimilarity measure are, for example:

- A *scoring matrix* for discrete alphabets $|\Sigma| < \infty$:
  Let $k = a_I \in \Sigma$, $m = b_J \in \Sigma$ be symbols from the respective sequences $\bar{a}, \bar{b}$. Then, the dissimilarity $d_\lambda(a_I, b_J) = \lambda_{km} \geq 0$ specifies the substitution costs if symbol k is aligned with symbol m.

- A *relevance weighting* for vectorial sequence entries:
  Let $\mathbf{a}_I, \mathbf{b}_J \in \Sigma = \mathbb{R}^n$ be vectorial elements from the respective sequences $\bar{a}, \bar{b}$. The notation a_I^r refers to the r-th entry in the vector $\mathbf{a}_I = (a_I^1, \ldots, a_I^n)$. Then, $d_\lambda(\mathbf{a}_I, \mathbf{b}_J) = \sum_{r=1}^{n} \lambda_r \cdot d_r(a_I^r, b_J^r)$ is a weighted sum of appropriate non-negative and symmetric dissimilarity measures d_r for each dimension. Therefore, the value $\lambda_r \geq 0$ specifies the 'relevance' of the r-th dimension for all sequence elements w.r.t. the given task.

Alignment incorporates the possibility of deletions and insertions to be able to compare two sequences of different lengths. For this purpose, the alphabet Σ is extended by a specific symbol, the gap "$-$". Similarly, the dissimilarity measure is extended to incorporate gaps, using the same symbol for simplicity:

$$d_\lambda : (\Sigma \cup \{-\})^2 \longrightarrow \mathbb{R}$$

specifying the gap costs

$$d_\lambda(a_I, -) = d_\lambda(-, a_I) \geq 0\,.$$

We exclude the case of two gaps being aligned, by the choice $d_\lambda(-,-) = \infty$.

Based on these definitions, a dissimilarity measure for sequences can be defined via alignment: A (global) *alignment* of sequences $\bar{a}$ and $\bar{b}$ consists of extensions $\bar{a}^* \in (\Sigma \cup \{-\})^*$ and $\bar{b}^* \in (\Sigma \cup \{-\})^*$ by gaps such that $|\bar{a}^*| = |\bar{b}^*|$. The overall costs of a fixed alignment is comprised of the sum of pairwise local distances $d(a_I^*, b_I^*)$. The optimal *alignment costs* (which we also refer to as *alignment dissimilarity*) are given by the minimal achievable costs

$$d^*(\bar{a}, \bar{b}) = \min \left\{ \sum_{I=1}^{|\bar{a}^*|} d_\lambda(a_I^*, b_I^*) \;\middle|\; (\bar{a}^*, \bar{b}^*) \text{ is alignment of } (\bar{a}, \bar{b}) \right\} . \tag{3.2}$$

Although this definition inherently considers all possible arrangements (which is an exponential number), these costs can be computed efficiently based on the following dynamic programming (DP) scheme. We use the shorthand notation $\bar{a}(I) = (a_1, \ldots, a_I)$ and $\bar{b}(J) = (b_1, \ldots, b_J)$ to denote the first I or J components of a sequence. Then, the following Bellman equality holds for the alignment costs of the parts $\bar{a}(I)$ and $\bar{b}(J)$:

$$\begin{aligned}
d^*(\bar{a}(0), \bar{b}(0)) &= 0\ , \\
d^*(\bar{a}(0), \bar{b}(J)) &= \sum_{J'=1}^{J} d_\lambda(-, b_{J'})\ , \\
d^*(\bar{a}(I), \bar{b}(0)) &= \sum_{I'=1}^{I} d_\lambda(a_{I'}, -)\ , \\
d^*(\bar{a}(I+1), \bar{b}(J+1)) &= \min \Big\{ A_{\texttt{Rep}} := d^*(\bar{a}(I), \bar{b}(J)) + d_\lambda(a_{I+1}, b_{J+1}), \\
&\qquad\qquad A_{\texttt{Ins}} := d^*(\bar{a}(I+1), \bar{b}(J)) + d_\lambda(-, b_{J+1}), \\
&\qquad\qquad A_{\texttt{Del}} := d^*(\bar{a}(I), \bar{b}(J+1)) + d_\lambda(a_{I+1}, -) \Big\}\ .
\end{aligned} \tag{3.3}$$

Note that the three terms $A_{\texttt{Rep}}, A_{\texttt{Ins}}, A_{\texttt{Del}}$, respectively, refer to the cases

- *replacement*: symbols a_{I+1}, b_{J+1} are aligned (called *match* if a_{I+1}=b_{J+1}),
- *insertion*: symbol b_{J+1} is aligned with a gap,
- *deletion*: symbol a_{I+1} is aligned with a gap.

This recursive scheme can be computed efficiently in time and memory $\mathcal{O}(|\bar{a}| \cdot |\bar{b}|)$ based on dynamic programming.

3.4. Learning scoring parameters from labeled data

Sequence alignment crucially depends on the local dissimilarities d_λ, which in turn are determined by the parameters λ. For a discrete alphabet, these parameters correspond to the scoring matrix which quantifies the costs of substituting a symbol by another one (i.e. for symbolic replacements, insertions, or deletions). We propose an adaptation of λ based on the RGLVQ error function, given labeled training data. This provides a way to automatically learn a suitable parameterization of the alignment dissimilarity for a given task.

We transfer the basic idea that was precedented for vectorial LVQ in [119]: simultaneously to prototype updates, the alignment parameters are optimized by means of a gradient descent based on the RGLVQ error. Thus, we consider the derivative of summand E^i_{RGLVQ} corresponding to a sequence $\bar{a}^i$ w.r.t. one parameter λ_q in λ:

$$\begin{aligned}\frac{\partial E^i_{\mathrm{RGLVQ}}}{\partial \lambda_q} = \quad & \Phi' \cdot \frac{2d^-(\bar{a}^i)}{(d^+(\bar{a}^i)+d^-(\bar{a}^i))^2} \cdot \frac{\partial d^+(\bar{a}^i)}{\partial \lambda_q} \\ - \; & \Phi' \cdot \frac{2d^+(\bar{a}^i)}{(d^+(\bar{a}^i)+d^-(\bar{a}^i))^2} \cdot \frac{\partial d^-(\bar{a}^i)}{\partial \lambda_q}\end{aligned} \tag{3.4}$$

with

$$\frac{\partial d(\bar{a}^i, \bar{w}^j)}{\partial \lambda_q} = \sum_k \alpha^j_k \partial d^*_{ik}/\partial \lambda_q - \frac{1}{2}\sum_{kl} \alpha^j_k \alpha^j_l \partial d^*_{kl}/\partial \lambda_q \tag{3.5}$$

where d^*_{ik} refers to the alignment dissimilarity of sequences i and k. An alignment $d^*(\bar{a}, \bar{b})$ as introduced above is not differentiable. Therefore, we consider an approximation, which we call *soft alignment*. We substitute min by

$$\mathrm{softmin}(v_1, \ldots, v_m) = \sum_i^m v_i \cdot \frac{\exp(-\beta v_i)}{\sum_j^m \exp(-\beta v_j)}$$

with the derivative

$$\mathrm{softmin}'(v_i) = \big(1 - \beta \cdot (v_i - \mathrm{softmin}(v_1, \ldots, v_m))\big) \cdot \frac{\exp(-\beta v_i)}{\sum_j^m \exp(-\beta v_j)} \;,$$

for a fixed *"crispness"* $\beta \geq 0,\ \beta \in \mathbb{R}$, where $\beta \to \infty$ corresponds to the discrete minimum function. The derivative $\partial d^*(\bar{a}, \bar{b})/\partial \lambda_q$ can be computed in a DP scheme analog to the

alignment:

$$
\begin{aligned}
\frac{\partial d^*(\bar{a}(0),\bar{b}(0))}{\partial \lambda_q} &= 0 \ , \\
\frac{\partial d^*(\bar{a}(0),\bar{b}(J))}{\partial \lambda_q} &= \sum_{J'=1}^{J} \frac{\partial d_\lambda(-, b_{J'})}{\partial \lambda_q} \ , \\
\frac{\partial d^*(\bar{a}(I),\bar{b}(0))}{\partial \lambda_q} &= \sum_{I'=1}^{I} \frac{\partial d_\lambda(a_{I'}, -)}{\partial \lambda_q} \ , \\
\frac{\partial d^*(\bar{a}(I+1),\bar{b}(J+1))}{\partial \lambda_q} &= \mathrm{softmin}'(A_{\texttt{Rep}}) \cdot \left(\frac{\partial d^*(\bar{a}(I),\bar{b}(J))}{\partial \lambda_q} + \frac{\partial d_\lambda(a_{I+1}, b_{J+1})}{\partial \lambda_q} \right) \\
&\quad + \mathrm{softmin}'(A_{\texttt{Ins}}) \cdot \left(\frac{\partial d^*(\bar{a}(I+1),\bar{b}(J))}{\partial \lambda_q} + \frac{\partial d_\lambda(-, b_{J+1})}{\partial \lambda_q} \right) \\
&\quad + \mathrm{softmin}'(A_{\texttt{Del}}) \cdot \left(\frac{\partial d^*(\bar{a}(I),\bar{b}(J+1))}{\partial \lambda_q} + \frac{\partial d_\lambda(a_{I+1}, -)}{\partial \lambda_q} \right)
\end{aligned}
\tag{3.6}
$$

The full derivation of Equation 3.6 is specified in the Appendix Section A.1.

The derivative $\partial d_\lambda\big(\bar{a}(I),\bar{b}(J)\big)/\partial \lambda_q$ depends on the choice of the dissimilarity measure d_λ. For the two particularly interesting cases of discrete symbolic, and vectorial sequence entries, we get:

- Dissimilarities for a discrete alphabet $d_\lambda(a_I, b_J)$, with scoring parameters λ_{km}:

$$
\begin{aligned}
\frac{\partial d_\lambda(a_I, b_J)}{\partial \lambda_{km}} &= \hat{\delta}(a_I, k) \cdot \hat{\delta}(b_J, m) \\
\frac{\partial d_\lambda(a_I, -)}{\partial \lambda_{km}} &= \hat{\delta}(a_I, k) \cdot \hat{\delta}(-, m) \\
\frac{\partial d_\lambda(-, b_J)}{\partial \lambda_{km}} &= \hat{\delta}(-, k) \cdot \hat{\delta}(b_J, m), \text{ with Kronecker-Delta } \hat{\delta}
\end{aligned}
$$

- Dissimilarities for a vector alphabet $d_\lambda(\mathbf{a}_I, \mathbf{b}_J) = \sum_{r=1}^{n} \lambda_r \cdot d_r(a_I^r, b_J^r)$, with relevance weights λ_r:

$$
\begin{aligned}
\frac{\partial d_\lambda(a_I, b_J)}{\partial \lambda_r} &= d_r(a_I^r, a_J^r) \\
\frac{\partial d_\lambda(a_I, -)}{\partial \lambda_r} &= d_r(a_I^r, -) \\
\frac{\partial d_\lambda(-, b_J)}{\partial \lambda_r} &= d_r(-, a_J^r)
\end{aligned}
$$

  where, in the latter case, parameterized gap costs are considered as a suitable extension of d_r. For real numbers, this can be chosen as $d_r(a^r, u)$ for some constant $u \in \mathbb{R}$ such as $u = 0$, for example.

The costs of computing the derivative $\partial d^*(\bar{a}, \bar{b})$ are $\mathcal{O}(|\bar{a}| \cdot |\bar{b}|)$, as for alignment itself. This, however, has to be performed for every possible parameter. Further, due to the implicit prototype representation as a convex combination, it has to be computed for all pairs of sequences to achieve a single update step, see Eq. 3.5. Hence, costs amount to $\mathcal{O}\big(|\lambda| \cdot N^2 \cdot \max\big\{|\bar{a}| \;\big|\; \bar{a} \text{ is sequence in the training set}\big\}^2\big)$ for an update, where N denotes the number of training sequences, which is infeasible. Therefore, we will present an efficient approximation in the following, where every prototype is substituted by a convex combination over a fixed number of `k` data instances only.

Approximation of prototypes by closest exemplars Equation 3.5 contains two sums which both refer to *all* sequences $\bar{a}^l$ in the given set, weighted by a corresponding coefficient α_l^j. Therefore, computing the update for one sample $\bar{a}^i$ requires the derivatives for all sequences $\bar{a}^l,\ l \in \{1, \ldots, N\}$.

To avoid this, we can transfer the principle of `k`-*approximation*, as introduced in Chapter 2, Subsection 2.2.7 for RGLVQ: we may limit the dependency of metric updates to only a few exemplar sequences per prototype, by restricting the coefficients α^j to their largest `k` components. The empirical results from Chapter 2 indicate that it works well for the positional updates of RGLVQ prototypes, even when choosing $\mathtt{k} \ll N$ in real data distributions.

Transferring this approximation to the representation of prototypes for metric adaptation, we calculate the derivative $\partial d(\bar{a}^i, \bar{w}^j)/\partial\lambda_q$ based only on a subset of sequences, namely the prototype's *exemplars* $\bar{a}^l,\ l \in \mathcal{E}_j$ where $\mathcal{E}_j$ is a set of indices with fixed size $\mathtt{k} = |\mathcal{E}_j|$. The indices $\mathcal{E}_j$ refer to the `k` largest components in the respective weight vector α^j. Therefore, the number of exemplars `k` is a meta-parameter in our method, which will be discussed further in Section 3.5.2. For the minimal choice $\mathtt{k} = 1$, the derivative reduces to the single term $\partial d^*_{il}/\partial\lambda_q$, i.e. a soft alignment derivative between the sample sequence $\bar{a}^i$ and only one exemplar $\bar{a}^l$. Even this coarse approximation seems to work well for practical data, as will be shown in later experiments. This approximation makes updates feasible, and allows for a user-controlled compromise between precision and speed of the metric adaptation. The complexity of a single update therefore reduces severely to $\mathcal{O}\big(|\lambda| \cdot \mathtt{k}^2 \cdot \max\big\{|\bar{a}| \;\big|\; \bar{a} \text{ is sequence in the training set}\big\}^2\big)$.

Hebbian learning as a limit case Finally, we want to point out that, in a limit case, the derived update rules strongly resemble Hebbian learning, hence the metric adaptation follows intuitive learning steps. We consider the limit where every prototype can be approximated by one data point, i.e. α_l^j is 0 for all but one l, so the approximation by $\mathtt{k} = 1$ is exact. Then, the derivative in Equation 3.5 is dominated by only one summand, namely the derivative of the alignment distance between a given training sequence and the corresponding prototype's single exemplar sequence. Further, the considered limit case refers to a crisp instead of a soft minimum, i.e. a softmin function with $\beta \to \infty$. Hence, only one path, the optimal alignment path, is relevant in the computation of

the alignment dissimilarity. On this path, the contribution of a considered parameter is measured, as follows:

- for a specified pair of symbols, in case of a discrete alphabet, it is the number of the alignments of this pair on an optimal alignment path,
- for a given dimension in case of vectorial sequence entries, it is the optimal alignment distance restricted to the dimension in question.

A more formal demonstration is given in the Appendix Section A.1.

For both settings, this number represents the learning stimulus, which (i) decreases the corresponding metric parameter if the labeling is correct, and (ii) increases the corresponding metric parameter if the labeling is incorrect. In general, normalization can take place, since the number of parameters $|\lambda|$ is fixed. Hence:

- For a discrete alphabet, in the limit, symbolic replacements are marked as costly if they contribute to a wrong labeling, while they become inexpensive if the labeling is correct.
- For vectorial alphabets, those vector dimensions are marked as relevant where the small values indicate a closeness to a correctly labeled prototype, while dimensions are marked as irrelevant otherwise.

3.5. Practical implementation

In this section, we will discuss the practical realization of the proposed metric learning strategy. First, we describe how the actual learning algorithm is implemented, followed by a discussion about meta-parameters and their influence. Thereafter, we investigate the algorithm's performance for artificial data in a first proof-of-concept evaluation and exemplify general characteristics of the error function.

3.5.1. Algorithm overview

To summarize our method, we provide pseudo-code in Algorithm 1 for the case of a discrete symbolic alphabet, i.e. the result of metric learning is a scoring matrix λ with entries λ_{km}. The algorithm works in a similar fashion for vectorial sequence entries. Since a learning step for the metric terms is more costly than an update of the prototypes, the former requiring alignment calculations, we always perform several prototype updates before addressing the metric parameters. We refer to this as a batch update since, typically, a batch of data points is considered. Similarly, metric parameter updates are performed in batches to avoid recurring alignments for sequences in the batch.

As an initial solution for λ, see Line 1, a simple configuration is applied, in the following referred to as *equal costs*: we set $\lambda_{km} = 1/|\Sigma|$ for all pairs $(k, m) \in (\Sigma \cup \{-\})^2, k \neq m$, and

add small random noise to break ties in the initial alignments. Only symbolic matches require no costs: $\lambda_{kk} = 0$. During the adaptation (see Line 12), small or negative values $\lambda_{km} < \epsilon = 0.005/|\Sigma|$ are reset to ϵ in order to keep $\mathbf{D}$ non-negative, and to ensure that an alignment always favors matches (k, k) over the trivial alternative of a deletion $(k, -)$ directly followed by an insertion $(-, k)$ or similar unnecessary replacements. RGLVQ requires symmetric dissimilarities in $\mathbf{D}$, which is ensured if the scoring matrix λ is itself symmetric. Therefore, we enforce the symmetry of λ after every update, in Line 13. We will refer to the part from Line 5 to 14 as one *epoch* of learning.

Algorithm 1: RGLVQ with metric adaptation

Data: a set of training sequences $\{\bar{a}^1, \ldots, \bar{a}^N\} = \mathcal{S} \ni \bar{a}^i$ over an alphabet Σ
Parameters: metric learning rate η, number of exemplars `k`, crispness β, classic RGLVQ parameters (e.g. number of prototypes M)
Result: a set of prototypes $\{\alpha^1, \ldots, \alpha^M\} \ni \alpha^j$, a scoring matrix λ

1 initialize parameters $\lambda \in \mathbb{R}^{(|\Sigma|+1)^2}$, e.g. with *equal costs* as in Sec. 3.5.1
2 calculate all dissimilarities $\mathbf{D}$ according to λ
3 initialize prototypes α^j near the center of the corresponding class
4 **for** *number of epochs* **do**
`// classic RGLVQ update:`
5 perform (batch) update of prototypes α^j acc. to Equation 2.2
`// find representative sequences for each prototype:`
6 **for** $j = 1$ **to** M **do**
7 determine `k` exemplar sequences $\bar{a}^l \in \mathcal{S}$ with indices $l \in \mathcal{E}_j$ for prototype α^j, as the `k` largest entries α^j_l
`// update of metric parameters:`
8 **for** $i = 1$ **to** N **do**
9 **foreach** *pair of symbols* $(k, m) \in (\Sigma \cup \{-\})^2, k \neq m$ **do**
10 gradient descent step: $\lambda_{km} := \lambda_{km} - \eta \cdot \frac{\partial E^i_{\mathrm{RGLVQ}}}{\partial \lambda_{km}}$
11 **if** $\lambda_{km} < \epsilon$ **then**
12 enforce small positive costs by setting: $\lambda_{km} := \epsilon$
13 symmetrize: $\lambda := (\lambda^\top + \lambda)\ /\ 2$
14 re-calculate dissimilarities $\mathbf{D}$ according to new λ

3.5.2. Meta-parameters

Since our metric adaptation scheme optimizes the RGLVQ error function via a stochastic gradient descent, there are several meta-parameters that influence this learning process:

(I) RGLVQ meta-parameters

(II) the learning rate η

(III) the number of exemplars `k`

(IV) the 'crispness' β in the *softmin* function

(I) The *RGLVQ meta-parameters* are comprised of the number of training epochs, the prototype learning rate, and the number of prototypes. It has been observed in experiments with RGLVQ, that the algorithm is not sensitive to its meta-parameters: few prototypes often yield excellent results, and there is a small risk of overfitting even when a large number of prototypes is considered [51].

The necessary number of epochs and prototype learning rate are correlated, requiring a higher number of epochs when a smaller learning rate is chosen, and vice versa. In all our experiments, the number of epochs was fixed to 10. This choice is well justified, since a plausible convergence was achieved within the given time frame: during the last training epoch, the absolute error changes by less than 2% of the final error value, in every experiment.

The number of prototypes is crucial to determine the complexity of classification boundaries in RGLVQ, as is generally the case in prototype-based classifiers. For multimodal classes, too few prototypes lead to a high classification error. However, in particular in the light of an adaptive and hence very flexible metric, a good starting point is to train the classifier in the most simplistic setting with only one prototype per class, and increasing the number when necessary. To automatically adjust the number of prototypes, quite a few incremental variants of LVQ have been proposed, see e.g. [33, 70, 142]. Interestingly, for a complex image segmentation task, only few prototypes (3-4 per class, on average) were generated, supporting the claim that rather small LVQ networks already show representative behavior in particular in the context of an adaptive metric [33].

In our experiments, we will generally focus our discussion on the choice of one prototype per class, which allows us to emphasize the capability of adding sufficient complexity to the classifier model via metric adaptation only. For comparison, we will report the classifier performance using more prototypes, in addition to the highlighted results.

(II) The *learning rate* η for metric parameters is, in contrast to the prototype learning rate, a sensitive meta-parameter for the optimization via stochastic gradient descent. Considering parameters for alignment scoring in particular, changes in the gap costs (i.e. for deletions λ_{k-} and insertions λ_{-m}) have a stronger influence on the overall

alignment than single pairwise replacement costs λ_{km}. Therefore, it can be advisable to assign separate learning rates $\eta_{\texttt{Gap}}$ and $\eta_{\texttt{Rep}}$ for the respective costs, similar to previous vectorial metric adaptation in the context of LVQ [119]. In this way, it is also possible to restrict the adaptation to parameters of interest, and limit the degrees of freedom for learning. In our experiments, we will not use this separation and generally maintain the simpler case of a single η.

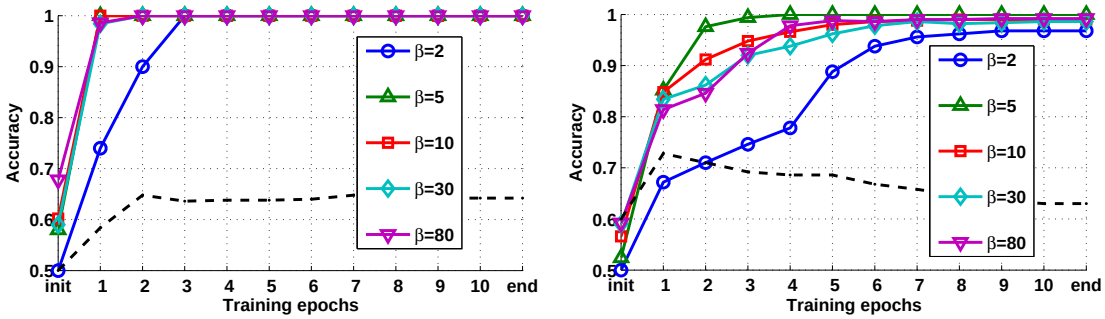
(III) The *number of exemplars* $\texttt{k}$ determines by how many real sequences a prototype $\mathbf{w}^j$ is represented in the update rule for metric parameter learning. As described in the end of Section 3.4, this is an approximation of the precise theoretical update where $\texttt{k} = N$. While a lower number could hypothetically decrease precision, it has shown to work well in practice, even for choices $\texttt{k} \ll N$, for example $\texttt{k} = 1$. Since the approximation strongly influences the computational demand of a single update step, the parameter has an immense impact on the overall runtime. The minimum choice of $\texttt{k} = 1$ yields the fastest update calculation, and usually provides sufficiently accurate results from our practical experience. In fact, all experiments presented in the remainder of this chapter rely on this setting, and we could achieve no considerable improvement in these cases, by choosing a larger number of exemplars $\texttt{k} > 1$.

(IV) The *crispness in the softmin function* β influences the classifier training progress. In the following Sections 3.5.3 and 3.5.5, its direct effect on the convergence characteristics are discussed in artificial data scenarios. In Figure 3.5, we can see how a lower crispness (e.g. for $\beta = 2$) generally slows down the adaptation, while higher values seem to facilitate a faster convergence, sometimes at the expense of robustness (see $\beta = 80$ in Figure 3.5b). Generally, we can observe that β directly affects the convergence characteristics, with an optimal value lying in a medium range.

3.5.3. Proof-of-concept with artificial data

We designed two artificial data sets with class structures that demonstrate the method's ability to adequately adapt (i) replacement costs and (ii) gap costs for the case of discrete sequence entries. Both data sets contain random sequences which follow deliberate structural patterns, such that a specific parameter configuration in the scoring matrix λ leads to a perfect class separation, while a naive choice of costs λ causes severe overlaps between classes.

Replacement data: In this data set, all strings have 12 symbols, randomly generated from the alphabet $\Sigma = \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}\}$ according to the regular expressions:
$(\mathsf{A}|\mathsf{B})^5$ $(\mathsf{A}|\mathsf{B})$ $(\mathsf{C}|\mathsf{D})$ $(\mathsf{C}|\mathsf{D})^5$ for the first, and $(\mathsf{A}|\mathsf{B})^5$ $(\mathsf{C}|\mathsf{D})$ $(\mathsf{A}|\mathsf{B})$ $(\mathsf{C}|\mathsf{D})^5$ for the second class. Hence, replacements of A or B by C or D are discriminative, while replacements A with B, and C with D are not. After the training of λ, we expect high costs for discriminative replacements, while other replacement costs in λ are close to zero. Also,



(a) *Replacement* data **(b)** *Gap* data

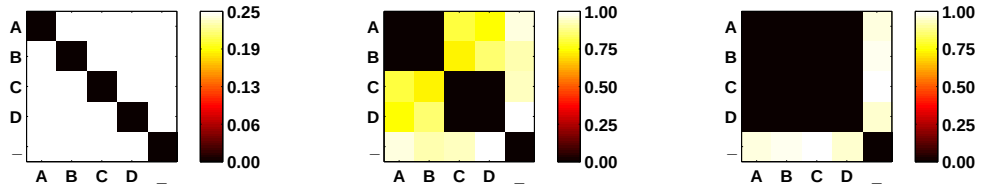
Figure 3.5.: The figures show the average test accuracy achieved during 10 epochs of RGLVQ training in a 5-fold cross-validation with 5 repeats on artificial data sets. The dashed black line represents the training without adapting λ, and serves as a baseline in which the classifier remains close to random guessing. The other curves show the accuracies achieved with the proposed metric adaptation scheme, for different settings of the 'crispness' parameter β. The adaptation yields nearly perfect results in all settings, while the convergence characteristics are slightly affected by β.

we expect positive gap costs, since gaps could otherwise circumvent the alignment of the discriminative middle parts.

Gap data: The second data set focuses on gap scoring. Strings in the first class are random sequences $\bar{a}^i \in \Sigma^{10}$ of length 10, whereas strings $\bar{a}^l \in \Sigma^{12}$ in the second class are longer by 2 symbols. Therefore, replacements of letters are not discriminative, while the introduction of any gaps discriminates classes. Thus, gap costs are expected to become high, while any other replacements should cost less.

Evaluation: For each data set, we created $N = 100$ sequences (50 for each class) and evaluated the average classifier performance in a 5-fold cross-validation with 5 repeats. RGLVQ was trained using one prototype per class, for 10 epochs. The learning rate for the adaptation of λ_{km} was set to $\eta = 1/N$, and the number of exemplars $\mathtt{k} = 1$. We use the aforementioned *equal costs* for the initial alignment parameters λ. Several settings of the 'crispness' β in the *softmin* function have been evaluated, but for now let us consider the intermediate setting of $\beta = 5$. We will discuss the influence of this meta-parameter later in this Section, and in Section 3.5.5.

The experimental results in Figure 3.5 show a drastically increased accuracy when adapting λ, for example, with $\beta = 5$ a perfect average test accuracy of 100% (with 0 deviation) was achieved after the 4th epoch. Consequently, the adapted λ represent ideal scoring matrices for both data sets, which exactly fulfill our aforementioned expectations: Figure 3.6 exemplarily shows the respective λ matrices before and after training from the last respective cross-validation run. For comparison, we trained RGLVQ in the



(a) Standard, initial λ **(b)** *Replacement* data **(c)** *Gap* data

Figure 3.6.: Visualizations of the scoring matrix λ, where color/intensity encodes the values. On the left is a standard choice of λ as *equal costs* which serves as the initial state for the training, the middle and right show the final state of λ after adaptation, fulfilling the expectations for the respective artificial data set.

classical fashion, based on fixed dissimilarities $\mathbf{D}$, without adapting the underlying scoring parameters. In this case, λ refers to the initial *equal costs*, which does not emphasize class-discrimination. As expected, classification remains close to random guessing in this setting, see the baseline in Figure 3.5: the average test accuracy after training was 64% for the *Replacement* data and 61% for the *Gap* data.

Figure 3.5 shows the progression of accuracy during training, for different values of the 'crispness' β. For lower settings (e.g. $\beta = 2$), we can see that the final level of accuracy is often achieved in later epochs, which indicates that the metric adaptation is slower. In contrast, higher values facilitate a faster adaptation, sometimes at the expense of robustness (see $\beta = 80$ in Figure 3.5b). In Section 3.5.5, we will demonstrate the influence of β in a soft alignment, implying its impact on the metric adaptation process and convergence characteristics.

From the proof-of-concept we can conclude that the proposed supervised metric adaptation strategy is indeed able to single-out discriminative parameters, which leads to a clear class separation and enables the training of a robust classifier model in our examples. The training arrives at the expected results, even for $\mathtt{k} = 1$, the most efficient approximation where each (virtual) prototype is represented by only one (tangible) exemplar sequence. In the following, we will first observe the characteristics of the RGLVQ error function w.r.t. metric parameters in our toy scenario, and thereafter, take a closer look on the crispness in a soft alignment.

3.5.4. RGLVQ error function surface

To get an impression of the characteristics of the RGLVQ error function with regard to metric parameters, we visualize its values for varying parameter settings as a 3-D surface. Therefore, we simplify our artificial data sets even further, to restrict to only a few degrees of freedom in the parameters λ. We obtained an adapted λ, as well as prototype positions α^1, α^2 from a single training run of 10 epochs ($\beta = 10$, $\eta = 0.07/N$). To evaluate various configurations of λ, a pair of entries $(\lambda_{km}, \lambda_{qr})$ will be iterated over

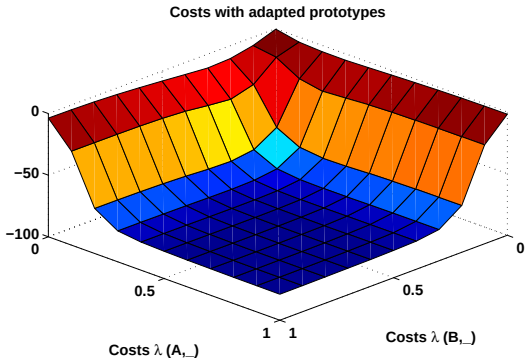


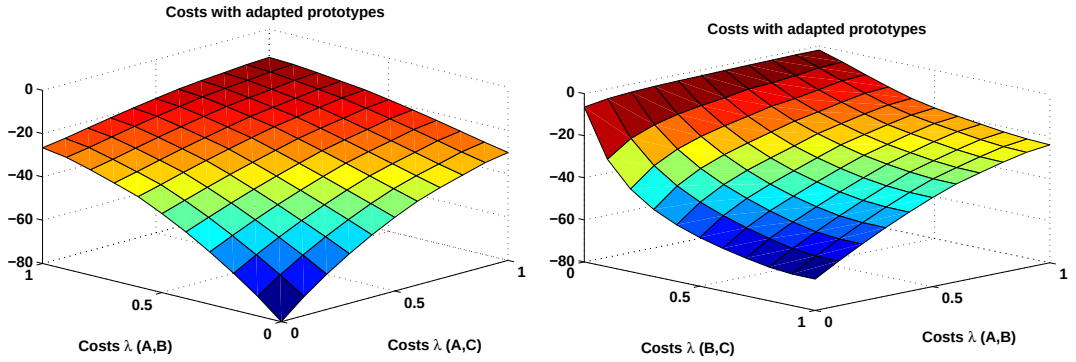
Figure 3.7.: The error E_{RGLVQ} for the *simplified Gap* data, evaluated with all parameter combinations $(\lambda_{\mathsf{A}-}, \lambda_{\mathsf{B}-})$ in steps of 0.1 over the interval $[0, 1]$, while replacement costs are at a low constant $\lambda_{\mathsf{AB}} \approx \epsilon$. As expected, the error surface drops smoothly to a low plateau, when both gap cost parameters are increased.

all combinations, while keeping the others fixed to their final state after training. Given the prototypes, we can visualize E_{RGLVQ} as a surface for all combinations $(\lambda_{km}, \lambda_{qr})$.

The *simplified Gap* data consists of random sequences over the two-letter alphabet $\Sigma = \{\mathsf{A}, \mathsf{B}\}$, as before with length 10 in the first, and length 12 in the second class, and $N = 100$. Again, the introduction of any gaps is crucial for class-discrimination, so a minimum of the error surface is expected for settings where both costs $\lambda_{\mathsf{A}-}$ and $\lambda_{\mathsf{B}-}$, become high. Figure 3.7 shows E_{RGLVQ} for configurations $(\lambda_{\mathsf{A}-}, \lambda_{\mathsf{B}-})$ in increasing steps of 0.1 over the interval $[0, 1]$. The remaining third parameter in λ is fixed to the final value after training, in this case it is close to the small constant $\lambda_{\mathsf{AB}} \approx \epsilon$. As expected, the error surface drops smoothly to a low plateau, when both gap costs are increased.

For the *simplified Replacement* data, we now use the three-letter alphabet $\Sigma = \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$, and regular expressions $(\mathsf{A}|\mathsf{B})^5\ \mathsf{B}\ \mathsf{C}\ (\mathsf{B}|\mathsf{C})^5$ and $(\mathsf{A}|\mathsf{B})^5\ \mathsf{C}\ \mathsf{B}\ (\mathsf{B}|\mathsf{C})^5$ to generate sequences in the first and second class, respectively. E_{RGLVQ} is then evaluated for all combinations of λ_{AB} and λ_{AC} (see Figure 3.8a), as well as λ_{AB} and λ_{BC} (Figure 3.8b). The respective remaining parameters in λ are constant at their final value from training, with low $\lambda_{\mathsf{AC}} \approx \epsilon$, and high $\lambda_{\mathsf{BC}}, \lambda_{\mathsf{A}-}, \lambda_{\mathsf{B}-}, \lambda_{\mathsf{C}-} > 0.7$. Since only replacements (B, C) and (C, B) are relevant for class-discrimination, we expect the error function to approach its minimum when λ_{AB} as well as λ_{AC} are low, and λ_{BC} is high. The surfaces in Figure 3.8 meet these expectations, with a monotonic decrease of error toward the optimum.

In a realistic scenario, the number of metric parameters is likely to be much higher. For sequence alignments with a scoring matrix for discrete alphabets (where we assume symmetry and a zero diagonal in λ), the number of free parameters is $(|\Sigma|^2 + |\Sigma|)/2$, i.e. it grows quadratically with the size of the alphabet. Their influence on the RGLVQ error can be rather complex, including intricate dependencies among the parameters themselves. Therefore, we can expect the error function to exhibit several local optima w.r.t. changes of metric parameters in a real data scenario.



(a) E_{RGLVQ} for λ_{AB} vs. λ_{AC} **(b)** E_{RGLVQ} for λ_{BC} vs. λ_{AB}

Figure 3.8.: Surfaces of E_{RGLVQ} for the *simplified Replacement* data, evaluated with parameter combinations λ_{AB} and λ_{AC} (left), as well as λ_{AB} and λ_{BC} (right) in steps of 0.1 over the interval $[0, 1]$. The respective remaining parameters are constant at the final value after training. As expected, the error approaches its minimum for $\lambda_{\mathsf{AB}} = \lambda_{\mathsf{AC}} = 0$ and $\lambda_{\mathsf{BC}} = 1$.

3.5.5. Influence of crispness on the alignment

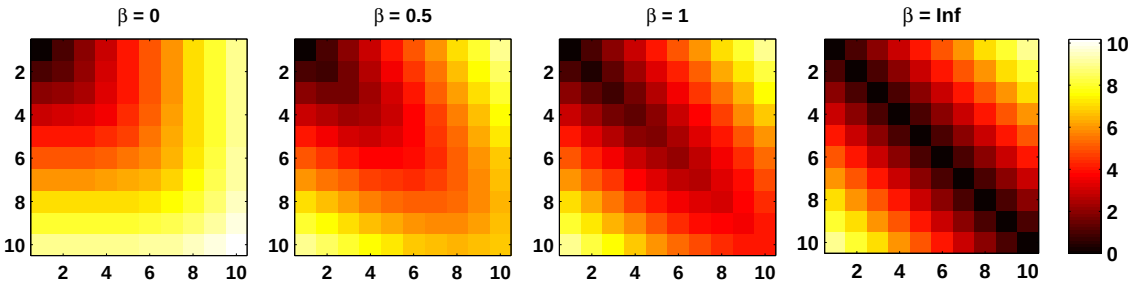
In this paragraph, we demonstrate, on a small example, how soft alignment (with its crucial parameter β) compares to classical sequence alignment (which is the limit case of soft alignment, for $\beta \to \infty$). Here, we address only the calculation of the alignment distance, not the learning of parameters. As described in Section 3.3, page 58, the alignment of two sequences can be calculated by DP via a recursive algorithm, see Equation 3.4. All different possibilities to partially align the sequences and accumulate costs can be assembled in a DP matrix:

$$\big[\mathbf{M}\big]_{(I,J)} = \mathbf{M}_{(I,J)} = d^*\big(\bar{a}(I), \bar{b}(J)\big) \quad \forall\ 0 \leq I \leq |\bar{a}|,\ 0 \leq J \leq |\bar{b}|$$

The upper left entry $\mathbf{M}_{(0,0)} = d^*\big(\bar{a}(0), \bar{b}(0)\big) = 0$ represents the initialization of the recursive calculation, while the bottom right entry contains the final accumulated costs for the full alignment $\mathbf{M}_{(|\bar{a}|,|\bar{b}|)} = d^*(\bar{a}, \bar{b})$.

In a crisp alignment (where $\beta \to \infty$), the accumulated cost at a position $\mathbf{M}_{(I+1,J+1)}$ is determined by selecting the discrete minimum among the choices $\{A_{\mathtt{Rep}}, A_{\mathtt{Ins}}, A_{\mathtt{Del}}\}$, see Equation 3.4. This means that every value $\mathbf{M}_{(I+1,J+1)}$ depends on only one of the preceding entries $\big\{\mathbf{M}_{(I,J)}, \mathbf{M}_{(I+1,J)}, \mathbf{M}_{(I,J+1)}\big\}$. In contrast, using a *softmin* function (with smaller β) means that all choices contribute to the result to a certain extent. Therefore, $\mathbf{M}_{(I+1,J+1)}$ depends on several preceding entries in the DP matrix. Accordingly, suboptimal alignment choices have an increasing influence on the accumulated cost if β is decreased.

To demonstrate the impact of parameter β, we investigate the characteristics of $\mathbf{M}$ in a simple example. Consider the alignment of a sequence $\bar{a} = (\mathsf{AAAAAAAAA})$ with itself



(a) $\beta = 0$ **(b)** $\beta = \frac{1}{2}$ **(c)** $\beta = 1$ **(d)** $\beta \to \infty$

Figure 3.9.: The images demonstrate the impact of different choices of 'crispness' β on the DP matrix $\mathbf{M}$ for a trivial alignment of $\bar{a} = (\mathsf{AAAAAAAAA})$ with itself, using the simple scoring scheme of $\lambda_{\mathsf{AA}} = 0$ and $\lambda_{\mathsf{A}-} = \lambda_{-\mathsf{A}} = 1$. Each figure shows a color-coded view of values in $\mathbf{M}$ for a setting $\beta \in \{0, \frac{1}{2}, 1, \infty\}$. While the diagonal is the optimal alignment path in all four settings, it becomes more distinguished as a low-cost path when β is high. With lower β values, sub-optimal alignment operations (in this case off-diagonal entries) get a higher contribution to the accumulated cost in the optimal path along the diagonal.

(i.e. $\bar{a} = \bar{b}$), using the simple scoring scheme $\lambda_{\mathsf{AA}} = 0$ and $\lambda_{\mathsf{A}-} = \lambda_{-\mathsf{A}} = 1$. Obviously, in a crisp sequence alignment, the optimal alignment path would match all symbols $(\bar{a}_I, \bar{b}_I) = (\mathsf{A}, \mathsf{A})$ without making use of deletions or insertions. This corresponds to the diagonal of $\mathbf{M}$, ending at a total cost of zero. Since only insertions or deletions can increase the accumulated cost in this case, the optimal alignment path (along the diagonal, using only matches) remains zero in every step, as can be seen in Figure 3.9d.

The three images on the left (Figures 3.9a-3.9c) show the corresponding DP matrix for values $\beta \in \{0, \frac{1}{2}, 1\}$: when increasing β from zero to one, the optimal path becomes more pronounced and stands out with significantly lower costs. Accordingly, the accumulated cost of the entire alignment drops for higher β.

For $\beta = 5$, the alignment approaches the *de facto* crisp condition. With $\lambda_{\mathsf{AA}} = 0, \lambda_{\mathsf{A}-} = \lambda_{-\mathsf{A}} = 1$, the weight by which a match operation $A_{\mathtt{Rep}}$ for (A, A) contributes to the *softmin* choice is

$$\operatorname{softmin}(A_{\mathtt{Rep}}, A_{\mathtt{Ins}}, A_{\mathtt{Del}}) = \frac{e^{(-5\cdot 0)}}{e^{(-5\cdot 1)} + e^{(-5\cdot 1)} + e^{(-5\cdot 0)}} = \frac{1}{2\,e^{(-5)} + 1} \approx 0.99\,.$$

Therefore, insertions and deletions only contribute 1% to the total soft alignment in this case. For other scoring schemes, a higher β might be required to achieve the de facto crisp alignment. It is therefore helpful to evaluate *softmin* values exemplarily, given a scoring λ, to assess the impact of a certain β setting.

3.6. Experiments with real-world data

In this section, we investigate the classification performance of our method on two real-world data sets. Additionally, we will take a look at general class-separation in the original and the adapted data dissimilarities, as well as interpretable aspects of the resulting adapted metric parameters.

3.6.1. Experimental procedure

Our experimental procedure, applied for both data sets, is summarized in the following. As before, the accuracy of an RGLVQ classifier with fixed metric parameters serves as a baseline, and is compared to the accuracy achieved via the proposed adaptation of metric parameters during RGLVQ training. This comparison directly reflects benefits which the classifier gains from metric adaptation. We report the respective average training and test accuracies (along with their standard deviation) obtained in a 5-fold cross-validation with 10 repeats.

To assess the overall class-separation without relying specifically on RGLVQ, we further evaluate the corresponding data dissimilarities before and after the metric is adapted. In the latter case, we use the adapted metric parameters resulting from the last respective cross-validation run of RGLVQ.

First, we report the average test accuracy of a *support vector machine* classifier (SVM), along with its corresponding average number of support vectors (#SV). The quantity of support vectors reflects the complexity of an SVM's classification boundary, where a lower number suggests that class-separation is easier in the given data, while higher values (up to the total number of given training data) indicate overlapping classes. In our practical implementation, we use the Open Source software *LIBSVM*[7] 3.18, and perform a 5-fold cross-validation with 10 repeats, based on the original, as well as the adapted metric. However, in order to apply SVM correctly, we need to obtain a valid kernel matrix from given dissimilarities d_{ij} in the matrix $\mathbf{D}$. Therefore, we first use Torgerson's *double centering*, see [128, p.258] to get similarities, as:

$$[\mathbf{S}]_{(i,j)} = s_{ij} = -\frac{1}{2} \cdot \left(d_{ij}^2 - (\mathbf{c}^j)^2 - (\mathbf{r}^i)^2 + o^2 \right)$$

where $\mathbf{c}^j, \mathbf{r}^i, o$ are the mean of the j-th column, of the i-th row, and of all values in $\mathbf{D}$, respectively. Thereafter, a kernel matrix $\mathbf{K}$ is created from $\mathbf{S}$ by correcting possible non-metric aspects in the given similarities, via 'flipping' negative Eigenvalues of $\mathbf{S}$, as described in [51].

Further, the accuracy of a simple *k-nearest-neighbor* classifier (k-NN) is evaluated, using $k = 5$ neighbors. Obviously, k-NN and SVM are expected to achieve a higher accuracy in general, since the model complexity in the sparse RGLVQ classifier is highly

[7] `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

restricted by using only one prototype per class. For these evaluation models, we will therefore focus on differences between fixed and adapted dissimilarities, instead of comparing the sparse RGLVQ model with SVM or k-NN classification in terms of accuracy.

As an additional quantitative indicator, independent of any particular classification model, we measure the ratio of mean intra-class distances to mean inter-class distances, in the following referred to as *separation ratio*. Here, smaller values indicate a clearer class-separation in general, which is an expected result from the metric adaptation procedure.

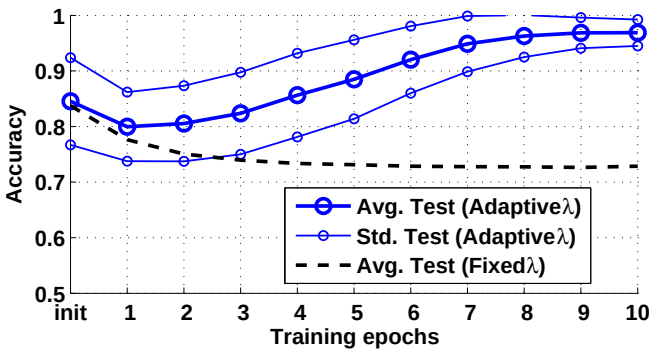
3.6.2. Copenhagen Chromosomes

We recall, from Subsection 2.2.7 in Chapter 2, the *Chromosomes* data set, which consists of sequences that represent band patterns from the Copenhagen Chromosomes database [91]. For this experiment, however, we will refer to a smaller subset of the database and use a different dissimilarity measure for the given sequences. Every sequence encodes the differential succession of density levels observed in gray-scale images of a human chromosome. Since 7 levels of density are distinguished, a 13-letter alphabet $\Sigma = \{\mathsf{f}, \ldots, \mathsf{a}, =, \mathsf{A}, \ldots, \mathsf{F}\}$ represents a difference coding of successive positions, where upper and lower case letters mark positive and negative changes respectively, and "=" means no change[8]. Table A.1 on page 137 in the Appendix Section A.2 lists all symbols with their associated difference levels, and the number of occurrences in the considered data set. From the database we use the *"CopChromTwo"* subset for binary classification, containing classes 4 and 5 with 200 sequences each ($N = 400$). In the literature, these two classes have been reported to yield a lower recognition rate than the others, see [40]. The authors in [40] used an organized ensemble of multilayer perceptrons to classify all 22 chromosomes in the Copenhagen database, and list the classification accuracies for individual classes. For the chromosomes 4 and 5, they report 91% and 89% accuracy on the test set, respectively, whereas the overall average is 95.86%. However, since every class is addressed by a one-versus-all classifier, these values are not directly comparable to the binary classification task on which we will focus in the following. To handle the full 22-class database, a local scoring matrix λ^j for every prototype α^j would be necessary, which is the subject of ongoing work, see Section 3.7.

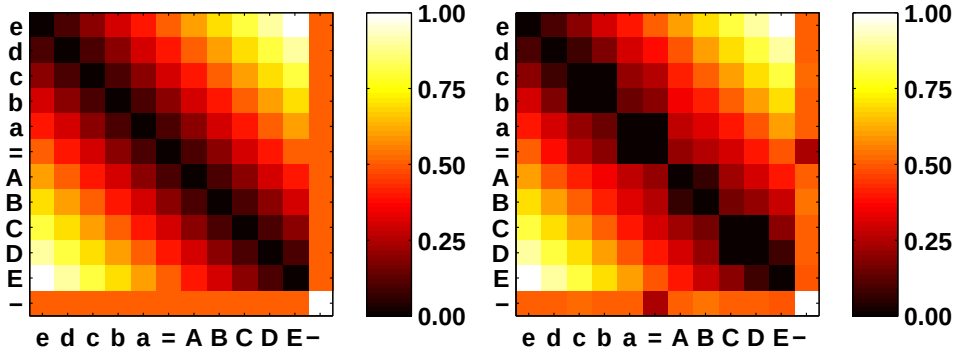
For the Copenhagen Chromosomes data, assumptions about a meaningful parameterization of the metric are available in [64]. The authors propose a weighting scheme for the edit distance, where replacement costs are the absolute difference of corresponding density changes: for example, $\lambda_{\mathsf{ae}} = |-1-(-5)| = 4$, and $\lambda_{\mathsf{fF}} = |-6-6| = 12$. The introduction of any gaps requires half the maximum of replacement costs, i.e. $\lambda_{k-} = \lambda_{-m} = 6$ for all $k, m \in \Sigma$. See Figure 3.10b for the full cost matrix[9]. Therefore, we compared two different options to initialize metric parameters λ in our experiment: (i) using the cost pattern from [64], and (ii) using the simple initialization with *equal costs*. For both

[8] For details, see `http://algoval.essex.ac.uk/data/sequence/copchrom/`

[9] Symbols f, F did not occur in the *CopChromTwo* subset and were thus not considered.



(a) Classification accuracies



(b) Initial λ **(c)** Adapted λ

Figure 3.10.: Metric parameters and results for the *Chromosomes* data set. The bottom left shows the initial alignment scoring pattern in the parameters λ according to a weighting scheme from [64]. The bottom right shows an adaptation of λ where costs for an insertion or deletion of the most frequent symbol "=" are strongly reduced, and replacements of neighboring symbols are decreased slightly. The graphs in the top figure show that the adaptation improves classification accuracy over 10 learning epochs in a repeated 5-fold cross-validation.

cases, we compare the classification performance of RGLVQ with and without metric adaptation, as before. In the latter case, RGLVQ training uses fixed dissimilarities based on the respective initial costs λ. The choice of meta-parameters was optimized w.r.t. the data in a 5-fold cross-validation with 10 repeats, setting crispness $\beta = 20$, and learning rate $\eta = 0.45/N$ to adapt λ. In order to minimize the computational effort, we chose $\mathtt{k} = 1$, which prove to be sufficient.

First, we consider the initialization of λ according to the weighting scheme from [64]. The results are displayed in Figure 3.10, and Table 3.1 (the two top rows). With fixed metric parameters, the final classification accuracies are rather low with 73% average test accuracy, see the baseline in Figure 3.10a. This is expectable, since the number of prototypes for RGLVQ was deliberately chosen to be only one per class, which implies

Method	*Init.*	Train (Std)	Test (Std)	SVM (#SV)	5-NN	Sep.
Fixed λ	*prior*	74% (2%)	73% (6%)	97% (174)	95%	0.93
Adapted λ	*prior*	97% (2%)	97% (2%)	98% (129)	98%	0.91
Fixed λ	*equal*	89% (3%)	89% (6%)	96% (210)	97%	0.99
Adapted λ	*equal*	95% (1%)	95% (3%)	97% (139)	97%	0.93

Table 3.1.: Performance of RGLVQ for the *Chromosomes* data set, comparing fixed vs. adaptive metric parameters λ, measured by the average **Train**ing and **Test** accuracies and their standard deviation (**Std**) in a 5-fold cross-validation with 10 repeats. From the respective last run, dissimilarities induced by λ are evaluated by **5-NN** classification accuracy, the separation ratio (**Sep.**, where smaller is better), as well as the average test accuracy of **SVM** (and number of support vectors **#SV**) from a repeated 5-fold cross-validation. All results are reported for two initialization methods for λ: *equal* costs, and a weighting scheme from [64] using *prior* knowledge.

minimal complexity of the classification boundary. Using more prototypes yields some improvements: with 3, 5, and 7 prototypes per class, 79%, 79%, and 81% average test accuracy is achieved, respectively.

However, metric adaptation with only one prototype enables a nearly perfect average accuracy of 97% for training and test sets. This demonstrates how a problem-adapted metric can alleviate the given classification task, even without a complex classifier model. (We observed no considerable benefit, when more prototypes are used.) Interestingly, this major improvement was achieved by subtle changes in λ from the initial scoring (see Figure 3.10b) to the final state after training (see Figure 3.10c, taken from the last cross-validation run). The adaptation mainly changes the replacement costs for neighboring scales of difference: many values on the first off-diagonals become nearly zero, signifying that these symbols are interchangeable within classes. At the same time, the gap costs for the symbol "=" become lower, which can be attributed to the fact that it is the most frequent symbol in the set (see Table A.1 in the Appendix Section A.2).

Comparing the class-separation in the fixed vs. the adapted dissimilarities, we observe that the separation ratio drops and the 5-NN accuracy improves, as reported in Table 3.1. Also, the average number of support vectors used in the trained SVM classifiers decreases drastically, which indicates a less complex classification boundary. This underlines our hypothesis, that metric learning can greatly facilitate class-discrimination, especially when the parameterization of the underlying dissimilarity measure is complex.

In the next step, we consider the case of initializing λ with *equal costs* (meta-parameters set to $\beta = 7$, $\eta = 0.45/N$, $\mathtt{k} = 1$). Surprisingly, this very simple setting of λ yields higher accuracies than the cost pattern proposed in [64], see the bottom two rows in Table 3.1. Without any adaptation of the alignment costs, RGLVQ achieves 89% average test accuracy for one prototype per class. This can be improved by increasing the model complexity: 93%, 95%, and 95% average test accuracy are achieved with 3, 5, and 7

prototypes per class, respectively. Learning the metric parameters provides the same level of improvement to 95%, however, with only one prototype, since the underlying data representation is tuned according to the classification task. Thereby, the resulting model is very sparse, and it offers the possibility to inspect and interpret the adapted metric parameterization. In this case, increasing the number of prototypes also results in a slightly better classification performance: for example, 5 prototypes per class yield 96% average test accuracy. Although the 5-NN accuracy is not increased, the separation ratio is improved by the metric adaptation, and the average number of support vectors for SVM drops, again supporting our claim.

3.6.3. Intelligent tutoring systems for Java programming

In the context of educational technology, *intelligent tutoring systems* (ITSs) have greatly advanced in recent years. The goal of these systems is to provide intelligent, one-on-one, computer-based support to students, as they learn to solve problems in a type of instruction that is often not available because of scarce (human) resources [133, 29, 136]. One approach to facilitate ITSs is based on the automatic clustering and classification of student solutions, see [46] and [C13b]. Therefore, a crucial ingredient is a reliable (dis)similarity measure for pairs of solutions [C13b]. While a solution could be represented in many forms, we will focus here on a representation as a (multi-dimensional) sequence. In this experiment, we will consider the dissimilarity between Java programs, as an example pointing towards the idea of adaptive metrics in an ITS for Java programming, as described in [C13b]. Given the complexity of syntactic structures, we demonstrate how the parameterization of such a dissimilarity measure can be adapted to facilitate a classification task.

To properly model Java programs as sequential data, we no longer consider discrete symbolic sequences as before, but instead refer to sequences with more complex, multi-modal entries. Each entry, in the following called a *node*, holds a collection of *properties*, where the number of properties K is fixed a priori for the given data set. For every single property, either a finite discrete symbolic alphabet, or a numeric domain is defined a priori. Given the property number $\kappa \in \{1, \ldots, K\}$, we refer to its designated alphabet or domain as Σ_κ, i.e. the set of all possible values for that property. A multi-modal sequence is denoted by: $\widetilde{a} = (a_1, \ldots, a_I, \ldots, a_{|\widetilde{a}|})$ where a_I is a node, and a_I^κ refers to the (symbolic or numeric) content of property number κ in this node.

In the case of Java, the nodes represent syntactic building blocks of a Java program, which were extracted from the abstract syntax tree via the official Oracle Java Compiler API. Properties are, for example, the node's position in the source code file (`codePosition`, an array of integers indicating the starting and ending line and column), the `type` of this node (e.g. a method declaration, a variable declaration, an assignment, etc.), or more specific properties like the `name` of a variable, method or class that is declared.

To define a parameterized alignment measure for such multi-modal sequences, we will use a generalization of the two alignment scenarios described in Section 3.3, on page 56. We adopt the *relevance weighting* from vectorial sequence alignment, but loose the restriction to numerical vectors containing real-valued entries, in favor of a more general notion of nodes containing multiple properties. Therefore, we replace the 'inner' dissimilarity measures for each vector dimension with a measure for each property: assume a symmetric dissimilarity $d^\kappa : \Sigma_\kappa \times \Sigma_\kappa \to [0, 1]$ for each property κ. This measure can be parameterized by some λ (denoted as d^κ_λ), but if this is not specified, we simply assume $d^\kappa(s, t) = 0 \Leftrightarrow s = t$, and $d^\kappa(s, t) = 1 \Leftrightarrow s \neq t$ for all symbols or values $s, t \in \Sigma_\kappa \cup \{-\}$ in the corresponding symbolic alphabet or numeric domain. This means that costs for replacements, insertions, and deletions can be specified, which corresponds to the case of a *scoring matrix* in Section 3.3, but now individually for every property κ. Then, we redefine the 'outer' dissimilarity between single nodes as:

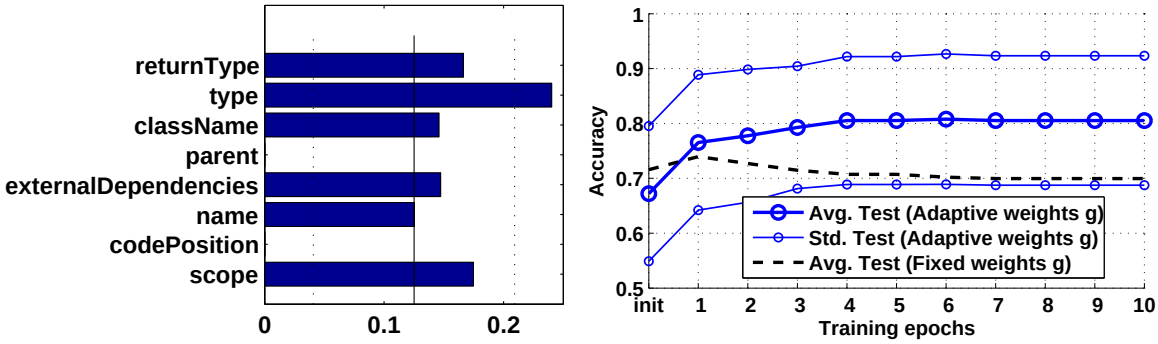
$$d_{\mathbf{g}}(a_I, b_J) = \sum_{\kappa=1}^{K} g_\kappa \cdot d^\kappa_\lambda(a^\kappa_I, b^\kappa_J) \ .$$

The vector $\mathbf{g} = (g_1, \ldots, g_K)$ contains the *relevance weights* g_κ for each property κ, which lie in the interval $[0, 1]$ and are normalized to sum up to one. Thus, for $g_\kappa = 1$, no other property is considered, but κ. If $g_\kappa = 0$, the property κ does not contribute to the alignment at all.

In this experiment, our goal is to learn the metric parameters of both, the outer and inner dissimilarity, i.e. g_κ and λ, respectively. Therefore, we use a two-stage consecutive process: first, we fix the 'inner' parameterization λ for all properties κ, and adapt the 'outer' parameters g_κ. Thereafter, the resulting weights g_κ remain fixed, and the 'inner' parameters λ are adapted for the property with the highest relevance, given by $\arg\max_{\kappa \in \{1,\ldots,K\}}(g_\kappa)$. For the adaptation in each stage, we can directly transfer the respective learning scheme for *relevance weights* and *scoring matrices*, as described in Section 3.4.

The *Sorting* data set consists of a collection of Java programs, which are freely available in online code-sharing platforms. The programs implement two different algorithms to sort sets of integer numbers: we collected $N = 78$ programs in total, of which 44 implement the *BubbleSort* algorithm, and 34 realize *InsertionSort*. From each program, the sequence of syntactic nodes was extracted by a parser, where 8 properties are defined for every node. A list of the properties is given in Figure 3.11a.

Initially, every property $\kappa \in \{1, \ldots, 8\}$ is weighted equally, with $g_\kappa = 1/8$. First, we aim to learn a configuration of weights g_κ to facilitate class discrimination. The RGLVQ classifier was trained, with and without metric adaptation, for 10 epochs with one prototype per class, in a 5-fold cross-validation with 5 repeats. The meta-parameters for metric learning were set to: $\eta = 0.002/N$, $\mathtt{k} = 1$, and $\beta = 200$ (i.e. using a de facto crisp alignment).



(a) Adapted weights g_κ

(b) Classification accuracies

Figure 3.11.: Results for the *Sorting* data set, in which the (semantically sound) adaptation of weights g_κ for properties κ (left) yields an improvement of 11% in average test accuracy (right) in a 5-fold cross-validation with 5 repeats.

The results in Figure 3.11 and Table 3.2 (the two top rows) show that RGLVQ with metric adaptation improves the average test accuracy by 11%, compared to the default metric with all equal weights g_κ. Under the initial metric paramterization, even a higher number of prototypes in RGLVQ does not yield comparable performance: for 7 prototypes per class, the average accuracy is 72% on the test set. The 5-NN accuracy and separation ratio are also improved by the metric adaptation, and the average number of support vectors for SVM decreases, indicating a simpler classification boundary.

The resulting weights are reported in the bar graph in Figure 3.11a and can be interpreted as semantically sound for the classification task: `type` is weighted as the most relevant property, while `parent` and `codePosition` are deactivated entirely. This is justified, since `type` holds the most important semantic information by specifying one out of 29 possible categorial values encoding the basic functionality of the respective syntax part. (The alphabet for this property refers to token types in the abstract syntax tree of a program, a full list of symbols and corresponding Java code examples can be found in Section A.3 in the Appendix.) In contrast, (i) `parent` and (ii) `codePosition` clearly introduce noise w.r.t. classification, since they encode (i) the index of the previous node in the syntax tree, and (ii) the position in the raw source code file, both of which can drastically change from minor alterations in the program and are thus not discriminative. The intermediate weights for the remaining properties like `className` and `returnType` are also justified, since they convey valuable information about the semantics, like the class of (return) variables, such as *Integer* or *String*. However, since they are empty for many nodes, the lower relevance, as compared to `type`, can be explained. Interestingly, the property `name` refers to textual definitions for variable, class, and function names, freely chosen by the programmer. Of course, such names are not guaranteed to be meaningful for class-discrimination, and could potentially introduce noise in the data. However, since our set contains programs from an educational context, these names are likely to

Method	*Init.*	Train (Std)	Test (Std)	SVM (#SV)	5-NN	Sep.
Fixed g	λ *equal*	75% (3%)	70% (14%)	75% (58)	82%	0.93
Adapted g	λ *equal*	81% (3%)	81% (12%)	87% (49)	87%	0.81
Adapted λ	g *prior*	83% (2%)	82% (9%)	87% (49)	87%	0.81

Table 3.2.: Performance of RGLVQ for the *Sorting* data set, comparing fixed vs. adaptive metric parameters (λ and g), measured by the average **Train**ing and **Test** accuracies and their standard deviation (**Std**) in a 5-fold cross-validation with 5 repeats. From the respective last run, dissimilarities (induced by λ or g) are evaluated with the **5-NN** classification accuracy, the separation ratio (**Sep.**, where smaller is better), as well as the average test accuracy of **SVM** (and number of support vectors **#SV**) from a repeated 5-fold cross-validation. The results refer to a two-stage learning process: the adaptation of property weights g (in the two top rows) where λ is set to *equal costs*, and the subsequent adaptation of costs λ (in the bottom row) where g is fixed to the result of the previous learning procedure.

be defined in an explanatory fashion, which justifies the intermediate weighting.

To facilitate the classification further, we assume the trained weights g_κ as fixed, and subsequently adapt the metric again, now by learning the parameters d^κ_λ for the most relevant property `type`. Thus, κ refers to the index of property `type` in the following, and we focus on the alphabet Σ_κ of 29 symbols, as listed in the Appendix Section A.3. We therefore return to the learning scheme for alignment scoring parameters d^κ_λ, in the following denoted as λ_{km} for symbols $k, m \in \Sigma_\kappa \cup \{-\}$. The subsequent adaptation improves the results again, by 1% for the average test accuracy. While this is only a moderate quantitative enhancement, it can be seen as a refinement of the metric, since the standard deviation of training and test accuracies was reduced, suggesting a higher robustness, see Table 3.2 (the bottom row).

3.6.4. Reducing computational demand

Besides an approximation of prototypes by its `k` most prominent exemplars, a variety of further approximation techniques can be integrated to enhance the computational performance. While these methods are not mandatory for standard data sets, they become necessary as soon as larger data sets, e.g. $N > 500$ are addressed: The complexity of RGLVQ scales quadratically with the number of data points, so that it becomes infeasible for large data sets. We will discuss two options for speedup, which consider two different bottlenecks of the computational load:

- The first part addresses the computation of derivatives with respect to metric parameters: soft alignment requires the consideration of the full alignment path for every metric parameter, while crisp alignment reduces to contributions of the optimal path only. We will consider an approximation scheme, which disregards

small contributions of the alignment paths, enabling a computation of the derivatives in linear time with respect to sequence length in the best case, as compared to squared complexity. This approximation is particularly relevant for long input sequences.

- The second part addresses the computation of dissimilarities by means of a reference to the full dissimilarity matrix $\mathbf{D}$: the full dissimilarity matrix scales quadratically with the number of data. We can approximate $\mathbf{D}$ by a low rank matrix via the popular Nyström approximation. Since $\mathbf{D}$ is used in matrix vector operations only, a low rank approximation speeds up these operations to linear instead of quadratic time with respect to the number of sequences. This approximation is particularly relevant if a large number of training sequences is considered.

Approximated alignment derivative As before, we exemplarily consider the setting of a discrete alphabet and the adaptation of the scoring matrix, parameterized by λ_{km}. The overall runtime for online learning of metric parameters is strongly affected by the computational effort to calculate a single alignment derivative: given a set of sample sequences $\mathcal{S}$ and the set of exemplars $\mathcal{E}_j$ for one prototype α^j in one learning epoch, the derivative $\partial d^*(\bar{a}^i, \bar{w}^l)/\partial d_\lambda$ is calculated for all pairs of samples $\bar{a}^i \in \mathcal{S}$ and exemplars $\bar{w}^l \in \mathcal{E}_j$, i.e. it is done $|\mathcal{S}| \times |\mathcal{E}_j| = N \cdot \mathtt{k}$ times for the update w.r.t. one prototype in one epoch alone.

Therefore, we empirically evaluate the speedup gained from dropping alignment paths with a small contribution, as follows: In the limit $\beta \to \infty$, contributions restrict to the best alignment path, hence derivatives $\partial d^*(\bar{a}, \bar{b})/\partial \lambda_{km}$ for all λ_{km} can be computed in time $\mathcal{O}(|\bar{a}| + |\bar{b}|)$ based on the DP matrix. In general, derivatives are weighted sums corresponding to alignments of the symbols k and m at some position (I, J) of the matrix. Weighting takes into account all possible paths which include this pair according to the path eligibility measured by $\mathrm{softmin}'(A_i)$ for actions $A_i \in \{A_{\mathtt{Rep}}, A_{\mathtt{Ins}}, A_{\mathtt{Del}}\}$ on the path. The worst case complexity is $\mathcal{O}(|\bar{a}| \cdot |\bar{b}| \cdot |\Sigma|^2)$, using backtracing in the alignment matrix. We propose an approximation based on the observation that a small $\mathrm{softmin}'(A_i)$ leads to a small weight of paths including A_i. Hence, we store the 3 terms $A_{\mathtt{Rep}}$, $A_{\mathtt{Del}}$, $A_{\mathtt{Ins}}$ together with the distances $\mathrm{softmin}(A_{\mathtt{Rep}}, A_{\mathtt{Del}}, A_{\mathtt{Ins}})$ in the data structure of the DP matrix, and we cut all values $\mathrm{softmin}'(A_i) < \theta$ for a fixed threshold $\theta \geq 0$. Backtracing depends on the nonzero values only, so that a speedup to linear complexity is possible in the best case.

The threshold θ therefore determines that values $\mathrm{softmin}'(A_i) < \theta$ are ignored in the backtracing of alignment paths. Since the impact of θ depends on the alphabet size and sequence length, it should be tuned according to good classification results for the given data set. Typical values lie in the interval $\theta \in [0.01, 0.2]$. As a simple test scenario, we created several sets of random sequences, each consisting of 10 sequences $\bar{a}^i \in \Sigma^L$ with $\Sigma = \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}\}$, with four different choices of length L. For various thresholds θ, we

tracked the runtime of calculating alignment derivatives for all 100 sequence pairs on a standard laptop computer with an *Intel Core i7* processor (4 cores at 2.9 GHz, and calculations done in parallel). The results in Table 3.3 clearly show how increasing θ drastically reduces the computational effort, especially for longer sequences.

To demonstrate that reasonable approximations do not impede classifier performance in practice, we performed single training runs on the *Chromosomes* data from Section 3.6.2, using a random split of 80% training and 20% test data, with various settings of θ. The training was performed on a server computer with two *Intel Xeon X5690* processors (each with 6 cores at 3.47 GHz), using the same meta-parameter settings as in the original experiment. Table 3.4 lists the achieved test accuracies and runtimes, in comparison with the original result from Section 3.6.2. The values show that a slight approximation with $\theta = 0.1$ already reduces the average runtime for one learning epoch by 21%, without decreasing the classification accuracy. More crude degrees of approximation yield further, but marginal speedup, which stagnates for settings $\theta \geq 0.2$, while the accuracy drops continuously to 90% for the extreme setting of $\theta = 0.65$. In general, choices $\theta > 1/3$ carry the risk of loosing potentially valuable learning stimuli, since all three values $A_{\texttt{Rep}}, A_{\texttt{Ins}}, A_{\texttt{Del}}$ could be lower than θ for certain steps in the soft alignment, and therefore this entire alignment path would be ignored.

Table 3.3.: Runtimes (in seconds) to calculate the alignment derivatives for all pairs of random strings $\bar{a}^i \in \Sigma^L$, $i \in \{1 \ldots 10\}$, using different thresholds θ and $\beta = 10$. (Note, that this is not a classification task to discriminate labeled data, but a plain runtime test using all pairs of sequences. Therefore, no classification accuracies are reported.)

Sequence length L	100	150	200	250
Runtime for $\theta = 0$	7.2	24.6	87.6	426
Runtime for $\theta = 0.15$	4.2	13.2	31.8	98.4
Runtime for $\theta = 0.2$	1.8	6.6	13.8	27.0
Runtime for $\theta = 0.25$	1.2	3.6	7.2	13.2

Table 3.4.: RGLVQ with metric adaptation, evaluated in single training runs on the *Chromosomes* data from Section 3.6.2, using an approximation technique to calculate the alignment derivative. The degree of approximation is controlled via the threshold θ by which marginally contributing alignment paths are neglected in the derivative calculation, i.e. with higher θ the approximation becomes more crude. For each setting of θ, the final test accuracy is reported, along with the average runtime for one learning epoch (in seconds). For small settings of θ, the approximation yields speedup without sacrificing accuracy.

Degree of approximation	no approx.	$\theta = 0.1$	$\theta = 0.2$	$\theta = 0.35$	$\theta = 0.65$
Epoch runtime in s (%)	303 (100%)	240 (79%)	225 (74%)	228 (75%)	228 (75%)
Test accuracy	avg. 95%	97%	95%	94%	90%

Nyström approximation In our algorithm, metric parameters λ are adapted in every epoch. They induce a different dissimilarity measure, thus $\mathbf{D}$ needs to be re-calculated according to this new parameterization, see Line 14 in Algorithm 1 on page 62. To avoid the repeated alignment between *all* sequence pairs, we refer to the *Nyström* technique to approximate the full matrix as $\mathbf{D}^{\nu} \approx \mathbf{D}$.

We recall, from Chapter 2, Subsection 2.2.5, that the Nyström approximation can be integrated into RGLVQ to achieve considerable speed-up. To demonstrate the suitability of Nyström approximation for our method, we replaced the corresponding dissimilarity calculations in our algorithm, and consider single training runs on the *Chromosomes* data from Section 3.6.2. Using this data set, with $N = 400$ sequences, we can showcase the validity of the Nyström technique in principle. However, since the approximation trades the $\mathcal{O}(N^2)$ complexity for $\mathcal{O}(V^2 \cdot N)$ based on the chosen number of landmarks V, the benefits become more apparent in large-scale scenarios, where the number of sequences is very high, e.g. $N > 1000$, and a choice $V \ll N$ is justified. For the selection of appropriate landmark sequences, we use a random sample of the data in the corresponding epoch. This simple strategy relies on no assumptions about the data structure, and has shown to work well in our experiment. However, more informed selection plans can be found in the literature, see [141] for example.

The results in Table 3.5 show that the average runtime for one training epoch is decreased by 17%, when 70% of the data are chosen as landmarks, i.e. $V = \lfloor 0.7 \cdot N \rfloor$. However, this causes a slight drop in test accuracy, from 95% to 89%. Interestingly, we observe that the accuracy does not decrease monotonically with the crudeness of the approximation: using 80% of all sequences as landmarks yields a better accuracy than using 90%. In the context of Nyström approximation, this effect has been observed in the literature for the Copenhagen Chromosomes data, see [143]. A plausible explanation would be that noise in the data representation is suppressed at a certain level of approximation. Our experiment shows, that the Nyström approximation is a valid technique to decrease the computational effort of the proposed algorithm. A more elaborate evaluation is the subject of ongoing work, since a realistic application scenario would involve larger data sets.

Table 3.5.: RGLVQ with metric adaptation, evaluated in single training runs on the *Chromosomes* data, using the Nyström approximation to (repeatedly) calculate the dissimilarity matrix $\mathbf{D}^{\nu} \approx \mathbf{D}$. The degree of approximation is controlled via the number of landmarks V, in relation to the total number of sequences N, i.e. with lower V the approximation becomes more crude. For each setting of V, the final test accuracy is reported, along with the average runtime for one learning epoch (in seconds).

Degree of approximation	no approx.	$V = \lfloor 0.9 \cdot N \rfloor$	$V = \lfloor 0.8 \cdot N \rfloor$	$V = \lfloor 0.7 \cdot N \rfloor$
Epoch runtime in s (%)	303 (100%)	274 (91%)	261 (87%)	251 (83%)
Test accuracy	95%	91%	93%	89%

3.7. Discussion

The metric adaptation scheme proposed in this chapter transfers the principle of relevance learning to the structural domain, by addressing sequence alignment in particular. However, the approach opens the way towards efficient metric adaptation schemes for distance-based methods in other discrete structure spaces, such as trees or graph structures. A similar metric learning becomes possible, provided the metric is differentiable with respect to its significant parameters. Note that, unlike the approach [66], we do not assume differentiability of the dissimilarity measure with respect to the data structures itself, but differentiability with respect to the adaptive metric parameters only. Hence, discrete structure spaces are covered by our proposed technique.

So far, our method relies on a global metric with one set of parameters λ. This can be problematic, if relevant structural constituents change, depending on their position in the data space, as is common e.g. for classification problems with more than two classes. In this context, it could be beneficial to use class-specific parameter sets λ^j associated with every prototype $\mathbf{w}^j$. For vectorial LVQ, this 'local' metric learning has been proposed in [119, 51], and it could be transferred to the relational setting. However, its computational efficiency becomes problematic, due to the computational demand for calculating individual dissimilarity matrices for every parameter set λ^j. In this context, it might be worthwhile to investigate efficient low-rank metric approximations only.

Another important question arises in the context of the interpretability of metric parameters: it is not clear whether parameter configurations are unique, and whether invariances exist, caused e.g. by structural invariances. In the latter case, metric parameters do not necessarily relate to the true relevance of these structural constituents, rather random effects can occur. This property has recently been observed in the vectorial setting, when dealing with very high data dimensionalities. In this case, high relevance can be related to correlations of data dimensions in some cases, falsely suggesting a high feature relevance if interpreted directly [123]. Therefore, before relying on the interpretation of metric parameters, a normalization of the representation with respect to structural invariances is mandatory. For structural data, similar effects can be expected. Therefore, it is the subject of ongoing work to exactly identify these invariances for a given metric, and to devise unique representatives of the resulting equivalence classes for valid interpretation.

Chapter 4.

Unsupervised suitability assessment for data representations

Chapter overview *This chapter presents quantitative criteria and visualization tools to assess the suitability of data representations in an unsupervised scenario. Since low-dimensional Euclidean embeddings of data can serve as an alternative representation, and help humans to investigate the data's neighborhood structure, we demonstrate how their reliability can be measured and integrated into a visualization. Additionally, we transfer this principle to achieve an unsupervised comparison between different dissimilarity measures for the same given data set.*

Parts of this chapter are based on:

[J13] B. Mokbel, W. Lueks, A. Gisbrecht, and B. Hammer. Visualizing the quality of dimensionality reduction. *Neurocomputing*, 112:109–123, 2013.

[C12f] B. Mokbel, S. Gross, M. Lux, N. Pinkwart, and B. Hammer. How to quantitatively compare data dissimilarities for unsupervised machine learning? In *ANNPR 2012*, volume 7477 of *LNCS*, pages 1–13, 2012.

[C11b] B. Hammer, M. Biehl, K. Bunte, and B. Mokbel. A general framework for dimensionality reduction for large data sets. In *WSOM 2011*, pages 277–287, 2011.

4.1. Motivation

In the previous chapters, we have discussed dissimilarity-based data representations as a means to tackle particular challenges of complex data. In this chapter, we will address several problems regarding complex data and dissimilarity data, which arise in the absence of a classification task, i.e. if class-labels are not available.

In Chapter 2, we raised the issue that many established machine learning methods are restricted to Euclidean vector spaces, often requiring vectorial input data. As one possibility to circumvent this limitation, we proposed extensions of popular prototype-based techniques to process (non-Euclidean) dissimilarity data directly. However, there is an alternative approach, which is independent of the applied learning scheme: given a dissimilarity matrix, an approximate vectorial representation can be calculated, via

so-called *dimensionality reduction* (DR) techniques, which replicates the original data structure and distribution. This *Euclidean embedding* may then serve as a substitute input for machine learning algorithms. The same principle can be helpful, if a feature-based data description is available, but it is very high-dimensional. By substituting the original feature vectors with lower-dimensional counterparts, we can avoid or diminish problems regarding high-dimensional spaces in the learning procedure, such as the curse of dimensionality, or computational demand that scales with the number of features.

4.1.1. Scientific contributions and structure of the chapter

In this chapter, we will discuss methods to evaluate the suitability of data representations for unsupervised learning tasks, addressing two important questions: (i) How can we assess the quality of low-dimensional embeddings, as a substitute for the given original data representation? (ii) How can we compare given dissimilarity-based data representations, without addressing a specific machine learning method?

The following key contributions are presented:

- A generic approach for quality assessment in DR, introduced by Lee and Verleysen in [81], is extended by a point-wise evaluation scheme. This allows for a direct integration of local quality ratings into a scatter plot for 2D or 3D embeddings.
- We refine the parameterization of the given quality measure to enable more fine-grained control in the evaluation process.
- We transfer the quality evaluation framework to enable the comparison of general dissimilarity matrices. This way, it is possible to assess, prior to learning, in how far two different dissimilarity measures or choices of parameters lead to different results, and if so, for which data they differ.

The following Section 4.2 briefly describes some well-known DR techniques to obtain a Euclidean embedding from high-dimensional vectors or pairwise dissimilarities[1]. As an interesting special case, DR methods are able to produce an embedding in a two- or three-dimensional Euclidean space, which allows for a direct visualization of the data set's neighborhood structure.

Since the substitute vectors are usually an approximation of the original data, we discuss methods to evaluate their individual reliability in an unsupervised setting, in Section 4.3, and thereby propose a conceptual extension of the evaluation framework established in [81]. This provides a tool to assess the suitability of the alternative data representation at a fine-grained level, which can be accompanied by intuitive visualizations of the data distribution.

[1] We recognize DR not only as a tool for visualization, but also to create an alternative, low-dimensional data representation for subsequent machine learning. Note, however, that DR methods are often themselves deemed machine learning, as they utilize similar concepts and optimization schemes.

In Section 4.4, we will address the problem that users who control the overall processing pipeline (see Chapter 1) are often faced with an abundance of design choices regarding dissimilarity measures. On the one hand, several different measures are usually available for a certain data type, e.g. for symbolic sequences. On the other hand, their parameterization can have a strong impact on the outcome, as was discussed in detail in Chapter 3. This increases the need to compare different dissimilarity-based data representations quantitatively. We propose a technique which realizes such a comparison per data instance, alongside visualizations obtained by DR.

4.2. Low-dimensional Euclidean embeddings

Dimensionality reduction (DR) is a common problem in the field of data analysis. In broad terms, the goal of DR is to find low-dimensional vectors $\{\mathbf{y}^1, \ldots, \mathbf{y}^N\} = Y \subset \mathbb{R}^L$, which resemble the distribution and structure of a data set $\{x^1, \ldots, x^N\} = X$. The target dimensionality L is determined a priori, typically by a user. The data X may be given explicitly in the form of high-dimensional vectors $x_i = \mathbf{x}_i \in \mathbb{R}^D$, or represented implicitly by providing a matrix of pairwise dissimilarities $d(x_i, x_j) = [\mathbf{D}]_{(i,j)}$. We refer to this as the *input data*. The resulting vectorial representation Y is accordingly called *output data*, or simply the *embedding*, in the following. If the inputs are vectors in $\mathbb{R}^D$, and thus their dimensionality is known to be D, we typically assume that $L < D$ is chosen, in order to achieve a reduction of the dimensionality. An important special case is to select $L = 2$ or $L = 3$, which allows to visualize the data distribution in a two- or three-dimensional scatter plot. This way, human experts can easily inspect neighborhood structures in large data collections, and gain additional insights about the data set's characteristics.

To realize a mapping $X \ni x^i \longrightarrow \mathbf{y}^i \in Y \subset \mathbb{R}^L$, a broad variety of techniques[2] has been proposed in the machine learning literature, see e.g. [80, 23, 131, 132] and [C11b] for overviews. Although the earliest techniques, e.g. multidimensional scaling, have been proposed more than 50 years ago, DR still constitutes a very active research field. Many methods have been proposed with a clear focus on visualization, i.e. an embedding in 2D or 3D space. However, the underlying principles are usually independent of a specific target dimensionality, so that an arbitrary $L < D$ may be chosen. Successful applications of DR can be found in diverse areas, such as robotics, medicine, the web, biology, etc., see [105, 5, 83, 68], for example.

While the embedding Y aims to adequately resemble the original data X, dimensionality reduction constitutes an ill-posed problem: not all the structure and relations

[2] Note, that some techniques aim to reduce the number of instances $|Y| < |X|$ in addition to the dimensionality reduction, resulting in a condensed representation of the original data, such as the GTM presented in Chapter 2. However, we will assume $|Y| = |X|$ in the following, meaning that every input has exactly one low-dimensional counterpart. This does not rule out that some vectors in Y may be equal, i.e. at the same position in the output space $\mathbb{R}^L$.

that exist in intrinsically high-dimensional data can be faithfully represented in a lower-dimensional space, and it is not clear which relations should be preserved. The compromises ultimately depend on the given data set and the application scenario at hand. In the following, we will assume that the intrinsic dimensionality of the original data is higher than the one of the embedding space. Therefore, the user faces the problem of choosing an appropriate DR technique, and an adequate configuration of its parameters, along with other design choices that affect the DR procedure, e.g. preprocessing steps.

The variety of possible strategies has resulted in the development of many different DR techniques. Often, the algorithms are based on the optimization of an objective function, in which the goal for preserving information is formalized in mathematical terms. In the following, we will briefly introduce three established DR methods, which constitute particular landmarks in the field of DR. For detailed explanations, please refer to the overviews in [80, 23, 131, 132], and the individual references therein. Other techniques will be mentioned briefly, without detailing the mathematical background. Given the premise of this chapter, we restrict our discussion to unsupervised DR techniques only.

In our formalization, we will denote distances in the original data space by $d\left(x^i, x^j\right) = d_{ij}$ and in the embedding by $\delta\left(\mathbf{y}^i, \mathbf{y}^j\right) = \delta_{ij}$. It is assumed, that δ_{ij} is the Euclidean distance, while d_{ij} can be a dissimilarity measure for raw data, as discussed in the previous chapters.

Principal component analysis (PCA)

One simple, and yet fundamental, DR method is the well-known *principal component analysis* (PCA), see [80][ch. 2] and [35][ch. 10]. PCA uses an unsupervised linear transformation of the data, which minimizes the loss of information, as measured by the sum of squared errors. The method is restricted to vectorial input data. PCA defines a linear mapping $\mathbf{y}^i = \mathbf{x}^i \cdot \mathbf{A}$ for input vectors $\mathbf{x}^i$, where the projection matrix $\mathbf{A}$ is the solution to the costs

$$\min \sum_{i}^{N} \left\| \mathbf{x}^i - \mathbf{x}^i \mathbf{A}\mathbf{A}^\intercal \right\|^2$$

for orthonormal vectors in the columns of $\mathbf{A}$. It can be shown algebraically, that these vectors correspond to the directions of the largest variance in the data set, i.e. the principal components. They are also the eigenvectors of the data covariance matrix. By specifying the number of principal components used in the projection, we can regulate the embedding dimensionality L. Due to its simplicity and well-understood behavior, PCA is often used for data visualization, although the linear mapping function severely restricts its versatility. Modern DR methods commonly use nonlinear mapping schemes. Several approaches have been proposed which can be interpreted as nonlinear extensions of PCA, such as kernel PCA [120], auto-encoding neural networks [59], or projections based on principal curves which pass through the 'center' of the high-dimensional data [44].

Multidimensional scaling (MDS)

The approach of PCA can be described as a global preservation strategy for pairwise inner products of the data, aiming for an embedding which reproduces the largest contributions of the metric in the data space, for vectorial input data. *Multidimensional scaling*[3] (MDS), see [80], formulates an error term to express the distortion of distances explicitly. However, it is often reasonable to focus on maintaining local neighborhoods (i.e. small distances) as good as possible, while the accurate replication of global neighborhoods (i.e. large distances) is less critical in the embedding. Therefore, it incorporates a weighting scheme to control the emphasis in the error function.

MDS aims to preserve distances d_{ij} by minimizing

$$E_{\mathrm{MDS}} = \sum_{ij}^{N} \mathcal{W}_{ij} \big(d_{ij} - \delta_{ij}\big)^2,$$

where the weights $\mathcal{W}_{ij}$ can be chosen appropriately, e.g. emphasizing the contribution of small distances by $\mathcal{W}_{ij} = 1/d_{ij}$, see [80]. Optimization can take place by gradient descent methods, typically starting with a random initial solution for the embedding points $\mathbf{y}^i$. Given the straightforward formalization of the preservation goal, MDS is a simple and well-established nonlinear mapping technique for general dissimilarity data.

Stochastic neighbor embedding (SNE) and t-distributed SNE (t-SNE)

The formulation of MDS aims to replicate the (weighted) pairwise distances, based on their exact numerical representation. Instead, *Stochastic neighbor embedding* (SNE) [58] uses a more abstract representation of pairwise proximity, by defining probabilities for point neighborhoods

$$p_{j|i} = \frac{\exp\left(\frac{-d_{ij}}{2\sigma_i}\right)}{\sum_{k\neq i} \exp\left(\frac{-d_{ik}}{2\sigma_i}\right)} \quad \text{and} \quad q_{j|i} = \frac{\exp\left(-\delta_{ij}\right)}{\sum_{k\neq i} \exp\left(-\delta_{ik}\right)}$$

in the data space ($p_{j|i}$) and the embedding ($q_{j|i}$), assuming squared data distances d_{ij}. Then, it minimizes the Kullback-Leibler divergence $E_{\mathrm{SNE}} = -\sum_{ij} p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$, where individual bandwidths σ_i are determined for every point, according to the so-called *perplexity* meta-parameter. The perplexity is an integer, which can be thought of as a soft k-ary neighborhood: it specifies the approximate number of neighbors with neighborhood probability $q_{j|i}$ which is not in the tail of the Gaussian function. The bandwidth σ_i for a given point is then automatically adjusted with a search algorithm to best meet this requirement. A gradient descent is used for the optimization of the cost function.

[3] While the name can address a whole class of algorithms, we will use the term *multidimensional scaling* for a type that is known as *non-metric* MDS in many textbooks, i.e. preserving distances by minimizing a weighted error term, measuring the differences between d_{ij} and δ_{ij}.

SNE suffers from the so-called crowding problem, which describes the particularly frequent issue that many points are crowded in the embedding, see [131]. The enhanced technique *t-distributed stochastic neighbor embedding* (t-SNE) avoids this issue, by assuming a long-tail distribution in the embedding space, the *student-t* distribution, see [131]. This allows for an approximation of medium-sized distances in the data by longer distances in the embedding, reflecting the fact that low-dimensional embedding spaces offer less degrees of freedom to depict a large number of medium distances correctly. The SNE cost function is slightly modified to $E_{\text{tSNE}} = \sum_i \sum_j p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right)$, where $p_{ij} = \frac{p_{j|i}+p_{i|j}}{2N}$ symmetrizes the conditional probabilities, and

$$q_{ij} = \frac{\left(1+\delta_{ij}/\varsigma\right)^{-\frac{\varsigma+1}{2}}}{\sum_{k\neq l}\left(1+\delta_{kl}/\varsigma\right)^{-\frac{\varsigma+1}{2}}}$$

is given by student-t with parameter $\varsigma = -1$, for example. Optimization takes place by means of a gradient method.

The t-SNE method is widely used today, and its underlying concept has spawned several similar techniques in recent years, taking a probabilistic approach to DR and drawing inspiration from information retrieval, such as the *neighborhood retrieval visualizer* (NeRV) [135], and the *Jensen-Shannon embedding* (JSE) [79]. These three state-of-the-art techniques (t-SNE, NeRV, and JSE) are rather similar, and their cost functions can be written in a unified form, where only components and parameters are altered, see [79].

General remarks about DR

Among the plethora of available DR techniques, the three methods described above stand out as important landmarks in the history of the research field. Although PCA and MDS are no longer considered state-of-the-art for visualization purposes, they are rather easy to explain, and are therefore popular outside the machine learning and DR research community, see [15, 4, 3]. In practical applications among several other disciplines, these methods are still widely used and appreciated for their simplicity. Among modern nonlinear embedding techniques, t-SNE is considered particularly popular and has been described as a leap forward in DR quality [79]. Hence, we will focus on these three techniques in our experiments and examples in the remainder of this chapter.

The different approaches result in qualitatively very different visualizations for a given data set, see the overview papers [80, 131] for direct comparisons. Therefore, it is not clear a priori, which DR technique is best suited for the task at hand. In addition, virtually all recent techniques have parameters to control (in some way) the preservation strategy for the embedding. Hence, depending on the chosen parameters, even a single DR method can lead to vastly diverse results. Moreover, many nonlinear DR techniques do not arrive at a unique solution due to random aspects of the algorithm. Instead,

they can produce different outputs in every run, corresponding to different local optima of the objective. Therefore, it is possible that qualitatively different solutions can be obtained by a single method with a single set of model parameters.

Assessing the reliability of an embedding

Usually it is not clear whether differences in the dimensionality reduction (between different methods, parameters or runs) represent different relevant aspects in the data or signify unsuitability of a method. Further, it can happen that suboptimal results are obtained simply because of numerical problems, such as (bad) local optima. At the same time, it is very hard for humans to judge the quality of a given embedding by visual inspection. The user cannot compare it against a ground truth, as this data is inaccessible due to its high dimensionality. Therefore, we need formal measures which judge the quality of a given data embedding. Such formal measures should evaluate, in an automated and objective way, in how far the structure of the original data is preserved in the low-dimensional representation. Apart from their importance for practical applications, quality measures are generally relevant to automatically evaluate and compare DR techniques for research. As reported in [135], a high percentage of publications on data visualization evaluates results in terms of visual impression only – in [135], about 40% of the 69 referenced papers did not use any quantitative evaluation criterion. Even if formal evaluation criteria are used, these differ from one application to the next, referring e.g. to a local misclassification error for labeled data [132, 135], the reconstruction error provided an inverse mapping is possible [81], or local preservation of neighborhoods [135]. Further, many popular benchmark data sets in the literature are artificially generated and thus are of limited use for a realistic evaluation of DR methods [132]. Although a few real-world data sets are currently available (see e.g. [131]), there does not exist a large variety of data encompassing different characteristics together with suitable evaluation criteria.

4.3. Quantitative quality assessment

Section overview In this Section, we will first give a short overview over existing approaches of quality assessment for dimensionality reduction in Subsection 4.3.1, and will discuss some fundamental features which distinguish their strategies, concluding with the general motivation for our own approach. Thereafter, we will focus on the *co-ranking matrix* from [81], which serves as a unifying framework to represent several other measures. In Subsection 4.3.2, we will briefly describe the co-ranking matrix itself, and, in Subsection 4.3.3, propose to augment data visualizations by point-wise quality contributions based on the co-ranking framework. Subsection 4.3.4 discusses how a fairly simple parameterization in established quality measures causes problems regarding the interpretability of the evaluation results. We propose a new parameterization which allows for more fine-grained control over the evaluation focus, and which facilitates a more specific analysis of the given embedding. After we demonstrate the benefits of our approach on several artificial examples, we show, in Subsection 4.3.5, how it performs in real-world visualization scenarios in comparison with the former model. In Subsection 4.3.6, we briefly summarize our findings and point out follow-up questions.

4.3.1. Principles of quality assessment for DR

Several quality criteria to evaluate DR have been proposed in recent years, see [81] for an overview of the more prominent measures. However, the problem to define formal criteria suffers from the ill-posedness of DR itself: it is not clear a priori which structural aspects of the data should be preserved in a given task. Generally, the existing quality measures evaluate in how far the original data relations agree with the ones of the embedding. A similar notion forms the basis for most objective functions in DR methods, but the specifics and priorities of the agreement calculation are a matter of ongoing research and debate. By formalizing an objective, every DR strategy gives rise to a perfect mapping in terms of the global optimum of this objective, and thus it incorporates a quality measure in itself. However, the goal of DR is to produce a representative embedding of the data and, thus, algorithmic aspects such as easy optimization are a prior motive, while a quality measure is used to gain insight into the properties of the embedding. Therefore, the quality measure can and should be general, as well as understandable, so the user can easily interpret its results, and thus judge the trustworthiness of the embedding. Regarding application scenarios, a formal quality evaluation can assist the user in two ways, to which we will refer in our further discussion:

(a) Given a data set, formal measures help to compare different DR methods along with their parameter settings, as demonstrated in extensive experiments, e.g. in [135, 81, 82, 79]. Therefore, by iterative comparison, a DR method's parameter configuration could be optimized interactively. A relatively coarse-grained, overall quality assessment seems sufficient for this purpose.

(b) Given a single embedding, formal measures can provide the user with information about the qualitative characteristics of the observed embedding. Since the user intends to gain knowledge about the original data, it is beneficial to analyze – in detail – the approximative representation in the embedding. For this task, fine-grained evidence is necessary, see [7, 129].

In the following, we will briefly discuss existing strategies for quality assessment in the literature and point out some basic distinguishing features. DR evaluation compares characteristics derived from the data X to corresponding characteristics derived from the embedded points Y. Since a formal mathematical characterization together with a unifying framework of many DR evaluation techniques has already been developed [81], we do not aim at a formal definition of the particular methods. Instead, we highlight distinguishing characteristics of the evaluation methods.

Distances vs. ranks Most DR evaluation techniques relate to pairwise distances of data in some way. One fundamental distinguishing aspect is whether the pairwise distances in the high-dimensional data are compared directly with the low-dimensional setting, or if only their order, i.e. ranks, is considered. All evaluation measures mentioned in [81, 124] use ranks, whereas in [135] the criteria *precision* and *recall* may be evaluated for any form of proximity measure, including ranks of distances, or distances itself. From the measures presented in [44], the *quality of point neighborhood preservation* and *quality of group compactness* are both based on K-nearest-neighbors, i.e. they consider ranks; while the *quality of distance mapping* can be evaluated for both, distances and ranks alike. While absolute distance information is lost when only the order of distances is considered, it has the benefit that any notion of pairwise proximity in the original data (e.g., distances, dissimilarities, similarities, neighborhood probabilities) is comparable with the Euclidean distances of the embedded data points, since ordering the neighbors of a point is possible in all these cases. Further, ranks are invariant to monotonic transformations of the distances.

Neighborhood scales Many quality measures aim to give an overview of the visualization's characteristics on different scales, by considering the agreement rates over varying neighborhood sizes (usually averaged over all data points). This facilitates to some extent the fine-grained analysis mentioned in our introductory statement (b). The neighborhoods are either defined via hyperspheres of a radius ϵ centered at each point, or, alternatively, as the K nearest neighbors of each point. All measures discussed in [81, 38, 124] use K-neighborhoods, while in [135] ϵ-hyperspheres are considered. Note, that the latter case is more general, since an ϵ-radius can serve as a boundary for any kind of proximity, including ranks[4].

[4]When distances are replaced by their respective ranks, limiting to a radius of size ϵ for a point means choosing its ϵ nearest neighbors.

Agreement evaluation Based on these neighborhoods for some fixed K or ϵ, there are different possibilities to evaluate the agreement between the characteristics in the high-dimensional data and its counterpart in the embedding. Some quality measures simply calculate the ratio of agreed points within these regions, see e.g. [81, 38]. Others consider a weighted combination of the agreement rate inside and outside of the neighborhoods, like the *mean relative rank errors* from [80]. A recent criterion known as *local strict rank order preservation* [124] counts strictly preserved ranks. Instead of counting the number of agreed neighbors, the *quality of distance mappping* [44] uses the correlation of pairwise distances between the original and the embedded data. For the *precision* and *recall* for DR, as defined in [135], one can use ranks instead of pairwise distances between the data, which leads to defining regions of ϵ nearest neighbors. Then, precision and recall are both equal to calculating the average number of agreeing neighbors, which coincides exactly with the *quality* Q_{NX} from [81], the *quality of point neighborhood preservation* in [44], as well as the *agreement rate* from [38]; criteria which were all proposed independently. Supplemental to the *quality* from [81], the *behavior* indicator gives insight about the types of errors which occur in the visualization: either points become closer in the embedding, or points are farther apart than in the original, called *intrusive* or *extrusive* behavior, indicated by values below or above zero respectively. While most of these concepts aim at our scenario described in (a), the behavior indicator reveals more details about the embedding's characteristics.

Aggregation of pointwise contributions Point-wise agreement rates (i.e. independently regarding every point's neighborhood) are usually aggregated to fewer numeric values, in order to deliver a compact evaluation result, like a curve over growing ϵ or K. For the aggregation, a simple average is often used. While this is beneficial when comparing several DR techniques for the same data set, as addressed in statement (a), the aggregation hides the local quality characteristics of the embedding, which would be beneficial regarding our statement (b).

Scale-independent criteria To obtain even fewer numeric values which subsume the quality on all (or some important) neighborhood scales, different possibilities can be found in the literature. In [82] averaging the quality curve $Q_{\mathrm{NX}}(K)$ over certain ranges of K has been proposed. A splitting point $K_{\max}$ is defined as the first maximum of the curve with respect to its baseline. Then, the mean quality for all $k \leq K_{\max}$ represents the local quality, whereas the mean quality over all $K > K_{\max}$ defines the global quality.

Recently, in [79, 78], Lee and colleagues presented a new averaging scheme for the quality curve $Q_{\mathrm{NX}}(K)$: First, the baseline is subtracted, and the quality value is weighted evenly for all neighborhood scales K, in order to gain a more equalized and intuitive representation. This results in a justified measure $R_{\mathrm{NX}}(K)$, for which a logarithmically weighted average, or area under the curve, can be considered, indicating where the gross mass of the $R_{\mathrm{NX}}(K)$ curve is distributed. The logarithmic weighting emphasizes local

neighborhood preservation in the measure. See [79] for examples of the adjusted quality curves and the resulting average indicators.

Other measures to judge the overall topology at once are, for example, the *quality of distance mapping* proposed in [44] which calculates the Pearson or Spearman correlation coefficients between all pairwise distances in the high-dimensional versus the low-dimensional setting, yielding a single value. For efficiency, the authors propose to calculate the correlation only on a representative subset of pairwise distances, selected by the *natural PCA* procedure, see [44]. For the topographic mapping with *self-organizing maps* (SOMs) [73], the *topographic product* has been proposed, which yields a single value to assess the topographic disturbances in the map, see [10]. It basically considers the distances between pairs of nearest neighbors, hence it is easily possible to generalize the topographic product to any high- and low-dimensional point configurations (without the fixed lattice of a SOM).

A single quantitative quality rating greatly benefits overall comparison of DR methods as stated in (a), while a fine-grained analysis of a given visualization is not supported.

Supervised evaluation There are also measures which take class labels of the data into account, like e.g. the *quality of group compactness* in [44], the K-nearest-neighbor error of the embedding in [135], or which even introduce a local labeling in the original data space to judge the preservation of local neighborhoods via this labeling [132]. We will not discuss these further, since we focus strictly on an unsupervised evaluation scenario.

General remarks From the existing literature, we see that many quality measures are suitable for a situation where the user wants an overall comparison of a number of different embeddings of the same data, e.g. originating from different DR methods or different parameter settings, as described in statement (a) on page 90. However, there are only few approaches which aim at a more fine-grained analysis of a single visualization, mentioned in (b). Some measures are useful for compromises between (a) and (b), by evaluating the quality over all neighborhood scales, e.g. in [135, 81]. However, only the works [7, 129] aim fully towards the scenario (b), by integrating visual cues about local reliability directly into the embedding. Their idea is to provide the user with sufficient information to compensate for the distortions in the observed visualization, when reasoning about the original data. These approaches are, however, not directly linked to any of the referenced formal quality measures. Therefore, our goal is to extend the formal evaluation based on the well-established *co-ranking matrix* [81] toward a more fine-grained analysis.

After introducing the co-ranking matrix in the next section, we will utilize a decomposition into point-wise quality contributions in Subsection 4.3.3. In Subsection 4.3.4 we will point out certain disadvantages of the quality framework with regard to our purpose of fine-grained analysis and control, and propose to circumvent these disadvantages with a different parameterization.

4.3.2. Evaluating DR based on the co-ranking matrix

Referring to the high-dimensional data set $X = \{x^1, \ldots, x^N\} \subset \mathbb{R}^D$ and the low-dimensional data set $Y = \{\mathbf{y}^1, \ldots, \mathbf{y}^N\} \subset \mathbb{R}^L$, the rank of x^j with respect to x^i in $\mathbb{R}^D$ is given by

$$r_{ij} = |\{k \mid d_{ik} < d_{ij} \text{ or } (d_{ik} = d_{ij} \text{ and } 1 \leq k < j \leq N)\}| \,.$$

Analogously, the rank of $\mathbf{y}^j$ with respect to $\mathbf{y}^i$ in the low-dimensional space is

$$\rho_{ij} = |\{k \mid \delta_{ik} < \delta_{ij} \text{ or } (\delta_{ik} = \delta_{ij} \text{ and } 1 \leq k < j \leq N)\}| \,.$$

The differences $R_{ij} = \rho_{ij} - r_{ij}$ are the *rank errors*. The co-ranking matrix $\mathbf{C}$ [81] can be seen as a histogram of all rank errors, and is defined by

$$\mathbf{C}_{kl} = |\{(i,j) \mid r_{ij} = k \text{ and } \rho_{ij} = l\}|.$$

Pairs of points which change their rank between the original data and its projection are considered errors of the DR procedure. They result in non-zero off-diagonal entries in the co-ranking matrix. A point $\mathbf{y}^j$ with $r_{ij} > \rho_{ij}$ is called an *intrusion*, with $r_{ij} < \rho_{ij}$ it is an *extrusion*. Usually, a DR method cannot embed all relationships of data faithfully. Often, the focus is on the preservation of local relationships. The co-ranking matrix offers a framework, in which several existing evaluation measures can be expressed, as pointed out in [81]: *local continuity meta criterion* (LCMC) [27], *trustworthiness & continuity* (T&C) [134], and *mean relative rank errors* (MRRE) [80]. Essentially, these quality measures correspond to weighted sums of entries $\mathbf{C}_{kl}$ of the co-ranking matrix for regions $k \leq K$ and/or $l \leq K$, with a fixed neighborhood range K.
In [81], a comprehensible (unweighted) sum has been proposed, the *quality* Q_{NX}:

$$Q_{\mathrm{NX}}(K) = \frac{1}{KN} \sum_{k=1}^{K} \sum_{l=1}^{K} \mathbf{C}_{kl} = \frac{1}{KN} \sum_{i=1}^{N} |A_{x^i} \cap B_{\mathbf{y}^i}|. \tag{4.1}$$

where $A_{x^i} = \{j | r_{ij} \leq K\}$ and $B_{\mathbf{y}^i} = \{j | \rho_{ij} \leq K\}$ are the index sets of K nearest neighbors of point x^i in the high-dimensional data, and, respectively, $\mathbf{y}^i$ in the embedding. Hence, this normalized sum is simply the average ratio of K nearest neighbors coinciding in the original and the embedded data. Therefore, it summarizes all 'benevolent' points which maintain a rank below K, which are also called *mild* in- and extrusions. Figure 4.1 shows a schematic picture of how the co-ranking matrix is partitioned via K, and how intrusions and extrusions appear in the matrix.

To display the quality, usually a curve of $Q_{\mathrm{NX}}(K)$ is plotted for a range of different settings of K. An example is given in Figure 4.2 for the classical *swiss roll* data set, which has often been used for illustration purposes in the DR literature, see e.g. [80]. In our case, the original three-dimensional data consists of 1000 points sampled from the curled two-dimensional manifold, see Figure 4.2a. It was reduced to two dimensions

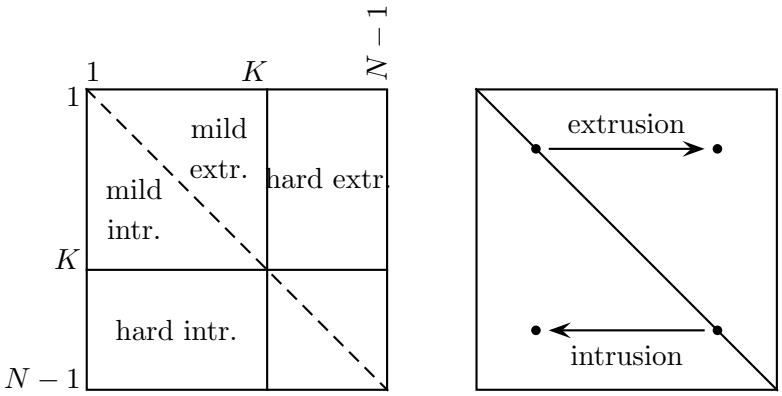


Figure 4.1.: Large-scale structure of the co-ranking matrix. On the left, the matrix is split into blocks to show different types of intrusions and extrusions. In a perfect mapping, the co-ranking matrix will be a diagonal matrix. The image on the right shows how rank differences will alter the matrix. If a neighbor moves further away in the embedding (an extrusion) it will appear to the right of the diagonal. Similarly, intrusions appear to the left of the diagonal.

using t-SNE [131] with the perplexity parameter set to 50, see Figure 4.2c. The 2D embedding produced a piecewise 'unrolled' view of the original spiral strip, where some continuous regions were separated and the data is depicted as three distinct patches[5]. In the embedding, most local neighbors stay in the proximity, corresponding to a quality close to 1, while not all neighbors are preserved in larger neighborhood sizes, see Figure 4.2b. Here, the nonlinear structure of the swiss roll comes into the play: while points on different ends of the swiss roll are relatively close as measured using the Euclidean distances in 3D, these are far away in the 2D unfolding of the spiral strip. The expected quality of a random mapping serves as a baseline for $Q_{\mathrm{NX}}(K)$, see [81, 82] and [38] for a formal derivation. It is displayed as the dotted line in Figure 4.2b. Therefore, with K approaching the total number of points, quality values of 1 are reached slowly, a necessity corresponding to the baseline.

4.3.3. Point-wise quality measure

We argue, that it is important to provide the user with information about the reliability of the displayed embedding, as mentioned in (b). Integrated visual cues indicating the

[5]Note, that we did not use pairwise geodesic distances to represent the original data, despite our knowledge of the underlying manifold. Instead, we calculated Euclidean distances in the original 3-dimensional space to reveal more clearly the effects of the quality evaluation. Moreover, the t-SNE method is generally more suited to embed data which is arranged in clusters, as opposed to data lying continuously on a manifold. We deliberately chose the method for this example to demonstrate the quality evaluation with the typical effects of separating or condensing neighboring points due to the method's inherent assumption of an underlying cluster structure.

reliability of every single mapped point can help the user reason about the original data based on the embedding. Ideally, the user is not only able to identify erroneous regions in the visualized data, but can also get an intuition about the structure of the original data. In a real visualization task, the expert user would have semantical knowledge about the data that might imply certain structural assumptions or expectations. Often, additional information is available, like the membership to semantically meaningful classes in the data. When the expert combines such semantic knowledge with the given visualization, the augmented display might help to distinguish whether the observed local errors are artifacts of the DR procedure, or structures which are in fact contradictory in low-dimensions. While the approaches presented in [7, 129] provide very effective heuristics to do so, surprisingly, none of the formal evaluation measures mentioned so far have been directly integrated into the visualization display. As mentioned, many of the measures explained in Subsection 4.3.1 are aggregated values consisting of point-wise quality contributions (or error rates analogously), and thus yield the possibility to be extended in such a way.

Pointwise co-ranking matrices In the following we will derive the point-wise quality contributions which are aggregated in the measure $Q_{\mathrm{NX}}(K)$. A co-ranking matrix can be seen as the joint histogram of ranks in the high- and low-dimensional data, as stated in [81]. For every single point, it contains the ranks of all its $N-1$ neighbors. Every co-ranking matrix $\mathbf{C}$ can therefore be decomposed into per-point permutation matrices $\mathbf{C}^{\mathbf{y}^i}$ for every point $\mathbf{y}^i \in X$ with $\mathbf{C} = \sum_{i=1}^{N} \mathbf{C}^{\mathbf{y}^i}$ where

$$\mathbf{C}^{\mathbf{y}^i}_{kl} = |\{j \,|\, \rho_{ij} = k \text{ and } r_{ij} = l\}| \,.$$

Hence, the point-wise contributions of the quality Q_{NX} directly follow as

$$Q^{\mathbf{y}^i}_{\mathrm{NX}}(K) = \frac{1}{K} \sum_{k \leq K} \sum_{l \leq K} \mathbf{C}^i_{kl} = \frac{1}{K} \left| A_{x^i} \cap B_{\mathbf{y}^i} \right|$$

which, averaged over all points, again yields the quality measure

$$Q_{\mathrm{NX}}(K) = \frac{1}{N} \sum_{i=1}^{N} Q^{\mathbf{y}^i}_{\mathrm{NX}}(K) \,.$$

Thus, every mapped point can be colored based on its quality $Q^{\mathbf{y}^i}_{\mathrm{NX}}(K)$ for relevant K. The parameter K is either chosen according to relevant structural criteria such as a local extremum of the curve Q_{NX}, or determined interactively according to the user's needs.

Figure 4.2d shows an example for the swiss roll data set, where the points are colored by $Q^{\mathbf{y}^i}_{\mathrm{NX}}(14)$ with $K = 14$ chosen according to the first local optimum of the quality curve. While it indicates small errors for almost all of the points in the inner parts of the patches, it also reveals the positions of stronger topological mismatches on the

borders of the visualized patches. These errors are caused by the unrolling and tearing of the original manifold, and are clearly revealed by the coloring via point-wise quality. Hence this augmentation of the DR according to local quality highlights those regions where the user cannot rely on the visualization.

4.3.4. Parameterization of the quality measure

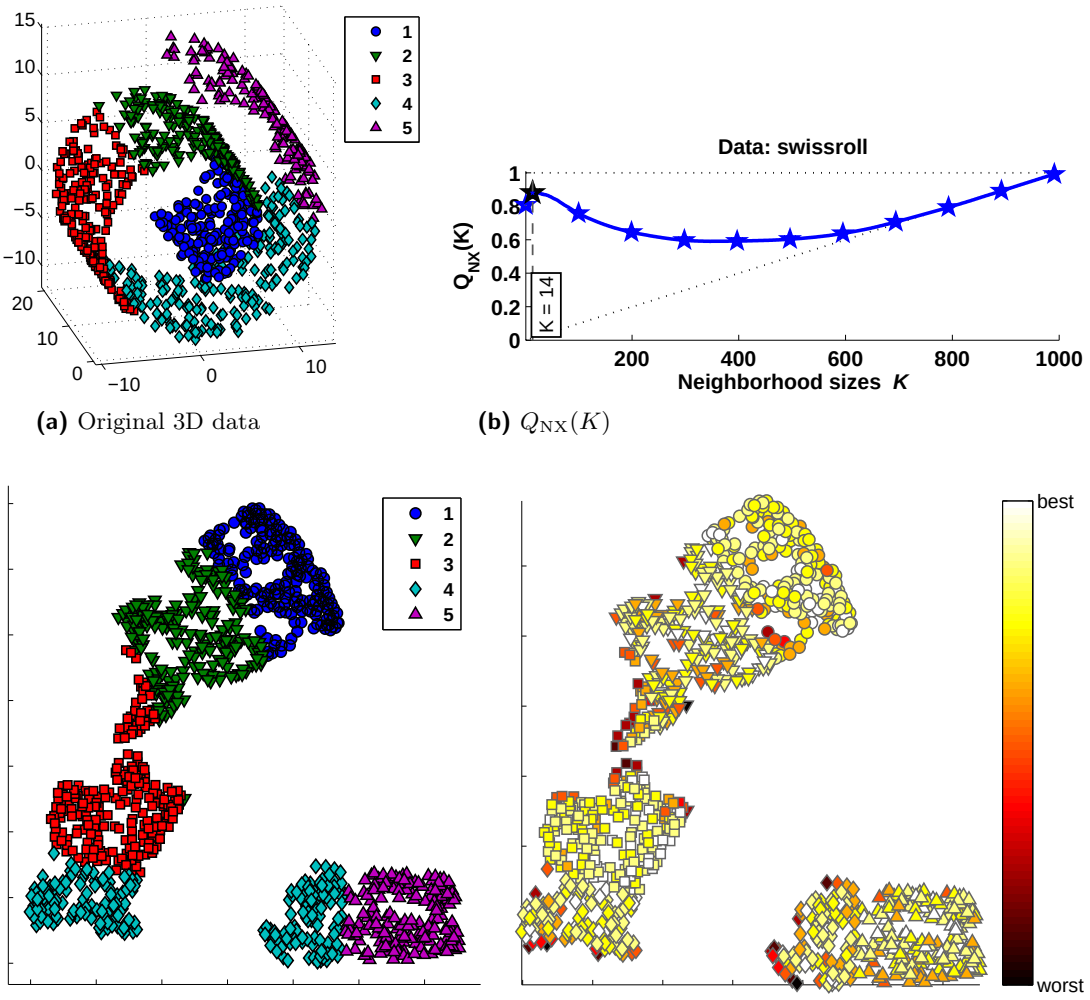
Even in very simple settings, however, it is difficult to interpret the shape of the curve $Q_{\mathrm{NX}}(K)$ and the related local quality measure $Q_{\mathrm{NX}}^{\mathbf{y}^i}(K)$, in particular the parameter K is a fairly simple, but sometimes unintuitive control mechanism. To demonstrate this problem, we will consider a few simple examples of two-dimensional toy data, where we performed no reduction of the dimensionality, but created artificial 'mappings' which map the input data to a different configuration of points in the plane. Although we have direct access to the original data structure as well as the specific characteristics of the mapping in these cases, we found that the types of our deliberately implanted errors are hard to recognize from the results of the quality measure Q_{NX}.

First, we consider a very simple scenario: A row of equidistant points is mapped to a row where the points are swapped in pairs, as depicted in Figure 4.3a. Any even number of points could be chosen arbitrarily. When examining this scenario, we find that the maximum absolute rank error between the original and the switched points is 4 for the entire data set, and independent from the total number of points (for example, when point d moves left, and its right neighbor e moves right)[6]. Intuitively, if we consider rank error sizes up to 4 as acceptable, this mapping is perfect. This is, however, not indicated by $Q_{\mathrm{NX}}(4)$ in the graph in Figure 4.3c (which displays the quality for a row of 20 points which are swapped in this manner): the quality is below one for most $K \geq 4$. It is hardly possible to gain insight about the characteristics of the errors based on the observation of $Q_{\mathrm{NX}}(K)$ and the mapped points alone, although the errors in this scenario can be fully characterized by local pairwise swapping.

The problem arises, because small rank errors can have an effect over larger ranges of K: regarding some reference point $\mathbf{y}^i$, let us consider a faraway neighbor $\mathbf{y}^j$ with the original rank $r_{ij} = K$ For the quality $Q_{\mathrm{NX}}(K)$, this point is considered benign (i.e. it adds to the quality) as long as its rank stays at K or intrudes to some lower rank $1 \leq \rho_{ij} < K$, whereas this neighbor would be regarded as erroneous immediately with just a slightly higher rank of $K + 1$ for instance. On the other hand, a close neighbor, e.g. with rank 1, is allowed to extrude up to a rank of K and still adds to the quality rating, although the rank difference can be rather large. This seems to be an unbalanced characteristic of the quality measure in general.

A look at the co-ranking matrix in Figure 4.3b reveals the distribution of rank changes

[6] Note, that for these equidistant points, ties in the pairwise distances need to be broken to arrive at proper ranks. In case of a tie, we define that the point with the lower alphabetical letter gets assigned the lower rank.



(a) Original 3D data

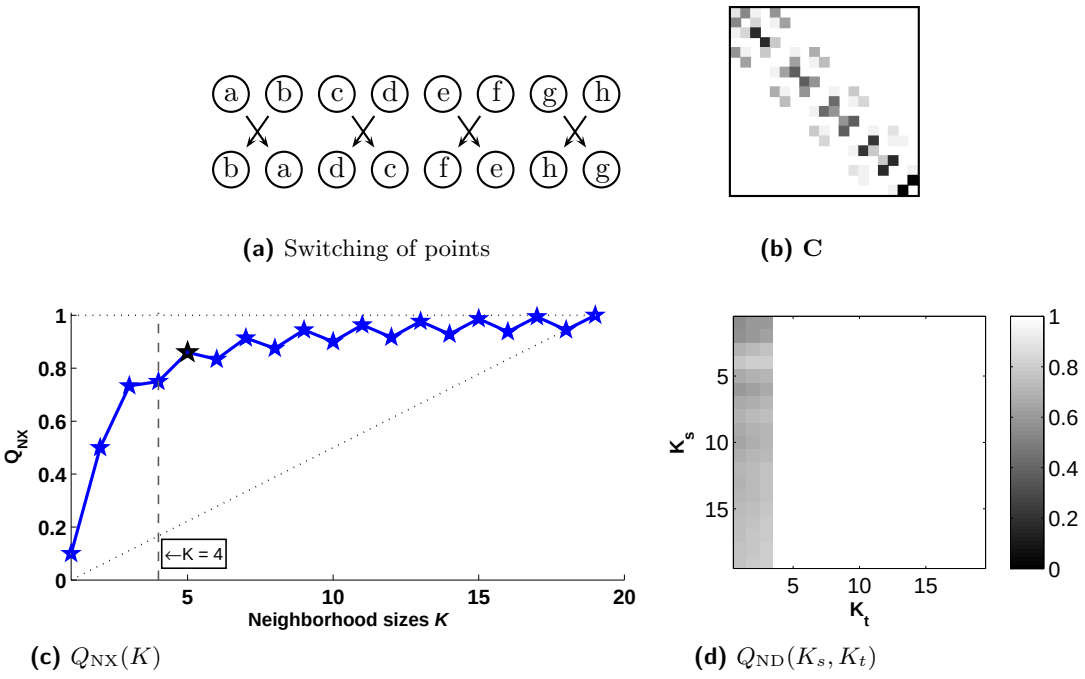
(b) $Q_{\mathrm{NX}}(K)$

(c) 2D t-SNE embedding

(d) Point-wise quality of t-SNE embedding

Figure 4.2.: An example of the qualitative evaluation for an embedding of the well-known artificial *swiss roll* data. On the upper left, the original 3D data is shown, and on the lower left is the 2D embedding obtained by the t-SNE method (with a perplexity of 50). The different symbols serve as a reference to the original positions on the spiral-shaped manifold. The upper right shows the classical evaluation via the quality graph over $Q_{\mathrm{NX}}(K)$. The lower right shows the embedding, colored by the proposed point-wise qualities $Q_{\mathrm{NX}}^{\mathbf{y}_i}(14)$. While the DR method mostly 'unrolled' the original manifold rather truthfully, the strip is torn into several pieces, and the locations of the tears are clearly indicated by the coloring.

for this simple example. Since the rank error is always smaller than 5, only 4 off-diagonals of the co-ranking matrix are not equal to 0, since the ith off-diagonal corresponds to rank errors of size i. However, the quality Q_{NX} is a sum over a square block of the co-ranking matrix, like many other DR evaluation measures described in [81]. This observation also suggests how the quality measure can be altered to achieve a more appropriate parameterization: rather than considering a rectangular sub-matrix, it should focus on a limited number of off-diagonals corresponding to the rank deviation that is considered acceptable.

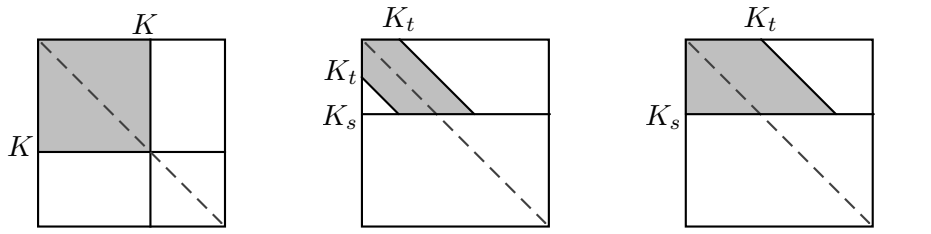


(a) Switching of points **(b)** C

(c) $Q_{\mathrm{NX}}(K)$ **(d)** $Q_{\mathrm{ND}}(K_s, K_t)$

Figure 4.3.: The upper left shows the artificial mapping, which is a simple switching scheme of a row of one-dimensional points. Obviously, rank errors are at most four (in case of tie breaks) in this setting. This is mirrored by the shape of a co-ranking matrix for the same setting with 20 points (upper right) for which four off-diagonals are non-vanishing. However, the established measure $Q_{\mathrm{NX}}(K)$ is below 1 for almost all K (on the bottom left), which is hard to link back to the mapping's characteristics. For our proposed measure (on the bottom right), $Q_{\mathrm{ND}}(K_s, K_t) = 1$ for all $K \geq 4$.

Looking at the rather comprehensible and straightforward definition of Q_{NX}, we find that the parameter K serves two different purposes: on the one hand, K identifies a region of interest by determining the size of the neighborhood of every point in the original data, namely $r_{ij} \leq K$. On the other hand, it determines the size and shape of errors which are tolerated for points in the region of interest: every $\rho_{ij} \leq K$ is acceptable and adds to the overall quality. This parameterization has the effect that small rank errors can contribute to the shape of the curve $Q_{\mathrm{NX}}(K)$ on every scale of



(a) original $Q_{\mathrm{NX}}(K)$ **(b)** $Q_{\mathrm{ND}}(K_s, K_t)$, K_t limits w.r.t. diagonal **(c)** $Q_{\mathrm{ND}}(K_s, K_t)$ for $K_s = K_t$

Figure 4.4.: Change of the summation area of the co-ranking matrix for precise control over the region of interest and the tolerated rank errors.

K. While this is no immediate drawback when merely comparing several DR methods, i.e. the usage scenario described in (a) on page 90, the effect can be problematic when a fine-grained analysis of a visualization is desired, like in case (b) on page 91. As stated in Subsection 4.3.1, the quality measure Q_{NX} is similar (or equal) to several other evaluation criteria which rely on the same part $\mathbf{C}_{kl}, k, l \leq K$ of the co-ranking matrix. Hence, this problem is present in all these evaluation measures.

To circumvent the described problem, we propose a different, more fine-grained assessment of quality based on the co-ranking matrix, which (i) identifies benign points by their amount of deviation from the original rank, rather than their absolute rank in the embedding, and (ii) allows for separate control over the region of interest and the size of the tolerated errors. We therefore replace the single parameter K by the pair (K_s, K_t), where K_s determines the region of interest (alias the significant ranks) and K_t is the size of tolerated rank errors. Further, rather than tolerating errors within a certain region of the projection, we explicitly consider a limit on the absolute rank errors. The new measure is defined as:

$$Q_{\mathrm{ND}}(K_s, K_t) = \frac{1}{K_s N} \sum_{i \leq K_s} \sum_{j:|i-j| \leq K_t} \mathbf{C}_{ij} \ .$$

Since the second sum is limited to entries $j : |i - j| \leq K_t$, i.e. rank errors $|i - j|$ smaller than K_t, we now sum over a part of the co-ranking matrix which is oriented according to the diagonal, see the schematic in Figure 4.4b. By controlling the two parameters of the quality measure, the user can assess the compliance with specific requirements for the embedding. For example, common tasks would be to assess:

(I) the preservation of local relationships (chosen by small K_s and small K_t);

(II) the amount of errors originating in fairly local neighborhoods, but are deviating largely from the original rank (small K_s, large K_t);

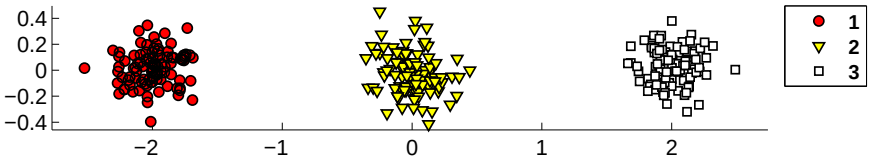
(III) the preservation of global relationships in the data (large K_s, smaller K_t).

To get a rich impression of a visualization's qualitative characteristics, the quality $Q_{\mathrm{ND}}(K_s, K_t)$ is now parameterized by two values. Hence, rather than in a single curve, the results are now represented by a surface. The full quality surface can easily be displayed as a colored matrix, where the position (K_s, K_t) is assigned a color value according to $Q_{\mathrm{ND}}(K_s, K_t)$, see Figure 4.3d for an example. The matrix in Figure 4.3d shows the results for our example of 20 swapped points. It clearly reveals that all entries for $K_t \geq 4$ yield the maximum quality, which is the expected behavior.

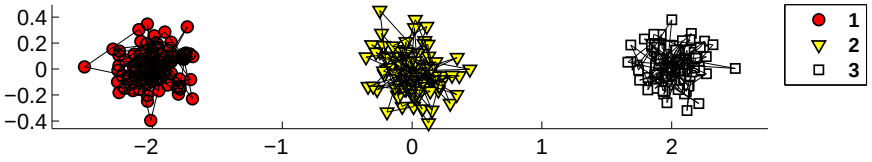
In the following artificial example, we will further demonstrate the more directly controllable characteristics of our approach. We consider three simple scenarios, mapped from two-dimensional points to a new point distribution in 2D. The original data consist of three well-separated Gaussian clusters, containing 100 points each, see Figure 4.5a. As a 'mapping', we consider the points obtained by (i) a random permutation of the points within every cluster, see Figure 4.5b, (ii) a switch of the two leftmost clusters, see Figure 4.5c, and (iii) the middle and leftmost cluster stacked on top of each other, see Figure 4.5d. These artificial mappings represent typical behavior of DR embeddings since they capture (i) local distortions, (ii) a tearing of regions, and (iii) an overlay of regions, which are common effects due to the low dimensionality of the projection space.

The resulting curves for Q_{NX} are depicted in Figure 4.6a. Although we know the exact behavior of the mapping in this case, it is not easy to link the entire shape of the curves to the characteristics of the respective mapping. In setting (i), the random permutation of points within the clusters causes a vast number of local errors, which is clearly indicated by the low quality for $K < 100$. Farther neighbors change their rank as well, because of the permutation within the neighboring clusters. However, the absolute size of all rank errors in the mapping is strictly below 100, when considering only a single cluster, which cannot be inferred on the basis of the quality curve. The quality matrix for the new measure Q_{ND} in Figure 4.6b clearly shows the errors which are present rather steadily over all scales of K_s, whereas the quality is perfect for all pairs $(K_t > 100, K_s < 100)$, which implies that the absolute size of rank errors caused by the mapping for a single cluster is below this range. When considering large neighborhoods of interest with $K_s > 100$, the quality is very good for $100 < K_t < 150$, and perfect for all $K_t > 150$. The type of errors that appear here are more rare, the extreme case would be, that a point on the very right of a cluster moves to the very left of its cluster, and a neighbor originally on the left, moves to the very right of its cluster. The absolute rank error for this type cannot exceed half of the total number of points, as indicated by $Q_{\mathrm{ND}}(K_s, K_t) = 1$ for all $K_t > 150$. This mapping refers to the evaluation tasks (I) and (III) as described on page 100, i.e. the upper left part of the quality surface for (I), and the lower left/middle part for (III).

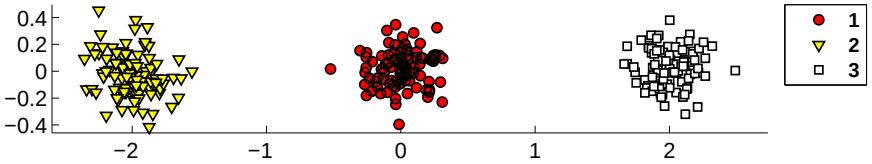
For mapping (ii), the curve of Q_{NX} in Figure 4.6a reveals that there are no errors on a small neighborhood scale (below the cluster size of 100), whereas the quality drops severely beyond this scale. The corresponding matrix of Q_{ND} in Figure 4.6c gives us the



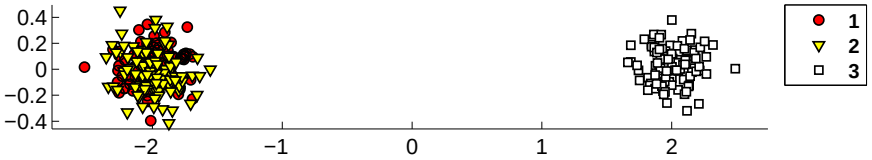
(a) original clusters



(b) mapping (i): shuffled within clusters



(c) mapping (ii): left two clusters switched



(d) mapping (iii): left two clusters merged

Figure 4.5.: These scatter plots show a simple example of deliberately designed artificial mappings, which resemble typical effects of DR procedures and serve as a demonstration benchmark for our quality evaluation. The upper plot depicts the original data consisting of 3 Gaussian clusters in 2D. The plot below shows how the data was randomly shuffled within each cluster, where black lines are drawn from every point to its original position, to demonstrate the permutation. The last two plots show, respectively, how two clusters are switched, and stacked on top of each other.

same information, but also reveals that the absolute size of rank errors is below 200 by showing a perfect quality for all $K \geq 200$. This is expected, since there are only two clusters involved in errors. We also see a sharp rise in quality at $K_t = 100$, because for the points of the rightmost cluster, two thirds of all neighbors (the points of the other clusters) change their ranks by exactly 100 due to the switching. From the perspective of the leftmost cluster, there are also some rank errors of size 100 to 200, which is indicated by the slight coloring in the region $100 \leq K_t < 200$. Mapping (ii) refers to the evaluation task (III) on page 100, i.e. the lower left to middle part of the surface.

In the evaluation for mapping (iii), the curve of Q_{NX} shows a steadily diminished quality until $K \approx 125$, a scale which cannot be linked to the structural knowledge about the data. Thereafter, the curve steadily rises to the maximum. From this, we can gather that there are relatively little global errors, however, the matrix for Q_{ND} in Figure 4.6d gives more insight: we can see that the errors originating in small regions (small K_s) are rather small, i.e. there are errors only for $K_t < K_s$ approximately. On larger scales, the number of errors increases along with the tolerance K_t, which implies that the absolute size of rank errors increases. This is expected, since the stacking of the clusters causes small deviations from the original ranks when considering small neighborhoods, as well as large errors when considering large neighborhoods. However, the quality is perfect for $K_t > 200$ which, again, suggests that there are only two clusters involved in the occurring errors. This mapping is linked to the evaluation tasks (I) and (II).

In order to reduce the computational cost of calculating $Q_{\mathrm{ND}}(K_s, K_t)$ for all pairs $(K_s, K_t) \in \{1, \ldots, N-1\}^2$ in a practical evaluation scenario, it is reasonable to calculate only the quality values for the following three curves instead of the full surface:

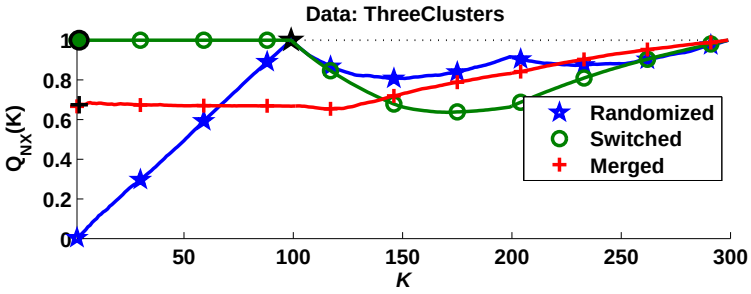
- the values $Q_{\mathrm{ND}}(K_s, K_t)$ with $K_s = K_t \in \{1, \ldots, N-1\}$, which resembles the original curve from $Q_{\mathrm{NX}}(K)$ over growing neighborhood sizes, but with a different area of summation, taking the error size into account; the corresponding part of the co-ranking matrix is depicted in the schematic in Figure 4.4c. This means that the size of tolerated errors is growing as the considered region of interest gets larger. We then denote the measure by $Q_{\mathrm{ND}}(K_{st})$ with $K_{st} := K_s = K_t$.

- $Q_{\mathrm{ND}}(K_s, K_t)$ for all $K_s \in \{1, \ldots, N-1\}$ and fixed K_t, i.e. the mapping's characteristics under a fixed assumption about the failure tolerance, as only the region of interest grows.

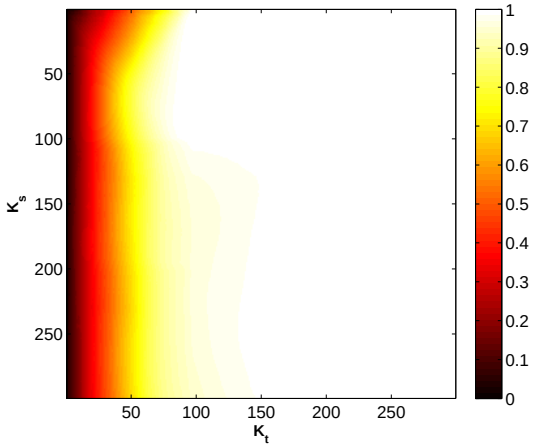
- $Q_{\mathrm{ND}}(K_s, K_t)$ for all $K_t \in \{1, \ldots, N-1\}$ with a constant K_s, i.e. for a fixed neighborhood size of interest, as the failure tolerance increases.

In the latter two cases, the respective fixed parameters can be selected according to the user's prospect, e.g. on which scale the visualization is required to be trustworthy. Figure 4.7 shows how the combination of these curves for Q_{ND} offers an adequate approximation of the full surfaces from Figure 4.6.

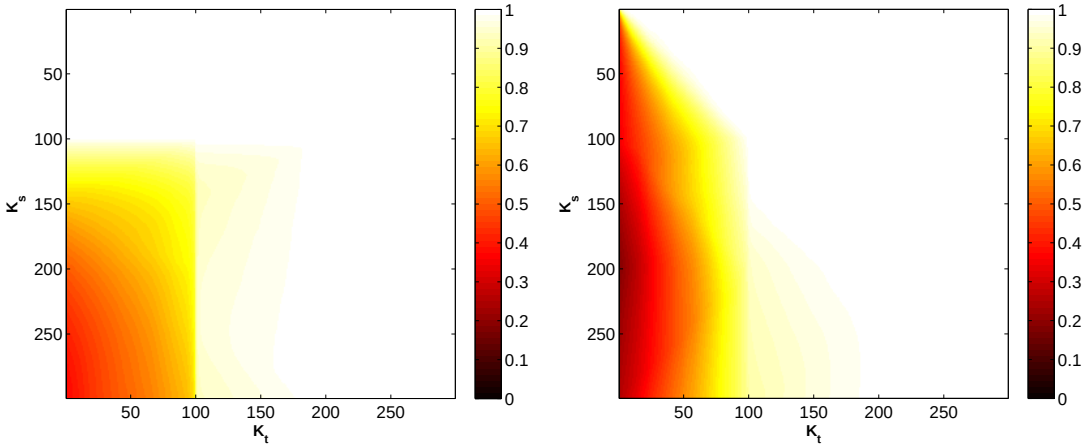
The evaluation in these artificial cases is simplified by the assumption of equal cluster sizes, which yield a natural threshold for the parameters. While this was helpful to clarify the proposed parameterization, the benefits of the two parameters become apparent when the aggregated overview of the mapping quality is combined with a point-wise evaluation, which is introduced in the following section.



(a) $Q_{\mathrm{NX}}(K)$



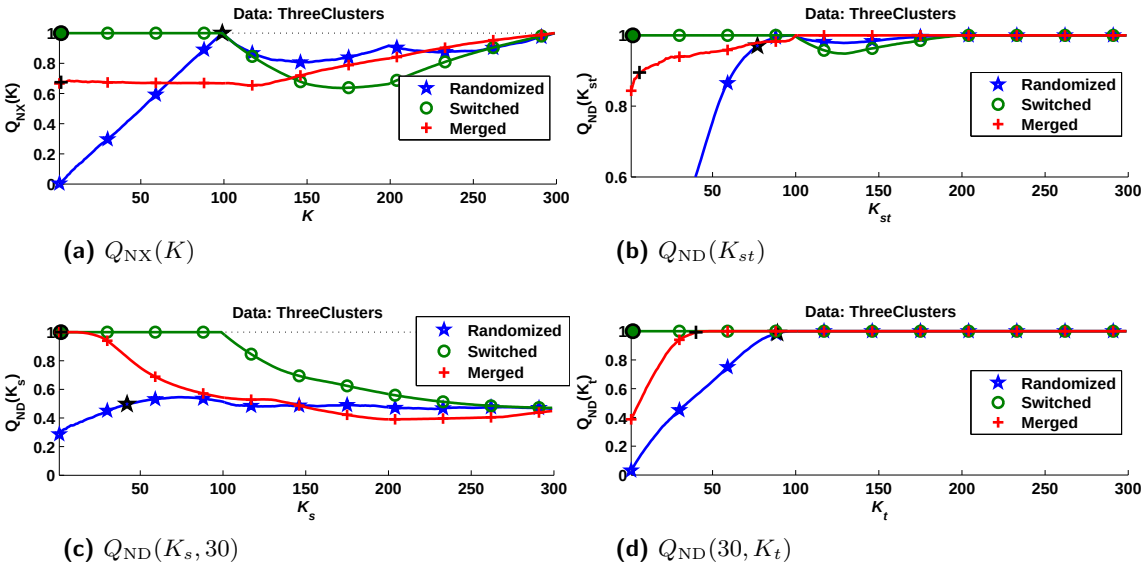
(b) $Q_{\mathrm{ND}}(K_s, K_t)$, shuffled within clusters



(c) $Q_{\mathrm{ND}}(K_s, K_t)$, left two clusters switched

(d) $Q_{\mathrm{ND}}(K_s, K_t)$, left two clusters merged

Figure 4.6.: The figures show the evaluation for the artificial mappings from Fig. 4.5, first with curves obtained by the classical quality measure Q_{NX} (the top figure), and with the quality surfaces resulting from the newly proposed Q_{ND} measure (in the three remaining figures), each being a counterpart to one of the curves above.



(a) $Q_{\mathrm{NX}}(K)$ **(b)** $Q_{\mathrm{ND}}(K_{st})$

(c) $Q_{\mathrm{ND}}(K_s, 30)$ **(d)** $Q_{\mathrm{ND}}(30, K_t)$

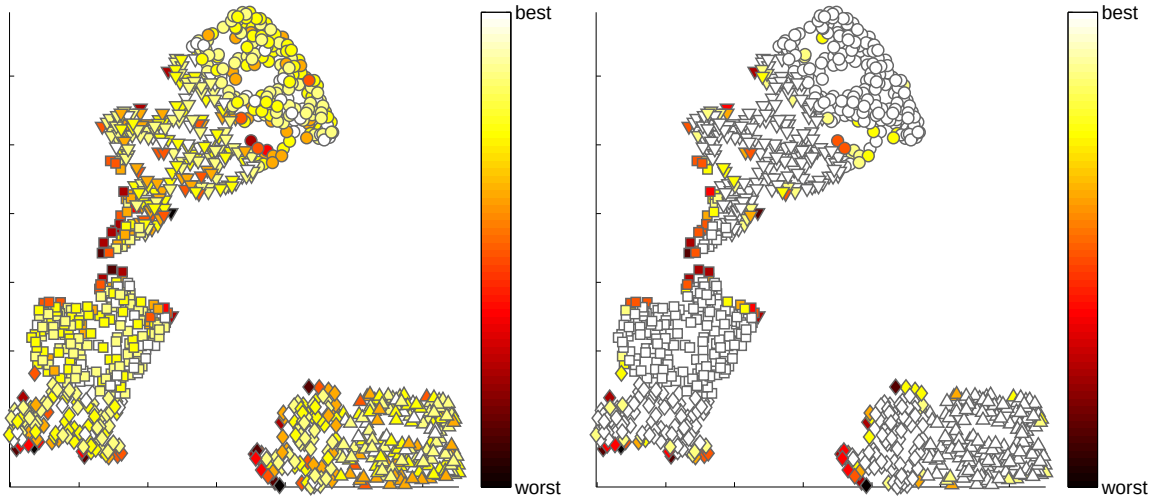
Figure 4.7.: This figure shows a different representation of the qualitative evaluation presented in Fig. 4.6 for the artificial mappings of three clusters shown in Fig. 4.5. In the upper left are the curves for $Q_{\mathrm{NX}}(K)$ (the same as in Fig. 4.6a). The other three plots show an alternative, lean representation of the Q_{ND} measure, which needs less computational effort. The upper right shows $Q_{\mathrm{ND}}(K_{st})$ for all $K_{st} := K_s = K_t \in \{1, \dots, N-1\}$, meaning that the measure tolerates all absolute rank errors which are smaller than the current neighborhood of interest. (Here, the y-axis is scaled differently to highlight the details.) The graphs in the bottom left capture the mapping's characteristics under a fixed assumption about the failure tolerance, as only the region of interest grows, in this case for an error tolerance of 30. The bottom right shows the graphs for $Q_{\mathrm{ND}}(30, K_t)$ for all $K_t \in \{1, \dots, N-1\}$, i.e. the average quality of the 30 nearest neighbors, as the failure tolerance increases. These graphs capture the essential content of the surfaces shown in Fig. 4.6, requiring far less computational effort.

Controllable point-wise quality For the new measure Q_{ND}, the definition of the point-wise quality is analogous to the one from Subsection 4.3.3:

$$Q^{\mathbf{y}^i}_{\mathrm{ND}}(K_s, K_t) = \frac{1}{K_s} \sum_{k \le K_s} \sum_{l : |k-l| \le K_t} \mathbf{C}^i_{kl}$$

where $Q_{\mathrm{ND}}(K_s, K_t) = \sum_{i=1}^{N} Q^{\mathbf{y}^i}_{\mathrm{ND}}(K_s, K_t)/N$. Here, the benefits of the new parameterization are particularly noticeable, since the user is able to tune the parameters to make specific types of embedding errors directly visible. We consider the example from Figure 4.2, and show how the previous point-wise quality compares to the new definition in Figure 4.8. The problem of Q_{NX} described above becomes apparent when looking at $Q^{\mathbf{y}^i}_{\mathrm{NX}}$: at a scale K, the measure considers very different types of errors at once. In this case, we see small rank errors caused by fairly local permutations of points within the unfolded pieces of the strip, which exhibit a lighter color. Also, we observe a small

number of strongly colored points on the edges where tearing occurred and lead to larger rank errors. In contrast, the new measure $Q_{\mathrm{ND}}^{\mathbf{y}^i}$ exclusively singles out the tearing, since in this case the small rank errors within the unfolded patches are below the tolerance threshold of $K_t = 14$, while larger errors which originate in small regions of 14 nearest neighbors (K_s) are caused by the tearing only, and diminish the quality in this parameter configuration.



(a) $Q_{\mathrm{NX}}^{\mathbf{y}^i}(14)$ **(b)** $Q_{\mathrm{ND}}^{\mathbf{y}^i}(14, 14)$

Figure 4.8.: This figure compares the two proposed point-wise quality measures for the same embedding of the *swiss roll* data by t-SNE, as introduced in Fig. 4.2. On the left, we show the points colored according to $Q_{\mathrm{NX}}^{\mathbf{y}^i}(14)$ (the same as in Fig. 4.2d). On the right, the points' color coding is obtained by $Q_{\mathrm{ND}}^{\mathbf{y}^i}(14, 14)$. Both measures highlight the tearing of the original manifold, but $Q_{\mathrm{ND}}^{\mathbf{y}^i}$ shows only the torn regions and almost no local errors within the unrolled patches, since absolute rank errors below 14 are explicitly tolerated. (The sequence of class labels from the inside to the outside of the original spiral-shaped manifold is: ∘▽□◊△, see Fig. 4.2a.)

4.3.5. Experiments with real-world data

In this section, we demonstrate the quality evaluation framework on two real-world data sets, and showcase the augmented visualization along with the classical evaluation by the quality curve Q_{NX}. For the dimensionality reduction of the data, we applied the standard linear technique PCA projecting the data on the first two principal components, as well as the well-known modern nonlinear method t-SNE.

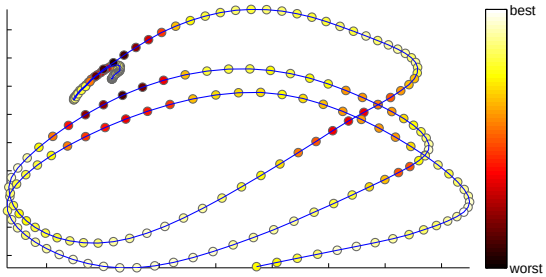
Runner data The first data set is a motion capture sequence, freely available from the *Open Motion Data Project* at the Ohio State University[7]. It contains the three-

[7] sequence *Figure Run 1* from `http://accad.osu.edu/research/mocap/mocap_data.htm`

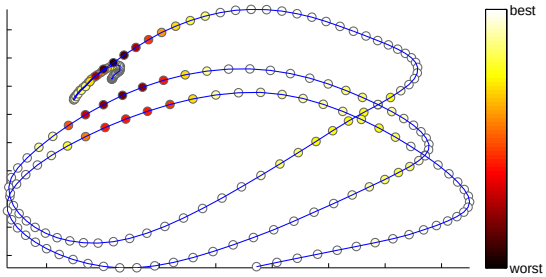
dimensional positions of 34 tracking markers over 217 time steps. The sequence shows a person, who begins to run from a forward-leaning position, and takes about five strides during which the inclination of the body becomes upright. Figure 4.9 shows 2D-embeddings of the original 102-dimensional points. To clarify the sequential relation of the visualized data, we connected points from consecutive frames by a line. We deliberately chose this data set, because here the user has additional knowledge about the original underlying manifold, and can directly inspect where a DR technique did not represent the manifold truthfully. Therefore, the visual augmentation to detect tearing and overlapping of the manifold is superfluous. However, since data lying on an (unknown) underlying manifold structure are common in general practical applications, this showcases the insights we can gain from the point-wise quality evaluation.

The embeddings of both, PCA and t-SNE, show a similar shape of a 'tail' which leads into a spiral structure. This can be explained by the sequence starting from a leaning posture (the tail) and progressing to several strides of upright running (the spiral). In case of PCA, the sequence of points is overlapping at several positions, while the t-SNE method splits some of the consecutive points apart but shows less overlap in general. The t-SNE embedding also produced some crowds and zig-zag shapes along the point sequence. We report the corresponding quality curves in Figure 4.10.

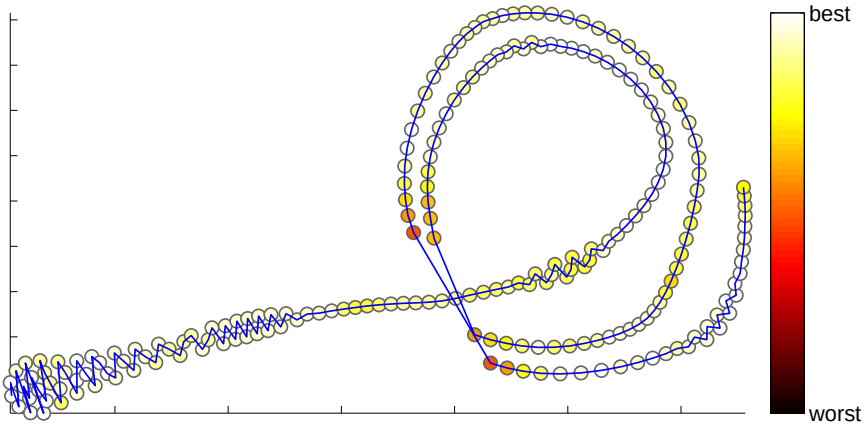
In case of the PCA, both measures $Q_{\mathrm{NX}}^{\mathbf{y}^i}$ and $Q_{\mathrm{ND}}^{\mathbf{y}^i}$ show, that some of the overlapping regions have a reduced quality. However, the measure Q_{ND} identifies less regions to be severely erroneous, due to the tolerance of $K_t = 20$, compare, for example, the overlap on the left of the scatter plots in Figures 4.9a and 4.9b. This indicates that the rank error for these points must be below 20, while the highlighted regions contain larger errors. Depending on the practical purpose, the user may want to be aware of the severe mismatches, neglecting tolerable errors. Similar characteristics can be observed in the Figures 4.9c and 4.9d for t-SNE. The tearing of the sequence is distinctly highlighted as erroneous in the coloring by $Q_{\mathrm{ND}}^{\mathbf{y}^i}$. Here, the advantage of the parameterization becomes particularly apparent: the user may choose via K_t not to highlight the tolerable local errors which are caused by the crowding and zig-zag patterns.



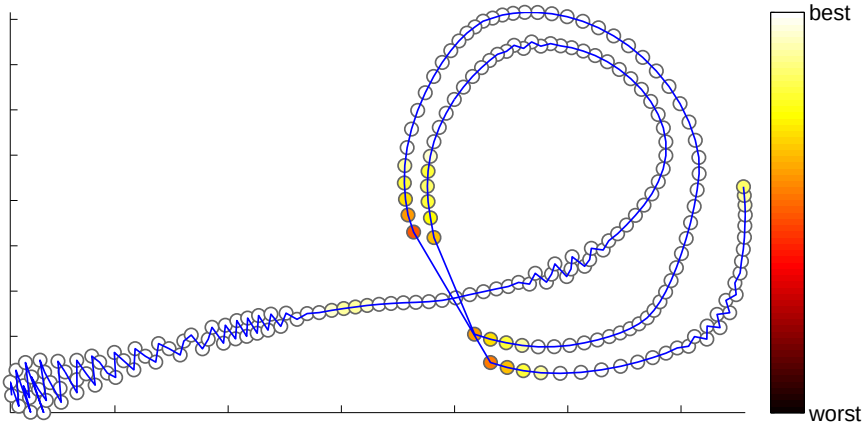
(a) PCA, $Q_{\mathrm{NX}}^{\mathbf{y}^i}(20)$



(b) PCA, $Q_{\mathrm{ND}}^{\mathbf{y}^i}(20,20)$

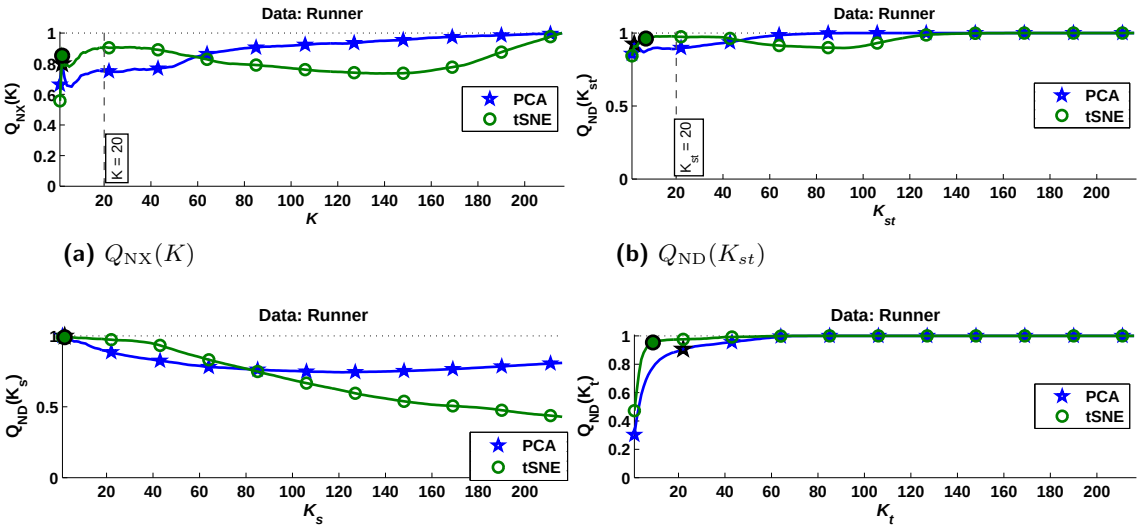


(c) t-SNE, $Q_{\mathrm{NX}}^{\mathbf{y}^i}(20)$



(d) t-SNE, $Q_{\mathrm{ND}}^{\mathbf{y}^i}(20,20)$

Figure 4.9.: The figure shows a PCA and a t-SNE embedding (with perplexity 30) of the *runner* data, each colored by the pointwise quality $Q_{\mathrm{NX}}^{\mathbf{y}^i}(20)$ and $Q_{\mathrm{ND}}^{\mathbf{y}^i}(20,20)$, respectively. Points from consecutive motion capture frames are connected by a line. For $Q_{\mathrm{NX}}^{\mathbf{y}^i}(20)$, various types of errors arising in neighborhoods of 20 points are highlighted all at once, while $Q_{\mathrm{ND}}^{\mathbf{y}^i}(20,20)$ is able to identify where the neighbors deviate from their original rank by more than 20. This gives a clearer indication of where the underlying manifold is not truthfully represented, e.g. torn apart in case of t-SNE.



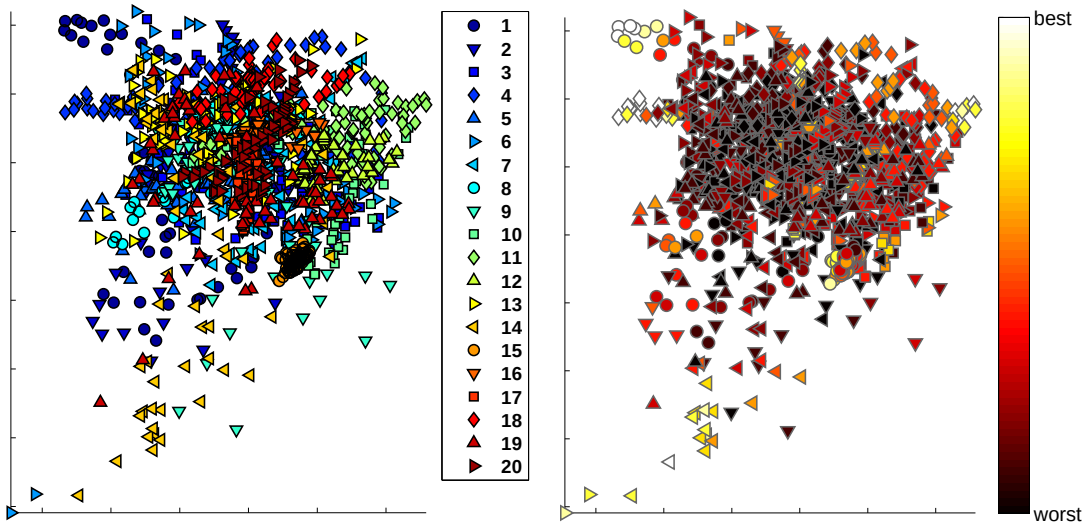
(a) $Q_{\mathrm{NX}}(K)$ **(b)** $Q_{\mathrm{ND}}(K_{st})$

(c) $Q_{\mathrm{ND}}(K_s, 20)$ **(d)** $Q_{\mathrm{ND}}(20, K_t)$

Figure 4.10.: The quality curves of Q_{NX} (a) and Q_{ND} (b,c,d) for the PCA and the t-SNE embedding of the *runner* data, see Fig. 4.9. The curves show that the t-SNE embedding is more truthful in displaying the local relationships, whereas PCA preserves more of the ranks in the larger neighborhoods. The curves for $Q_{\mathrm{ND}}(K_s, 20)$ in the lower left (for a fixed error tolerance of 20) reveal that there are many errors in the t-SNE embedding when considering larger neighborhoods. On the other hand, the curves over $Q_{\mathrm{ND}}(20, K_t)$ in the lower right show that the errors in the t-SNE embedding are only of very small magnitude when considering 20 nearest neighbors.

COIL-20 data The second data set, the *Columbia University Image Library* from [102], consists of 1440 gray-value images in a resolution of 128×128, which show 20 small objects, photographed from 72 consecutive rotation angles. Each class in the data corresponds to photos of one particular object. Because of the consecutive angles, we can assume that the original data is clustered by their class membership, and that within a class, the data are situated on a ring-shaped manifold.

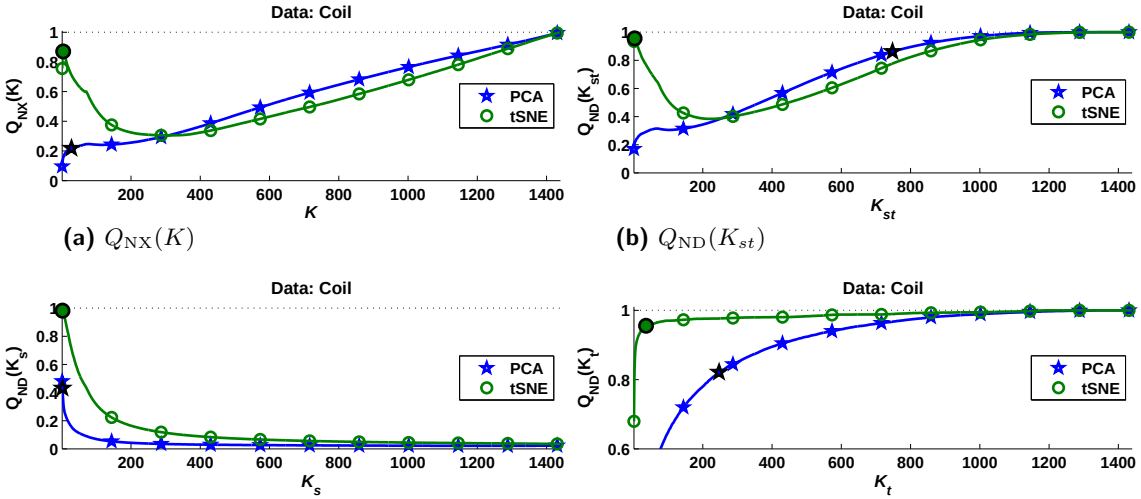
In Figure 4.11 and 4.13, we show an embedding by PCA and t-SNE, respectively, with points colored by their quality $Q_{\mathrm{ND}}^{\mathbf{y}^i}(10, 10)$. Since we chose $K_s = K_t = 10$, the measure highlights absolute rank errors larger than 10, originally situated among the 10 nearest neighbors of a point. Figure 4.12 shows the corresponding quality curves. The PCA embedding is generally of a low quality, as indicated by the coloring in Figure 4.11b, and the curves in Figure 4.12. For the t-SNE embedding, the coloring in Figure 4.13b reveals distinct defects in the clusters, seemingly either caused by the tearing or contracting of the original manifold within a cluster, or by overlaying separated clusters.



(a) PCA embedding

(b) $Q_{\mathrm{ND}}^{\mathbf{y}^i}(10,10)$ for PCA embedding

Figure 4.11.: On the left, we show a PCA embedding of the *COIL-20* data where each point's class membership is represented by a distinct combination of a marker symbol and color. On the right, points are colored by their quality $Q_{\mathrm{ND}}^{\mathbf{y}^i}(10,10)$. The embedding exhibits a low quality at almost every position, since there are many rank errors > 10 occurring in the 10 nearest neighbors.



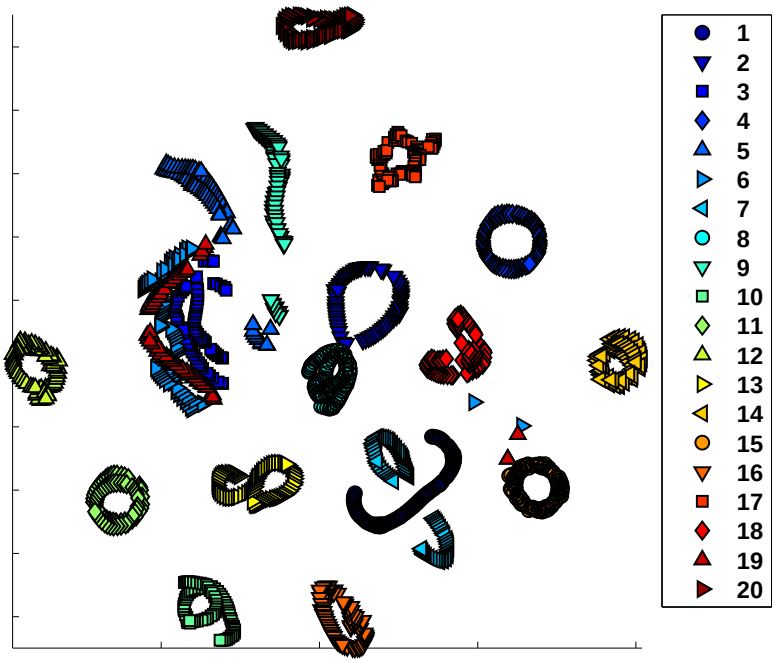
(a) $Q_{\mathrm{NX}}(K)$

(b) $Q_{\mathrm{ND}}(K_{st})$

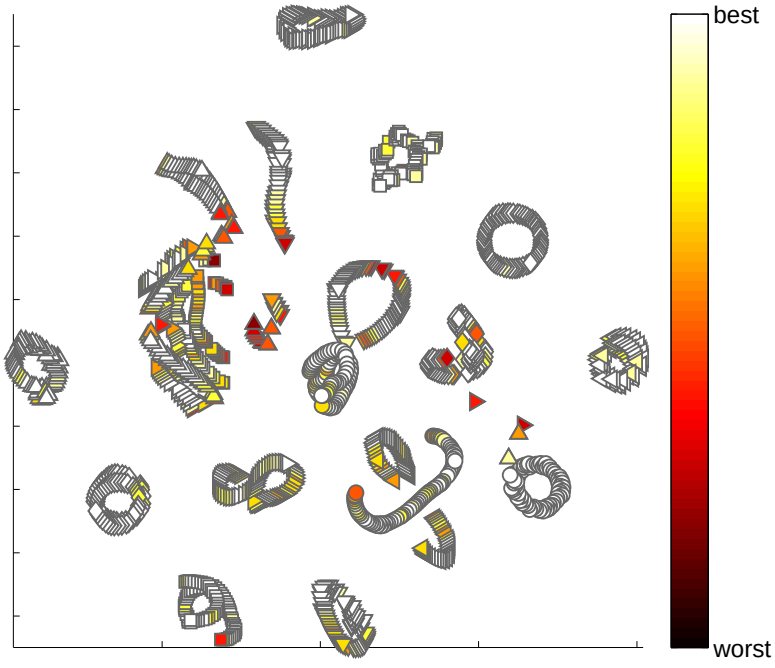
(c) $Q_{\mathrm{ND}}(K_s, 10)$

(d) $Q_{\mathrm{ND}}(10, K_t)$

Figure 4.12.: This figure shows quality evaluations for the PCA and t-SNE embeddings of the *COIL-20* data (see Fig. 4.11a,4.13a). The curves in (a) are the quality Q_{NX}, while (b,c,d) show the quality curves resulting from the Q_{ND} measure. Both, Q_{NX} and Q_{ND} clearly identify that the PCA embedding fails to reliably represent the local relationships, while the t-SNE embedding sacrifices some of the global relationships but is generally depicting the smaller neighborhoods rather truthfully.

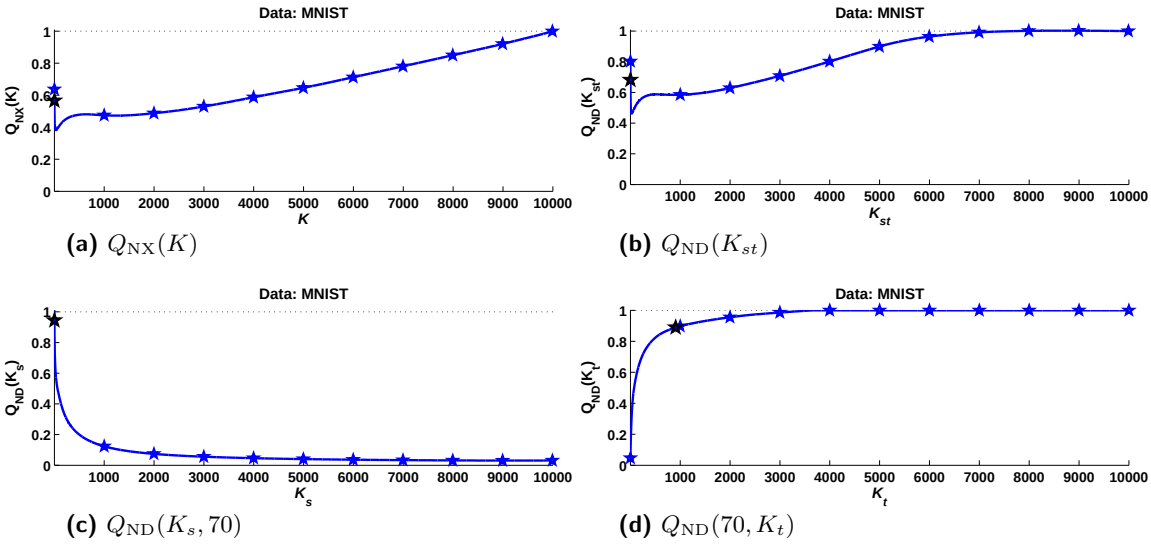


(a) t-SNE embedding



(b) $Q_{\mathrm{ND}}^{\mathbf{y}^i}(10,10)$ for t-SNE embedding

Figure 4.13.: The figure shows a t-SNE embedding of the *COIL-20* data, the perplexity parameter was set to 15. In the upper visualization, points are marked according to their classes by combinations of marker shape and color. The points in the lower picture are colored by their quality $Q_{\mathrm{ND}}^{\mathbf{y}^i}(10,10)$. Since the region of interest K_s as well as the failure tolerance K_t were both set to 10, we can see where some of the clusters, which are assumed to be on a ring-shaped manifold originally, have been torn or contracted severely.



(a) $Q_{\mathrm{NX}}(K)$ **(b)** $Q_{\mathrm{ND}}(K_{st})$

(c) $Q_{\mathrm{ND}}(K_s, 70)$ **(d)** $Q_{\mathrm{ND}}(70, K_t)$

Figure 4.14.: This figure shows the qualitative evaluations for the t-SNE embedding of the *MNIST* data (see Fig. 4.15). The curves in (a) are the quality Q_{NX}, while (b,c,d) show the quality curves resulting from the Q_{ND} measure. The curves indicate that the distortions in the mapping are generally quite large. Even in a range of 70 neighbors, there are many errors, and the rank errors have a size of up to 3000, see the curve for $Q_{\mathrm{ND}}(70, K_t)$. Only the very small neighborhood ranges are depicted rather truthfully, as seen on the very left of the curves for $Q_{\mathrm{NX}}(K)$ and $Q_{\mathrm{ND}}(K_{st})$.

MNIST data The third data set *MNIST* from [77] consists of 60,000 gray-value images of handwritten digits[8] from 0 to 9. Each image comes at a resolution of 28×28 and is therefore represented as a vector of 784 dimensions. Applying t-SNE on the full data set of 60,000 images is not feasible in terms of memory demand and computational effort. We therefore used a random sample of 10,000 points for our experiments. For this data set, we have no prior assumption about an underlying manifold structure, but we can assume that there are clusters according to the ten digits.

Figure 4.15 shows a visualization with t-SNE. We now omitted the corresponding PCA embedding since it shows a considerably inferior quality, similar to the case of the *COIL-20* data. In Figure 4.16, the embedding is colored by the point-wise quality $Q_{\mathrm{ND}}^{y^i}(300, 300)$, and Figure 4.14 shows the quality curves. The embedding shows the expected cluster structure according to the digits, however, the classes are only weakly separated and they are partially overlapping. Although the overall quality is diminished by these effects, we can see in the point-wise evaluation that the errors are less pronounced for the digits 0, 1, 6, and 7. The other digits show a lower quality, especially in the border regions, presumably caused by the stronger overlaps.

[8] For further information, see `http://yann.lecun.com/exdb/mnist/`

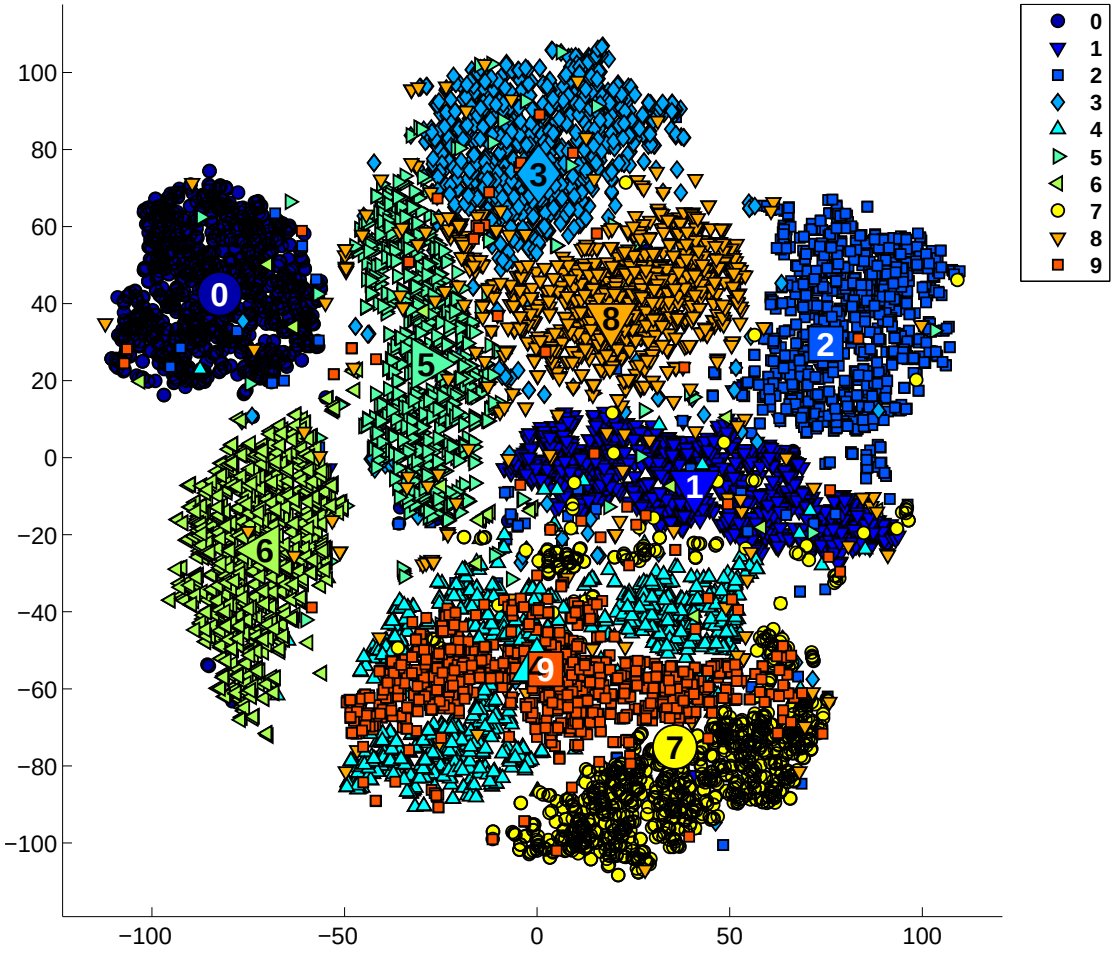


Figure 4.15.: This figure shows a t-SNE embedding of the *MNIST* data consisting of 10,000 data points, where the perplexity parameter was set to 30. Each point's class membership is represented by a distinct combination of a marker symbol and color. Additionally we highlighted the corresponding digit in each cluster center. We see that the data are arranged in clusters according to the classes, but are generally close together with some significant overlaps between classes.

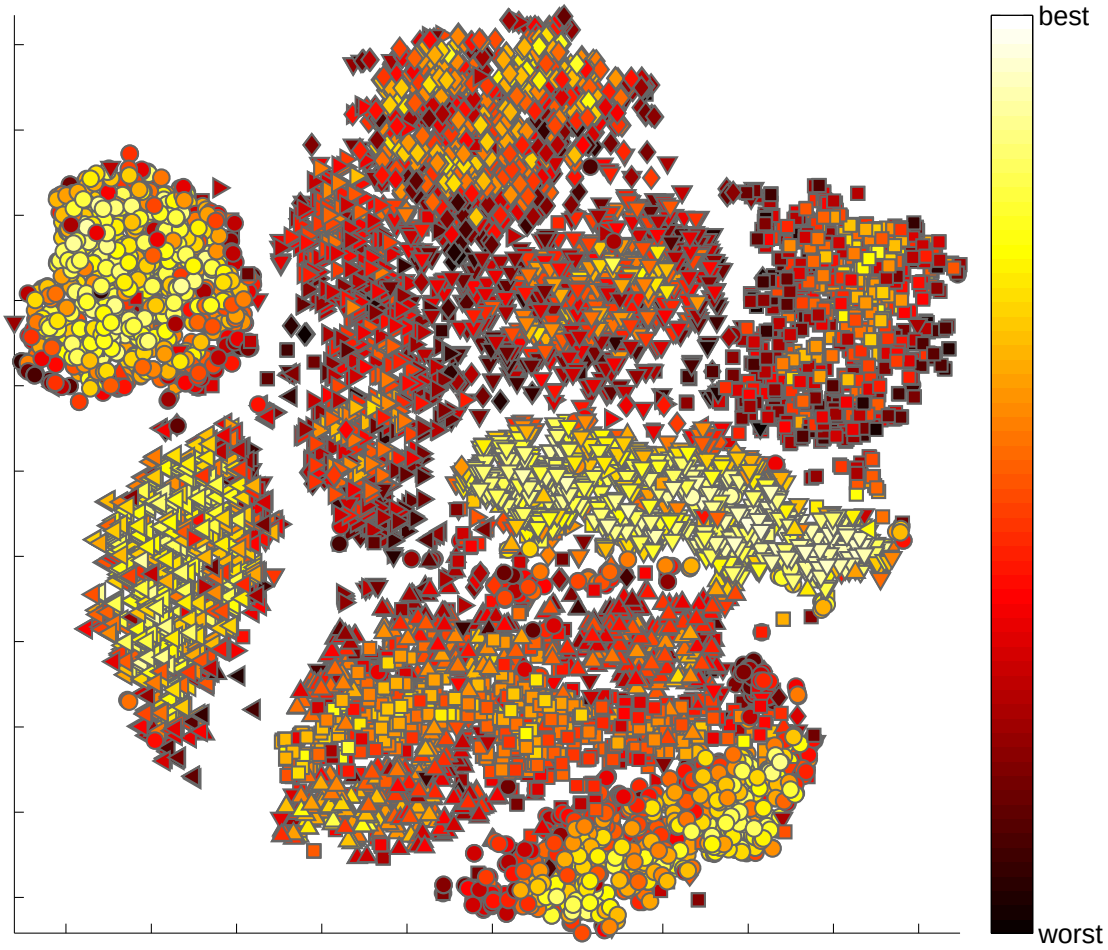


Figure 4.16.: The figure shows the point-wise quality $Q_{\mathrm{ND}}^{\mathbf{y}^i}(300, 300)$ for the t-SNE embedding of the *MNIST* data from Fig. 4.15. As expected from the curves of Fig. 4.14, we generally see many errors in the visualization. Furthermore, we can observe that the overlaps of the classes cause stronger errors. This is less pronounced for the digits 0, 1, 6, and 7. The classes 2, 3, 5, and 8 show many disturbances in the defined range of 300 neighbors, deviating from the original rank by more than 300.

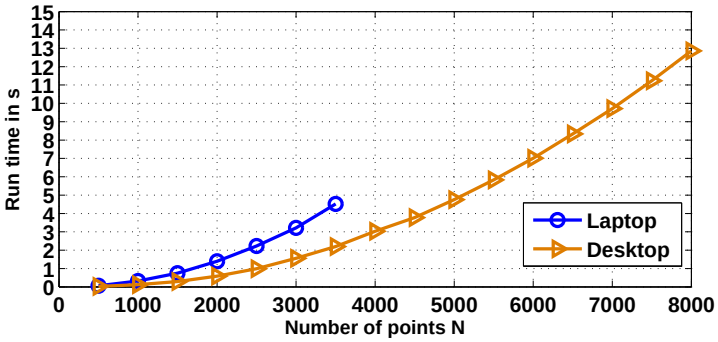
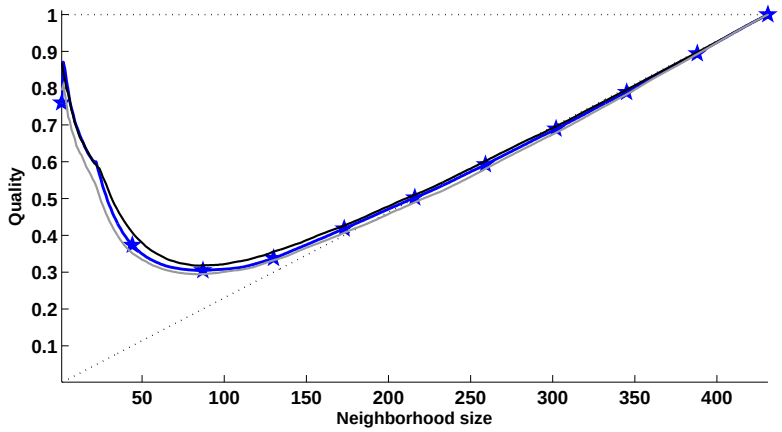


Figure 4.17.: The figure shows a small survey about the computing time to calculate the co-ranking matrix from given ranks r_{ij}, ρ_{ij}. The ranks were calculated from uniformly random points in ten dimensions which were randomly mapped to points in two dimensions. We used various data set sizes $N \in \{500, \ldots, 8000\}$ and tracked the run time of a standard Matlab implementation. One curve shows the computation times on a standard laptop machine with a 2.0 Ghz dual core processor and 2 GB of RAM, where the memory limitation allowed a maximum set size of 3500. The other curve represents a desktop computer using 4 CPU cores with 2.5 Ghz each, and 6 GB of memory.

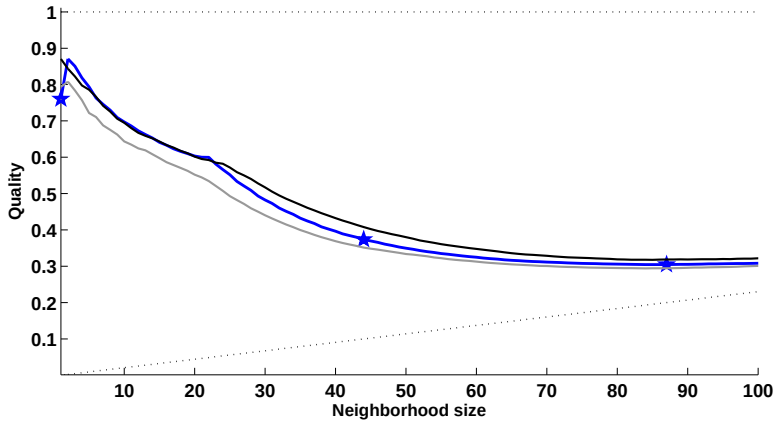
Computational effort and speedup In real-world data sets, such as *MNIST*, sizes in the order of several thousand data points become more and more common. Since the computational demand for the discussed quality evaluation is rather high, we address this topic shortly. If ranks are given, assembling the pointwise co-ranking matrices requires a lookup operation for every pair of points, therefore the time complexity is $\mathcal{O}(N^2)$. To give an impression of the practical computational effort, we tracked the run time to calculate the classical co-ranking matrix for random mappings of sizes between 500 and 8000 points on a standard laptop, as well as a modern desktop computer, see Figure 4.17.

To tackle this practical issue, we investigated in how far a random subsampling of the points affects the outcome of the quality. In a small experiment, we performed a subsampling of the t-SNE embedding of the *COIL-20* data from Figure 4.13, where we randomly sampled 30% of the points, i.e. 432 out of 1440 (using the same subset of the original as well as the embedded points). This procedure was repeated 20 times, evaluating the quality Q_{NX} every time. In Figure 4.18 we show the respective maximum and minimum of the resulting curves, together with the original quality curve as given in 4.12a. The figure shows that the deviation from the original curve is relatively small, from which we can conclude that subsampling seems to be a valid possibility to approximate the quality evaluation using less computational effort. While this shows only Q_{NX} exemplarily, we observed a similar effect for the Q_{ND} measure.

Subsampling could open the way towards an interactive graphical user interface, where the user can observe a given visualization augmented by the point-wise quality, and directly try different parameter settings for K_s, K_t. Since the computational effort and memory demands can be limited by sampling a fixed number of points, the interface can be updated instantly and the user can quickly browse various combinations. To-



(a) subsampling of Q_{NX}



(b) subsampling of Q_{NX}

Figure 4.18.: The figure shows the quality Q_{NX} for random subsamples of the t-SNE embedding from Fig. 4.13. As a reference, the line marked by pentagrams is the original quality from the full data set, as given in 4.12a. We sampled 20 times a random subset of 30% of the total points, and calculated the Q_{NX} based on this subset only. From the 20 iterations, the gray line shows the minimal outcoming value for the respective neighborhood size, while the black line shows the maximum. The neighborhood sizes of the original curve were aligned in relation to the respective value in the sampled case. The upper figure displays the graphs for all possible neighborhood sizes, and the lower figure shows a zoomed view, focusing on the neighborhoods up to 100 points only. The deviation from the original curve is fairly small, although the co-ranking matrix is based only on the subsample.

gether with techniques to accelerate the DR process itself, see e.g. [C12d], mapping and evaluation would become feasible even for very large data sets.

4.3.6. Discussion

In this section, we have discussed the established co-ranking framework, which unifies several quality criteria for dimensionality reduction. To yield a richer impression of the embedding's local characteristics, we proposed a point-wise quality measure, following directly from individual co-ranking matrices. These local quality ratings can be used to augment the given data embedding by meaningful color values which highlight distortions in the visualization for user-specified neighborhood scales. We further suggested to improve the parameterization of the original quality measure to enable more control over the evaluation's focus. In several artificial and real-world experiments, we demonstrated the benefits of our evaluation framework, and discussed possibilities to reduce the computational demand with an interactive user interface in mind.

From the presented work, one very important question arises: How valuable are the discussed quality criteria for human users? Ultimately, this question can only be answered by conducting user studies, which is a challenging topic of ongoing research. Given the ill-posed nature of DR for visualization, this question touches a wide area of related open problems and challenges. For example, there are no clearly structured benchmark scenarios for DR so far, which incorporate an explicit formulation of the expected outcome, or user expectations. This makes an evaluation of quality assessment highly challenging. User expectations, usability, and accessibility of DR methods are emerging topics in recent literature, see [86, 85, 21]. Related issues generate increasing interest and debate among researchers in the *information visualization* field (where topics often address human-computer interaction), and the machine learning community (where the attention is mainly focused on mathematical principles behind the embedding), see [68]. In this regard, a quantitative evaluation of DR methods constitutes a valuable asset for interactive user interfaces, in the future.

4.4. Comparing dissimilarity-based representations

In Chapter 1, we pointed out that for complex data, the pairwise (dis)similarity often serves as the interface of the application scenario to the machine learning tool. Hence, the learning process and its outcome are severely influenced by the choice of the dissimilarity measure. While dissimilarity measures for supervised settings can eventually be compared by the classification error, the situation is less clear in unsupervised domains where a clear objective is lacking. The question occurs, how to compare dissimilarity measures and their influence on the final result in such cases. In this section, we propose to use the quantitative measure introduced earlier in this chapter, to compare, whether (and on which scale) dissimilarities coincide for an unsupervised learning task. Essentially, the measure evaluates in how far neighborhood relations are preserved if evaluated based on rankings, which provides a robustness of the measure against scaling of data. Apart from a global comparison, local versions allow to highlight regions of the data where two dissimilarity measures induce the same results.

Section overview After a brief introduction, we will discuss different existing options for an unsupervised comparison of dissimilarity matrices in Subsection 4.4.2. The previously described quality assessment framework yields an adequate choice, since it compares two dissimilarity matrices based on their induced neighborhood structure. In Subsection 4.4.3 and 4.4.4, we will demonstrate this technique on examples, and end with concluding remarks in Subsection 4.4.5.

4.4.1. Introduction

The proposed framework can be beneficial to answer questions, such as: How can we decide whether a variation of the metric (or its parameters) results in changes of the data representation for the subsequent machine learning task? Are there possibilities to compare whether (and if so, in which regions) two metrics differ, regarding machine learning?

For supervised learning, a few extensive comparisons have been conducted in the literature, about how different dissimilarity measures influence the outcome, see, e.g. [89] for the performance of different dissimilarities for content-based image retrieval, [95] for an according study in the symbolic domain, [25] for the comparison of distances for probability measures, or [28] for the performance of classifiers on differently preprocessed dissimilarity matrices to arrive at a valid kernel. This clearly demonstrates how the difference of dissimilarity-based data representations has an impact on the results, as we detailed also in the previous Chapter 3.

The situation is less clear when dealing with unsupervised domains. Unsupervised learning is essentially ill-posed and the final objective depends on expert evaluation. The primary mathematical goal is often to cluster or visualize data, such that an underlying

structure becomes apparent. Quite a few approaches for unsupervised learning based on general dissimilarities have been proposed in the past: kernel clustering techniques, such as *kernel SOM* or *kernel NG* [139, 109]; or relational clustering as proposed for fuzzy-k-means, SOM, or NG [56, 50], as well as *relational GTM* discussed in Chapter 2. Further, many state-of-the art nonlinear visualization techniques like t-SNE are based on pairwise dissimilarities rather than vectors [131, 80].

4.4.2. How to compare dissimilarity measures?

We assume that data x^i are sampled from some underlying data space. These data are input to an unsupervised machine learning algorithm by means of pairwise dissimilarities $d_{ij} = d(x^i, x^j)$. Interestingly, although the chosen dissimilarity structure crucially determines the output of any machine learning algorithm based thereon, no framework of how to compare different dissimilarities for unsupervised domains is commonly accepted in the literature. The question occurs what is the relevant information contained in a dissimilarity, which guides the output of such an algorithm? Interestingly, even slight changes of the dissimilarity, such as a shift, can severely influence the result of an unsupervised algorithm, as shown in [50]. Apart from generic mathematical considerations, indications for the answer to this question may be taken from attempts to formalize axioms for unsupervised learning [1, 85, 135, 81]. Here, guidelines such as scale-invariance, rank-invariance, or information retrieval perspectives are formalized. In the following, we will discuss different possibilities to compare dissimilarity measures. We assume that pairwise dissimilarities d^1_{ij} and d^2_{ij} are given, which are to be compared.

Matrix comparison: The pairwise dissimilarities d^1_{ij} and d^2_{ij} give rise to two square matrices $\mathbf{D}^1$ and $\mathbf{D}^2$ respectively, which could directly be compared using some matrix norm. This possibility, however, is immediately ruled out when considering standard axioms for clustering [1], for example. One natural assumption is scale-invariance of the unsupervised learning algorithm. Scaling the matrix, however, does affect the resulting matrix norm. More generally, virtually any matrix norm severely depends on specific numeric choices of the representation rather than the global properties of the data.

Induced topology: An alternative measure which ignores numerical details but focuses on basic structures could be connected to the mathematical set-theoretic topology of a data space. Every distance measure induces a topology. Hence, it is possible to compare whether the topological structure induced by two metrics is equivalent. In mathematics, two metrics are called topologically equivalent if the inequality $c \cdot d^1(x^i, x^j) \leq d^2(x^i, x^j) \leq c' \cdot d^1(x^i, x^j)$ holds for all x^i, x^j for some constants $0 < c \leq c'$, since they induce the same topology in this case. It can easily be shown that any two metrics in a finite-dimensional real vector space are topologically equivalent. However, this observation shows that this notion is not appropriate to compare metrics with respect to their use for unsupervised

learning: topologically equivalent metrics such as the standard Euclidean metric and the maximum-norm yield qualitatively different clusters in practical applications, as we will demonstrate in an example in Subsection 4.4.3.

Rank preservation: One axiom of clustering, as formalized in [1], is the invariance to rank-preserving distortions. Indeed, many clustering or visualization techniques take into account the ranks induced by the given dissimilarity measure only, this way achieving a high robustness of the results. Examples include algorithms based on winner-takes-all schemes, or extensions such as vector quantization, NG, SOM, or similar approaches. Also, many visualization techniques try to preserve local neighborhoods as measured by the rank of data. How can rank-preservation be evaluated quantitatively? One way is to transform the matrices $\mathbf{D}^1$ and $\mathbf{D}^2$ into rank matrices, i.e. matrices which contain permutations of the numbers $\{0, \ldots, N-1\}$. Then, these two matrices could be compared by their column-wise correlation. However, usually the preservation of all ranks is not as critical as the preservation of a local neighborhood for most machine learning tools, such that different scales of the neighborhood size should be taken into account. In the previous Subsection 4.3, we explained the co-ranking framework which can be seen as a way to observe this rank-preservation property according to various neighborhood sizes of interest.

Information retrieval based comparison: Information retrieval constitutes a typical application area for unsupervised learning. Therefore a comparison of dissimilarity measures based on this perspective would be interesting. Assume a user queries a database for the neighborhood of x^i. What is the precision/recall, if d^2 is used instead of d^1? When defining the notion of neighborhood as the K nearest neighbors, precision and recall for a query x^i are both given by the term $|\{x^j \mid d^1(x^i, x^j) \leq K \wedge d^2(x^i, x^j) \leq K\}|$ normalized by K. Summing over all x^i and dividing by N yields an average of all possible queries. In fact, this instantiation of a quality measure coincides with an evaluation within the co-ranking framework, which we introduced in Section 4.3.

How can the co-ranking quality assessment measure be used to compare two dissimilarities? Since $Q_{\mathrm{NX}}(K)$ essentially evaluates in how far a rank-neighborhood induced by d_{ij} coincides with a rank-neighborhood induced by δ_{ij}, we can directly apply this measurement to two given dissimilarity measures d^1 and d^2, and obtain a quantitative statement about the rank-preservation of d^2 given d^1. Since $Q_{\mathrm{NX}}(K)$ is symmetric, the ordering of the dissimilarities is not important.

4.4.3. Comparison of metrics for the Euclidean vector space

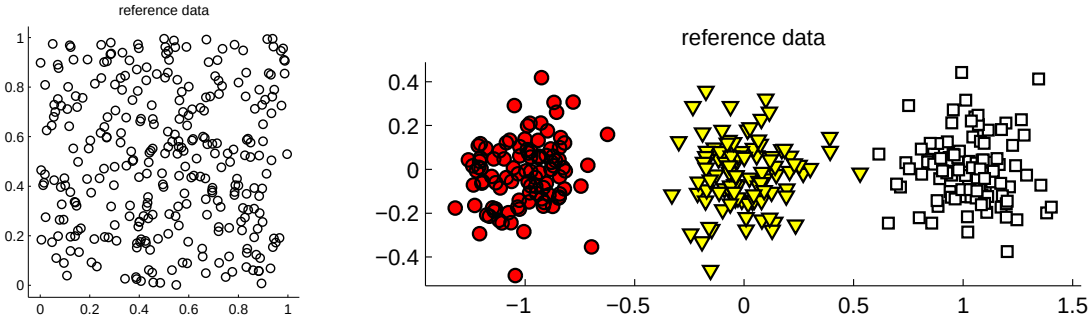
We start with an illustrative example, which shows that the measure $Q_{\mathrm{NX}}(K)$ allows to identify situations where dissimilarities induce similar/different results. We restrict to the two-dimensional Euclidean vector space where data are distributed uniformly,

or in clustered form, respectively, see Figure 4.19. For these data, we compare the Euclidean distance to the L_p norm, with $p \in \{1, 3, 6\}$ as well as the maximum-norm as the limit case. We can see the effect of these choices by using a metric multidimensional scaling (MDS) to project the data to the Euclidean plane, see Figures 4.20 and 4.21. Obviously, if data is distributed uniformly, a smooth transition from L_1 to L_∞ can be observed, as expected, whereby the global topological form does not change much. This observation is mirrored in the co-ranking evaluation, see Figure 4.22. The quality curves change smoothly and have a value near 1, indicating a good agreement of the topologies. Note that these metrics are topologically equivalent in the mathematical sense, which is supported by the observation made in this case.

The situation changes if more realistic settings are considered, i.e. if structure is present in the data. We consider three clusters and the same setting as before. Here, the metric L_1 and L_∞ yield very different behavior, as can be seen in the projection in Figure 4.21, as well as in the evaluation in Figure 4.22. Thus, equivalence in terms of mathematical topology does not imply that the overall neighborhood structures are similar, for realistic settings where cluster structures are present in the data distribution. The co-ranking framework mirrors the expected differences in these settings. Note, that due to the choice of K, differences at various scales are observable as well. In Figure 4.22, the underlying structure with cluster sizes of 100 can be clearly recovered from the quality curves.

4.4.4. Comparison of non-Euclidean settings

In the previous sections, we introduced a mathematical approach to compare two dissimilarity measures, and demonstrated it on artificial data sets. In this section, we use two real world data scenarios as a first proof-of-concept study, to show the usefulness of our approach given domain-specific – and possibly non-Euclidean – dissimilarity measures.



(a) Uniform data **(b)** Clustered data

Figure 4.19.: Original artificial data in the two-dimensional plane with, uniform distribution (a) and clustered distribution (b).

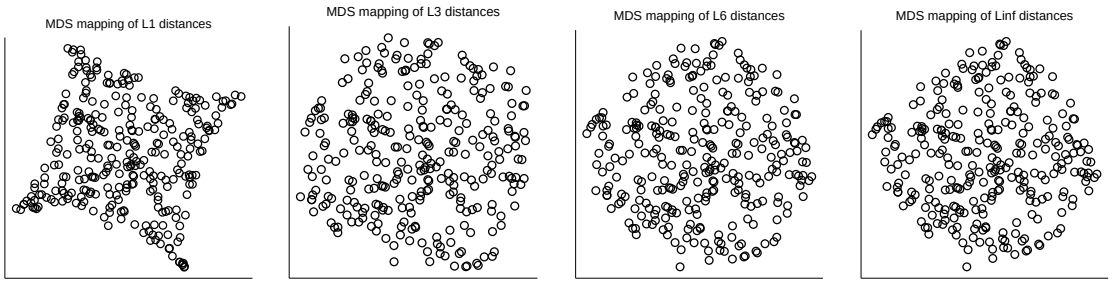


Figure 4.20.: Comparison of L_p-norms on uniform square data. (L_1, L_3, L_6, L_∞ l.t.r.)

App description texts Current research in the area of semantic web utilizes state-of-the-art machine learning and data visualization techniques, in order to automatically organize and represent vast data collections within user-friendly interfaces. Here, sophisticated dissimilarity measures for textual content play an important role. Our first experimental scenario relates to a typical machine learning task in this context. It consists of descriptions from 500 randomly collected applications, available on the online platform *Google Play*[9]. Google Play is a large distribution service for digital multimedia content which currently offers over 1.3 million downloadable programs (commonly referred to as *apps*) for the mobile operating system *Android*. Each app is attributed to one of 34 categories, while every category belongs to one of the two major branches "Games" or "Applications". The content of every app is summarized in a textual description of about 1200 characters on average. Our 500 apps come from two categories: 293 from "Arcade & Action" (in Games), and 207 from "Travel & Local" (in Applications). In the following they will be referred to as class 1 and 2, respectively. We consider three different measures to calculate dissimilarities between the descriptions:

(I) *Euclidean* distances on the tf-idf weights, where weight vectors are calculated from the frequencies of the appearing terms (tf) and their inverse frequency of occurrence in all documents (idf), see [111],

[9] `http://play.google.com`

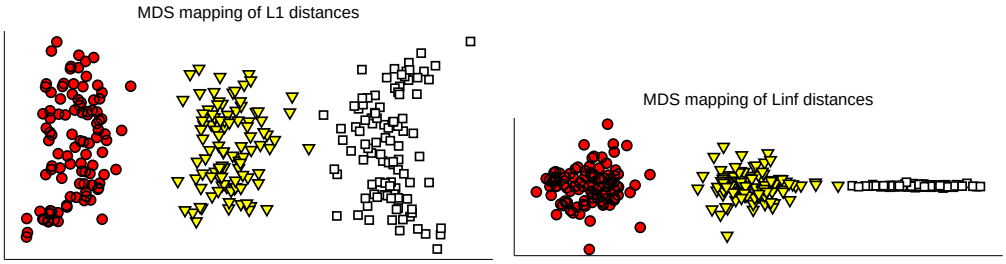


Figure 4.21.: MDS projection using L_p-norms on three clusters data. (L_1, L_∞)

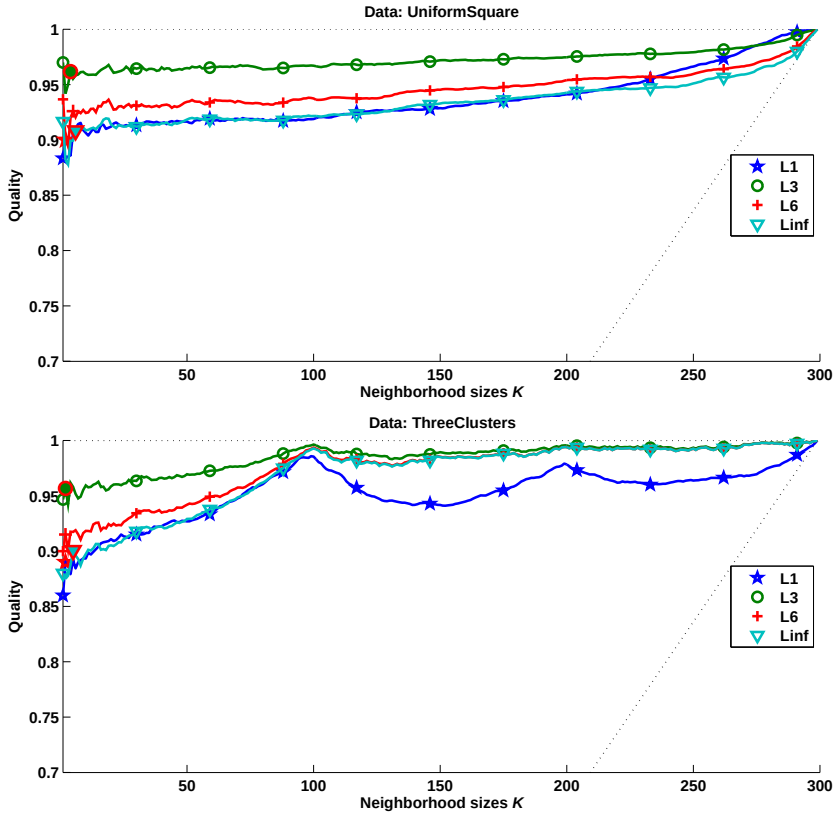


Figure 4.22.: Comparison of the dissimilarities using the co-ranking framework: uniform square (top) and three clusters (bottom).

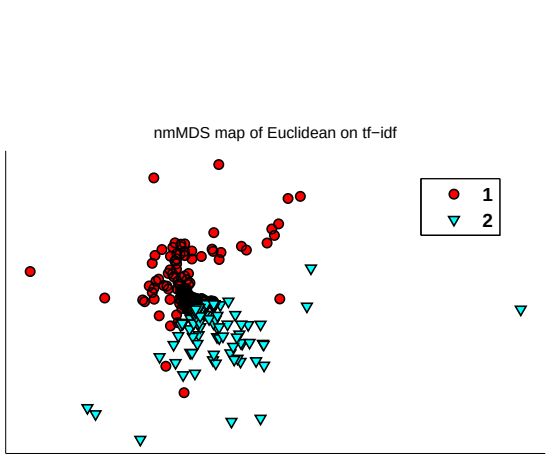
(II) the *Cosine* distance on the term frequencies, which is calculated as $c(\mathbf{a}, \mathbf{b}) := 1 - \big((\mathbf{a}^\intercal \mathbf{b})/(\pi \, \|\mathbf{a}\| \, \|\mathbf{b}\|)\big)$, where $\mathbf{a}$ and $\mathbf{b}$ are vectors of term frequencies for the two respective documents,

(III) the *normalized compression distance* (NCD) [87], which is a string dissimilarity measure described in Chapter 1, Subsection 1.2.4, in this case using the Lempel-Ziv-Markov chain compressor (LZMA).

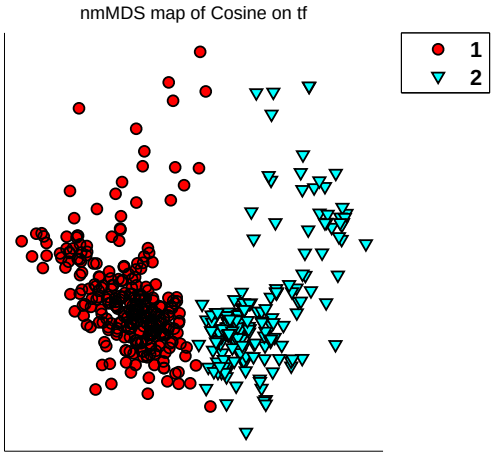
While the first two measures are based on basic word statistics, the NCD also takes structural aspects into account implicitly, since the lossless compressor utilizes recurring patterns in the texts to reduce the description length. Prior to applying the dissimilarity measures, we used a standard preprocessing workflow of stopword reduction and Porter stemming [108].

Figure 4.23 shows MDS visualizations of the three different dissimilarities, as well as evaluation curves from the comparison of Euclidean distances versus the Cosine and the NCD measure. For the visualizations in Figure 4.23a, 4.23b, 4.23d, we used non-metric MDS with squared stress. From the evaluation curves in Figure 4.23c we see that the

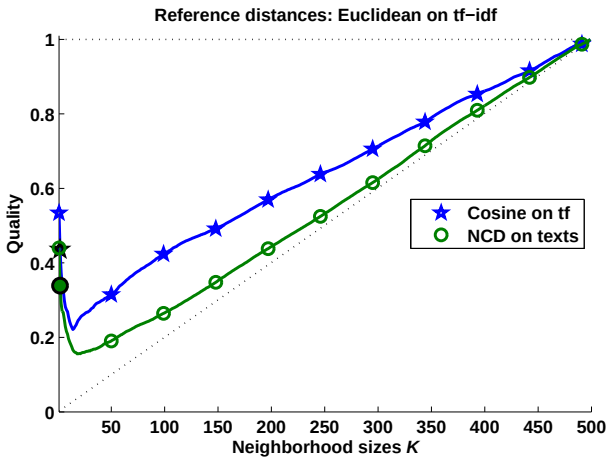
agreement of the Euclidean distances to the Cosine and NCD measure is low in general, with values below 0.6, even for very small neighborhood sizes. Although the visualizations indicate a qualitatively similar structure, the overall ranks seem to be rather different, which is also reflected in the visualizations to some extent: Figure 4.23a shows a small number of outliers, while there are fairly distinct clusters in Figure 4.23d; and Figure 4.23b shows both characteristics: similarly dense regions and some widespread outliers. In this real world data set, every pair of measures showed a low agreement when compared with the evaluation framework, with $Q_{\mathrm{NX}}(K) < 0.6$ for all $K < 100$.



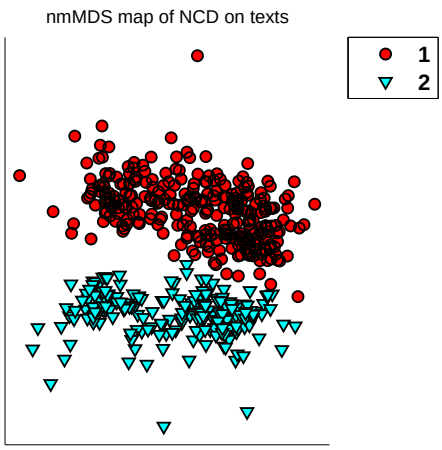
(a) MDS map of Euclidean distances



(b) MDS map of Cosine distances



(c) $Q_{\mathrm{NX}}(K)$ of Euclidean dist. vs. Cosine & NCD



(d) MDS map of NCD distances

Figure 4.23.: Comparison of three dissimilarity measures for a real-world showcase data set consisting of 500 textual descriptions of Android apps.

Java programs The second example is related to current challenges in the research of *intelligent tutoring systems* (ITS), as previously addressed in Chapter 3, Subsection 3.6.3. Our data set consists of 169 short Java programs which represent student solutions, originating from a Java programming class of first year students at Clausthal University of Technology, Germany. We used the open source plagiarism detection software *Plaggie* [2] to extract a tokenized representation (a *token stream*) from each given Java source code. Based on the token streams, we consider four different dissimilarity measures:

(I) Euclidean distances on the tf-idf weights like in the previous data set, however, tf and idf now refer to the occurrence of each token instead of term,

(II) the Cosine distance on the token frequencies,

(III) the normalized compression distance (NCD) on the token streams,

(IV) *Greedy String Tiling* (GST) which is the inherent similarity measure that Plaggie uses to compare the given sources [2, 138]; since GST yields a similarity matrix $\mathbf{S}$, with values in the interval $(0, 1)$ and self-similarities of 1, we converted $\mathbf{S}$ into dissimilarities by taking $\mathbf{D} = \sqrt{1 - \mathbf{S}}$, as proposed in [106].

Figure 4.24 shows the quality $Q_{\mathrm{NX}}(K)$ when comparing Euclidean distances to Cosine, GST, and NCD dissimilarities. The curves show the highest similarity to the Cosine distances, especially high in small neighborhood ranges, which is expected due to the fact that both are based on token frequencies. Interestingly, the curves of the Cosine and the GST measure show a similar shape in comparison to Euclidean distances, which may indicate a similar response to certain structural aspects in the data, in contrast to the steadily growing curve for NCD.

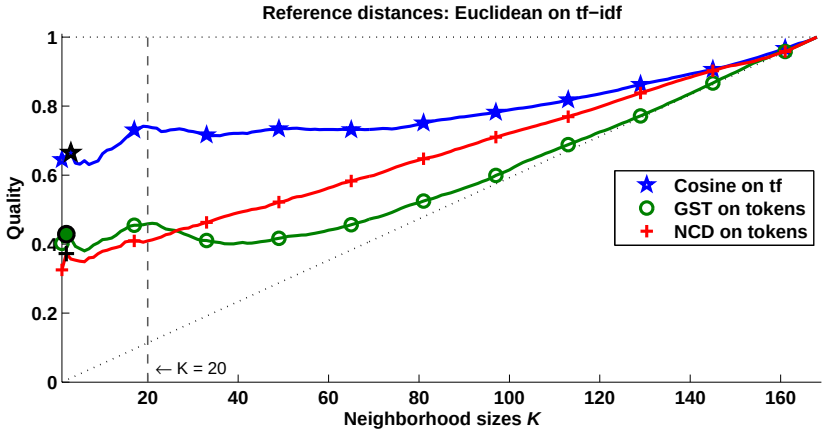
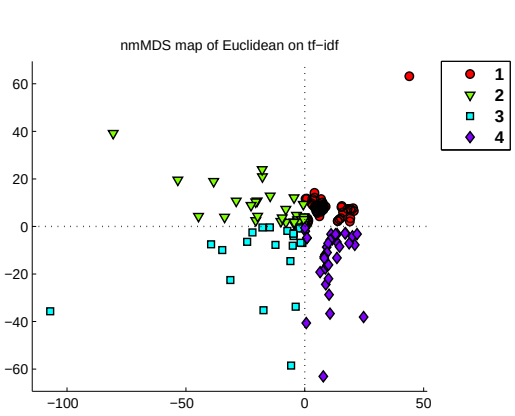


Figure 4.24.: $Q_{\mathrm{NX}}(K)$ when comparing Euclidean distances to Cosine, GST, and NCD dissimilarities used on our second real-world showcase data set consisting of 169 student solutions from a Java programming class.

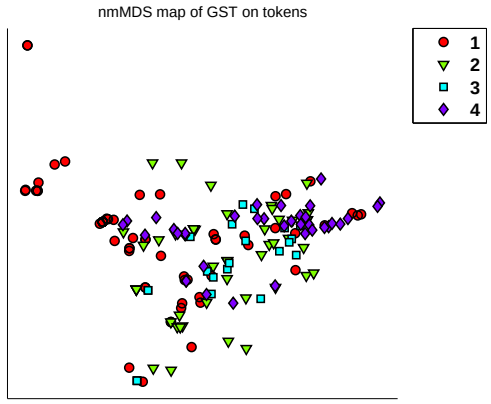
Figure 4.25 demonstrates our proposed framework for the pointwise comparison of dissimilarity measures on the same data scenario. The coloring in 4.25c and 4.25d refers to $Q_{\mathrm{NX}}^{x^i}(20)$, which is the agreement of the 20-neighborhood for every point x^i as compared to the other dissimilarity measure. To link the coloring scheme to the evaluation curves, $K = 20$ is highlighted on the graphs in Figure 4.24. The pointwise evaluation clearly reveals a region of data which is very close in the Euclidean case, but was considered very dissimilar by the GST measure.

4.4.5. Discussion

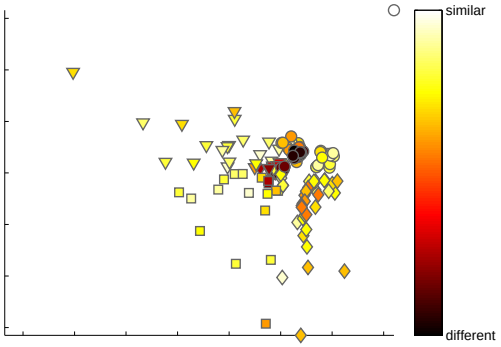
In this section, we have discussed the comparison of dissimilarity measures for unsupervised learning tasks, based on rank-preservation criteria. This opens the way for several topics of ongoing research: To test the proposed comparison scheme in the context of a typical machine learning workflow, one could refer to classification methods for relational data, such as RGTM and RGLVQ introduced in Chapter 2, and investigate how differences in the input data representation (i.e. from different dissimilarity measures) influence the output of the method. While different input matrices are compared in an unsupervised manner, one could refer to data sets where class labels are available, and explicitly track changes in the resulting classification accuracies. Another canonical application field are metric learning techniques, as presented in Chapter 3: When the parameters of a dissimilarity measure are adapted during training, the induced changes in the data representation could be observed directly, on a local and global scale. With the help of DR techniques, even a visual representation of these metric changes is possible, which is the subject of ongoing work.



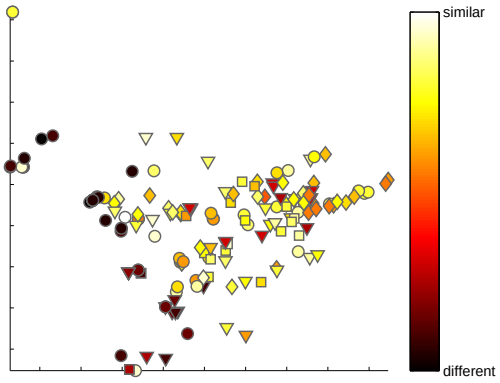
(a) MDS map of Euclidean distances



(b) MDS map of GST dissimilarities



(c) Euclidean distances, colored by $Q_{\mathrm{NX}}^{x^i}(20)$ vs. GST dissimilarities



(d) GST dissimilarities, colored by $Q_{\mathrm{NX}}^{x^i}(20)$ vs. Euclidean distances

Figure 4.25.: Pointwise comparison of dissimilarity measures used on a data set of 169 student solutions from a Java programming class. The dissimilarities from two measures (Euclidean and GST) are embedded in 2D using non-metric MDS, see (a) and (b). The different symbols for points in the visualizations do not correspond to given semantic class labels, but to the quadrants of the cartesian coordinate system in (a), to give some indication of how the point locations differ to the map of GST in (b). The pointwise coloring in (c) and (d) shows for each point, how much the neighbor ranks for $K = 20$ in the Euclidean case differ from the ranks given by GST.

Chapter 5.

Conclusion

Summary In this thesis, we discussed particular challenges of machine learning, regarding complex data and their representation. We identified three distinctive characteristics of complex data sets: high dimensionality in vectorial data description schemes, large set sizes due to contextual relationships among instances, and strongly pronounced aspects of compositionality in the encoding. In this regard, feature-based data representations are usually not well-suited, because they have limited capabilities to account for heterogeneous information and structured encoding schemes. One promising strategy is the use of domain-specific dissimilarity measures to obtain pairwise proximities between raw data instances, in which aspects of compositionality are treated explicitly in comparison to another instance. For a resulting dissimilarity-based data representation, the inherent dimensionality is bounded by the number of data points.

We demonstrated how such a relational data representation can be integrated in two classical prototype-based machine learning methods, for unsupervised and supervised scenarios. Several benchmark experiments proved the viability of this approach, showing competetive results compared to established kernel classifiers. Particular advantages are that prototype-based learning with relational data is not restricted to metric distances, and yields an interpretable model via representative exemplars in the data space.

Further, we addressed specific caveats of dissimilarity-based data representations, along with corresponding remedies. One problem is the inherent quadratic complexity, which can be circumvented via low-rank matrix approximations like the Nyström technique. In experiments with several benchmark data sets, we confirmed the suitability and efficiency of this approximation in case of the relational LVQ classifier. Another important issue is the intricate choice of parameters when applying dissimilarity measures on complex data sets. Therefore, we proposed a learning scheme, which adapts the parameters of a sequence alignment measure to the given classification problem. This has shown to facilitate the classification accuracy in experiments on real-world data sets, while simplifying the necessary classification boundary, i.e. decreasing the number of support vectors in an SVM.

For unsupervised learning scenarios, low-dimensional Euclidean embeddings of the

data offer interesting possibilities: they can serve as an alternative data representation, but also yield an approximative basis for visualizations, in which the data set's neighborhood structure becomes easily observable. In this context, quantitative measures are available to assess the reliability of a given embedding, independent of the learning task. We proposed an extension to integrate such a quality criterion into the corresponding visualization, on a point-wise basis. Further, a different parameterization was introduced, which allows for more fine-grained control in the assessment procedure. Intuitive results have been presented for reference data, illustrating the working principle and practical value of the measure, particularly in cases where strong assumptions about the original data structure were available as tentative ground truth. The same principle can be utilized to compare different dissimilarity measures for the same data set. To prove the concept, we presented results for artificial and real-world data sets with accompanying visualizations. These tools can help to assess the suitability of data representations, in the absence of an explicit classification goal.

Future work Based on the work presented in this thesis, we can point out several avenues of ongoing research. To avoid overlaps with specific open problems that were mentioned in the previous chapters, the prospects stated here are more general in nature.

In this thesis, we have focused on the principle of relational prototypes, in order to address input data represented by pairwise proximities, and how corresponding techniques can be derived from classical prototype-based machine learning models. A complementary strategy is the "kernelization" of learning methods, in which the central notion of proximity is based on inner products, and can be exchanged by an appropriate kernel function. While the latter is a more widespread approach, recent literature establishes relational methods as a distinct alternative, due to the advantage of a precise theoretical foundation for non-metric dissimilarities, see [51, 112]. Since kernel matrices can be converted to dissimilarities without loosing information, the relational approach is more general. Therefore, a new class of algorithms becomes available, for which thorough comparative studies should be considered. Apart from an empirical evaluation, there are open questions regarding the theoretical properties of non-Euclidean dissimilarities in the context of machine learning. One concrete example is the probabilistic interpretation for RGTM, from Chapter 2: while GTM yields a probabilistic model of the data in the Euclidean space, these properties are no longer guaranteed for the pseudo-Euclidean embedding of relational data. Similarly, it is not clear in how far fundamental concepts from learning theory can be transferred to prototype-based learning models in the pseudo-Euclidean space, see [50, 106].

In the previous chapters, we addressed several application scenarios, which bear potential for an integration into practical software environments in the future. In this regard, *intelligent tutoring systems* (ITSs), as mentioned in Chapter 2, are a particularly promising area to adopt machine learning techniques. An ITS is a specific type of educational software, that supports students in their learning process, e.g. to learn

Java programming skills, by providing individual feedback to students while they fulfill a given assignment. The research program *FIT*[1] is currently aiming to realize adaptive feedback mechanisms in a new, domain-agnostic ITS, using machine learning methods. The novelty of this approach is its independence of a particular tutoring domain: feedback strategies are abstract and rely only on the student's current solution, together with a database of finished examples, which are all encoded in a generic graph format. By training the metric parameters according to given data, a problem-adapted dissimilarity measure can identify meaningful examples for feedback in a particular domain. In our ongoing work, this generic approach will be integrated into a flexible software architecture that fits into existing landscapes of educational software, see [C13a], [J14]. Field studies about the plausibility of the automatically generated feedback are currently being conducted by experts in the fields of ITSs and computer-aided education [J14], [C13b]. Based on the resulting dissimilarity-based data representation, it becomes possible to investigate a student's learning progress over time, which will be subject of the follow-up research project *DynaFIT*. One important topic in the field of computer-aided education are *open learner models* [22, 92], whereby students can inspect their learning progress and current state, and teachers may observe individual learning characteristics. With increasingly complex ITSs, there is a strong demand for visualization techniques to create a compact display of learner model data, e.g. as visual feedback for students and teachers [36].

Dimensionality reduction (DR) offers great potential to visualize data in intuitive user interfaces. It is a promising tool to make data representations accessible and comprehensible for expert users, especially when combined with prototype-based classification models. While the resulting low-dimensional embeddings are often only depicted in simple scatter plots of the data, one can imagine more sophisticated, interactive display techniques to explore very large data collections easily. Current research of DR methods is concentrated in the machine learning community, with a strong focus on mathematical strategies to obtain meaningful embeddings of high-dimensional data. This encompasses recent algorithmic solutions to handle very large data sets efficiently, and thereby address ample real-world applications [130], and [C13d]. However, only few studies have been conducted so far, that thoroughly investigate the benefits of modern nonlinear DR from the viewpoint of *human computer interaction*, taking usability, perception, and cognition into account, see [86, 85, 21]. This open topic has recently spawned fruitful debate with researchers from the *information visualization* community, see [68]. In this regard, more user-centered studies and evaluations become necessary to substantiate the potential of DR for visualization. The combination of sophisticated data-driven mathematical approaches with the visual appeal and usability of information visualization techniques could offer mutual benefit.

[1] *Learning Feedback in Intelligent Tutoring Systems* (FIT) is a research program funded by the German Science Foundation (DFG) within the priority programme 1527 "Autonomous Learning".

Appendix A.

Additional information

A.1. Derivative of soft alignment

Recall the definition of the soft minimum:

$$\text{softmin}(x_1, \ldots, x_n) = \frac{1}{Z} \sum_i p_i \cdot x_i$$

where

$$p_i = \exp(-\beta \cdot x_i)$$
$$Z = \sum_j p_j$$

The derivative of the *soft sequence alignment dissimilarity* with respect to the parameter λ_q is given as:

$$\begin{aligned}
\frac{\partial}{\partial \lambda_q} \text{softmin}(x_1, \ldots, x_n) &= \frac{\partial}{\partial \lambda_q} \frac{1}{Z} \sum_i p_i \cdot x_i \\
&= \frac{1}{Z^2} \left[\left(\sum_i \frac{\partial}{\partial \lambda_q} p_i \cdot x_i \right) \cdot Z - \left(\sum_i p_i \cdot x_i \right) \cdot \left(\frac{\partial}{\partial \lambda_q} Z \right) \right] \\
&= \frac{1}{Z} \left[\sum_i \frac{\partial}{\partial \lambda_q} p_i \cdot x_i - \text{softmin}(x_1, \ldots, x_n) \cdot \frac{\partial}{\partial \lambda_q} Z \right] (*)
\end{aligned}$$

Furthermore:

$$\begin{aligned}
\frac{\partial}{\partial \lambda_q} p_i \cdot x_i &= \left(\frac{\partial}{\partial \lambda_q} p_i \right) \cdot x_i + p_i \cdot \left(\frac{\partial}{\partial \lambda_q} x_i \right) \\
&= p_i \cdot (-\beta) \cdot \left(\frac{\partial}{\partial \lambda_q} x_i \right) \cdot x_i + p_i \cdot \left(\frac{\partial}{\partial \lambda_q} x_i \right) \\
&= p_i \cdot \left(\frac{\partial}{\partial \lambda_q} x_i \right) \cdot (-\beta \cdot x_i + 1)
\end{aligned}$$

and:

$$\begin{aligned}\frac{\partial}{\partial\lambda_q}Z &= \sum_j \frac{\partial}{\partial\lambda_q}p_j \\ &= \sum_j p_j \cdot (-\beta) \cdot \frac{\partial}{\partial\lambda_q}x_j\end{aligned}$$

It follows:

$$\begin{aligned}(*) &= \frac{1}{Z}\left[\sum_i p_i \cdot \left(\frac{\partial}{\partial\lambda_q}x_i\right) \cdot (-\beta \cdot x_i + 1) - \text{softmin}(x_1,\ldots,x_n) \cdot \left(\sum_j p_j \cdot (-\beta) \cdot \frac{\partial}{\partial\lambda_q}x_j\right)\right] \\ &= \frac{1}{Z}\left[\sum_i p_i \cdot \left(\frac{\partial}{\partial\lambda_q}x_i\right) \cdot (-\beta \cdot x_i + 1) + \sum_i \text{softmin}(x_1,\ldots,x_n) \cdot \left(p_i \cdot \beta \cdot \frac{\partial}{\partial\lambda_q}x_i\right)\right] \\ &= \frac{1}{Z}\left[\sum_i p_i \cdot \left(\frac{\partial}{\partial\lambda_q}x_i\right) \cdot (-\beta \cdot x_i + 1) + \text{softmin}(x_1,\ldots,x_n) \cdot \left(p_i \cdot \beta \cdot \frac{\partial}{\partial\lambda_q}x_i\right)\right] \\ &= \frac{1}{Z}\sum_i p_i \cdot \left(\frac{\partial}{\partial\lambda_q}x_i\right) \cdot [-\beta \cdot x_i + 1 + \text{softmin}(x_1,\ldots,x_n) \cdot \beta] \\ &= \frac{1}{Z}\sum_i p_i \cdot \left(\frac{\partial}{\partial\lambda_q}x_i\right) \cdot [1 - \beta \cdot (x_i - \text{softmin}(x_1,\ldots,x_n))] \\ &= \sum_i \left(\frac{\partial}{\partial\lambda_q}x_i\right) \cdot \text{softmin}'(x_i)\end{aligned}$$

with

$$\text{softmin}'(x_i) = \frac{p_i}{Z} \cdot [1 - \beta \cdot (x_i - \text{softmin}(x_1,\ldots,x_n))]$$

This directly leads to Equation 3.6.

Hebbian learning as a limit case The derivative has a particular nice interpretation for $\beta \to \infty$. Consider:

$$\begin{aligned}\frac{p_i}{Z} &= \frac{\exp(-\beta \cdot x_i)}{\sum_j \exp(-\beta \cdot x_j)} \\ &= \frac{\exp(-\beta \cdot x_i) \cdot \exp(\beta \cdot \min(x_1,\ldots,x_n))}{\sum_j \exp(-\beta \cdot x_j) \cdot \exp(\beta \cdot \min(x_1,\ldots,x_n))} \\ &= \frac{\exp[-\beta \cdot (x_i - \min(x_1,\ldots,x_n))]}{\sum_j \exp[-\beta \cdot (x_j - \min(x_1,\ldots,x_n))]}\end{aligned}$$

Consider two distinct cases for x_j:

- $x_j = \min(x_1,\ldots,x_n)$. Then:

$$\exp[-\beta \cdot (x_j - \min(x_1,\ldots,x_n))] = \exp[-\beta \cdot 0] = 1$$

- $x_j > \min(x_1, \ldots, x_n)$. Then (using $\beta \to \infty$):

$$\exp[-\beta \cdot (x_j - \min(x_1, \ldots, x_n))] \approx 0$$

Let $i_1, \ldots, i_T$ be the indices, for which $x_{i_t} = \min(x_1, \ldots, x_n)$. Then it follows:

$$\sum_j \exp[-\beta \cdot (x_j - \min(x_1, \ldots, x_n)) \approx \sum_{t=1}^{T} \exp[-\beta \cdot (x_{i_t} - \min(x_1, \ldots, x_n)) = \sum_{t=1}^{T} 1 = T$$

which in turn leads to:

$$\frac{p_i}{Z} \approx \hat{\delta}_{\min}(x_i)$$

where

$$\hat{\delta}_{\min}(x_i) := \begin{cases} \frac{1}{T} & \text{if } x_i = \min(x_1, \ldots, x_n) \\ 0 & \text{otherwise} \end{cases}$$

Now it is obvious that softmin does indeed approach min for large β:

$$\text{softmin}(x_1, \ldots, x_n) = \frac{1}{Z} \sum_i p_i \cdot x_i \approx \sum_{t=1}^{T} \frac{1}{T} \cdot \min(x_1, \ldots, x_n) = \min(x_1, \ldots, x_n)$$

For $\text{softmin}'(x_{i_t})$ we get:

$$\begin{aligned} \text{softmin}'(x_{i_t}) &= \frac{p_i}{Z} \cdot [1 - \beta \cdot (x_i - \text{softmin}(x_1, \ldots, x_n))] \\ &\approx \frac{1}{T} \cdot [1 - \beta \cdot (x_{i_t} - \min(x_1, \ldots, x_n))] \\ &= \frac{1}{T} \cdot [1 - \beta \cdot 0] \\ &= \frac{1}{T} \end{aligned}$$

For all other x_j with $x_j > \min(x_1, \ldots, x_n)$:

$$\begin{aligned} \text{softmin}'(x_j) &= \frac{p_i}{Z} \cdot [1 - \beta \cdot (x_j - \text{softmin}(x_1, \ldots, x_n))] \\ &\approx 0 \cdot [1 - \beta \cdot (x_j - \min(x_1, \ldots, x_n))] \\ &= 0 \end{aligned}$$

Therefore: $\text{softmin}'(x_i) = \hat{\delta}_{\min}(x_i)$. Consider Equation 3.6 again, and plug in that result:

$$\begin{aligned}\frac{\partial d^*(\bar{a}(I+1), \bar{b}(J+1))}{\partial \lambda_q} &= \hat{\delta}_{\min}(A_{\texttt{Rep}}) \cdot \left(\frac{\partial d^* \left(\bar{a}(I), \bar{b}(J)\right)}{\partial \lambda_q} + \frac{\partial d_\lambda(a_{I+1}, b_{J+1})}{\partial \lambda_q} \right) \\ &+ \hat{\delta}_{\min}(A_{\texttt{Ins}}) \cdot \left(\frac{\partial d^* \left(\bar{a}(I+1), \bar{b}(J)\right)}{\partial \lambda_q} + \frac{\partial d_\lambda(-, b_{J+1})}{\partial \lambda_q} \right) \\ &+ \hat{\delta}_{\min}(A_{\texttt{Del}}) \cdot \left(\frac{\partial d^* \left(\bar{a}(I), \bar{b}(J+1)\right)}{\partial \lambda_q} + \frac{\partial d_\lambda(a_{I+1}, -)}{\partial \lambda_q} \right)\end{aligned}$$

Recall Equation 3.2 and consider the optimal extensions $\bar{a}^*$ and $\bar{b}^*$ using arg min instead of min. Using a simple inductive argument it clearly follows:

- For the case of symbolic sequences:

$$\frac{\partial}{\partial \lambda_{km}} d(\bar{a}, \bar{b}) = \sum_{i=1}^{|\bar{a}^*|} \hat{\delta}(a_i^*, k) \cdot \hat{\delta}(b_i^*, m)$$

- For the case of vectorial sequences:

$$\frac{\partial}{\partial \lambda_r} d(\bar{a}, \bar{b}) = \sum_{i=1}^{|\bar{a}^*|} d_r(a_i^{*r}, b_i^{*r})$$

  where $a_i^{*r} = \psi$ if $a_i^* = -$ and $b_i^{*r} = \psi$ if $b_i^* = -$.

This strongly resembles Hebbian learning as argued in Section 3.4.

A.2. Information about the Chromosomes data set

Σ	Δ	# occ.
f	-6	0
e	-5	32
d	-4	149
c	-3	468
b	-2	770
a	-1	2542
=	0	17675
A	+1	2746
B	+2	596
C	+3	318
D	+4	195
E	+5	114
F	+6	0

Table A.1.: The differential encoding for sequences from the *Chromosomes* data, described in Section 3.6: each symbol of the alphabet Σ represents a level of difference Δ in the density along a banded chromosome. The number of occurrences (# occ.) of each symbol are reported for the *"CopChromTwo"* set, which is a subset of the original Copenhagen Chromosomes database, see [91, 64].

A.3. Information about the Sorting data set

Σ_κ for property 'type'	Example
array access	`arr[4]`
array type	`int[]`
assignment	`tmp = arr[i]`
binary	`arr[i] > arr[i + 1]`
block	`{ ... }`
break	`break;`
class	`public class MyClass { ... }`
compilation unit	The entire program file
compound assignment	`a += 2`
do while loop	`do{ i++;} while(i < 10);`
expression statement	`tmp = arr[i];`
for loop	`for(i = 1; i < 10; i++) { ... }`
identifier	`i`
if	`if(i > 10){ ... }`
import	`import java.util.HashMap;`
literal	`5`
member select	`arr.length`
method	`int my_fun(int i) { ... }`
method invocation	`bubble(A, l, r)`
modifiers	`public`
new array	`new arr[4]`
new class	`new ArrayList<Integer>()`
parameterized type	`new ArrayList<Integer>`
parenthesized	`(arr[i] > arr[i + 1])`
primitive type	`int`
return	`return arr;`
unary	`i++`
variable	`int i = 0;`
while loop	`while (swapped) { ... }`

Table A.2.: The alphabet Σ_κ for property `type` used in the *Sorting* data set, in Section 3.6: every symbol of the alphabet (left), with an example Java code snippet illustrating the respective type (right).

Appendix B.

Publications in the context of this thesis

Journal articles

[J15] Bassam Mokbel, Benjamin Paassen, Frank-Michael Schleif, and Barbara Hammer. Metric learning for sequences in relational LVQ. *Neurocomputing*, (accepted/in press), 2015.

[J14] Sebastian Gross, Bassam Mokbel, Benjamin Paassen, Barbara Hammer, and Niels Pinkwart. Example-based feedback provision using structured solution spaces. *International Journal of Learning Technology*, 9(3):248–280, 2014.

[J13] Bassam Mokbel, Wouter Lueks, Andrej Gisbrecht, and Barbara Hammer. Visualizing the quality of dimensionality reduction. *Neurocomputing*, 112:109–123, 2013.

[J12] Andrej Gisbrecht, Bassam Mokbel, Frank-Michael Schleif, Xibin Zhu, and Barbara Hammer. Linear time relational prototype based learning. *International Journal of Neural Systems*, 22(5), 2012.

[J11] Andrej Gisbrecht, Bassam Mokbel, and Barbara Hammer. Relational generative topographic mapping. *Neurocomputing*, 74(9):1359–1371, April 2011.

Conference articles

[C14c] Bassam Mokbel, Benjamin Paassen, and Barbara Hammer. Efficient adaptation of structure metrics in prototype-based classification. In S. Wermter, C. Weber, W. Duch, T. Honkela, P. D. Koprinkova-Hristova, S. Magg, G. Palm, and A. E. P. Villa, editors, *Artificial Neural Networks and Machine Learning - ICANN 2014 - 24th International Conference on Artificial Neural Networks, Hamburg, Germany, September 15-19, 2014. Proceedings*, volume 8681 of *Lecture Notes in Computer Science*, pages 571–578. Springer, 2014.

[C14b] Bassam Mokbel, Benjamin Paassen, and Barbara Hammer. Adaptive distance measures for sequential data. In M. Verleysen, editor, *22th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2014, Bruges, Belgium, April 23-25, 2014*, pages 265–270. i6doc.com, 2014.[1]

[1] Winner of the *best student paper* award at ESANN 2014.

[C14a] Sebastian Gross, Bassam Mokbel, Barbara Hammer, and Niels Pinkwart. How to select an example? A comparison of selection strategies in example-based learning. In S. Trausan-Matu, K. E. Boyer, M. E. Crosby, and K. Panourgia, editors, *Intelligent Tutoring Systems - 12th International Conference, ITS 2014, Honolulu, HI, USA, June 5-9, 2014. Proceedings*, volume 8474 of *Lecture Notes in Computer Science*, pages 340–347. Springer, 2014.

[C13d] Andrej Gisbrecht, Barbara Hammer, Bassam Mokbel, and Alexander Sczyrba. Nonlinear dimensionality reduction for cluster identification in metagenomic samples. In E. Banissi, H. Azzag, M. W. McK. Bannatyne, S. Bertschi, F. Bouali, R. Burkhard, J. Counsell, A. Cuzzocrea, M. J. Eppler, B. Hammer, M. Lebbah, F. T. Marchese, M. Sarfraz, A. Ursyn, G. Venturini, and T. G. Wyeld, editors, *17th International Conference on Information Visualisation, IV 2013, London, United Kingdom, July 16-18, 2013*, pages 174–179. IEEE, 2013.

[C13c] Sebastian Gross, Bassam Mokbel, Barbara Hammer, and Niels Pinkwart. Towards providing feedback to students in absence of formalized domain models. In H.C. Lane, K. Yacef, J. Mostow, and P. Pavlik, editors, *Proceedings of 16th International Conference on Artificial Intelligence in Education (AIED)*, pages 644–648, 2013.

[C13b] Bassam Mokbel, Sebastian Gross, Benjamin Paassen, Niels Pinkwart, and Barbara Hammer. Domain-independent proximity measures in intelligent tutoring systems. In S. K. D'Mello, R. A. Calvo, and A. Olney, editors, *Proceedings of the 6th International Conference on Educational Data Mining (EDM)*, pages 334–335, 2013.

[C13a] Sebastian Gross, Bassam Mokbel, Barbara Hammer, and Niels Pinkwart. Towards a domain-independent ITS middleware architecture. In N.-S. Chen, R. Huang, Kinshuk, Y. Li, and D. G. Sampson, editors, *Proceedings of the 13th IEEE International Conference on Advanced Learning Technologies (ICALT)*, pages 408–409, 2013.

[C12g] Sebastian Gross, Bassam Mokbel, Barbara Hammer, and Niels Pinkwart. Feedback provision strategies in intelligent tutoring systems based on clustered solution spaces. In J. Desel, J. M. Haake, and C. Spannagel, editors, *DeLFI 2012: Die 10. e-Learning Fachtagung Informatik*, pages 27–38, Hagen, Germany, 2012. Köllen.

[C12f] Bassam Mokbel, Sebastian Gross, Markus Lux, Niels Pinkwart, and Barbara Hammer. How to quantitatively compare data dissimilarities for unsupervised machine learning? In N. Mana, F. Schwenker, and E. Trentin, editors, *Proceedings of the 5th IAPR-TC3 Workshop on Artificial Neural Networks in Pattern Recognition*, volume 7477 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2012.

[C12e] Frank-Michael Schleif, Bassam Mokbel, Andrej Gisbrecht, Leslie Theunissen, Volker Dürr, and Barbara Hammer. Learning relevant time points for time-series data in the life sciences. In A. E. P. Villa, W. Duch, P. Erdi, F. Masulli, and G. Palm, editors, *Proceedings of the 22nd International Conference on Artificial Neural Networks (ICANN), Lausanne, Switzerland, September 11-14, 2012, Proc. Part II*, volume 7553 of *Lecture Notes in Computer Science*, pages 531–539. Springer, 2012.

[C12d] Andrej Gisbrecht, Bassam Mokbel, and Barbara Hammer. Linear basis-function t-sne for fast nonlinear dimensionality reduction. In *Proceedings of the 12th International Joint Conference on Neural Networks*, IJCNN, pages 1–8. IEEE, 2012.

[C12c] Bassam Mokbel, Wouter Lueks, Andrej Gisbrecht, Michael Biehl, and Barbara Hammer. Visualizing the quality of dimensionality reduction. In M. Verleysen, editor, *20th European Symposium on Artificial Neural Networks, ESANN 2012, Bruges, Belgium, April 25-27, 2012*, pages 179–184. ESANN 2012, 2012.

[C12b] Andrej Gisbrecht, Wouter Lueks, Bassam Mokbel, and Barbara Hammer. Out-of-sample kernel extensions for nonparametric dimensionality reduction. In M. Verleysen, editor, *20th European Symposium on Artificial Neural Networks, ESANN 2012, Bruges, Belgium, April 25-27, 2012*, pages 531–536. ESANN 2012, 2012.

[C12a] Barbara Hammer, Bassam Mokbel, Frank-Michael Schleif, and Xibin Zhu. White box classification of dissimilarity data. In E. Corchado, V. Snáel, A. Abraham, M. Wozniak, M. Graña, and S.-B. Cho, editors, *Hybrid Artificial Intelligent Systems*, volume 7208 of *Lecture Notes in Computer Science*, pages 309–321. Springer, 2012.

[C11c] Barbara Hammer, Bassam Mokbel, Frank-Michael Schleif, and Xibin Zhu. Prototype-based classification of dissimilarity data. In *Proceedings of the 10th international conference on Advances in intelligent data analysis X*, IDA'11, pages 185–197. Springer, 2011.

[C11b] Barbara Hammer, Michael Biehl, Kerstin Bunte, and Bassam Mokbel. A general framework for dimensionality reduction for large data sets. In *Proceedings of the 8th international conference on Advances in self-organizing maps*, WSOM'11, pages 277–287. Springer, 2011.

[C11a] Barbara Hammer, Andrej Gisbrecht, Alexander Hasenfuss, Bassam Mokbel, Frank-Michael Schleif, and Xibin Zhu. Topographic mapping of dissimilarity data. In *Proceedings of the 8th international conference on Advances in self-organizing maps*, WSOM'11, pages 1–15. Springer, 2011.

[C10d] Bassam Mokbel, Andrej Gisbrecht, and Barbara Hammer. On the effect of clustering on quality assessment measures for dimensionality reduction. In *Proceedings of NIPS Workshop on Challenges in Data Visualization*, Whistler, Canada, 2010.

[C10c] Andrej Gisbrecht, Bassam Mokbel, and Barbara Hammer. The Nyström approximation for relational generative topographic mapping. In *Proceedings of NIPS Workshop on Challenges in Data Visualization*, Whistler, Canada, 2010.

[C10b] Andrej Gisbrecht, Bassam Mokbel, Alexander Hasenfuss, and Barbara Hammer. Visualizing dissimilarity data using generative topographic mapping. In *Proceedings of the 33rd annual German conference on Advances in artificial intelligence*, KI'10, pages 227–237. Springer, 2010.

[C10a] Andrej Gisbrecht, Bassam Mokbel, and Barbara Hammer. Relational generative topographic map. In *ESANN 2010, 18th European Symposium on Artificial Neural Networks, Bruges, Belgium, April 28-30, 2010, Proceedings*, pages 277–282. d-side publications, Belgium, 2010.

[C09a] Bassam Mokbel, Alexander Hasenfuss, and Barbara Hammer. Graph-based representation of symbolic musical data. In A. Torsello, F. Escolano, and L. Brun, editors, *7th IAPR-TC-15 International Workshop on Graph-Based Representations in Pattern Recognition (GbRPR)*, volume 5534 of *Lecture Notes in Computer Science*, pages 42–51. Springer, 2009.

Non-refereed publications

[TR12] Bassam Mokbel, Mario Heinz, and Georg Zentgraf. Analyzing motion data by clustering with metric adaptation. In Th. Villmann and F.-M. Schleif, editors, *Machine Learning Reports 01/2012 – Proceedings of the German-Polish Workshop on Computational Biology, Scheduling and Machine Learning (ICOLE 2011)*, number MLR-01-2012 in Machine Learning Reports, pages 70–79, 2012. ISSN:1865-3960.

[TR11] Wouter Lueks, Bassam Mokbel, Michael Biehl, and Barbara Hammer. How to evaluate dimensionality reduction? - improving the co-ranking matrix. *CoRR*, abs/1110.3917, 2011.

Bibliography

[1] M. Ackerman, S. Ben-David, and D. Loker. Towards property-based classification of clustering paradigms. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 10–18, 2010.

[2] A. Ahtiainen, S. Surakka, and M. Rahikainen. Plaggie: GNU-licensed source code plagiarism detection engine for java exercises. In *Proceedings of the 6th Baltic Sea conference on Computing education research: Koli Calling 2006*, Baltic Sea '06, pages 141–142, New York, NY, USA, 2006. ACM.

[3] G. Alexe, G. S. Dalgin, D. Scanfeld, P. Tamayo, J. P. Mesirov, C. DeLisi, L. Harris, N. Barnard, M. Martel, A. J. Levine, S. Ganesan, and G. Bhanot. High expression of lymphocyte-associated genes in node-negative HER2+ breast cancers correlates with lower recurrence rates. *Cancer Res*, 67(22):10669–10676, Nov. 2007.

[4] G. Alexe, R. Vijaya Satya, M. Seiler, D. Platt, T. Bhanot, S. Hui, M. Tanaka, A. Levine, and G. Bhanot. Pca and clustering reveal alternate mtdna phylogeny of n and m clades. *Journal of Molecular Evolution*, 67(5):465–487, 2008.

[5] A. Angelova, L. Matthies, D. M. Helmick, and P. Perona. Dimensionality reduction using automatic supervision for vision-based terrain learning. In W. Burgard, O. Brock, and C. Stachniss, editors, *Robotics: Science and Systems*. The MIT Press, 2007.

[6] W. Arlt, M. Biehl, A. E. Taylor, S. Hahner, R. Libe, B. A. Hughes, P. Schneider, D. J. Smith, H. Stiekema, N. Krone, E. Porfiri, G. Opocher, J. Bertherat, F. Mantero, B. Allolio, M. Terzolo, P. Nightingale, C. H. L. Shackleton, X. Bertagna, M. Fassnacht, and P. M. Stewart. Urine steroid metabolomics as a biomarker tool for detecting malignancy in adrenal tumors. *J. of Clinical Endocrinology and Metabolism*, 96:3775–3784, 2011.

[7] M. Aupetit. Visualizing distortions and recovering topology in continuous projection techniques. *Neurocomputing*, 70(7-9):1304 – 1330, 2007.

[8] D. Bacciu, A. Micheli, and A. Sperduti. Modeling bi-directional tree contexts by generative transductions. In C. Loo, K. Yap, K. Wong, A. Teoh, and K. Huang, editors, *Neural Information Processing - 21st International Conference, ICONIP 2014, Kuching, Malaysia, November 3-6, 2014. Proceedings, Part I*, volume 8834 of *Lecture Notes in Computer Science*, pages 543–550. Springer International Publishing, 2014.

[9] A. Backhaus and U. Seiffert. Classification in high-dimensional spectral data: Accuracy vs. interpretability vs. model size. *Neurocomputing*, 131:15–22, 2014.

[10] H.-U. Bauer, K. Pawelzik, and T. Geisel. A topographic product for the optimization of self-organizing feature maps. In *NIPS*, pages 1141–1147, 1991.

[11] A. Bellet, A. Habrard, and M. Sebban. A survey on metric learning for feature vectors and structured data. *CoRR*, abs/1306.6709, 2013.

[12] M. Bernard, L. Boyer, A. Habrard, and M. Sebban. Learning probabilistic models of tree edit distance. *Pattern Recognition*, 41(8):2611–2629, 2008.

[13] M. Biehl, K. Bunte, and P. Schneider. Analysis of flow cytometry data by matrix relevance learning vector quantization. *Plos One*, 8(3), 2013.

[14] M. Biehl, A. Ghosh, and B. Hammer. Dynamics and generalization ability of LVQ algorithms. *Journal of Machine Learning Research*, 8:323–360, 2007.

[15] M. Biehl, B. Hammer, E. Merényi, A. Sperduti, and T. Villman. Learning in the context of very high dimensional data (Dagstuhl Seminar 11341). *Dagstuhl Reports*, 1(8):67–95, 2011.

[16] C. M. Bishop. *Pattern Recognition and Machine Learning*. Information science and statistics. Springer, 2006.

[17] C. M. Bishop, M. Svensén, and C. K. I. Williams. GTM: The generative topographic mapping. *Neural Computation*, 10:215–234, 1998.

[18] B. Boeckmann, A. Bairoch, R. Apweiler, M.-C. Blatter, A. Estreicher, E. Gasteiger, M. J. Martin, K. Michoud, C. O'Donovan, I. Phan, S. Pilbout, and M. Schneider. The swiss-prot protein knowledgebase and its supplement trembl in 2003. *Nucleic Acids Research*, 31(1):365–370, 2003.

[19] R. Boulet, B. Jouve, F. Rossi, and N. Villa. Batch kernel SOM and related Laplacian methods for social network analysis. *Neurocomputing*, 71(7-9):1257–1273, Mar. 2008.

[20] L. Boyer, Y. Esposito, A. Habrard, J. Oncina, and M. Sebban. Sedil: Software for edit distance learning. *Lect. Notes Comput. Sci.*, 5212 LNAI(2):672–677, 2008.

[21] M. Brehmer, M. Sedlmair, S. Ingram, and T. Munzner. Visualizing dimensionally-reduced data: Interviews with analysts and a characterization of task sequences. In *Proceedings of the Fifth Workshop on Beyond Time and Errors: Novel Evaluation Methods for Visualization*, BELIV '14, pages 1–8, New York, NY, USA, 2014. ACM.

[22] S. Bull and R. Vatrapu. *Supporting Collaborative Interaction with Open Learner Models: Existing Approaches and Open Questions*, volume 2, pages 761–765. ISLS: International Society of the Learning Sciences, 2011.

[23] K. Bunte, M. Biehl, and B. Hammer. A general framework for dimensionality-reducing data visualization mapping. *Neural Computation*, 24(3):771–804, 2012.

[24] K. Bunte, P. Schneider, B. Hammer, F.-M. Schleif, T. Villmann, and M. Biehl. Limited rank matrix learning, discriminative dimension reduction and visualization. *Neural Networks*, 26:159–173, 2012.

[25] S.-H. Cha. Comprehensive survey on distance/similarity measures between probability density functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, 1(4):300–307, 2007.

[26] A. B. Chan, N. Vasconcelos, and G. R. G. Lanckriet. Direct convex relaxations of sparse svm. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 145–153, New York, NY, USA, 2007. ACM.

[27] L. Chen and A. Buja. Local multidimensional scaling for nonlinear dimension reduction, graph drawing, and proximity analysis. *Journal of the American Statistical Association*, 104(485):209–219, 2009.

[28] Y. Chen, E. K. Garcia, M. R. Gupta, A. Rahimi, and L. Cazzanti. Similarity-based classification: Concepts and algorithms. *J. Mach. Learn. Res.*, 10:747–776, June 2009.

[29] A. T. Corbett, K. R. Koedinger, and J. R. Anderson. Intelligent tutoring systems. In M. G. Helander, T. K. Landauer, and P. Prabhu, editors, *Handbook of Human-Computer Interaction*, pages 859–874. The Netherlands: Elsevier Science, 1997.

[30] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sept. 1995.

[31] K. Crammer, R. Gilad-Bachrach, A. Navot, and N. Tishby. Margin analysis of the LVQ algorithm. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15, NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada]*, pages 462–469. MIT Press, 2002.

[32] S. Das, L. Trutoiu, A. Murai, D. Alcindor, M. Oh, F. De la Torre, and J. Hodgins. Quantitative measurement of motor symptoms in Parkinson's disease: A study with full-body motion capture data. In *Engineering in Medicine and Biology Society,EMBC, 2011 Annual International Conference of the IEEE*, pages 6789–6792, Sept. 2011.

[33] A. Denecke, H. Wersing, J. J. Steil, and E. Körner. Online figure-ground segmentation with adaptive metrics in generalized lvq. *Neurocomputing*, 72(7-9):1470–1482, Mar. 2009.

[34] R. Douglas and T. Sejnowski. Future challenges for the science and engineering of learning - final NSF workshop report. NSF workshop report, National Science Foundation, 2008.

[35] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification.* Wiley, New York, 2. edition, 2001.

[36] C. D. Epp, S. Bull, and M. D. Johnson. Visualising uncertainty for open learner model users. In I. Cantador, M. Chi, R. Farzan, and R. Jäschke, editors, *Proceedings of the 22nd Conference on User Modeling, Adaptation, and Personalization (UMAP2014), Aalborg, Denmark, July 7-11, 2014.*, volume 1181 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2014.

[37] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta. A survey of kernel and spectral methods for clustering. *Pattern recognition*, 41(1):176–190, 2008.

[38] S. France and D. Carroll. Development of an agreement metric based upon the rand index for the evaluation of dimensionality reduction techniques, with applications to mapping customer data. In *Proceedings of MLDM '07*, pages 499–517, Berlin, Heidelberg, 2007. Springer-Verlag.

[39] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315:972–976, 2007.

[40] S. Gagula-Palalic and M. Can. An organized committee of artificial neural networks in the classification of human chromosomes. *International Journal of Computer Applications*, 80(8):38–41, 2013.

[41] T. Gärtner. A survey of kernels for structured data. *SIGKDD Explor. Newsl.*, 5(1):49–58, July 2003.

[42] E. Gasteiger, A. Gattiker, C. Hoogland, I. Ivanyi, R. D. Appel, and A. Bairoch. ExPASy: The proteomics server for in-depth protein knowledge and analysis. *Nucleic acids research*, 31(13):3784–3788, July 2003.

[43] A. Gisbrecht and B. Hammer. Relevance learning in generative topographic mapping. *Neurocomput.*, 74:1351–1358, April 2011.

[44] A. Gorban and A. Zinovyev. Principal manifolds and graphs in practice: from molecular biology to dynamical systems. *Int. J. Neural Syst.*, 20(3):219–232, 2010.

[45] T. Graepel and K. Obermayer. A stochastic self-organizing map for proximity data. *Neural Computation*, 11(1):139–155, Jan. 1999.

[46] S. Gross, X. Zhu, B. Hammer, and N. Pinkwart. Cluster based feedback provision strategies in intelligent tutoring systems. In *Proceedings of the 11th international conference on Intelligent Tutoring Systems*, ITS'12, pages 699–700, Berlin, Heidelberg, 2012. Springer-Verlag.

[47] B. Haasdonk and C. Bahlmann. Learning with distance substitution kernels. In C. E. Rasmussen, H. H. Bülthoff, B. Schölkopf, and M. A. Giese, editors, *Pattern Recognition, 26th DAGM Symposium, August 30 - September 1, 2004, Tübingen, Germany, Proceedings*, volume 3175 of *Lecture Notes in Computer Science*, pages 220–227. Springer, 2004.

[48] A. Habrard, J. Iñesta, D. Rizo, and M. Sebban. Melody recognition with learned edit distances. *Lect. Notes Comput. Sci.*, 5342 LNCS:86–96, 2008.

[49] B. Hammer. *Compositionality in Neural Systems*, pages 244–248. Handbook of Brain Theory and Neural Networks. MIT Press, 2002.

[50] B. Hammer and A. Hasenfuss. Topographic mapping of large dissimilarity data sets. *Neural Computation*, 22(9):2229–2284, 2010.

[51] B. Hammer, D. Hofmann, F.-M. Schleif, and X. Zhu. Learning vector quantization for (dis-)similarities. *Neurocomputing*, 131:43–51, 2014.

[52] B. Hammer and B. J. Jain. Neural methods for non-standard data. In *ESANN 2004, 12th European Symposium on Artificial Neural Networks, Bruges, Belgium, April 28-30, 2004, Proceedings*, pages 281–292, 2004.

[53] B. Hammer, A. Micheli, A. Sperduti, and M. Strickert. A general framework for unsupervised processing of structured data. *Neurocomputing*, 57:3–35, 2004.

[54] B. Hammer and T. Villmann. Generalized relevance learning vector quantization. *Neural Netw.*, 15:1059–1068, October 2002.

[55] R. W. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal, The*, 29(2):147–160, April 1950.

[56] R. J. Hathaway and J. C. Bezdek. Nerf c-means: Non-euclidean relational fuzzy clustering. *Pattern Recognition*, 27(3):429–437, 1994.

[57] S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919, 1992.

[58] G. E. Hinton and S. T. Roweis. Stochastic neighbor embedding. In *Advances in Neural Information Processing Systems 15*, pages 833–840. MIT Press, 2003.

[59] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313:504 – 507, 2006.

[60] D. Hofmann, A. Gisbrecht, and B. Hammer. Efficient approximations of robust soft learning vector quantization for non-vectorial data. *Neurocomputing*, 147:96–106, 2015.

[61] P. J. Ingram, M. P. Stumpf, and J. Stark. Network motifs: structure does not determine function. *BMC genomics*, 7(1):108, 2006.

[62] P. Jaccard. The distribution of the flora in the alpine zone. *New Phytologist*, 11(2):37–50, 1912.

[63] Y. Jin and B. Hammer. Computational intelligence in big data [guest editorial]. *IEEE Comp. Int. Mag.*, 9(3):12–13, 2014.

[64] A. Juan and E. Vidal. On the use of normalized edit distances and an efficient k-nn search technique (k-aesa) for fast and accurate string classification. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 2, pages 676–679 vol.2, 2000.

[65] M. Kästner, B. Hammer, M. Biehl, and T. Villmann. Functional relevance learning in generalized learning vector quantization. *Neurocomput.*, 90:85–95, Aug. 2012.

[66] M. Kästner, D. Nebel, M. Riedel, M. Biehl, and T. Villmann. Differentiable kernels in generalized matrix learning vector quantization. In *ICMLA, p. 132-137*, 2012.

[67] D. A. Keim, F. Mansmann, J. Schneidewind, J. Thomas, and H. Ziegler. Visual analytics: Scope and challenges. In S. Simoff, M. H. Boehlen, and A. Mazeika, editors, *Visual Data Mining: Theory, Techniques and Tools for Visual Analytics*. Springer, 2008. Lecture Notes in Computer Science (LNCS).

[68] D. A. Keim, F. Rossi, T. Seidl, M. Verleysen, and S. Wrobel. Information Visualization, Visual Data Mining and Machine Learning (Dagstuhl Seminar 12081). *Dagstuhl Reports*, 2(2):58–83, 2012.

[69] T. C. Kietzmann, S. Lange, and M. Riedmiller. Incremental GRLVQ: Learning relevant features for 3d object recognition. *Neurocomput.*, 71(13-15):2868–2879, Aug. 2008.

[70] T. C. Kietzmann, S. Lange, and M. A. Riedmiller. Computational object recognition: a biologically motivated approach. *Biological Cybernetics*, 100(1):59–79, 2009.

[71] S. Kirstein, A. Denecke, S. Hasler, H. Wersing, H.-M. Gross, and E. Körner. A vision architecture for unconstrained and incremental learning of multiple categories. *Memetic Computing*, 1(4):291–304, 2009.

[72] S. Kirstein, H. Wersing, H.-M. Gross, and E. Körner. A life-long learning vector quantization approach for interactive learning of multiple categories. *Neural Networks*, 28:90–105, 2012.

[73] T. Kohonen. *Self-organizing maps*. Springer, 1995.

[74] T. Kohonen and P. Somervuo. How to make large self-organizing maps for nonvectorial data. *Neural Networks*, 15(8-9):945–952, 2002.

[75] B. Kulis. Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4):287–364, 2013.

[76] J. Laub, V. Roth, J. M. Buhmann, and K.-R. Müller. On the information and representation of non-euclidean pairwise data. *Pattern Recogn.*, 39(10):1815–1826, Oct. 2006.

[77] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[78] J. A. Lee, D. H. Peluffo-Ordóñez, and M. Verleysen. Multiscale stochastic neighbor embedding: Towards parameter-free dimensionality reduction. In M. Verleysen, editor, *22th European Symposium on Artificial Neural Networks, ESANN 2014, Bruges, Belgium, April 23-25, 2014*. i6doc.com, 2014.

[79] J. A. Lee, E. Renard, G. Bernard, P. Dupont, and M. Verleysen. Type 1 and 2 mixtures of Kullback-Leibler divergences as cost functions in dimensionality reduction based on similarity preservation. *Neurocomput.*, 112:92–108, July 2013.

[80] J. A. Lee and M. Verleysen. *Nonlinear dimensionality reduction*. Springer, 2007.

[81] J. A. Lee and M. Verleysen. Quality assessment of dimensionality reduction: Rank-based criteria. *Neurocomput.*, 72(7-9):1431–1443, 2009.

[82] J. A. Lee and M. Verleysen. Scale-independent quality criteria for dimensionality reduction. *Pattern Recognition Letters*, 31:2248–2257, October 2010.

[83] A. Lehrmann, M. Huber, A. C. Polatkan, A. Pritzkau, and K. Nieselt. Visualizing dimensionality reduction of systems biology data. *Data Mining and Knowledge Discovery*, 27(1):146–165, 2013.

[84] C. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: A string kernel for SVM protein classification. In *Proceedings of the Pacific Symposium on Biocomputing*, volume 7, pages 566–575, 2002.

[85] J. Lewis, M. Ackerman, and V. D. Sa. Human cluster evaluation and formal quality measures. In *Proc. of the 34th Annual Conference of the Cognitive Science Society*, 2012.

[86] J. M. Lewis, V. R. de Sa, and L. van der Maaten. Divvy: Fast and intuitive exploratory data analysis. *Journal of Machine Learning Research*, 14:3159–3163, 2013.

[87] M. Li, X. Chen, X. Li, B. Ma, and P. M. B. Vitányi. The similarity metric. In *IEEE Transactions on Information Theory*, pages 863–872, 2003.

[88] M. Li and P. M. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer, 3 edition, 2008.

[89] H. Liu, D. Song, S. Rüger, R. Hu, and V. Uren. Comparing dissimilarity measures for content-based image retrieval. In *Proceedings of the 4th Asia information retrieval conference on Information retrieval technology*, AIRS'08, pages 44–50, Berlin, Heidelberg, 2008. Springer-Verlag.

[90] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *J. Mach. Learn. Res.*, 2:419–444, Mar. 2002.

[91] C. Lundsteen, J. Phillip, and E. Granum. Quantitative analysis of 6985 digitized trypsin G-banded human metaphase chromosomes. *Clin. Genet.*, 18:355–370, 1980.

[92] A. Mabbott and S. Bull. Alternative views on knowledge: Presentation of open learner models. In J. C. Lester, R. M. Vicari, and F. Paraguacu, editors, *Intelligent Tutoring Systems, 7th International Conference, ITS 2004, Maceio, Alagoas, Brazil, August 30 - September 3, 2004, Proceedings*, volume 3220 of *Lecture Notes in Computer Science*, pages 689–698. Springer, 2004.

[93] D. MacDonald and C. Fyfe. The kernel self-organising map. In R. J. Howlett and L. C. Jain, editors, *Fourth International Conference on Knowledge-Based Intelligent Information Engineering Systems & Allied Technologies, KES 2000, Brighton, UK, 30 August - 1 September 2000, Proceedings, 2 Volumes*, pages 317–320. IEEE, 2000.

[94] T. Maier, S. Klebel, U. Renner, and M. Kostrzewa. Fast and reliable MALDI-TOF MS-based microorganism identification. *Nature Methods*, 3, 2006.

[95] D. Malerba, F. Esposito, V. Gioviale, and V. Tamma. Comparing dissimilarity measures for symbolic data analysis. In *Pre-proceedings of the ETK-NTTS 2001 Conference, Hersonissos (Crete), 18-22 June*, pages 473–481, 2001.

[96] M. Martin-Merino and A. Munoz. Extending the SOM algorithm to non-euclidean distances via the kernel trick. In N. Pal, N. Kasabov, R. Mudi, S. Pal, and S. Parui, editors, *Neural Information Processing*, volume 3316 of *Lecture Notes in Computer Science*, pages 150–157. Springer Berlin Heidelberg, 2004.

[97] T. M. Martinetz and K. J. Schulten. A "neural gas" network learns topologies. In T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, editors, *Proceedings of the International Conference on Artificial Neural Networks 1991, Espoo, Finland*, pages 397–402. Amsterdam; New York: North-Holland, 1991.

[98] G. D. S. Martino, N. Navarin, and A. Sperduti. A memory efficient graph kernel. In *The 2012 International Joint Conference on Neural Networks (IJCNN), Brisbane, Australia, June 10-15, 2012*, pages 1–7. IEEE, 2012.

[99] G. D. S. Martino and A. Sperduti. Mining structured data. *IEEE Comp. Int. Mag.*, 5(1):42–49, 2010.

[100] H. T. Mevissen and M. Vingron. Quantifying the Local Reliability of a Sequence Alignment. *Protein Engineering*, 9(2):127–132, 1996.

[101] M. Müller. *Information retrieval for music and motion*. Springer, 2007.

[102] S. A. Nene, S. K. Nayar, and H. Murase. Columbia Object Image Library (COIL-20). Technical report, Department of Computer Science, Columbia University, Feb 1996.

[103] M. Neuhaus and H. Bunke. *Bridging the Gap Between Graph Edit Distance and Kernel Machines*. Series in machine perception and artificial intelligence. World Scientific, 2007.

[104] V.-H. Nguyen, F. Merienne, and J.-L. Martinez. An efficient approach for human motion data mining based on curves matching. In L. Bolc, R. Tadeusiewicz, L. Chmielewski, and K. Wojciechowski, editors, *Computer Vision and Graphics*, volume 6374 of *Lecture Notes in Computer Science*, pages 163–184. Springer Berlin Heidelberg, 2010.

[105] M. Pechenizkiy, A. Tsymbal, and S. Puuronen. Local dimensionality reduction within natural clusters for medical data analysis. In *Computer-Based Medical Systems, 2005. Proceedings. 18th IEEE Symposium on*, pages 365–370, June 2005.

[106] E. Pekalska and B. Duin. *The Dissimilarity Representation for Pattern Recognition. Foundations and Applications*. World Scientific, 2005.

[107] O. Penner, P. Grassberger, and M. Paczuski. Sequence alignment, mutual information, and dissimilarity measures for constructing phylogenies. *PloS One*, 6(1), 2011.

[108] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[109] A. K. Qinand and P. N. Suganthan. Kernel neural gas algorithms with application to cluster analysis. In *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 4 - Volume 04*, ICPR '04, pages 617–620, Washington, DC, USA, 2004. IEEE Computer Society.

[110] C. E. Rasmussen and C. K. Williams. *Gaussian Processes for Machine Learning*. Adaptative computation and machine learning series. University Press Group Limited, 2006.

[111] S. Robertson. Understanding inverse document frequency: On theoretical arguments for IDF. *Journal of Documentation*, 60(5):503–520, 2004.

[112] F. Rossi. How many dissimilarity/kernel self organizing map variants do we need? In T. Villmann, F.-M. Schleif, M. Kaden, and M. Lange, editors, *Advances in Self-Organizing Maps and Learning Vector Quantization (Proceedings of the 10th International Workshop on Self Organizing Maps, WSSOM 2014)*, volume 295 of *Advances in Intelligent Systems and Computing*, pages 3–23, Mittweida (Germany), 7 2014. Springer International Publishing.

[113] Y. Rubner, C. Tomasi, and L. J. Guibas. A metric for distributions with applications to image databases. In *Proceedings of the Sixth International Conference on Computer Vision*, ICCV '98, pages 59–, Washington, DC, USA, 1998. IEEE Computer Society.

[114] A. Sato and K. Yamada. Generalized learning vector quantization. In D. S. Touretzky, M. Mozer, and M. E. Hasselmo, editors, *NIPS*, pages 423–429. MIT Press, 1995.

[115] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[116] F.-M. Schleif and A. Gisbrecht. Data analysis of (non-)metric proximities at linear costs. In E. R. Hancock and M. Pelillo, editors, *SIMBAD*, volume 7953 of *Lecture Notes in Computer Science*, pages 59–74. Springer, 2013.

[117] F.-M. Schleif, B. Hammer, M. Kostrzewa, and T. Villmann. Exploration of mass-spectrometric data in clinical proteomics using learning vector quantization methods. *Briefings in Bioinformatics*, 9(2):129–143, 2008.

[118] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[119] P. Schneider, M. Biehl, and B. Hammer. Distance learning in discriminative vector quantization. *Neural Computation*, 21:2942–2969, 2009.

[120] B. Schölkopf, A. J. Smola, and K. R. Müller. Kernel principal component analysis. *Advances in kernel methods: support vector learning*, pages 327–352, 1999.

[121] S. Seo and K. Obermayer. Soft learning vector quantization. *Neural Computation*, 15(7):1589–1604, 2003.

[122] V. Sperschneider. *Bioinformatics*. Springer, 2008.

[123] M. Strickert, B. Hammer, T. Villmann, and M. Biehl. Regularization and improved interpretation of linear data mappings and adaptive distance measures. In *IEEE SSCI CIDM 2013*, pages 10–17. IEEE Computational Intelligence Society, 2013.

[124] J. Sun, C. Fyfe, and M. Crowe. Extending sammon mapping with bregman divergences. *Inf. Sci.*, 187:72–92, Mar. 2012.

[125] A. Takasu, D. Fukagawa, and T. Akutsu. Statistical learning algorithm for tree similarity. In *IEEE Int. Conf. on Data Mining, ICDM*, pages 667–672, 2007.

[126] P. Tiňo, A. Kabán, and Y. Sun. A generative probabilistic approach to visualizing sets of symbolic sequences. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 701–706, New York, NY, USA, 2004. ACM.

[127] M. E. Tipping. Sparse bayesian learning and the relevance vector machine. *J. Mach. Learn. Res.*, 1:211–244, Sept. 2001.

[128] W. Torgerson. *Theory and methods of scaling*. Wiley, 1958.

[129] A. Ultsch and H. Siemon. Kohonen's self organizing feature maps for exploratory data analysis. In *Proceedings of INNC'90*, pages 305–308. Kluwer, 1990.

[130] L. van der Maaten. Accelerating t-SNE using tree-based algorithms. *Journal of Machine Learning Research*, 15:3221–3245, 2014.

[131] L. van der Maaten and G. Hinton. Visualizing high-dimensional data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, November 2008.

[132] L. van der Maaten, E. Postma, and H. van den Herik. Dimensionality reduction: A comparative review. Technical report, Tilburg University Technical Report, TiCC-TR 2009-005, 2009.

[133] K. Vanlehn. The behavior of tutoring systems. *International Journal of Artificial Intelligence in Education*, 16:227–265, August 2006.

[134] J. Venna and S. Kaski. Local multidimensional scaling. *Neural Netw.*, 19:889–899, 2006.

[135] J. Venna, J. Peltonen, K. Nybo, H. Aidos, and S. Kaski. Information retrieval perspective to nonlinear dimensionality reduction for data visualization. *J. Mach. Learn. Res.*, 11:451–490, 2010.

[136] E. Wenger. *Artificial intelligence and tutoring systems: computational and cognitive approaches to the communication of knowledge*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.

[137] C. K. I. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA*, pages 682–688. MIT Press, 2000.

[138] M. Wise. Running Karp-Rabin matching and greedy string tiling. Technical report, University of Sydney, Basser Dept. of Computer Science, 1993.

[139] H. Yin. On the equivalence between kernel self-organising maps and self-organising mixture density networks. *Neural Netw.*, 19(6):780–784, July 2006.

[140] Y. Zhai, Y.-S. Ong, and I. Tsang. The emerging "big dimensionality". *Computational Intelligence Magazine, IEEE*, 9(3):14–26, Aug 2014.

[141] K. Zhang and J. T. Kwok. Clustered Nyström method for large scale manifold learning and dimension reduction. *IEEE Transactions on Neural Networks*, pages 1576–1587, 2010.

[142] X. Zhu. *Adaptive prototype-based dissimilarity learning*. PhD thesis, Bielefeld University, Faculty of Technology, Bielefeld, Germany, 2015.

[143] X. Zhu, A. Gisbrecht, F.-M. Schleif, and B. Hammer. Approximation techniques for clustering dissimilarity data. *Neurocomputing*, 90:72–84, 2012.