Adaptive prototype-based dissimilarity learning

Xibin Zhu

Dissertation

Cognitive Interaction Technology Center of Excellence (CITEC)
Theoretical Computer Science (TCS) &
Research Institute for Cognition and Robotics (Cor-Lab)
Bielefeld University
Germany

October 2014

Vorgelegt zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften
Technische Fakultät, Universität Bielefeld

Gedruckt auf alterungsbeständigem Papier nach DIN-ISO 9706

Abstract

In this thesis we focus on prototype-based learning techniques, namely three unsupervised techniques: *generative topographic mapping* (GTM), *neural gas* (NG) and *affinity propagation* (AP), and two supervised techniques: *generalized learning vector quantization* (GLVQ) and *robust soft learning vector quantization* (RSLVQ). We extend their abilities with respect to the following central aspects:

- **Applicability on dissimilarity data**: Due to the increased complexity of data, in many cases data are only available in form of (dis)similarities which describe the relations between objects. Classical methods can not directly deal with this kind of data. For unsupervised methods this problem has been studied, here we transfer the same idea to the two supervised prototype-based techniques such that they can directly deal with dissimilarities without an explicit embedding into a vector space.
- **Quadratic complexity issue**: For dealing with dissimilarity data, due to the need of the full dissimilarity matrix, the complexity becomes quadratic which is infeasible for large data sets. In this thesis we investigate two linear approximation techniques: *Nyström approximation* and *patch processing*, and integrate them into unsupervised and supervised prototype-based techniques.
- **Reliability of prototype-based classifiers**: In practical applications, a reliability measure is beneficial for evaluating the classification quality expected by the end users. Here we adopt concepts from *conformal prediction* (CP), which provides point-wise confidence measure of the prediction, and we combine those with supervised prototype-based techniques.
- **Model complexity**: By means of the confidence values provided by CP, the model complexity can be automatically adjusted by adding new prototypes to cover low confidence data space.
- **Extendability to semi-supervised problems**: Besides its ability to evaluate a classifier, conformal prediction can also be considered as a classifier. This opens a way that supervised techniques can be easily extended for semi-supervised settings by means of a *self-training* approach.

Acknowledgements

Here, I would like to thank my supervisor Barbara Hammer for helping me and guiding me during my Ph.D journey, as well as all my colleagues (alphabetical order w.r.t. the first names :-)) Andrej Gisbrecht, Alexander Schulz, Bassam Mokbel, Daniela Hofmann and Frank-Michael Schleif. Especially, Dr. Frank-Michael Schleif who shared the office with me in the past over 3 years, thanks for hearing and answering (sometimes very native) questions.

Special thank to my parents for supporting me all the time, to my lovely wife Jingjing for understanding and encouraging me, as well as to our cute little daughter for bringing more 'fun' to my life. ;-)

Financially, I appreciate the CITEC GradSchool and DFG under grant number HA2719/4-1 for funding me at the early phase of my Ph.D, and later on as a part of its-owl project (subproject self-optimization), I also appreciate the German Federal Ministry of Education and Research (BMBF) within the Leading-Edge Cluster Competition and the Project Management Agency Karlsruhe (PTKA) by which the its-owl project is funded and managed.

Notations

n	the size of the training data set
m	the dimensionality of data space
$\mathbb{R}^m$	$m-$dimensional data space
$\mathbf{x}_i \in \mathbb{R}^m$	i-te data point in a m-dimensional data space $\mathbb{R}^m$
$\mathbf{w}_j \in \mathbb{R}^m$	j-te prototype in a m-dimensional space $\mathbb{R}^m$
$R(\mathbf{w}_j)$	the receptive field of the prototype $\mathbf{w}_j$
$\mathbb{L} := \{1, \ldots, L\}$	set of a finite number of different labels/classes
$\mathbb{X}$	training data set
$\mathbb{W}$	set of prototypes
K	the number of prototypes
L	the number of labels
M	patch size
α_{ji}	coefficient of the data point $\mathbf{x}_i$ representing prototype $\mathbf{w}_j$ in a linear combination
m_i	multiplicity of the data point $\mathbf{x}_i$
A	non-conformity measure
a_i^y	non-conformity value of data point $\mathbf{x}_i$ with respect to label y
$\mathcal{A}_T^{\mathbb{L}}$	set of non-conformity values of elements in T for all labels in $\mathbb{L}$
$\mathcal{P}_T^{\mathbb{L}}$	set of p-values of elements in T for all labels in $\mathbb{L}$
cf_i	confidence value of $\mathbf{x}_i$
cr_i	credibility value of $\mathbf{x}_i$
$\mathcal{CF}_T$	set of confidence values of elements in T
$\mathcal{CR}_T$	set of credibility values of elements in T
$\mathbf{1}$	a vector of all ones
$\mathbf{0}$	a vector of zeros or the origin in a space
I	identity matrix

Abbreviations

i.i.d.	independent and identically distributed
psd	positive semi-definite

Abbreviations of algorithms

GLVQ	Generalized Learning Vector Quantization
RSLVQ	Robust Soft Learning Vector Quantization
AP	Affinity Propagation
GTM	Generative Topographic Mapping
NG	Neural Gas
RGLVQ	Relational Generalized Learning Vector Quantization
RRSLVQ	Relational Robust Soft Learning Vector Quantization
RGTM	Relational Generative Topographic Mapping
RNG	Relational Neural Gas
PRGLVQ	Patch Relational GLVQ
PRGTM	Patch Relational GTM
PRNG	Patch Relational NG
PAP	Patch Affinity Propagation
AC-RGLVQ	Adaptive Conformal RGLVQ
S3-C-RGLVQ	Secure Semi-Supervised Conformal RGLVQ

Contents

Chapter 1

Introduction

In almost every field from industry to academia the size of data is continuously increasing. Machine learning techniques are playing a very important and irreplaceable role, in this context: they revolutionized the possibility to deal with large and complex data sets by offering powerful tools to automatically extract a regularity from given data. Albeit techniques such as the support vector machine (SVM) or Gaussian processes provide efficient state-of-the-art techniques with excellent classification ability, it is often not easy to manually inspect the way in which decisions are taken by the model. Hence, it is hardly possible to visualize its decisions to domain experts in such a way that the results can be interpreted, and relevant information can be inferred based thereon. In contrast, prototype-based methods provide an intuitive working style and interpretability of the final model in the sense that they represent their decisions in terms of typical representatives contained in the input space. Since prototypes can be directly inspected by humans in the same way as data points, an intuitive access to the decision becomes possible: the responsible prototype and its (dis)similarities to the given data determine the output.

There are different ways to learn prototypes from the data: Unsupervised techniques such as k-means [43], neural gas [63], or self organizing map [53] and its statistical counterparts such as generative topographic mapping [9] infer prototypes based on input data only. Supervised techniques take also class label information into account and find decision boundaries according to the label information. Learning vector quantization (LVQ) [53] and its cost function based variants [81, 90] constitute popular techniques in this context. In this thesis we will address several prototype-based techniques: neural gas (NG) [63], generative topographic mapping (GTM) [9] and affinity propagation (AP) [24] for the unsupervised case, and generalized LVQ (GLVQ) [81] and robust soft LVQ (RSLVQ) [90] for supervised problems.

Besides the size of data continuously getting larger, rapid developments in modern sensor technologies, dedicated data formats, and data storage continue to pose new challenges to this field: Data are often not given as vectors for which the features are determined by domain experts and coded as numerical values, rather, they are characterized by more complex structures and a problem-specific (dis)similarity measure for which only the relations between data points are defined. Examples include biological

sequences alignment techniques such as FASTA [59], BLAST [2], mass spectra, metabolic networks, where, e.g. complex alignment techniques, background information or general information theoretical principles drive the comparison of data points [74, 61, 49]. Some other (dis-)similarity measures are, for example, the travel time from one place to another, edit distances, tangent distance as popular in computer vision [19], shape matching distance [6], etc. or the widely used cosine similarity between term frequency-inverse document frequency (tf-idf) in document classification. In such cases, the data are only characterized by means of pairwise (dis)similarities without an explicit vectorial representation.

To deal with this kind of data, dissimilarity- or similarity-based machine learning techniques have been proposed, such as nearest neighbor classifiers which 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 point. Because of this simplicity, (dis)similarity techniques enjoy a large popularity in application domains, whereby the methods range from simple k-nearest neighbor classifiers up to advanced techniques such as affinity propagation which represents a clustering in terms of typical exemplars [53, 24]. With respect to kernel methods such as SVM, similarity-based learning has been deeply investigated such as [12]. This work makes a comprehensive study in terms of different preprocessing techniques to turn an arbitrary similarity matrix to a valid kernel matrix and vectorial representation together with linear and Gaussian kernel, respectively, used in conjunction with a standard SVM.

(Dis-)similarity-based techniques can be distinguished according to different criteria: (i) The number of data points used to represent the classifier ranging from dense models such as k-nearest neighbor to sparse representations such as prototype-based methods. To arrive at easily interpretable models, a sparse representation in terms of few data points is necessary. (ii) The degree of supervision ranging from clustering techniques such as AP up to supervised learning. In this thesis, we will deal with both, supervised and unsupervised techniques for dissimilarity data. (iii) The complexity of the dissimilarity measure the methods can deal with ranging from vectorial techniques restricted to Euclidean spaces, adaptive techniques which learn the underlying metrics, up to tools which can deal with arbitrary (dis)similarities [86, 77]. Typically, Euclidean techniques are well suited for simple classification scenarios, but they fail if high-dimensionality or complex structures are encountered. For unsupervised cases, the approaches [38, 29] have investigated how to deal with arbitrary (probably non-Euclidean) dissimilarity data by means of prototype-based clustering techniques. In this thesis we follow the concept of these approaches, and extend it to supervised prototype-based learning methods, especially two cost function based LVQs, generalized LVQ (GLVQ) and the statistical counterpart, robust soft LVQ (RSLVQ).

The extensions of prototype-based learning techniques for dissimilarity data open a way to more broadly apply prototype-based methods on modern applications such as the bioinformatics domain where sometimes only the relations between objects, e.g. protein sequences, can be obtained by some sophisticated measuring techniques. However, one main drawback of dissimilarity learning is its quadratic complexity due to the dependency on the dissimilarity matrix, the computation of the dissimilarity often constituting the

bottleneck in real-life applications. By integrating approximation techniques such as Nyström approximation [106] the effort can be reduced to linear time. In this thesis except Nyström approximation we also investigate another intuitive approach, namely *patch processing*, which process the data iteratively instead of taking the whole data matrix, to achieve linear complexity, and we investigate both techniques based on non-Euclidean data sets from the bioinformatics domain.

Although prototype-based learning techniques have many advantages in terms of interpretability and its intuitive work style, they have generally two limitations. First, the complexity of the model, i.e. the number of prototypes, has to be defined in advance. This is impossible if the prior knowledge of the data is unknown. Some approaches have been published to automatically adjust the model complexity [14, 94, 52], but most of them are heuristic based and not in a statistical sense. Besides, in many practical applications, especially for life science, medical applications, not only the learning accuracy is important, but also some kind of reliability measure, similar to p- or q-value from statistics. This would provide more information about the prediction besides the predicted class label. Only few attempts exist to enhance prototype methods by reliability estimates or rejection options based thereon [14, 94, 23]. In this thesis we adopt the concept of *conformal prediction* (CP) [105] which has a solid statistical foundation and provides confidence value about the classification. It can be considered as a reliability measure and integrated into prototype-based classifiers. By means of the confidence values we can identify situations when the model complexity is not sufficiently large to describe the data. In this case, a more complex model is expected and the model will be adjusted accordingly, resulting a self-adaptive version thereof while providing a confidence value for each prediction.

The reliability measure should give more insides about the resulting predictions to find where the predictions have to be paid more attention due to the lower reliability level. By means of the confidence value of conformal prediction we can not only adjust the model complexity on the one hand, but on the other hand, we can easily extend prototype-based classifiers to semi-supervised tasks which will also discussed in this thesis.

Table 1.1 summarizes the main contributions of this thesis (marked by $\surd$): In Chapter 2 we will briefly review five prototype-based methods, namely, three unsupervised methods: Neural Gas (NG), Generative Topographical Mapping (GTM) and Affinity Propagation (AP), and two supervised techniques, Generalized Learning Vector Quantization (GLVQ) and Robust Soft LVQ (RSLVQ). In Chapter 3 we discuss the theoretical background of a dissimilarity representation of data highlighting their properties, i.e. metric and Euclidean, correction techniques making them Euclidean and the ways to deal with dissimilarities. After this foundational part, we will move to the parts of the thesis which present novel algorithmic approaches as extensions of prototype techniques for non-vectorial data, its efficiency and reliability. In Chapter 4 we introduce extensions of prototype-based classifiers to deal with dissimilarity data in a implicit way without explicit embedding into some vector space. Due to the quadratic complexity for dissimilarity-based learning, in Chapter 5 we propose a very intuitive method, i.e. patch processing, for large scale problems and compare it with the *Nyström* approximation technique. Chapter 6 mainly focuses on introducing the concept of conformal

Chapter		***Supervised***		***Unsupervised***		
		GLVQ	*RSLVQ*	*NG*	*AP*	*GTM*
2	*cost function*	margin	likelihood ratio	topographical QE	QE	log-likelihood
	complexity	$\mathcal{O}(n)$	$\mathcal{O}(n)$	online: $\mathcal{O}(n)$ batch: $\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
	data type	vectorial	vectorial	vectorial	vectorial	vectorial
	working style	online	online	batch/online	online	batch
4	*relational data*	$\surd$	$\surd$	[37]	similarity	[29]
5	*patch*	$\surd$	analog	[37]	$\surd$	$\surd$
	nyström	$\surd$	analog	$\surd$	-	[28]
6	*adaptive*	$\surd$	analog	-	-	-
7	*semi-super.*	$\surd$	analog	-	-	-

Table 1.1: Tabular summary of this thesis

prediction and shows how to combine conformal prediction with relational GLVQ to get a self-adaptive extension with respect to model complexity by means of point-wise confidence values provided by conformal prediction. In Chapter 7 we will extend conformal prediction to relational GLVQ for semi-supervised problems resulting in a self-adaptive semi-supervised version thereof. In the last chapter we conclude this thesis and discuss possible future work.

While pursuing the PhD, I had the opportunity to present parts of the work to an international audience in the following journal and conference publications:

Publication list

Journal papers

- Xibin Zhu, Frank-Michael Schleif, and Barbara Hammer. *Adaptive conformal semi-supervised learning vector quantization for dissimilarity data.* Pattern Recognition Letters, 49:138-145, 2014.

- Xibin Zhu, Andrej Gisbrecht, Frank-Michael Schleif, and Barbara Hammer. *Approximation techniques for clustering dissimilarity data.* Neurocomputing, 90:72-84, 2012.

- Frank-Michael Schleif, Xibin Zhu, and Barbara Hammer. *Sparse conformal prediction for dissimilarity data.* Annals of Mathematics and Artificial Intelligence (AMAI), 2014.
  Doi:http://dx.doi.org/10.1007/s10472-014-9402-1

- Barbara Hammer, Daniela Hofmann, Frank-Michael Schleif, and Xibin Zhu. *Learning vector quantization for (dis-)similarities.* Neurocomputing, 131:43-51, 2014.
  Doi: http://dx.doi.org/10.1016/j.neucom.2013.05.054

- 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.

Conference and workshop papers

- Xibin Zhu, Frank-Michael Schleif, and Barbara Hammer. *Secure semi-supervised vector quantization for dissimilarity data.* In I. Rojas, G. Joya, and J. Cabestany, editors, 12th International Work-Conference on Artificial Neural Networks (IWANN), Part I, volume 7902 of LNCS, pages 347-356. Springer, 2013.

- Xibin Zhu, Frank-Michael Schleif, and Barbara Hammer. *Semi-supervised vector quantization for proximity data.* In Michel Verleysen, editor, 21st European Symposium on Artificial Neural Networks (ESANN), pages 89-94. Ciaco-i6doc.com, 2013.

- Xibin Zhu, Frank-Michael Schleif, and Barbara Hammer. *Patch processing for relational learning vector quantization.* In Jun Wang, Gary G. Yen, and Marios M. Polycarpou, editors, Advances in Neural Networks - 9th International Symposium on Neural Networks (ISNN), volume 7367 of Lecture Notes in Computer Science, pages 55-63. Springer, 2012.

- Xibin Zhu and Barbara Hammer. *Patch affinity propagation.* In Michel Verleysen, editor, 19th European Symposium on Artificial Neural Networks (ESANN), pages 99-104. Ciaco-i6doc.com, 2011.

- Xibin Zhu, Frank-Michael Schleif, and Barbara Hammer. *Relational extensions of learning vector quantization.* Workshop: New Challenges in Neural Computation (NC^2) in German Conference on Pattern Recognition (GCPR), 2011

- Frank-Michael Schleif, Xibin Zhu, and Barbara Hammer. *A conformal classifier for dissimilarity data.* In Lazaros S. Iliadis, Ilias Maglogiannis, Harris Papadopoulos, Kostas Karatzas, and Spyros Sioutas, editors, Artificial Intelligence Applications and Innovations (AIAI) (2), volume 382 of IFIP Advances in Information and Communication Technology, pages 234-243. Springer, 2012.

- Frank-Michael Schleif, Xibin Zhu, and Barbara Hammer. *Sparse prototype representation by core sets.* In et.al Hujun Yin, editor, Intelligent Data Engineering and Automated Learning (IDEAL), 2013.

- Frank-Michael Schleif, Xibin Zhu, and Barbara Hammer. *Soft competitive learning for large data sets.* In Mykola Pechenizkiy and Marek Wojciechowski, editors, New Trends in Databases and Information Systems, volume 185 of Advances in Intelligent Systems and Computing, pages 141-151. Springer Berlin Heidelberg, 2013.

- Frank-Michael Schleif, Xibin Zhu and Barbara Hammer. *Prior knowledge for core vector data description.* Workshop: New Challenges in Neural Computation (NC^2) in German Conference on Pattern Recognition (GCPR), 2014

- Sebastian Gross, Xibin Zhu, Barbara Hammer, and Niels Pinkwart. *Cluster based feedback provision strategies in intelligent tutoring systems.* In Stefano A. Cerri, William J. Clancey, Giorgos Papadourakis, and Kitty Panourgia, editors, Intelligent Tutoring Systems - 11th International Conference (ITS 2012), volume 7315 of Lecture Notes in Computer Science, pages 699-700. Springer, 2012.

- Frank-Michael Schleif, Thomas Villmann, and Xibin Zhu. *High dimensional matrix relevance learning.* IEEE International Conference on Data Mining series (ICDM), Workshop on High Dimensional Data Mining (HDM), 2014. (Accepted)

- Barbara Hammer, Bassam Mokbel, Frank-Michael Schleif, and Xibin Zhu. *White box classification of dissimilarity data.* In Emilio Corchado, Vaclav Snasel, Ajith Abraham, Michal Wozniak, Manuel Grana, and Sung Bae Cho, editors, Hybrid Artificial Intelligent Systems (HAIS)(1) - 7th International Conference, volume 7208 of Lecture Notes in Computer Science, pages 309-321. Springer, 2012.

- Barbara Hammer, Bassam Mokbel, Frank-Michael Schleif, and Xibin Zhu. *Prototype-based classification of dissimilarity data.* In Joao Gama, Elizabeth Bradley, and Jaakko Hollmen, editors, Advances in Intelligent Data Analysis (IDA) - 10th International Symposium, volume 7014 of Lecture Notes in Computer Science, pages 185-197. Springer, 2011.

- Barbara Hammer, Andrej Gisbrecht, Alexander Hasenfuss, Bassam Mokbel, Frank-Michael Schleif, and Xibin Zhu. *Topographic mapping of dissimilarity data.* In Jorma Laaksonen and Timo Honkela, editors, Advances in Self-Organizing Maps - 8th International Workshop (WSOM), volume 6731 of Lecture Notes in Computer Science, pages 1-15. Springer, 2011.

- Andrej Gisbrecht, Frank-Michael Schleif, Xibin Zhu, and Barbara Hammer. *Linear time heuristics for topographic mapping of dissimilarity data.* In Hujun Yin, Wenjia Wang, and Victor J. Rayward-Smith, editors, Intelligent Data Engineering and Automated Learning (IDEAL), volume 6936 of Lecture Notes in Computer Science, pages 25-33. Springer, 2011.

- Andrej Gisbrecht, Barbara Hammer, Frank-Michael Schleif, and Xibin Zhu. *Accelerating kernel clustering for biomedical data analysis.* In Clare Bates Congdon, Steven M. Corns, and Jennifer A. Smith, editors, Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB), pages 154-161. IEEE, 2011.

Chapter 2

Prototype-based learning

Chapter overview

Prototype-based learning constitutes a very intuitive machine learning technique to help people to analyze the data. In this chapter we shortly review five popular prototype-based learning techniques, namely, two supervised methods: generalized learning vector quantization (GLVQ), robust soft LVQ (RSLVQ), and three unsupervised methods: neural gas (NG), affinity propagation (AP) and generative topographic mapping (GTM), which serve as basis methods in this thesis.

2.1 Introduction

Unlike many black-box algorithms in machine learning, prototype-based learning techniques offer an intuitive interface to given data sets since prototypes can be directly inspected by experts. The basic idea of prototype based learning is representing the given data by prototypes in the same space. The number of prototypes is significantly smaller than the size of data which results in a compact representation of the data but while keeping the statistic of the data as much as possible.

There exist different possibilities to infer appropriate prototypes from data: Unsupervised leaning techniques such as simple k-means [43], fuzzy-k-means [7], topographic mapping, neural gas (NG)[63], or the self-organizing map (SOM) [53], and its statistical counterparts such as the generative topographic mapping (GTM) [9] infer prototypes based on input data only. Supervised techniques incorporate the information of class labels and find decision boundaries which describe priorly known class labels, one of the most popular learning algorithm in this context being learning vector quantization (LVQ). Albeit original LVQ has been introduced on somewhat heuristics grounds [53], extensions thereof are derived from explicit const functions or statistical models [81, 90]. In addition, recent developments of prototype-based models provide a solid mathematical derivation of generalization ability and learning dynamics: explicit large margin generalization bounds of LVQ classifiers are available [16, 86]; 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, [81, 90]. Interestingly, already the dynamics of simple LVQ as proposed by Kohonen provably

leads to surprisingly good generalization curves when investigated in the framework of the theory of online learning [8].

Besides these mathematical derivations, these learning algorithms share several fundamental aspects: they represent data in a sparse way by means of prototypes, they form decisions based on the (dis)similarity of data to prototypes, and training is often very intuitive based on Hebbian principles [45]. Further, prototypes offer a compact and efficient representation of the important aspects of given data which very naturally allows to extend the basic algorithms into an incremental life-long learning paradigm, treating prototypes as a compact representation of all already seen data. This aspect has been used in diverse scenarios which deal with incremental settings or very large data sets [18, 52, 1].

In this chapter we review five prototype-based learning techniques, i.e. three unsupervised techniques, namely Generative Topographic Mapping (GTM) [9] as a probabilistic counterpart of self organizing maps (SOM), Neural Gas (NG) [63] which incorporates neighborhood information to quantization error, and Affinity Propagation (AP) [24] which directly optimizes the quantization error and in which prototypes are exemplars from given data, and two supervised techniques, namely two cost function based LVQ variants, Generalized LVQ [81], probably the most popular cost function based LVQ, and Robust Soft LVQ [90], a probabilistic version by means of mixture of Gaussians.

2.2 Prototype-based clustering

The aim of clustering is to decompose data into subsets, such that the data within the clusters are as similar as possible, while the different clusters are well separated. Generally, for unsupervised learning, no additional class information of the data is available. The unsupervised prototype-based methods aim to optimize the quantization error, which results in prototype positions which are representative for the underlying data distribution. Given a set of n data points $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$, clustering divides this set into K groups. Usually, the number of clusters K is fixed and predefined by the user. As an example of prototype-based clustering, we consider k-means clustering, one of the most popular clustering methods. It depends on the assumption that data are given in some Euclidean space e.g. $\mathbf{x}_i \in \mathbb{R}^m$, and it builds clusters represented by prototypes $\mathbf{w}_j \in \mathbb{R}^m$ which define their receptive fields by

$$R(\mathbf{w}_j) := \{\mathbf{x}_i \in \mathbb{R}^m \mid d(\mathbf{x}_i, \mathbf{w}_j) \leq d(\mathbf{x}_i, \mathbf{w}_k), \forall k \neq j\}\,.$$

That means, the receptive field of prototype $\mathbf{w}_j$ consists of all data points $\mathbf{x}_i$ which are closest to $\mathbf{w}_j$ as measured by the squared Euclidean distance $d(\mathbf{x}_i, \mathbf{w}_j) = \|\mathbf{x}_i - \mathbf{w}_j\|^2$, breaking ties deterministically. This example explains the basic idea of prototype-based learning which represent clusters in terms of prototypical representatives. The goal of clustering can be formalized as minimizing the *quantization error*

$$E_{\mathrm{QE}} := \sum_{i,j:\mathbf{x}_i \in R_j} d(\mathbf{x}_i, \mathbf{w}_j)$$

to obtain prototypes which are as representative for their receptive field as possible. There exist different classical methods which achieve this goal: a direct optimization by

means of a gradient descent as present in online vector quantization, or more advanced methods which take a neighorhood structure into account or which rely on a probabilistic interpretation of the model [53, 24, 9]. The latter techniques often yield much more stable results. Further, they possibly provide additional information such as the ability to visualize the prototypes such as in the generative topographic mapping or a neighborhood structure of the clusters such as in neural gas. We will consider three typical clustering techniques in this context:

(I) The generative topographic mapping (GTM) which constitutes a generative statistical model. It models data by means of a constraint mixture of Gaussians induced by a mapping from a low-dimensional latent space. In the latent space visualization is possible.

(II) The neural gas (NG) which models data by means of representative prototypes which represent data in relation to the rank of its distance. This way a very robust algorithm is obtained which is widely independent from scaling issues.

(III) Affinity propagation (AP) which reformulates the quantization error as a likelihood function. This can be decomposed as factor graph for which the max-sum algorithm can be used [24].

We shortly describe these three models in the following. Thereby, we refer to distances such as the Euclidean distance, or similarities such as the Euclidean dot product as required. Similarities and dissimilarities can be transferred into each other using classical double centering (see e.g. [70] or section 3).

2.2.1 Generative topographic mapping

The GTM has been proposed in [9] as a probabilistic counterpart to self-organizing maps (SOM) [53]. It models given data $\mathbf{x} \in \mathbb{R}^m$ by a constrained mixture of Gaussians induced by a low dimensional latent space. More precisely, in GTM, lattice points $\mathbf{u}_j \in \mathbb{R}^d$ ($d \ll m$) on a regular lattice in a low dimensional latent space are given. These are mapped to prototypes in the data space

$$\mathbf{w}_j = \Phi(\mathbf{u}_j) \cdot W \tag{2.1}$$

by means of a generalized linear regression model. In principle, the base functions Φ could be chosen as nonlinear functions such that their linear combination has sufficient flexibility to map the low dimensional lattice to appropriate positions. At the same time, the base functions control the degree of topology preservation by the stiffness of (2.1). This is usually obtained by picking only a small number of base functions or by a strong regularization of $\mathbf{w}$. In practice, the base functions Φ are often chosen as equally spaced Gaussian functions. W denotes the weight parameters of the mapping which are determined during learning.

This mapping induces a constrained mixture of Gaussians in the data space in the following way. Every prototype $\mathbf{w}_j$ induces a Gaussian which variance is determined by

the parameter β

$$p(\mathbf{x}|\mathbf{w}_j, \beta) = \left(\frac{\beta}{2\pi}\right)^{m/2} \cdot \exp\left(-\frac{\beta}{2} d(\mathbf{x}, \mathbf{w}_j)^2\right) . \tag{2.2}$$

These Gaussians are combined in a mixture model with equal prior. Training optimizes the data log-likelihood with respect to W and β. This way the parameters W and β are determined. It is possible to derive an expectation maximization (EM) algorithm as detailed in [9]. It in turn optimizes the *responsibilities* of component j for data point i

$$R_{ij}(W, \beta) = \frac{p(\mathbf{x}_i|\mathbf{w}_j, \beta)}{\sum_{j'} p(\mathbf{x}_i|\mathbf{w}_{j'}, \beta)} , \tag{2.3}$$

and the parameters W and β by solving the linear equations

$$\mathbf{\Phi}^T \mathbf{G}_{\text{old}} \mathbf{\Phi} W_{\mathbf{new}}^T = \mathbf{\Phi}^T \mathbf{R}_{\text{old}} \mathbf{V} , \tag{2.4}$$

$$\frac{1}{\beta_{\text{new}}} = \frac{1}{nm} \cdot \sum_{i,j} R_{ij}(W_{\text{old}}, \beta_{\text{old}}) \cdot d(\mathbf{x}_i, \Phi(\mathbf{u}_j) \cdot W_{\text{new}}) . \tag{2.5}$$

n denotes the number of data points, $\mathbf{G}$ is the diagonal matrix with entries $G_{ii} = \sum_j R_{ij}(W, \beta)$, $\mathbf{R}$ is the matrix of responsibilities, $\mathbf{\Phi}$ is the matrix of base functions evaluated at all lattice points, and $\mathbf{V}$ is the matrix of data points. GTM is initialized by aligning the lattice and the first two data principal components.

2.2.2 Neural gas

Neural gas (NG) extends the quantization error to incorporate neighborhood cooperation:

$$E_{\text{NG}} := \sum_{ij} h_\sigma(k_{ij}) d(\mathbf{x}_i, \mathbf{w}_j) , \tag{2.6}$$

where $h_\sigma(t) = \exp(-t/\sigma)$ exponentially scales the neighborhood range. k_{ij} denotes the rank of prototype $\mathbf{w}_j$ with respect to $\mathbf{x}_i$, i.e. the number of prototypes $\mathbf{w}_k$ with $k \neq j$ which are closer to $\mathbf{x}_i$ as measured by the Euclidean distance d. Classical NG optimizes this objective by means of a stochastic gradient descent. The neighborhood range σ is annealed during training such that, in the limit, the standard quantization error is approximated [63]. There exists a faster batch optimization scheme as introduced in [15] which in turn optimizes prototype locations and assignments similar to an EM scheme:

$$\text{determine } k_{ij} \text{ based on } d(\mathbf{x}_i, \mathbf{w}_j) , \tag{2.7}$$

$$\text{set } \mathbf{w}_j := \frac{\sum_i h_\sigma(k_{ij}) \mathbf{x}_i}{\sum_i h_\sigma(k_{ij})} . \tag{2.8}$$

These steps are iterated until convergence.

2.2.3 Affinity propagation

Affinity propagation (AP) [24] constitutes an exemplar based clustering method, i.e. prototype locations are restricted to data points $\mathbf{w}_j \in \{\mathbf{x}_i \mid i = 1, \ldots, n\}$. Further, it deals with similarities $s(\mathbf{x}_i, \mathbf{w}_j)$ rather than dissimilarities such as the Euclidean dot product, for example. Obviously, the quantization error can be formulated accordingly.

Since prototypes are located at discrete positions, the quantization error can no longer be optimized by means of gradient techniques. Therefore, the quantization error is first rephrased as

$$E_{\text{QE\_AP}} := \sum_i s(\mathbf{x}_i, \mathbf{x}_{I(i)}) + \sum_i \delta_i(I)\,. \tag{2.9}$$

Note that there is no longer a fixed number of clusters K given. Rather, cluster assignments are defined by means of a function $I : \{1, \ldots, n\} \to \{1, \ldots, n\}$. Every data point picks a prototype by means of this function. Since prototypes are exemplars contained in the data set itself, this can be written as $\mathbf{x}_{I(i)}$ for data point $\mathbf{x}_i$. $\delta_i(I)$ punishes invalid assignments, i.e.

$$\delta_i(I) = \begin{cases} -\infty & \text{if } \exists j\, I(j) = i, I(i) \neq i\,, \\ 0 & \text{otherwise.} \end{cases}$$

During training, these assignments have to be adapted; basically the data points have to negotiate to arrive at a valid assignment function which optimizes the cost function.

The number of clusters K is no longer specified a priori. A trivial valid solution of the cost function would be given by the identity $I(i) = i$, i.e. every data point forms an exemplar. To avoid this trivial solution, costs have to be introduced as soon as a data point becomes an exemplar. This can be achieved by adjusting the self-similarities $s(\mathbf{x}_i, \mathbf{x}_i)$, indicating the *preferences* of data point $\mathbf{x}_i$ becoming an exemplar. In the limit, for low preferences, only one cluster will be found, and for high preferences, every data point will be an exemplar. In between, different numbers of clusters can be reached. Typically, there are two different strategies to set the self-similarities appropriately: either they are set to a reasonable fixed value such as the median of the given similarities. Alternatively, binary search can take place until a desired number of clusters is reached. We will use the latter strategy in this thesis for fair comparisons with other methods.

To solve this novel optimization problem (Eq. 2.9), the function $E_{\text{EQ\_AP}}$ is modeled as factor graph. This can be optimized by means of the max-sum algorithm. In turn, two kinds of messages between the data points are exchanged during the training, namely *responsibilities*

$$r_{ij} := s(\mathbf{x}_i, \mathbf{x}_j) - \max_{j' \neq j} \{a_{ij'} + s(\mathbf{x}_i, \mathbf{x}_{j'})\}$$

and *availabilities*

$$\begin{aligned} a_{ij} &:= \min\{0, r_{jj} + \sum_{i' \neq i,j} \max\{0, r_{i'j}\}\}\,, \\ a_{ii} &:= \sum_{i' \neq i} \max\{0, r_{i'i}\} \end{aligned}$$

which can be combined at any stage of the training to decide the assignments

$$I(i) = \text{argmax}_j(a_{ij} + r_{ij})$$

as detailed in [24]. Since the factor graph is cyclic, there is no guarantee to obtain the global optimum or even convergence. For this reason, small random values are added to the similarities in every run to avoid cycles. This way, usually, convergence to a fixed point is observed.

Note that AP does not rely on Euclidean similarities. Rather the cost function (2.9) and the corresponding optimization is valid for every general similarity measure s due to the restriction of prototypes to exemplars. In [24], this fact has been demonstrated by various applications involving microarray data and text data, for example. AP has the drawback that additional functionality such as neighborhood preservation of the clusters is not available.

2.3 Prototype-based classification

For the supervised case, label information about the data is available. The goal of prototype-based classification techniques is to find prototypes which represent a given data set as accurately as possible by means of the label information. For supervised learning, data $\mathbf{x}_i$ are equipped with class labels $c(\mathbf{x}_i) \in \mathbb{L} = \{1, \ldots, L\}$. Similarly, every prototype is equipped with a priorly fixed label $c(\mathbf{w}_j)$. A data point is classified according to the class of its closest prototype. The classification error of this mapping is given by the term

$$\sum_{j=1}^{K} \sum_{\mathbf{x}_i \in R(\mathbf{w}_j)} \delta(c(\mathbf{x}_i) \neq c(\mathbf{w}_j))$$

with the Kronecker delta function δ. This cost function cannot easily be optimized explicitly due to vanishing gradients and discontinuities. Therefore, different ways have to be considered.

Learning vector quantization (LVQ) proposed by Kohonen constitutes a very popular class of intuitive prototype-based learning algorithms with successful applications ranging from telecommunications to robotics [53]. The idea is that iteratively positioning prototypes in the data space during training such that the given data can be represented as accurately as possible. In this way the prototypes can be considered as a sparse representation of original data. To classify an unknown data point, the new sample is assigned to the class label of the closest prototype, so called *nearest prototype classification* (NPC) or *winner takes all* (WTA) in terms of prototypes.

Basic algorithms as proposed in [53] include LVQ1 which is directly based on Hebbian learning [45] to learn a set of prototypes from training data: given $\mathbf{x}$, its nearest prototype is adapted only. The adaptation depends on the classification: If the prototype has the same label as $\mathbf{x}$, it will be attracted by $\mathbf{x}$, otherwise it will be repelled. Some improvements exist such as LVQ2.1, LVQ3, or OLVQ which aim at a higher convergence speed or better approximation of the Bayesian borders. For example, LVQ2.1 introduces a "window" condition which is defined around the midplane of two nearest prototypes according to a data point $\mathbf{x}$ with different labels in the sense that one of them has the same label as $\mathbf{x}$ and the anther has different label. If the data point $\mathbf{x}$ falls into the "window", these two nearest prototypes are updated by attractive (for the one with the same label) and

repulsive (for the other with different label) forces from this data point. The LVQ2.1 is based on the idea of shifting the decision boundaries toward the Bayes limits with a attractive and repulsive forces from $\mathbf{x}$. However, no attention is paid to what might happen to the location of the prototype, so the prototypes diverge in the long run. LVQ3 has been proposed to ensure that the prototypes continue approximating the class distribution, but it must be noted that if only one reference vector is assigned to each class, LVQ3 is the same as LVQ2.1, and the problem of divergence remains unsolved.

These types of LVQ schemes have in common that their learning rule is essentially heuristically motivated and a valid cost function does not exist. One of the first attempts to derive LVQ from a cost function can be found in [81] with an exact computation of the validity at class boundaries in [87]. Later, a very elegant LVQ scheme [90] which is based on a probabilistic model and can be seen as a more robust probabilistic extension of LVQ2.1. We will briefly review these two cost function based LVQs.

2.3.1 Generalized learning vector quantization

Generalized LVQ [81] is one of the most popular cost function based LVQs which is derived from a cost function which can be related to the generalization ability of LVQ classifiers [86]:

$$E_{\mathrm{GLVQ}} = \sum_{i=1}^{n} \Phi \left(\frac{d(\mathbf{x}_i, \mathbf{w}_+(\mathbf{x}_i)) - d(\mathbf{x}_i, \mathbf{w}_-(\mathbf{x}_i))}{d(\mathbf{x}_i, \mathbf{w}_+(\mathbf{x}_i)) + d(\mathbf{x}_i, \mathbf{w}_-(\mathbf{x}_i))} \right) \tag{2.10}$$

where Φ is a differentiable monotonic function such as the hyperbolic tangent, and $\mathbf{w}_+(\mathbf{x}_i)$ refers to the prototype closest to $\mathbf{x}_i$ with the same label as $\mathbf{x}_i$, $\mathbf{w}_-(\mathbf{x}_i)$ refers to the closest prototype with a different label. This way, for every data point, its contribution to the cost function is small if and only if the distance to the closest prototype with a correct label is smaller than the distance to a wrongly labeled prototype, resulting in a correct classification of the point.

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

$$\begin{aligned} \Delta\mathbf{w}_+(\mathbf{x}_i) &\sim -\Phi'(\mu(\mathbf{x}_i)) \cdot \mu_+(\mathbf{x}_i) \cdot \nabla_{\mathbf{w}_+(\mathbf{x}_i)} d(\mathbf{x}_i, \mathbf{w}_+(\mathbf{x}_i)) \\ \Delta\mathbf{w}_-(\mathbf{x}_i) &\sim \Phi'(\mu(\mathbf{x}_i)) \cdot \mu_-(\mathbf{x}_i) \cdot \nabla_{\mathbf{w}_-(\mathbf{x}_i)} d(\mathbf{x}_i, \mathbf{w}_-(\mathbf{x}_i)) \end{aligned}$$

where

$$\mu(\mathbf{x}_i) = \frac{d(\mathbf{x}_i, \mathbf{w}_+(\mathbf{x}_i)) - d(\mathbf{x}_i, \mathbf{w}_-(\mathbf{x}_i))}{d(\mathbf{x}_i, \mathbf{w}_+(\mathbf{x}_i)) + d(\mathbf{x}_i, \mathbf{w}_-(\mathbf{x}_i))} ,$$

$$\mu_+(\mathbf{x}_i) = \frac{2 \cdot d(\mathbf{x}_i, \mathbf{w}_-(\mathbf{x}_i))}{(d(\mathbf{x}_i, \mathbf{w}_+(\mathbf{x}_i)) + d(\mathbf{x}_i, \mathbf{w}_-(\mathbf{x}_i))^2} ,$$

$$\mu_-(\mathbf{x}_i) = \frac{2 \cdot d(\mathbf{x}_i, \mathbf{w}_+(\mathbf{x}_i)}{(d(\mathbf{x}_i, \mathbf{w}_+(\mathbf{x}_i)) + d(\mathbf{x}_i, \mathbf{w}_-(\mathbf{x}_i))^2} .$$

For the squared Euclidean norm, 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, i.e. they adapt the closest correct prototypes towards a given data point, while pushing the closest incorrect prototype away from the data point.

A detailed derivation of the update rules can be found in [81] and as also pointed out by Sato and Yamada GLVQ guarantees convergence. Classification of a novel data point takes place by a WTA scheme or NPC as mentioned before:

$$\mathbf{x} \mapsto c(\mathbf{w}_j) \text{ where } d(\mathbf{x}, \mathbf{w}_j) \text{ is minimum} \tag{2.11}$$

with squared Euclidean distance. Next, we will introduce another version of LVQ which is derived based on a labeled Gaussian mixture model.

2.3.2 Robust soft learning vector quantization

Although GLVQ is a popular cost function based LVQ variant, its cost function is still somewhat heuristically motivated. Another cost function based LVQ was proposed by [90], Robust Soft LVQ (RSLVQ), which is based on a statistical modeling of the data, i.e. the probability density of unknown distribution of the data is described by a Gaussian mixture model. The probability density of the data set is given by a mixture model

$$p(\mathbf{x}|\mathbb{W}) = \sum_{y=1}^{L} \sum_{\{j:c(\mathbf{w}_j)=y\}}^{K} p(\mathbf{x}|j)p(j) \,,$$

where $\mathbf{x} \in \mathbb{R}^m$ and $\mathbb{W} := \{\mathbf{w}_1, \ldots, \mathbf{w}_K\}$ is a set of labeled prototypes. $p(j)$ is the prior probability that data points are generated by a particular component j. $p(\mathbf{x}|j)$ is the conditional probability that the component j generates data point $\mathbf{x}$. It is a function of prototype $\mathbf{w}_j$, which can be considered as the representative for all data points generated by $\mathbf{w}_j$ and later assigned to $\mathbf{w}_j$. It is chosen as Gaussian

$$p(\mathbf{x}|j) = (2\pi\sigma_j^2)^{-m/2} \cdot \exp\Big(\frac{-d(\mathbf{x}, \mathbf{w}_j)}{\sigma_j^2}\Big).$$

Since classification depends on only the distances between data points and prototypes, one often assumes every component has the same width and strength $\sigma_j^2 = \sigma^2$ and prior probability $p(j) = \frac{1}{K}$.

For a data point $\mathbf{x}$ with true label y, the following probability densities are considered

$$p(\mathbf{x}, y|\mathbb{W}) = \sum_{\{j:c(\mathbf{w}_j)=y\}} p(\mathbf{x}|j)p(j)$$

$$p(\mathbf{x}, \bar{y}|\mathbb{W}) = \sum_{\{j:c(\mathbf{w}_j)\neq y\}} p(\mathbf{x}|j)p(j)$$

$p(\mathbf{x}, y|\mathbb{W})$ is the probability density that a data point $\mathbf{x}$ is generated by a mixture model for the "correct" label. $p(\mathbf{x}, \bar{y}|\mathbb{W})$ is the probability density that a data point $\mathbf{x}$ is generated by a mixture model for the "incorrect" label. Therefore the likelihood ratio L_r constitutes

$$L_r = \prod_{i=1}^{n} \frac{p(\mathbf{x}_i, y_i|\mathbb{W})}{p(\mathbf{x}_i, y_i|\mathbb{W}) + p(\mathbf{x}_i, \bar{y}_i|\mathbb{W})} = \prod_{i=1}^{n} \frac{p(\mathbf{x}_i, y_i|\mathbb{W})}{p(\mathbf{x}_i|W)}.$$

The RSLVQ aims at optimizing the logarithm of the likelihood ratio L_r

$$E_{\text{RSLVQ}} = \log(Lr) = \sum_{i=1}^{n} \log \frac{p(\mathbf{x}_i, y_i|W)}{p(\mathbf{x}_i|W)} \tag{2.12}$$

Using stochastic gradient ascent, the learning rule can be obtained, given data point $\mathbf{x}_i$ with label y_i

$$\Delta\mathbf{w}_j \sim \begin{cases} (P_{y_i}(j|\mathbf{x}_i) - P(j|\mathbf{x}_i)) \cdot \frac{\partial d(\mathbf{x}_i, \mathbf{w}_j)}{\partial \mathbf{w}_j} & \text{if } c(\mathbf{w}_j) = y_i \\ -P(j|\mathbf{x}_i) \cdot \frac{\partial d(\mathbf{x}_i, \mathbf{w}_j)}{\partial \mathbf{w}_j} & \text{if } c(\mathbf{w}_j) \neq y_i \end{cases} \tag{2.13}$$

with the probabilities

$$P_{y_i}(j|\mathbf{x}_i) = \frac{\exp(\frac{-d(\mathbf{x}_i, \mathbf{w}_j)}{\sigma^2})}{\sum_{j:c(\mathbf{w}_j)=y_i} \exp(\frac{-d(\mathbf{x}_i, \mathbf{w}_j)}{\sigma^2})}$$

and

$$P(j|\mathbf{x}_i) = \frac{\exp(\frac{-d(\mathbf{x}_i, \mathbf{w}_j)}{\sigma^2})}{\sum_{j} \exp(\frac{-d(\mathbf{x}_i, \mathbf{w}_j)}{\sigma^2})}$$

Notice that the parameter σ^2, the variance of the Gaussians, is an additional hyperparameter of the algorithm and it has a crucial influence on the learning ability. It needs to be optimized, e.g. using cross-validation procedure. Further research about the parameter has been investigated [88, 89]. In the situation of vanishing bandwidth $\sigma \to 0$, RSLVQ reduces to a crisp learning from mistake algorithm: in case of erroneous classification, the two nearest prototypes are updated according to (2.13), which is a learning rule similar to LVQ2.1.

Given a novel data point $\mathbf{x}$, its class label can be determined by means of the most likely label y corresponding to a maximum value $p(y|\mathbf{x}, W) \sim p(\mathbf{x}, y|W)$. For typical settings, this rule can usually be approximated by a simple winner takes all rule as in GLVQ Eq. (2.11). It has been shown in [90], for example, that RSLVQ often yields excellent results while preserving interpretability of the model due to prototypical representatives of the classes in terms of the parameters $\mathbf{w}_j$.

2.4 Advantages of prototype-based methods

The resulting model of prototype-based methods is represented by embedded prototypes in the data space which can be inspected by humans in the same way as the input

data: prototypes show typical class or cluster representatives, and usually they are very sparse which is more beneficial since a direct interpretation of big data is even impossible. Relevance learning of prototype-based classifiers [42] reveals the importance of each input dimension of data, furthermore, matrix learning [86, 87] reveals the correlations of the input dimensions which contribute a better description of the data not only for single classes but also for the global classification. Multi-class problems can be naturally treated by LVQ without paying extra effort. Furthermore, unlike support vector machines or other neural classification techniques which suffer from black box characteristic, prototype-based classifiers offer a very intuitive way to understand the training process by means of the update rules of prototypes.

2.5 Evaluation measures

For supervised learning, the *classification accuracy* constitutes a very important evaluation measure about classification. It measures the ratio between the number of correctly classified data and the total number of data points:

$$Acc := \frac{|\{\mathbf{x}_i | l_i = y_i, i = 1, \ldots, n\}|}{n} \tag{2.14}$$

l_i denotes the predicted label of data point $\mathbf{x}_i$, y_i is the true label of $\mathbf{x}_i$ and n denotes the size of the data set.

For the unsupervised case, since label information is not available, some other evaluation measures have been used for evaluating the quality of clusters. The *quantization error* (QE) is the most common one,

$$E_{\mathrm{QE}} := \sum_{i,j:\mathbf{x}_i \in R(\mathbf{w}_j)} d(\mathbf{x}_i, \mathbf{w}_j) \tag{2.15}$$

which measures in how far the prototypes can represent the given data as measured by the averaged dissimilarity of prototypes to points in their receptive field. All clustering methods reviewed in the previous section optimize a cost function which can be related to the quantization error: GTM optimizes the data log-likelihood which, neglecting topological constraints due to the restricted form of the topographic mapping, would boil down to a mixture of Gaussians. Similarly, for a small neighborhood $\sigma \to 0$, the cost function of NG directly resembles the quantization error. AP optimizes the quantization error under the restriction that prototypes are exemplars. Therefore, the evaluation of the quantization error where the terms $d(\mathbf{x}_i, \mathbf{w}_j)$ are taken as dissimilarities, allows to directly evaluate the affect of the approximation techniques (this will be discussed in chapter 5) on an underlying cost function.

Another adopted evaluation measure for prototype-based clustering techniques is the *dual quantization error* given by

$$E_{\mathrm{dualQE}} = \sum_{i=1}^{n} \frac{1}{4 \cdot |R(\mathbf{w}_j)|} \sum_{i,i' \,:\, \mathbf{x}_i \in R(\mathbf{w}_j), \mathbf{x}_{i'} \in R(\mathbf{w}_j)} d(\mathbf{x}_i, \mathbf{x}_{i'}) \,, \tag{2.16}$$

Chapter		*Supervised*		*Unsupervised*		
		GLVQ	*RSLVQ*	*NG*	*AP*	*GTM*
2	*cost function*	margin	likelihood ratio	topographical QE	QE	log-likelihood
	complexity	$\mathcal{O}(n)$	$\mathcal{O}(n)$	online: $\mathcal{O}(n)$ batch: $\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
	data type	vectorial	vectorial	vectorial	vectorial	vectorial
	working style	online	online	batch/online	online	batch

Table 2.1: Summary of chapter 2: Prototype-based learning methods

which evaluates the quality of the decomposition of data points into clusters rather than the specific prototype locations. It is especially adopted to compare the quality of the approximation techniques of the reviewed methods, which we will discuss in chapter 5. It has been shown in [37], that the quantization error and its dual coincide if the prototypes are located at the centers of the receptive fields in the data space $\mathbf{w}_j = \sum_{i\,:\,\mathbf{x}_i \in R(\mathbf{w}_j)} \mathbf{x}_i / |R(\mathbf{w}_j)|$. Notice that in contrast to quantization error, for the dual quantization error it is not necessary to know the positions of prototypes, but only the cluster assignments, therefore it can be applied for any clustering technique.

Often, clustering or, more precisely, a prototype-based representation of data serves as a first step towards a classification of data in practical applications. If label information is available for the given training samples, prototypes can easily be assigned a label by means of a majority vote in its receptive field. In such cases, it is possible to evaluate the classification error of a given clustering by posterior labeling. Obviously, however, there is no reason to assume that the class boundaries of priorly known classes coincide with cluster boundaries. Therefore, this evaluation technique can judge the underlying clustering only to a limited degree.

2.6 Conclusions

We conclude this chapter by Table 2.1. In this chapter we gave an overview of five popular supervised and unsupervised prototype-based techniques. Table 2.1 summarizes the properties of the reviewed methods with respect to cost function, complexity, type of data that the method can deal with and its working style. From the cost function point of view we can see that GLVQ tries to maximize the margin between the two nearest prototypes with respect to the observed data point, RSLVQ optimizes the likelihood ratio between the class-specific probability density of data generated by the mixture model and the probability density of the full data, NG is based on the quantization error which also takes the local topology into account, AP reformulates the quantization error in another way that it can be modeled as factor graph and solved by the max-sum algorithm, and GTM, as a generative model, models data by a constrained mixture of Gaussians whose parameters can be optimized by the EM algorithm.

All these reviewed methods except AP are online methods and have time complexity $\mathcal{O}(n)$, n denotes the number of data points, and they can directly deal with vectorial

data. AP has the complexity $\mathcal{O}(n^2)$ because of the need of pairwise similarities of the data, and AP can indirectly deal with vectorial data by computing their pairwise negative Euclidean distances as similarities. Moreover, kernel variants of these methods have been proposed, for instances, kernel GLVQ [78, 84], kernel RSLVQ and sparse approximation techniques [39, 47], and the kernel version of NG [77]. Since AP does not constrain the similarities to be a valid kernel, kernel data can be directly dealt with by AP.

However, if data are inherently non-Euclidean, for example given in a form of a dedicated non-Euclidean dissimilarity measure such as dynamic time warping for time series, or alignment for symbolic strings [34], etc., they can not be applied. In the next chapter we will discuss this kind of data and the challenges for machine learning algorithms.

Chapter 3

Challenges of dissimilarity data

Chapter overview

Many classic machine learning techniques such as SVM are restricted to Euclidean vector spaces such that their suitability for complex non-Euclidean data sets are highly limited, as well as prototype-based techniques discussed before. In this chapter we focus on the dissimilarity representation of data in the sense that data are only given by pairwise dissimilarities between objects without an explicit vector representation. The properties of the dissimilarity representation of data and techniques to deal with non-Euclidean dissimilarity data will be discussed.

3.1 Introduction

This chapter is not about the methods, it is about the data, or the representation of data. In the classical machine learning approach, data are represented by features which are determined based on domain knowledge and encoded as numerical variables. Together, they constitutes a feature vector space, usually Euclidean, in which each object is represented as a vector of feature values. Thus, learning is inherently restricted to mathematical methods in this vector space, equipped with additional algebraic functions such as inner product, norm or distance. However, in modern applications, data are often no longer vectorial, rather, accompanied by more complex structures for which a problem specific similarity or dissimilarity measure has been designed. A simple example is the travel time from one place to another. Other examples include edit distances, in computer vision the tangent distance [19] and shape matching distance [6], etc., in bioinformatic sequences alignment techniques such as FASTA [59], BLAST [2], mass spectra, or metabolic networks, where complex alignment techniques, background information, or general information theoretical principles, for example, drive the comparison of data points [74, 61, 49], or in document classification the widely used cosine similarity between term frequency-inverse document frequency (tf-idf). In this case, it is possible to compute pairwise similarities or dissimilarities of the data rather than to arrive at an explicit vectorial representation. Classical methods such as SVM as well as prototype-based methods can not directly deal with this kind of data. Although some kernel extensions thereof have been proposed which can deal with valid kernels of the data, some preprocessing

steps have to be done beforehand to transform arbitrary (dis-)similarities into a valid kernel. Since similarity and dissimilarity can be faithfully transformed to each other [70], we use the term 'dissimilarity' throughout this thesis.

Some specified dissimilarity measure, on which dissimilarities between objects are obtained, is not necessarily a metric in the strict mathematical sense. It quantifies the dissimilarity between two objects by taking small values for two similar objects and large values for two different objects. As no vectorial representation of the objects is provided, the challenge is how to use dissimilarities in a meaningful way. [70] gives a comprehensive theoretical discussion about dissimilarity data. In this chapter we mainly follow the concept of [70] and extract the most important parts for this thesis in the following. First, we will discuss the two properties of dissimilarity data, *metric* and *Euclideaness*. Then a general embedding technique of dissimilarity data into a so-called pseudo-Euclidean space will be discussed. In the end we will discuss some correction techniques to make dissimilarity data Euclidean and the ways to deal with dissimilarities.

3.2 Properties of dissimilarity data

In a dissimilarity representation of data, data are given as pairwise dissimilarities $d(\mathbf{x}, \mathbf{y})$ between objects in some space. A dissimilarity measure d describes the commonality between two objects, namely, a smaller value indicates that the objects are more similar, a larger value indicates that the objects are more different. Before we go into the techniques dealing with dissimilarity data, some properties will be discussed. First of all, dissimilarity measures are usually defined on some space, for instance, the most commonly used and well known dissimilarity measure is the *Euclidean distance measure*,

$$d(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||_2 = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2}, \ \mathbf{x}, \mathbf{y} \in \mathbb{R}^m \tag{3.1}$$

based on a *Euclidean space* $\mathbb{R}^m$. For the sake of simplicity, two most important properties of dissimilarities are discussed in the following, namely *metric and* Euclideaness.

Definition 1 (*Metric space*)**.** A *metic space* is a pair (X, d), X is a set and d is a *metric*, i.e. a function on X, $d : X \times X \to \mathbb{R}^0_+$ which satisfies the following properties for all $\mathbf{x}, \mathbf{y}, \mathbf{z} \in X$

1. $d(\mathbf{x}, \mathbf{x}) = 0$ (*Reflexivity*)
2. $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$ (*symmetry*)
3. $d(\mathbf{x}, \mathbf{y}) = 0$ iff $\mathbf{x} = \mathbf{y}$ (*definiteness*)
4. $d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z}) \geq d(\mathbf{x}, \mathbf{z})$ (*triangle inequality*)

An example of a metric space is the Euclidean space $(\mathbb{R}^m, d)$ with the 2-norm distance d (3.1). X can also be $\mathbb{Z}^m$, $[a, b]^m$, etc.. If X is a finite set, e.g. $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, then d is specified by an $n \times n$ dissimilarity matrix $D = (d_{ij}) = (d(\mathbf{x}_i, \mathbf{x}_j)), i, j = 1, \ldots, n$. Consequently, the matrix D is nonnegative, symmetric and has a zero diagonal. Generally,

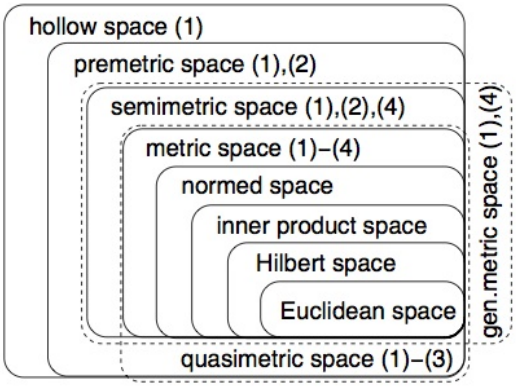


Figure 3.1: Generalized metric spaces (Source: [70])

a dissimilarity matrix usually satisfies *reflexivity*, often *positivity*, sometimes *symmetry* and rarely the *triangle inequality*.

Depending on the satisfaction of different axioms above more general spaces can be described as follows:

- *hollow space*: a space satisfying the reflexivity condition (1)
- *premetric space*: a hollow space satisfying the symmetry condition (1, 2)
- *semimetric space*: a premetric space satisfying the triangle inequality constrain (1, 2, 4)
- *quasimetric space*: a premetric space satisfying the definiteness constrain (1, 2, 3)
- *generative metric space*: a space satisfying 1 and 4 , see Figure 3.1.

Definition 2 (*Metric for dissimilarity matrix D*)**.** Let D be a symmetric dissimilarity matrix with positive off-diagonal elements. D is *metric* if the triangle inequality $d_{ij}+d_{jk} \geq d_{ik}$ holds for all triplets (i, j, k).

Note: If the triangle inequality does not hold, then the matrix D is called *quasimetric*. In this case the matrix D can be corrected to be metric by adding a constant c to D:

$$D' = D + c(\mathbf{1}\mathbf{1}^T - I), \text{ where } c \geq \max_{i,j,k} |d_{ij} + d_{ik} - d_{jk}|$$

$\mathbf{1}$ denotes a vector of all ones, I is the identity matrix. More details about the proof can be found in [70] (p. 106). c measures the '*non-metricness*'. If c is relatively small, the matrix D is only slightly non-metric. If c is large, then its non-metric properties should be considered. As mentioned by [70], in practice c is not the smallest value that can make D metric. If c has been estimated, a smaller value can be found by the bisection method.

Another important issue refers to transformations of a metric dissimilarity matrix D such that the metric properties can be preserved. If D is metric, then for $c > 0$ the

following matrices are also metric: (cd_{ij}), (d_{ij}^r) with $r \in (0,1]$, $(\log(1+d_{ij}))$, etc. See also [70] for more transformations and theoretical details. Besides the metric property another important property of dissimilarity is the "*Euclideaness*", both properties are very correlated.

Definition 3 (*Euclidean).* An $n \times n$ dissimilarity matrix $D = (d_{ij})$ is *Euclidean* if it can be embedded in a Euclidean space $(\mathbb{R}^m, d)$, where $m < n$, which means that a set of vectors $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ can be determined in $\mathbb{R}^m$ such that the dissimilarities can be preserved by Euclidean distances in $\mathbb{R}^m$, i.e. $d_{ij} = d(\mathbf{x}_i, \mathbf{x}_j) = ||\mathbf{x}_i - \mathbf{x}_j||_2$, $\forall i, j$.

There are several ways to check the Euclideaness of a given dissimilarities matrix. The most common way is: First, transform the squared dissimilarities $D^{*2} = (d_{ij}^2)$ to similarities S by means of *double centering*

$$S = -\frac{1}{2} J D^{*2} J, \text{ where } J = I - \frac{1}{n}\mathbf{1}\mathbf{1}^t, \tag{3.2}$$

then check whether the similarity matrix S is *positive semidefinite* (psd). If and only if S is psd, then the corresponding dissimilarity matrix D is Euclidean, i.e. it can be embedded into a Euclidean space. Note: If S is not psd, some correction techniques can make it psd, such as *clip*, *flip*, *shift*, which we will discuss later on. In this case, S has p positive eigenvalues, q negative eigenvalues, $(p, q, n-p-q)$ is referred to as *signature* which describes the non-Euclideaness of the corresponding dissimilarity data. Actually, this way is a special case of the following theorem to check whether a dissimilarity matrix is Euclidean:

Theorem 1. An $n \times n$ dissimilarity matrix D is *Euclidean* iff the matrix $S = -\frac{1}{2} J D^{*2} J^T$ with $D^{*2} = (d_{ij}^2)$ and $J = (I - \mathbf{1}s^T)$ is *positive semidefinite* (psd) for $s^T\mathbf{1} = 1$.

J is known as the centering matrix. Double centering (Eq. (3.2)) is a special case for $s = \frac{1}{n}$. Another special case of theorem 1 is for $s = e_i$, where $e_i \in \mathbb{R}^n$ is a standard basis vector. In [70] more theorems about checking the Euclidean behavior of matrix D are reviewed. Most of them are very closely related to Theorem 1 or some variations theirof.

Furthermore, if the similarity matrix S in Theorem 1 is a matrix of inner products (Gram matrix) of some vector representation X of n vectors $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ in a Euclidean space $\mathbb{R}^m$, usually $m \leq n$, i.e. $XX^T = S = -\frac{1}{2} J D^{*2} J^T$, then X is a vector representation of the corresponding dissimilarity D. More details will be discussed in the next section.

3.3 Linear embedding of dissimilarity data

Dissimilarity data can be embedded into a (pseudo-)Euclidean space in different ways. Beside approximate embedding techniques, we are more interested in a faithful embedding such that the distances are isometrically preserved. Since it is not alway possible to isometrically embed dissimilarities into a Euclidean space, a pseudo-Euclidean embedding will be discussed as well.

3.3.1 Euclidean embedding

For a Euclidean dissimilarity matrix D, to determine a vector representation (denoted as $X := \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$), the relation between Euclidean distances and inner products are used as mentioned before. The Gram matrix of X, $G = XX^T$, can be expressed by means of D, $G = -\frac{1}{2}JD^{*2}J$, where J is the centering matrix $J = I - \frac{1}{n}11^T \in \mathbb{R}^{n\times n}$ (Theorem 1). The eigendecomposition of G is

$$G = Q\Lambda Q^T, \tag{3.3}$$

where Λ is a diagonal matrix with nonnegative eigenvalues in descending order, and Q is the orthonormal matrix of corresponding eigenvectors. Taking k ($k \leq n$) non-zero eigenvalues, a k-dimentional vector represention X of D can be found as

$$X = Q_k\Lambda_k^{\frac{1}{2}}, \tag{3.4}$$

$\Lambda_k^{\frac{1}{2}} \in \mathbb{R}^{k\times k}$ constains the square roots of the k largest eigenvalues and $Q_k \in \mathbb{R}^{n\times k}$ is the matrix consisting of corresponding eigenvectors.

3.3.2 Pseudo-Euclidean embedding

If the dissimilarity matrix D is not Euclidean, the Euclidean space is not 'large enough' to reconstruct the dissimilarities, due to the negative eigenvalues of the corresponding Gram matrix. In this case a so called *pseudo-Euclidean space* was proposed by [30, 31], in which any premetric (only the reflexivity and symmetry properties are satisfied) finite dissimilarities can be embedded.

Definition 4 (*Pseudo-Euclidean space [70]*)**.** A pseudo-Euclidean space $\xi = \mathbb{R}^{(p,q)}$ is a real vector space equipped with a non-degenerate, indefinite inner product $\langle ., . \rangle_\xi$. ξ admits a direct orthogonal decomposition $\xi = \xi_+ \oplus \xi_-$ where $\xi_+ = \mathbb{R}^p$ and $\xi_- = \mathbb{R}^q$ and the inner product is positive definite on ξ_+ and negative definite on ξ_-. The space ξ is therefore characterized by the signature (p, q).

Pseudo-Euclidean is a more general version of Euclidean space with correction of non-Euclideaness: The first p components ($\mathbb{R}^p$) can be considered as standard Euclidean contribution whereas the next q components ($\mathbb{R}^q$) serve as a correction. Therefore, the inner product in this space becomes

$$\langle \mathbf{x}, \mathbf{y} \rangle_{pq} = x_1y_1 + \ldots + x_py_p - x_{p+1}y_{p+1} - \ldots - x_{p+q}y_{p+q}. \tag{3.5}$$

It can be negative definite due to the negative correction terms. If $q = 0$, the space is a normal Euclidean space.

To find the embedding, the same principle as in the Euclidean space can be applied. The difference is taking the negative inner product into account. Given a dissimilarity matrix D, the corresponding Gram matrix $G = -\frac{1}{2}JD^{*2}J$ in pseudo-Euclidean space becomes

$$G = X\mathcal{J}_{pq}X^T, \quad \mathcal{J}_{pq} = \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix} \tag{3.6}$$

where $\mathcal{J}_{pq}$ is a diagonal matrix with first p elements 1 and next q elements -1. Then the proper eigendecomposition of G in a pseudo-Euclidean space becomes

$$X\mathcal{J}_{pq}X^T = G = Q\Lambda Q^T = Q|\Lambda|^{\frac{1}{2}} \begin{bmatrix} \mathcal{J}_{pq} & \\ & 0 \end{bmatrix} |\Lambda|^{\frac{1}{2}} Q^T. \tag{3.7}$$

Λ is a diagonal matrix with eigenvalues and the corresponding eigenvectors are in the matrix Q. By multiplication of the term $\begin{bmatrix} \mathcal{J}_{pq} & \\ & 0 \end{bmatrix}$ and sorting, the eigenvalues in Λ can be presented in the following order: first the positive eigenvalue in deceasing order, then the negative one with decreasing magnitudes followed by zeros.

Therefore, the vector representation X of D can be determined in a pseudo-Euclidean space $\mathbb{R}^{(p,q)}$ with characteristic signature (p,q)[1] as

$$X = Q_k|\Lambda_k|^{\frac{1}{2}} \tag{3.8}$$

where $k = p+q$ and only k non-zero eigenvalues in Λ and the corresponding eigenvectors Q_k are taken into account. Otherwise, additional zero eigenvalues would describe X in a degenerate indefinite inner product space as explained in [70] (Sec. 2.7).

The square dissimilarities can be reconstructed by computing square distances in a pseudo-Euclidean space $\mathbb{R}^{(p,q)}$ which is realized by computing the square Euclidean distance in the 'positive' space $\mathbb{R}^p$ and subtracting the square Euclidean distance in the 'negative' space $\mathbb{R}^q$:

$$d^2(\mathbf{x},\mathbf{y}) = ||\mathbf{x}-\mathbf{y}||^2_{p,q} = (x_1-y_1)^2+\ldots+(x_p-y_p)^2-(x_{p+1}-y_{p+1})^2-\ldots-(x_{p+q}-y_{p+q})^2. \tag{3.9}$$

In a sense the square distances in the 'positive' space are overestimated such that the 'negative' space is applied to correct them, i.e. to make them non-Euclidean. In this work, we focus on non-negative dissimilarities which means in the resulting pseudo-Euclidean space the positive contributions dominate over the negative contributions. In [70] some examples have been mentioned that in practice many summation-based dissimilarity measures are close to Euclidean, and measures based on minimum or maximum operations may give rise to large (in magnitude) negative eigenvalues. Note that any symmetric dissimilarity matrix can be embedded into a pseudo-Euclidean space. If the matrix is asymmetric, it can be easily transformed into a symmetric one by

$$D' = \frac{1}{2}(D + D^T). \tag{3.10}$$

3.3.3 Correction of non-Euclidean dissimilarity data

The 'negative' space makes the dissimilarities non-Euclidean and the corresponding Gram matrix G indefinite and not psd. Since G has negative eigenvalues, D can not be isometrically embedded into a Euclidean space. In spite of the fact that in this case a pseudo-Euclidean embedding can be found, some correction techniques have been

[1] The signature can also be expressed as $(p,q,n-p-q)$, where $n-p-q$ denotes the number of zero eigenvalues.

proposed such that the corresponding G becomes psd and an Euclidean embedding is possible. This is especially useful when the negative eigenvalues are rather small meaning that the dissimilarities are nearly Euclidean. In such cases, the negative eigenvalues can be considered as noise or some error in the measurement. If the negative eigenvalues are relatively large (in magnitude), by neglecting them important information might be ignored, which can relegate the learning performance. Here we discuss some techniques.

Probably, the most widely used approaches are the *clip*-, *flip*- and *shift*-operation which come from kernel-based learning in order to transform non-psd similarity matrix into a valid kernel such that kernel-based learning methods, e.g. SVM, can be applied , see e.g. [12]. The *clip*-operation sets all negative eigenvalues to zero, i.e. $\Lambda_{\text{clip}} = \text{diag}(\max(\lambda_1, 0), \ldots, \max(\lambda_n, 0))$. The approach can be considered as a denoising step as mentioned in [12]. It is mathematically justified that the corrected psd matrix $G_{\text{clip}} = U^T \Lambda_{\text{clip}} U$ is the nearest psd matrix to G in terms of Frobenius norm [46]. In contrast to the consideration that the negative eigenvalues are caused by noise, some researchers show that negative eigenvalues can contain useful information for learning tasks [57, 56, 32, 73]. In order to use the negative eigenvalues, the *flip*-operation takes the absolute values, i.e. $\Lambda_{\text{flip}} = \text{diag}(|\lambda_1|, \ldots, |\lambda_n|)$, and then the corrected psd matrix becomes $G_{\text{flip}} = U^T \Lambda_{\text{flip}} U$. Another approach is *shift*-operation which increases all eigenvalues by the absolute value of the minimal eigenvalue, since $G + \lambda I = U^T(\Lambda + \lambda I)U$, any indefinite matrix G can be easily shifted by a given value λ. In this case λ is set to the absolute value of the minimum eigenvalue $|\lambda_{\min}|$ of G: $\Lambda_{\text{shift}} = \Lambda + |\lambda_{\min}| I$, and the corrected G turns to be $G_{\text{shift}} = U^T \Lambda_{\text{shift}} U$. Compared with clip and flip, the shift-operation only increases the 'self-similarities' by the amount of $|\lambda_{\min}|$ and the similarities between different objects are changed. It can be shown that for specific clustering tasks the shift-operation preserves the group structure [80]. In [12] an extensive comparison of these preprocessing methods in connection to SVM is performed for a variety of benchmarks. Interestingly, some operations such as shift do not affect the location of global optima of important cost functions such as the quantization error [57], albeit the transformation can severely affect other performance measures of different optimization algorithms [37]. The Euclidean dissimilarity matrix D can be obtained by transformation of G_*

$$D^{*2} = \mathbf{g}\mathbf{1}^T + \mathbf{1}\mathbf{g}^T - 2G_* \tag{3.11}$$

G_* is the corrected psd matrix by one of the above operations and $\mathbf{g}$ is the diagonal of G_*.

Other techniques are reviewed in [70]. For example, taking only p positive eigenvalues into account, i.e. $X = Q_p \Lambda_p^{1/2}$, results in a p-dimensional Euclidean vector representation. This might be a reasonable approach if the negative eigenvalues are relatively small with respect to positive ones or caused noise such as direct measurement of the dissimilarities. The same as clip-operation, the noise can be ignored which might purify the data. Other techniques are based on finding some extreme eigenvalue and add it to the original dissimilarity matrix: For instance, find $\tau \geq -\lambda_{\min}$, where $\lambda_{\min}$ is the smallest negative eigenvalue of G, then by adding τ to D^{*2} in such a way that $D_{2\tau} = [D^{*2} + 2\tau(\mathbf{1}\mathbf{1}^T - I)]^{*\frac{1}{2}}$ is Euclidean. More details and alternative approaches can be found in [70] (Theorem 3.9 and Theorem 3.19).

3.4 Ways to deal with dissimilarity data

As we have discussed a way to check the 'Euclideaness' of a given dissimilarity matrix D is as follows: If the corresponding similarity matrix S obtained by double centering (Eq. 3.2) is psd, then the dissimilarity matrix is Euclidean. This gives us the first opportunity to deal with dissimilarity data. If S is psd and it can be considered as a valid kernel matrix K, then standard kernel-based methods, e.g. SVM, can be applied. However, the cost of such transformation is $\mathcal{O}(n^3)$. If S is not psd, correction techniques in Sec. 3.3.3 can be applied to make it to be a valid kernel. The cost of such corrections is $\mathcal{O}(n^2)$. The analysis in [71] indicates that for non-Euclidean dissimilarities corrections like above should be avoided.

A further alternative is to embed the dissimilarity matrix into the so called dissimilarity space by taking all dissimilarities of a point $\mathbf{x}_i$ to all other points or a subset only, resulting in an at most n-dimensional vector representation of $\mathbf{x}_i$ [21]. Similar approach for similarities has been analyzed by [12]. However, this view is changing the original data representation and leads to a finite data space, limited by the number of samples.

As shown in Sec. 3.3.2, given a symmetric dissimilarity matrix D with zero diagonal, an embedding of D in a pseudo-Euclidean vector space determined by the eigenvector decomposition of S is always possible. A symmetric bilinear form in this space is given by $\langle \mathbf{x}, \mathbf{y} \rangle_{p,q} = \mathbf{x}^T \mathcal{J}_{pq} \mathbf{y}$ where $\mathcal{J}_{pq}$ is a diagonal matrix with p entries 1 and q entries -1. Taking the eigenvectors of the corresponding Gram matrix together with the square root of the absolute value of the eigenvalues, we obtain vectors $\mathbf{x}_i$ in pseudo-Euclidean space such that $d(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i - \mathbf{x}_j, \mathbf{x}_i - \mathbf{x}_j \rangle_{p,q}$ holds for every pair of data points. If the number of data is not limited, a generalization of this concept to Krein spaces with according decomposition is possible [70].

Vector operations can be directly transferred to the pseudo-Euclidean space, i.e. standard methods can be applied. Hence we can use prototype-based learning explicitly in pseudo-Euclidean space since it relies on vector operations only. One problem of this explicit embedding is given by the computational complexity which is $\mathcal{O}(n^3)$, and further, by the fact that out-of-sample extensions to new data points characterized by pairwise dissimilarities are not immediate. Because of this fact, we are more interested in efficient techniques which implicitly refer to this embedding only, which is the topic of the next chapter. As a side product, such algorithms are invariant to coordinate transforms in pseudo-Euclidean space.

3.5 Data sets

In this thesis we will use the following data sets in our experiments. Table 3.1 gives a summary of data sets with respect to size, number of labels, signature, source where the data are taken from and in which chapter each data set is used in the experiment part. The signature is reported for each data set whereby a cutoff at 0.0001 is made to account for numerical errors of the eigenvalue solver. Since some of these matrices correspond to similarities rather than dissimilarities, we use standard preprocessing as presented in [70]. The detailed descriptions are in the following:

Data	Size	#Labels	Signature	Source	Chapter
Amazon47	204	47	(192, 1, 11)	[12]	4
Aural Sonar	100	2	(61, 38, 1)	[12]	4
Face Rec.	945	139	(45, 0, 900)	[12]	4
Patrol	241	7	(54, 66, 121)	[12]	4
Protein	213	4	(169, 38, 6)	[12]	4
Voting	435	2	(16, 1, 418)	[12]	4
Cat Cortex	65	5	(41, 23, 1)	[35]	4
Sonatas	1068	5	(966, 30, 72)	[65]	7
Vibrio	1100	49	(499, 502, 99)	Bruker	4, 5, 6, 7
Bacteria	2007	30	(930, 795, 282)	Bruker	6
Zongker	2000	10	(647, 523, 830)	[20]	6, 7
ProDom	2604	53	(1502, 682, 420)	[20]	6
Chromosomes	4200	21	(1951, 2206, 43)	[60]	4, 5, 7
SwissProt	10988	32	(2028, 2, 8958)	[110]	5
SwissProt10	5791	10	(2043, 12, 3736)	[10]	7

Table 3.1: A summary of data sets

- *Amazon47*: This data set consists of 204 books written by 47 different authors. The similarity is determined as the percentage of customers who purchase book j after looking at book i. Class labeling of a book is given by the author.

- *Aural Sonar*: This data set consists of 100 wide band solar signals corresponding to two classes, observations of interest versus clutter. Similarities are determined based on human perception, averaging over 5 random probands for each signal pair. Class labeling is given by the two classes: target of interest versus clutter.

- *Face Recognition*: 945 images of faces of 139 different persons are recorded. Images are compared using the cosine-distance of integral invariant signatures based on surface curves of the 3D faces. The labeling corresponds to the 139 different persons.

- *Patrol*: 241 samples representing persons in seven different patrol units are contained in this data set. Similarities are based on responses of persons in the units about other members of their groups. Class labeling corresponds to the seven patrol units.

- *Protein*: 213 proteins are compared based on evolutionary distances comprising four different classes according to different globin families. Labeling is given by four classes corresponding to globin families.

- *Voting*: This data set contains 435 samples with categorical data compared by means of the value difference metric. Class labeling into two classes is present.

- *Cat Cortex* : This data set 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 [35] was used.

- *Sonatas*: This data set contains complex symbolic data similar to [65]. It is comprised of pairwise dissimilarities between 1,068 sonatas from the classical period (by Beethoven, Mozart and Haydn) and the baroque era (by Scarlatti and Bach). The musical pieces were given in the MIDI file format, taken from the online MIDI collection *Kunst der Fuge*[2]. Their mutual dissimilarities were measured with the normalized compression distance (NCD) [13], using a specific preprocessing, which provides meaningful invariances for music information retrieval, such as invariance to pitch translation (transposition) and time scaling. This method uses a graph-based representation of the musical pieces to construct reasonable strings as input for the NCD [65]. The musical pieces are classified according to their composer.

- *Vibrio*: This data set consists of 1,100 samples of vibrio bacteria populations characterized by mass spectra.[3] 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 [61]. As usual, mass spectra display strong functional characteristics due to the dependency of subsequent masses, such that problem adapted similarities such as described in [5, 61] are beneficial. In our case, similarities are calculated using a specific similarity measure as provided by the BioTyper software[61]. The Vibrio similarity matrix S has a maximum score of 3. The corresponding dissimilarity matrix is obtained as $D = 3 - S$.

- *Bacteria*: This data set consists of 2007 samples of bacteria mass spec fingerprints in 30 classes taken as a subset from a commercial database provided by [61][4]. The selected bacteria classes are the most prominent ones, consisting of 22 up-to 203 entries. The underlying similarity measure and data generation are discussed in [61]. Basically, the similarities are measures of the alignment of two different spectra and the spectra encode a peptide snapshot of the considered bacterium.

- *Zongker*: The digit dissimilarity data (2000 samples in 10 classes) from [20] is based on deformable template matching. The dissimilarity measure was computed between 2000 handwritten NIST digits in 10 classes, with 200 entries each, as a result of an iterative optimization of the non-linear deformation of the grid [50].

- *ProDom*: The *ProDom* data set consists of 2604 protein sequences with 53 labels. It contains a comprehensive set of protein families and appeared first in the work of [79]. The pairwise structural alignments are computed by [79]. Each sequence belongs to a group labeled by experts, here we use the data as provided in [20].

[2] `http://www.kunstderfuge.com`

[3] We would like to thank Dr. Markus Kostrzewa, Dr. Karl-Otto Kräuter, Dr. Stephan Klebel and Dr. Thomas Maier, all Bruker Daltonik GmbH, Germany, for providing the Vibrio data and support regarding the analysis of the mass spectra with the BioTyper environment.

[4] The database is not public but part of the sold product the article references to, here we use the version with 3034 bacteria groups. Details can be obtained by contacting the authors at Bruker.

- *Chromosomes*: The Copenhagen chromosomes data set has been introduced in [60] as a benchmark for cytogenetics. 4,200 human chromosomes from 21 classes (the autosomal chromosomes) are given by grey-valued images. The images can be represented as strings measuring the thickness of the silhouettes of the chromosomes. These strings are compared using edit distance with insertion/deletion costs 4.5 [66].

- *SwissProt*: The *SwissProt* data set consists of 10,988 samples of protein sequences in 32 classes taken as a subset from the SwissProt database [10]. The considered subset of the SwissProt database refers to the release 37 mimicking the setting as proposed in [55]. The full database consists of 77,977 protein sequences. A typical protein sequence consists of a string of the form $MSKAKEGDYGSIKKVSGPVVV\ldots$ where the letters refer to the amino acids, and the length of the full sequences varies between 30 to more than 1000 amino acids depending on the sequence. The 32 most common classes such as Globin, Cytochrome a, Cytochrome b, Tubulin, Protein kinase st, etc. provided by the Prosite labeling [25] where taken leading to 10,988 sequences. Due to this choice, an associated classification problem maps the sequences to their corresponding prosite labels. These sequences are compared using Smith-Waterman which computes a local alignment of sequences [34]. Popular alternatives could rely on global alignment as provided by Needleman-Wunsch, or linear time heuristics such as BLAST or FASTA [34]. This database is the standard source for identifying and analyzing protein sequences such that an automated classification and processing technique would be very desirable.

- *SwissProt10*: A small version of SwissProt which consists of $5,791$ samples of protein sequences from 10 most common lasses (such as Globin, Cytochrome b, etc.) from the original SwissProt data set described above.

3.6 Conclusions

In this chapter we discussed the dissimilarity representation of data, from the properties, metric and Euclideaness, to how to deal with it. Actually, Euclideaness is a more strict property beyond the metric property. If a given dissimilarity matrix is Euclidean, it can be isometrically embedded into Euclidean space. If not, a pseudo-Euclidean embedding can be found. Besides direct embedding, different ways to deal with dissimilarity data have also been discussed: For Euclidean dissimilarity data, kernel methods can be adopted on the corresponding kernel matrix. If the dissimilarity matrix is non-Euclidean, some correction techniques can be applied first. In this case extra transformation steps have to be paid. Otherwise, dissimilarities can also be considered as features resulting in a dissimilarity-based vector representation of the data, on which standard vector-based methods can be applied. One crucial problem of these approaches is that the out-of-sample extensions for new data is not immediate and extra effort has to be paid. Thus, we are interested in a more efficient way to deal with dissimilarity data and combine it to prototype-based learning, which we will discuss in the next chapter.

Note that this chapter only gives a very brief review of dissimilarity data. For a more comprehensive and theoretical view please refer to the book of Pekalska and Duin [70].

Chapter 4

Prototype-based learning for dissimilarity data

Chapter overview

In this chapter we extend original prototype-based supervised methods such that they can directly deal with dissimilarity data by means of an implicit embedding in pseudo-Euclidean space. The resulting methods give promising results on various benchmark data sets as shown in the experiment part. Parts of this chapters are based on the publications [40, 39].

4.1 Introduction

Unlike Affinity Propagation (AP), Neural Gas (NG), Generalized Topographic Mapping (GTM), as well as diverse LVQs are defined for the Euclidean setting only: they rely on vector operations in the data space $\mathbb{R}^m$. Thus, it is unsuitable for complex or heterogeneous data sets where input dimensions have different relevance or a high dimensionality yields to accumulated noise which disrupts the classification. This problem can partially be avoided by appropriate metric learning, see e.g. [86], or by kernel variants, see e.g. [77]. However, if data are inherently non-Euclidean, these techniques cannot be applied. For example, in modern applications, data are often addressed using dedicated non-Euclidean dissimilarities such as dynamic time warping for time series, alignment for symbolic strings, the compression distance to compare sequences based on an information theoretic ground, and similar. These settings do not allow a Euclidean representation of data at all, rather, data are given implicitly in terms of pairwise dissimilarities or relations.

Several ways to deal with dissimilarity data have been discussed in the previous chapter. Probably, the most obvious or popular way is using kernel variations such as kernel GLVQ [78], kernel NG [77], or kernel GTM [68], just to name few. But an extra preprocessing step has to be paid for, namely transforming dissimilarity to kernels, this step can be very costly, i.e. $\mathcal{O}(n^3)$, n denotes the size of the data set. On the other hand, if the resulting kernel is not psd, there is no guarantee of learning performance. In this case some correction techniques have to be applied to guarantee the psd characteristic

of the kernel as described in the previous chapter.

In this chapter we introduce a technique to directly deal with dissimilarities without extra preprocessing step to transform to a kernel or to embed into a vector space (i.e. pseudo-Euclidean space, Section 3.3). This technique can be easily integrated to LVQs. This way, LVQ techniques become directly applicable for relational data sets which are characterized in terms of a symmetric dissimilarity matrix only. The key ingredient is taken from recent approaches for relational data processing in the unsupervised domain [37, 36] which will be also briefly reviewed first in this chapter: If prototypes are represented implicitly as linear combinations of data in the pseudo-Euclidean embedding, the relevant distances of data and prototypes can be computed without an explicit reference to a vectorial data representation. This principle holds for every symmetric dissimilarity matrix and thus, allows us to formalize a valid objective of GLVQ and RSLVQ for relational data. Based on this observation, optimization can take place using gradient techniques.

4.2 A Review: relational prototype-based clustering

Recently, extensions of NG and GTM have been proposed which can take into account a general dissimilarity matrix [37, 29]. The key observation is given by the fact that a general dissimilarity matrix corresponds to vectorial data embedded in pseudo-Euclidean space (Def. 4) which refers to a vector space equipped with a bilinear form which induces the given dissimilarities. Unlike the standard Euclidean space, the form can be indefinite. In this space, standard vectorial operations are possible. An implicit calculation of the corresponding updates avoids an explicit computation of the embedding [44].

4.2.1 Relational neural gas

Relational neural gas (RNG) as introduced in [37] assumes that a symmetric dissimilarity matrix D with entries d_{ij}, describing pairwise dissimilarities of data, is available. As shown in Section 3.3, there exists a pseudo-Euclidean embedding of a given set of points characterized by pairwise symmetric dissimilarities. That means, there exists a vector space with a symmetric bilinear form $\langle \cdot, \cdot \rangle$ and vectors $\mathbf{x}_i$ such that $d_{ij} = d(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i - \mathbf{x}_j, \mathbf{x}_i - \mathbf{x}_j \rangle$. Without loss of generality, we can restrict prototypes to the span of the data points in this space:

$$\mathbf{w}_j = \sum_{l=1}^{n} \alpha_{jl} \mathbf{x}_l \text{ with } \sum_{l=1}^{n} \alpha_{jl} = 1 \,. \tag{4.1}$$

Under this constraint, dissimilarities can be computed by means of the formula

$$d(\mathbf{x}_i, \mathbf{w}_j) = [D\alpha_j]_i - \frac{1}{2} \cdot \alpha_j^T D \alpha_j \tag{4.2}$$

where $[\cdot]_i$ refers to component i of the vector. This allows a direct transfer of batch NG [15] to general dissimilarities:

$$\text{determine } k_{ij} \text{ based on } d(\mathbf{x}_i, \mathbf{w}_j)\,, \tag{4.3}$$

$$\text{set } \alpha_{jl} := \frac{h_\sigma(k_{lj})}{\sum_l h_\sigma(k_{lj})}\,. \tag{4.4}$$

These steps are iterated until convergence. This algorithm can be interpreted as neural gas in pseudo-Euclidean space for a given symmetric dissimilarity matrix D. It performs the updates implicitly without referring to the vectorial notation $\mathbf{x}_i$ in pseudo-Euclidean space. This way the computational complexity is reduced to $\mathcal{O}(n^2)$ instead of $\mathcal{O}(n^3)$ for a full embedding. If the bilinear form is indefinite, convergence is not guaranteed, albeit it is usually observed in practice [37].

4.2.2 Relational generative topographic mapping

Relational GTM (RGTM) [29] relies on the same principle as RNG: prototypes are restricted to linear combinations of data points

$$\mathbf{w}_j = \sum_{l=1}^{n} \alpha_{jl}\mathbf{x}_l \text{ with } \sum_{l=1}^{n} \alpha_{jl} = 1\,.$$

This way, vectorial operations become possible in the pseudo-Euclidean space, referring only implicitly to the corresponding embedding. Again, distances are computed by means of Eq. 4.2. Since prototypes are represented indirectly by means of weighting factors, the generalized regression model maps from the latent space to the space of coefficients $\alpha_k = \Phi(\mathbf{u}_k) \cdot W$ where the restriction $\sum_l [\Phi(\mathbf{u}_k) \cdot W]_l = 1$ is enforced. Using Lagrange optimization, it can be seen that this restriction is automatically fulfilled for the generic optima obtained by means of EM optimization. Hence, the EM scheme of GTM can directly be transferred to RGTM. The data matrix $\mathbf{V}$ is the identity in the space of coefficients. Typically, GTM is initialized based on the first two eigenvectors of the data. This can directly be extended to dissimilarity data by referring to MDS which corresponds to a PCA in the space of similarities. Detailed formulas for the resulting updates can be found in [29].

4.3 Relational prototype-based classification

This observation constitutes the key to transfer GLVQ and RSLVQ to relational data without an explicit embedding in pseudo-Euclidean space. In the following, we introduce and compare this novel technique. A Prototype $\mathbf{w}_j$ is represented implicitly by means of the coefficient vectors α_j as in (4.1). Then, we can use the equivalent characterization of distances in the GLVQ and RSLVQ cost function leading to the costs of relational GLVQ (RGLVQ) and relational RSLVG (RSLVQ), respectively.

4.3.1 Relational GLVQ

Generalized LVQ (GLVQ) [81] has been introduced in Section 2.3.1. Here we recall the cost function (2.10) based on distances between data points and prototypes:

$$E_{\mathrm{GLVQ}} = \sum_{i=1}^{n} \Phi \left(\frac{d(\mathbf{x}_i, \mathbf{w}_+(\mathbf{x}_i)) - d(\mathbf{x}_i, \mathbf{w}_-(\mathbf{x}_i))}{d(\mathbf{x}_i, \mathbf{w}_+(\mathbf{x}_i)) + d(\mathbf{x}_i, \mathbf{w}_-(\mathbf{x}_i))} \right)$$

by substitute the distance calculation between data points and prototypes by Eq. (4.2), we naturally get a variant of GLVQ for dissimilarity data in the pseudo-Euclidean space:

$$E_{\mathrm{RGLVQ}} = \sum_{i} \Phi \left(\frac{[D\alpha_+]_i - \frac{1}{2} \cdot (\alpha_+)^T D\alpha_+ - [D\alpha_-]_i + \frac{1}{2} \cdot (\alpha_-)^T D\alpha_-}{[D\alpha_+]_i - \frac{1}{2} \cdot (\alpha_+)^T D\alpha_+ + [D\alpha_-]_i - \frac{1}{2} \cdot (\alpha_-)^T D\alpha_-} \right), \tag{4.5}$$

where as before the closest correct and wrong prototype are referred to, corresponding to the coefficients α_+ and α_-, respectively. A stochastic gradient descent leads to adaptation rules for the coefficients α_+ and α_- in this version of GLVQ: component k of these vectors is adapted as

$$\begin{aligned} \Delta\alpha_{+k} \sim \quad & -\Phi'(\mu(\mathbf{x}_i)) \cdot \mu_+(\mathbf{x}_i) \cdot \frac{\partial\left([D\alpha_+]_i - \frac{1}{2} \cdot (\alpha_+)^T D\alpha_+\right)}{\partial \alpha_{+k}} \\ \Delta\alpha_{-k} \sim \quad & +\Phi'(\mu(\mathbf{x}_i)) \cdot \mu_-(\mathbf{x}_i) \cdot \frac{\partial\left([D\alpha_-]_i - \frac{1}{2} \cdot (\alpha_-)^T D\alpha_-\right)}{\partial \alpha_{-k}} \end{aligned}$$

where $\mu(\mathbf{x}_i)$, $\mu_+(\mathbf{x}_i)$, and $\mu_-(\mathbf{x}_i)$ are the same as in the original case but presented by Eq. (4.2) in the following:

$$\mu(\mathbf{x}_i) = \frac{[D\alpha_+]_i - \frac{1}{2} \cdot (\alpha_+)^T D\alpha_+ - [D\alpha_-]_i + \frac{1}{2} \cdot (\alpha_-)^T D\alpha_-}{[D\alpha_+]_i - \frac{1}{2} \cdot (\alpha_+)^T D\alpha_+ + [D\alpha_-]_i + \frac{1}{2} \cdot (\alpha_-)^T D\alpha_-},$$

$$\mu_+(\mathbf{x}_i) = \frac{2 \cdot [D\alpha_-]_i + \frac{1}{2} \cdot (\alpha_-)^T D\alpha_-}{([D\alpha_+]_i + \frac{1}{2} \cdot (\alpha_+)^T D\alpha_+ + [D\alpha_-]_i + \frac{1}{2} \cdot (\alpha_-)^T D\alpha_-)^2},$$

$$\mu_-(\mathbf{x}_i) = \frac{2 \cdot [D\alpha_+]_i + \frac{1}{2} \cdot (\alpha_+)^T D\alpha_+}{([D\alpha_+]_i + \frac{1}{2} \cdot (\alpha_+)^T D\alpha_+ + [D\alpha_-]_i + \frac{1}{2} \cdot (\alpha_-)^T D\alpha_-)^2}.$$

The partial derivative yields

$$\frac{\partial \left([D\alpha_j]_i - \frac{1}{2} \cdot \alpha_j^T D\alpha_j \right)}{\partial \alpha_{jk}} = d_{ik} - \sum_l d_{lk}\alpha_{jl} \tag{4.6}$$

We refer to this extension of GLVQ as *Relational GLVQ* (RGLVQ).

4.3.2 Relational RSLVQ

Similarly, the cost function of RSLVQ (2.12)

$$E_{\mathrm{RSLVQ}} = \log(Lr) = \sum_{i=1}^{n} \log \frac{p(\mathbf{x}_i, y_i|W)}{p(\mathbf{x}_i|W)}$$

can be presented:

$$E_{\mathrm{RRSLVQ}} = \sum_i \log \frac{\sum_{\alpha_j : c(\alpha_j)=c(\mathbf{x}_i)} p(\mathbf{x}_i|\alpha_j)/K}{\sum_{\alpha_j} p(\mathbf{x}_i|\alpha_j)/K} \tag{4.7}$$

where $p(\mathbf{x}_i|\alpha_j) = \mathrm{const} \cdot \exp\left(-\left([D\alpha_j]_i - \frac{1}{2}\cdot \alpha_j^T D\alpha_j\right)/2\sigma^2\right)$, since every component has the same width and prior probability. K is the number of prototypes. A stochastic gradient descent leads to the adaptation rule

$$\Delta\alpha_{jk} \sim \quad -\frac{1}{2\sigma^2}\cdot\left(\frac{p(\mathbf{x}_i|\alpha_j)}{\sum_{j:c(\alpha_j)=c(\mathbf{x}_i)} p(\mathbf{x}_i|\alpha_j)} - \frac{p(\mathbf{x}_i|\alpha_j)}{\sum_j p(\mathbf{x}_i|\alpha_j)}\right)\cdot\frac{\partial\left([D\alpha_j]_i - \frac{1}{2}\alpha_j^T D\alpha_j\right)}{\partial\alpha_{jk}},\ \text{if } c(\mathbf{x}_i) = c(\alpha_j),$$

$$\Delta\alpha_{jk} \sim \frac{1}{2\sigma^2}\cdot\frac{p(\mathbf{x}_i|\alpha_j)}{\sum_j p(\mathbf{x}_i|\alpha_j)}\cdot\frac{\partial\left([D\alpha_j]_i - \frac{1}{2}\alpha_j^T D\alpha_j\right)}{\partial\alpha_{jk}},\ \text{if } c(\mathbf{x}_i) \neq c(\alpha_j).$$

This method is referred to as *Relational RSLVQ* (RRSLVQ).

For both extensions, after every adaptation step, normalization takes place to guarantee $\sum_i \alpha_{ji} = 1$. The prototypes are initialized as random vectors, i.e we initialize α_{ij} with small random values such that the sum is one. It is possible to take class information into account by setting all α_{ij} to zero which do not correspond to the class of the prototype. The prototype labels can then be determined based on their receptive fields before adapting the initial decision boundaries by means of supervised learning vector quantization.

Out-of-sample extension

Out-of-sample extension of the classification to new data is possible based on the following observation [37]: for a novel data point $\mathbf{x}$ characterized by its pairwise dissimilarities $D(\mathbf{x})$ to the data used for training, the dissimilarity of $\mathbf{x}$ to a prototype modeled by α_j can be calculated by

$$d(\mathbf{x}, \mathbf{w}_j) = D(\mathbf{x})^T \cdot \alpha_j - \frac{1}{2}\cdot\alpha_j^T D\alpha_j \tag{4.8}$$

Then by finding the closest/most similar prototype based on the distances/dissimilarities to all prototypes calculated by (4.8) the new data point will be classified by the label of the closest prototype.

Note that, for GLVQ, a kernelized version has been proposed in [77]. However, this refers to a kernel matrix only, i.e. it requires Euclidean similarities instead of general symmetric dissimilarities. In particular, it must be possible to embed data in a possibly high dimensional Euclidean feature space. Here we extended GLVQ and RSLVQ to relational data characterized by a general symmetric dissimilarities which might be induced by strictly non-Euclidean data.

Interpretability: k-approximation of prototypes

As discussed before relational variants of LVQ represent prototypes indirectly by means of coefficient vectors which are not directly interpretable since they correspond to typical positions in the pseudo-Euclidean space. However, because of their representative

	RGLVQ	RRSLVQ	best SVM	#Proto.
Amazon47	0.81(0.01)	**0.83**(0.02)	0.82*	94
Aural Sonar	**0.88**(0.02)	0.85(0.02)	0.87*	10
Face Rec.	**0.96**(0.00)	**0.96**(0.00)	**0.96***	139
Patrol	0.84(0.01)	0.85(0.01)	**0.88***	24
Protein	0.92(0.02)	0.94(0.06)	**0.97***	20
Voting	**0.95**(0.01)	0.92(0.04)	**0.95***	20
Cat Cortex	0.93(0.01)	0.94(0.01)	**0.95**	12
Vibrio	**1.00**(0.00)	0.94(0.08)	**1.00**	49
Chromosome	0.93(0.00)	0.80(0.01)	**0.95**	63

Table 4.1: Results of prototype based classification in comparison to SVM for diverse dissimilarity data sets. The classification accuracy obtained in a repeated cross-validation is reported, the standard deviation is given in parenthesis. SVM results marked with * are taken from [12]. For Cat Cortex, Vibrio, Chromosome, the respective best SVM result is reported by using different preprocessing mechanisms clip, flip, shift, and similarities as features with linear and Gaussian kernel.

character, we can approximate these positions in pseudo-Euclidean space by its closest exemplars, i.e. data points originally contained in the training set. Unlike prototypes, these exemplars can be directly inspected in the same way as data. We refer to such an approximation as *k-approximation*, if a prototype is substituted by its k closest exemplars, the latter being directly accessible to humans. In the following chapters we will see its usability in combination with patch processing to achieve linear time complexity for large data sets and its capability to improve the interpretability of relational prototype-based methods, e.g. on analyzing biomedical data. For other approximation possibilities, the work [48] investigated various approximation techniques with respect to kernel robust soft LVQ.

4.4 Experiments

We evaluate the algorithms RGLVQ and RRSLVQ for several benchmark data sets where data are characterized by pairwise (dis)similarities. On the one hand, we consider six data sets used in [12]: *Amazon47*, *Aural-Sonar*, *Face Recognition*, *Patrol*, *Protein* and *Voting*. As point out in [12], these data sets cover a diverse range of different characteristics such that they constitute well suited test benchmarks to evaluate the learning ability of algorithms for (dis)similarity data. In additional we consider three data sets from biomedical domain, the *Cat Cortex* from [35], the *Copenhagen Chromosomes* data [66] and one own data set, the *Vibrio* data. They constitute interesting applications per se. More details about the data sets can be found in Section 3.5. For each data set we also report the number of prototypes which mirrors the number of classes, representing each class by only few prototypes relating to the choices taken in [37].

Note that Amazon47, Face Rec. and Voting are almost Euclidean, while all other

Chapter		***Supervised***		*Unsupervised*		
		GLVQ	*RSLVQ*	*NG*	*AP*	*GTM*
2	*cost function*	margin	likelihood ratio	topographical QE	QE	dual likelihood
	complexity	$\mathcal{O}(n)$	$\mathcal{O}(n)$	online: $\mathcal{O}(n)$ batch: $\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
	data type	vectorial	vectorial	vectorial	vectorial	vectorial
	working style	online	online	batch/online	online	batch
4	***relational data***	✓	✓	[37]	similarity	[29]

Table 4.2: Summary of chapter 4: Relational extensions of GLVQ and RSLVQ

data sets are rather non-Euclidean. For relational RSLVQ, we adopted the bisection method to find the appropriate σ of the range from 0.1 to 100. The evaluation of the results is done by means of the classification accuracy as evaluated on the test set in a ten fold repeated cross-validation (nine tenths of date set for training, one tenth for testing) with ten repeats. The results are reported in Tab. 4.1.

In addition, we report the best results obtained by SVM after diverse preprocessing techniques [12], which investigates the possibility to deal with similarity/dissimilarity data which is non-Euclidean with the SVM. Since the corresponding Gram matrix is not positive semidefinite, according preprocessing steps (such as clip, flip, shift, see Section 3.3.3) have to be done which make the SVM well defined. These steps can change the spectrum of the Gram matrix or they can treat the dissimilarity values as feature vectors which can be processed by means of a standard kernel.

Interestingly, in most cases, the results which are comparable to the best SVM as reported in [12] can be found, whereby making preprocessing as done in [12] superfluous. Further, unlike for SVM which is based on support vectors in the data set, solutions are represented as typical prototypes.

4.5 Conclusions

In this chapter, as summarized in Table 4.2, we focus on supervised cases and have presented extensions of supervised prototype-based techniques to general possibly non-Euclidean data sets characterized by arbitrary symmetric dissimilarity matrices, by means of an implicit embedding in pseudo-Euclidean space and the corresponding extensions of the cost function of GLVQ and RSLVQ to this setting. As a result, a very powerful learning algorithm can be derived which, in most cases, achieves results which are comparable to SVM but without the necessity of according preprocessing, since relational LVQ can directly deal with possibly non-Euclidean data whereas SVM requires a positive semidefinite Gram matrix.

However, similar to SVM, relational LVQs have quadratic complexity due to the dependency on the full dissimilarity matrix. This makes the methods infeasible for large data sets. A speed-up to linear techniques e.g. by means of patch processing and Nyström approximation for dissimilarity data is the focus of the next chapter.

Chapter 5

Speed-up techniques for large scale problems

Chapter overview

Relational extensions of prototype-based learning techniques can directly deal with dissimilarity data. However, they suffer from the quadratic complexity due to their dependence on the full dissimilarity matrix. In this chapter, we investigate two different linear time approximation techniques, patch processing and Nyström approximation. We apply these approximations to prototype-based learning techniques for dissimilarities, where possible, and compare the results for diverse data sets. Parts of this chapter are based on the publications [110, 112].

5.1 Introduction

Diverse high quality prototype-based learning techniques have been developed which can directly deal with data sets given by general pairwise dissimilarities rather than standard Euclidean vectors. Examples include affinity propagation (AP), relational neural gas (RNG), or relational generative topographic mapping (RGTM) for the unsupervised case, relational GLVQ and relational RSLVQ for the supervised learning. These algorithms allow to generalize classical learning algorithms to general dissimilarities. However, a principled drawback occurs: since the methods rely on the full similarity or dissimilarity matrix, respectively, their effort is quadratic with respect to the number of data. Even more severely, the full quadratic dissimilarity matrix has to be available to apply these methods. Because of these facts, the techniques cannot be used for large data sets.

Diverse approximations to get around a squared complexity in similar settings are available in the literature: Kernel approaches can be accelerated to linear techniques by means of the Nyström approximation [106] which approximates the full Gram matrix by a low rank approximation. By integrating this approximation into the learning algorithms, an overall linear complexity results. We will show that the Nyström approximation can be extended to dissimilarity data and an integration into RNG and RGTM is possible. A linear time approximation results provided a fixed approximation quality of the matrix. We will show that, depending on the nature of the dissimilarity matrix, reasonable results

can be obtained this way.

The Nyström technique has two drawbacks: it requires a representative set of examples for the low rank approximation, such that it cannot be used for online settings where data display a clear trend. Further, in order to arrive at linear techniques, the approximation has to be integrated into the learning algorithm. Hence this method does not constitute an option for techniques where the entries of the dissimilarity matrix are used in a distributed way such as AP.

Patch processing has been proposed as an alternative approximation scheme. It offers a powerful linear time and limited memory approximation for streaming data sets [1]. In the article [37], it has been used to speed up RNG. The resulting technique, patch RNG (PRNG) is linear time. It requires a direct access to the dissimilarities. In this contribution, we transfer this technique to AP and RGTM, resulting in patch AP (PAP) and patch RGTM (PRGTM). We show that, depending on the given setting, a good approximation quality can be achieved. Patch processing can even deal with streaming data which display a clear trend, unlike the Nyström approximation.

First, we introduce the basic idea behind patch processing and the Nyström technique, and we show how these techniques can be applied to relational clustering techniques. Then, we compare the techniques using three benchmarks from bioinformatics: image data in the context of cytogenetics, mass spectra characterizing bacteria, and a part of the classical SwissProt database of protein sequences. In all cases, we compare the behavior of the techniques for data which are directly accessible form versus a presentation as streaming data.

Finally, with the same principle, we transfer both techniques to a supervised technique, relational GLVQ, and compare the results based on the benchmarks again.

5.2 Patch processing

Patch processing was first introduced to k-means in [22]. It is a simple and efficient strategy for clustering with restricted buffer where data are processed consecutively in patches of predefined size. It has been proposed in [1] in the context of the application of NG to streaming data, and interestingly, it even gives good results in this setting. The principled algorithm is depicted in Algorithm 1. A fixed size of the patches M is chosen. Then patches of data are processed consecutively. Given a dissimilarity matrix, the patch p corresponds to the values of the matrix describing the pairwise dissimilarities along the diagonal: $d(\mathbf{x}_i, \mathbf{x}_j)$ where $i, j \in \{p \cdot M + 1, \ldots, (p+1) \cdot M\}$. In addition to this patch, all previous patches are represented in compressed form by means of the prototypes resulting from clustering the previous patches. For this purpose, the dissimilarities of data and these prototypes need to be available. In patch processing, these are computed or retrieved from the dissimilarity matrix on the fly.

Note that it is not clear how to compute these dissimilarities efficiently if prototypes are of a general form, i.e. they correspond to arbitrary vectors in pseudo-Euclidean space as for RNG and RGTM. For an implicit representation of prototypes by means of a linear combination of data points such as (4.1), the dissimilarity computation would depend on the full dissimilarity matrix. To avoid this problem, we assume that prototypes are

Algorithm 1 Principled algorithm for patch clustering

init: $E := \emptyset$; ▷ exemplars
  $p := 0$; ▷ patch number
repeat
  $P_{M,M} := \{d(\mathbf{x}_i, \mathbf{x}_j)\} \mid i, j \in \{p \cdot M + 1, \ldots, (p+1) \cdot M\}\}$; ▷ patch size M
  $P_{M,|E|} := \{d(\mathbf{x}_i, \mathbf{x}_j) \mid p \cdot M < i \leq (p+1) \cdot M, \mathbf{x}_j \in E\}$;
  ▷ dissimilarities of patch and exemplars
  $P_{|E|,|E|} := \{d(\mathbf{x}_i, \mathbf{x}_j) \mid \mathbf{x}_i, \mathbf{x}_j \in E\}$; ▷ dissimilarities of exemplars
  $P := \begin{pmatrix} P_{M,M} & P_{M,|E|} \\ P_{M,|E|}^T & P_{|E|,|E|} \end{pmatrix}$; ▷ full matrix for loop
  change P_{ii} for all i, if necessary; ▷ diagonal entries of P
  $m_i := 1$ for $\mathbf{x}_i$ in $P_{M,M}$; ▷ standard points in this patch
  perform *Patch Clustering* on P with multiplicities m_i; ▷ using RNG, RGTM or AP
  approximate prototypes by k closest exemplars (k-approximation), if necessary;
    $E :=$ set of exemplars obtained this way; ▷ new exemplars
    $m_i :=$ size of receptive field (or divided by k, if necessary) counted as multiplicities for $\mathbf{x}_i \in E$; ▷ new exemplars as multiple points for next patch
  p:=p+1; ▷ next patch
until all dissimilarities are read

approximated by a small number of exemplars such as mentioned in previous chapter, the *k-approximation* of prototypes. Under this assumption, the pairwise distances of these exemplars as well as the distances of these exemplars and the new patch can be retrieved from the dissimilarity matrix in constant time on the fly. This extended dissimilarity matrix, representing patches and representative exemplars, serves as the input for the clustering step of the next patch.

Exemplars which represent the previous clustering results represent a large set of data. Thus, it is vital to weight their relevance correspondingly. In the patch algorithm, this problem is solved by assigning a *multiplicity* to all exemplars which represent clusters of points, which corresponds to the size of its receptive field. This means, we assume that the corresponding exemplars are contained in the data set not only once but multiple times.

Note that patch processing requires constant space, provided the patch size M and the approximation quality of prototypes by k exemplars is fixed. Further, it requires linear time, provided M is fixed and the corresponding metaparameters of the algorithm such as the required number of epochs depend on the size of M only. Unlike the Nyström approximation which we will discuss later, it is not clear a priori which linear part of the dissimilarity matrix is used for training, since the relevant exemplars which represent clusters become available during patch processing only. Thus, we need a way to compute or retrieve the relevant entries of the dissimilarity matrix on the fly. This is easily possible if the dissimilarity is characterized by an algorithmic function such as e.g. Smith-Waterman to align biological sequences by means of a global alignment, or alternatives

such as local alignment by Needleman-Wunsch or approximations such as FASTA or BLAST [34]. Because all relevant information is kept either directly or in compressed form, patch processing can also deal with streaming data, as we will see in the experiments.

5.2.1 Patch relational neural gas and patch relational topographic mapping

Patch relational neural gas (PRNG) has been proposed in [37]. Here we extend RGTM to patch processing, patch RGTM (PRGTM). The algorithmic steps which have to be clarified are the rows 9, 11, and 12 in Algorithm 1.

For PRNG and PRGTM, the diagonal (row 9) is simply set to zero. It is easy to extend RNG to multiplicities (row 11). Assume data point $\mathbf{x}_i$ has multiplicity m_i. Then we exchange the update (4.4) by

$$\alpha_{jl} := \frac{m_l \cdot h_\sigma(k_{lj})}{\sum_l m_l \cdot h_\sigma(k_{lj})} \,.$$

RGTM can be extended to multiplicities by exchanging the update equations (2.4) and (2.5) of original RGTM such that they are valid for multiple data points: In (2.4), the matrices $\mathbf{G}$ and $\mathbf{R}$ which relate to responsibilities and sums of responsibilities, respectively, weight these responsibilities $R_{ij}(W, \beta)$ with m_i. In the update (2.5) for β, the summand according to data point $\mathbf{x}_i$ is weighted with m_i, and n is the number of data counted with multiplicities $\sum_i m_i$. Similarly, the MDS initialization can be extended to multiple data points.

To account for row 12, we need to approximate the prototypes computed by RNG and RGTM, since these entities refer to all given data points. As proposed in [37] we use a simple approximation of the prototypes by means of a fixed number of exemplars taken from the data processed in the current patch. The k-approximation $\mathbf{w}_j^k$ of a prototype $\mathbf{w}_j$ based on a given set of points $\mathbb{X}$ corresponds to the closest k data points in $\mathbb{X}$: $\mathbf{w}_j^k := \{\mathbf{x}_i \in \mathbb{X} \mid d(\mathbf{x}_i, \mathbf{w}_j) \leq d(\mathbf{x}_{i'}, \mathbf{w}_j)$ for all but k indices $i\}$. We simply substitute every prototype by the k closest exemplars of the current patch as measured by the given dissimilarity. These exemplars are counted with multiplicities according to the size of their receptive fields divided by k.

5.2.2 Patch affinity propagation

Since AP constitutes an exemplar based clustering method, the transfer of the meta algorithm is immediate: we just repeatedly apply AP to a given patch and the exemplars as provided by the previous patch, counted with multiplicities. Thus, the row 12 of Algorithm 1 is trivial since prototypes already correspond to exemplars with multiplicities according to the size of their receptive fields.

To apply patch processing to AP obtaining patch AP (PAP), we need to extend AP such that it can deal with multiple data points in row 11. One simple way to do so would exist in a simulation of the update equations provided by standard AP. We can include points according to their multiplicities to the data set. Since these points are exactly

identical, their responsibilities and availabilities can be shared. Thus, we only have to compute the corresponding updates of responsibilities and availabilities once.

Unfortunately, this naive procedure is not possible: AP consists of the max-sum algorithm applied to a cyclic factor graph, hence convergence is not guaranteed. In particular, cycles occur if symmetries are present in the problem formulation. Symmetries can be induced, for example, by exactly identical similarities. For this reason, slight noise is added to the similarities in the original code for AP[1]. If multiple points are present, as in our setting, however, symmetries necessarily occur. Since the updates of these identical points are shared, there is no simple way to cure this problem: the corresponding responsibilities and availabilities are exactly identical since there is only one variable representing the values.

Therefore, we use a different approach to extend AP to multiple points. Assume data point x_i is contained in the data set with multiplicity m_i. Then the quantization error is given as follows

$$E_{\text{QE\_AP}} = \sum_{i,j:\mathbf{x}_i \in R(\mathbf{w}_j)} m_i \cdot s(\mathbf{x}_i, \mathbf{w}_j)\,. \tag{5.1}$$

Obviously, we obtain the same costs if every point is contained only once in the data set, but similarities are given by the values $\tilde{s}(\mathbf{x}_i, \mathbf{x}_j) = m_i \cdot s(\mathbf{x}_i, \mathbf{x}_j)$.

Therefore, we can treat the problem to cluster the points $\mathbf{x}_i$ with multiplicities m_i as a standard problem for standard AP with simple points $\mathbf{x}_i$ where the similarities are given as $\tilde{s}(\mathbf{x}_i, \mathbf{x}_j)$. This way, the usual convergence of AP is observed using small random values to avoid symmetries. The update for the responsibilities becomes

$$r_{ik} := m_i \cdot s_{ik} - \max_{k' \neq k} \left\{ a_{ik'} + m_i \cdot s_{ik'} \right\}, \tag{5.2}$$

where $s_{ik} = s(\mathbf{x}_i, \mathbf{x}_k)$ refers to the original similarities of data points. The update of the availabilities is not changed.

The initialization of the diagonal terms, row 10 of the Algorithm 1, should also be adapted accordingly, putting a bias towards points with large multiplicities. We achieve this by a division of the preferences along the diagonal by the respective multiplicities m_i.

5.3 Nyström approximation

Beside patch processing, an alternative technique is the *Nyström approximation* as presented in [106] which substitutes a given Gram matrix S by a low rank approximation such that linear techniques result. As discussed in [28], this principle can be generalized to dissimilarities. Here we briefly review this method, more technical details and an application of Nyström for correcting of non-psd kernels can be found in [83].

By the Mercer theorem, one can find an expansion of the form

$$s(\mathbf{w}, \mathbf{x}) = \sum_{i=1}^{\infty} \lambda_i \Psi_i(\mathbf{w}) \Psi_i(\mathbf{x})$$

[1] See `http://www.psi.toronto.edu/index.php?q=affinity%20propagation`

for a given kernel s with eigenfunctions Ψ_i and eigenvalues λ_i. These values are solutions of the equation

$$\int s(\mathbf{w}, \mathbf{x})\Psi_i(\mathbf{x})p(\mathbf{x})d\mathbf{x} = \lambda_i\Psi_i(\mathbf{w}),$$

where $p(\mathbf{x})$ is the probability density of $\mathbf{x}$. In the Nyström approximation, this integral is approximated by sampling $\mathbf{x}_k$ i.i.d according to $p(\mathbf{x})$:

$$\frac{1}{M} \cdot \sum_k s(\mathbf{w}, \mathbf{x}_k)\Psi_i(\mathbf{x}_k) \approx \lambda_i\Psi_i(\mathbf{w}) \,,$$

M is the sample size. Using the matrix eigenproblem $S^{(M)}\mathbf{U}^{(M)} = \mathbf{U}^{(M)}\Lambda^{(M)}$ of the $M \times M$ Gram matrix $S^{(M)}$, we obtain

$$\lambda_i \approx \lambda_i^{(M)}/M \,,$$
$$\Psi_i(\mathbf{w}) \approx \frac{\sqrt{M}}{\lambda_i^{(M)}} \cdot (s(\mathbf{x}_1, \mathbf{w}), \ldots, s(\mathbf{x}_M, \mathbf{w}))\mathbf{u}_i^{(M)} \,,$$

where $\mathbf{u}_i^{(M)}$ is the ith column of $\mathbf{U}^{(M)}$.

For a given $n \times n$ Gram matrix S we randomly choose M rows and respective columns. The corresponding indices are also called *landmarks*. The corresponding rows are denoted by $S_{M,n}$ and columns by $S_{n,M}$. Hence we obtain

$$\tilde{S} = \sum_{i=1}^{M} 1/\lambda_i^{(M)} \cdot S_{n,M} \cdot \mathbf{u}_i^{(M)}(\mathbf{u}_i^{(M)})^T S_{M,n},$$

where $\lambda_i^{(M)}$ and $\mathbf{u}_i^{(M)}$ correspond to the $M \times M$ eigenproblem. Hence

$$\tilde{S} = S_{n,M}S_{M,M}^{-1}S_{M,n} \tag{5.3}$$

where $S_{M,M}^{-1}$ denotes the Moore-Penrose pseudo inverse. This matrix $\tilde{S}$ offers a low rank approximation of S.

We will see that the corresponding matrix S is used in algorithms such as RNG and RGTM in the form $S^T\mathbf{x}_i$ where $\mathbf{x}_i$ is an n-dimensional vector. Then, performing multiplication from right to left, the complexity $\mathcal{O}(M^3 + nM + M^2 + Mn)$ results when computing $\tilde{S}^T\mathbf{x}_i$. Hence the computation is of complexity $\mathcal{O}(M^3 n)$ instead of $\mathcal{O}(n^2)$ for the original matrix.

Similarly, dissimilarities D can be approximated if D is symmetric. Since D is symmetric, it can be diagonalized. Hence it can be interpreted as operator $d(\mathbf{x}, \mathbf{w}) = \sum_i \lambda_i\Psi_i(\mathbf{x})\Psi_i(\mathbf{w})$. Thus, the same mathematical treatment as before is possible, the only difference being that eigenvalues are allowed to be negative.

Nyström RGTM and Nyström RNG

For RGTM, this yields an approximation of dissimilarities calculation (4.2)

$$d(\mathbf{x_i}, \mathbf{w_j}) \approx [D_{n,M}(D_{M,M}^{-1}(D_{M,n} \cdot \alpha_j))]_i - \frac{1}{2} \cdot (\alpha_j^T D_{n,M}) \cdot (D_{M,M}^{-1}(D_{M,n} \cdot \alpha_j)) \tag{5.4}$$

which is $\mathcal{O}(M^3 n)$. Thereby, initialization of RGTM is done based on a corresponding landmark MDS. Similarly, the Nyström approximation can be integrated into RNG approximating the distance computation in the same way. In both cases, by evaluating the matrix multiplications consecutively, a linear time algorithm results provided the sample size M is fixed. Note that the same approximation does not lead to a reduction of the computational complexity for AP since the similarity values are distributed in the algorithmic computations rather than treated in matrix form.

The Nyström approximation is exact if the number of samples M is chosen according to the rank of D. If a subsample is chosen, bounds on the quality of the Nyström approximation can be derived as presented e.g. in [108]. Note that the quality of the approximation depends on the rank of the approximation as compared to the original matrix. Thus, it is vital that a representative subsample is chosen. We will see in experiments, that the size M can often be chosen as a small set to arrive at good results. The accuracy degrades severely, however, if streaming data are dealt with for which the chosen subset is not representative, as we will show in experiments.

If we assume a fixed size M and a fixed complexity of the algorithm depending on the rank of the approximation (i.e. a fixed number of epochs etc.), a linear time approximation results. Note that the part of the dissimilarity matrix which is required for the Nyström approximation is fixed as soon as the representative subsample is chosen. Thus, the required dissimilarities can be precomputed before applying the algorithm.

5.4 Experiments on biomedical applications

We evaluate and compare the following algorithms:

- Affinity Propagation (**AP**) and the approximation Patch Affinity Propagation (**PAP**),
- Relational Neural Gas (**RNG**) and the two approximations Patch Relational Neural Gas (**PRNG**) and Nyström Relational Neural Gas (**RNG (Ny)**),
- Relational Generative Topographic Mapping (**RGTM**) and the two approximations Patch RGTM (**PRGTM**) and Nyström RGTM (**RGTM (Ny)**).

on three data sets stemming from biomedical applications: *Copenhagen Chromosomes*, *Vibrio*, and *SwissProt* as described in Section 3.5. These three data sets constitute typical examples of non-Euclidean data which occur in biomedical domains. The dissimilarity measures are inherently non-Euclidean and cannot be embedded isometrically in a Euclidean vector space.

We evaluate the data sets using three different evaluation measures as introduced in Section 2.5:

- The *quantization error* (QE) (2.15) which measures in how far the prototypes can represent the given data as measured by the averaged dissimilarity of prototypes to points in their receptive field. All methods optimize a cost function which can be related to the quantization error as mentioned in Section 2.5. Therefore,

the evaluation of the quantization error where the terms $d(\mathbf{x}_i, \mathbf{w}_j)$ are taken as dissimilarities, allows to directly evaluate the affect of the approximation techniques on an underlying cost function.

- The *dual quantization error* (dualQE) is given by (2.15). The quantization error and its dual coincide if prototypes are located at centers of receptive fields in pseudo-Euclidean space $\mathbf{w}_j = \sum_{i\,:\,\mathbf{x}_i \in R_j} \mathbf{x}_i / |R_j|$. Hence the quantization error and its dual are identical for RNG. For RGTM, AP, or approximations of RNG, prototypes are no longer located at the center of gravity due to restrictions of the prototypes or approximations of the dissimilarities, respectively. In such cases, the quantization error can be large, albeit the decomposition of data into receptive fields is hardly affected. In these cases, the dual quantization error allows us to compare the quality of the decomposition of data points into clusters rather than the specific prototype locations.

- The *classification accuracy* is obtained by posterior labeling.

- For the settings, we also report the *CPU time* in seconds taken for one run on the full data sets on a 12 Intel(R) Xeon X5690 machine with 3.47GHz processors and 48 GB DDR3 1333MHz memory.

The parameter choices are as follows:

- *Repeats*: Per default, we average every result over ten repeats, reporting the mean value and standard deviation. Due to time issues, less repeats are taken in the following settings: RGTM (only one repeat for the cross-validation), RGTM (Ny with 10%) (only 4 repeats for the cross-validation).

- *Cross-validation*: The classification error is evaluated on the full data set, and in a ten-fold repeated cross-validation.

- *Patch sizes*: For patch processing, ten patches are chosen, i.e. every patch is of the size given by the number of data points / 10.

- *k-approximation:* The value k for the k-approximation for patch processing is taken in $\{1, 3, 5\}$.

- *Number of clusters:* The number of clusters K is roughly chosen according to the priorly known number of classes: For Chromosomes, we use 60 clusters for AP and RNG and its approximations, and 20×20 lattice points for RGTM and its approximations, to account for topological constraints of the latter. For the Vibrio data set, we use 50 clusters for AP and RNG, and 20×20 lattice points for RGTM. For SwissProt, we use 250 Prototypes for AP and RNG, and a 40×40 lattice for RGTM.

- *Nyström approximation*: A fraction of 1%, 2%, and 10% of the data is used to approximate the full matrix.

- *RGTM*: For RGTM and its approximations, we use the default initialization with MDS (landmark MDS for the Nyström approximation). The number of base functions F is picked according to the data set: We use 10×10 base functions with bandwidth 1 for the chromosomes and vibrio data and bandwidth 0.2 for Swissprot. 50 epochs are used for EM training.

- *RNG*: For RNG and its approximations, we use an exponential annealing schedule for the neighborhood range starting from $K/2$. 100 epochs are used for training.

- *AP*: For AP, self-similarities are set by binary search starting from the median in repeated runs such that a fixed number of exemplars/prototypes as specified above is obtained. We accept partial deviations thereof, an exact fit of a given number of clusters often requiring additional trials. Usually, 1-8 repeats are necessary to obtain the correct number of exemplars. The necessary number of repetitions is included in the CPU time measurement. Adaptation is done until convergence is observed.

We test the algorithms in two settings: On the one hand, we present the data in epochs subject to random permutations (standard setting). This way, the approximation techniques can access representative data for the approximation. In addition, we present the data in a fixed permutation referred to as *streaming data*; for the latter, data are sorted according to priorly given class labels and presented to the algorithm in this order - this resembles the fact that, for large data sets, it is often not feasible to assume a truly i.i.d. distribution of data; rather data can be addressed as they are stored in (probably distributed) memory; they are often ordered according to specific criteria such as e.g. the class labels. Therefore, it would be advantageous if algorithms could directly deal with the data as stored in the memory and the ordering would not have much influence on the outcome. For the Nyström approximation and patch processing, respectively, specific ordering might have an influence since the approximation is based on the ordering of the data: landmarks are taken from the first examples for the Nyström approximation, and, similarly, the first patches are restricted to the first data points for patch processing. We will test the feasibility of the approximation in such settings, referred to as 'streaming data'.

Since AP, unlike GTM and NG, requires similarities, we use standard double centering to transform similarities to dissimilarities and vice versa.

Results for the Chromosomes data

The results of the algorithms on the Chromosomes data set are displayed in Tab. 5.1 when referring to the full data set, and in Tab. 5.2 when referring to an evaluation of the classification accuracy in a repeated cross-validation. For the latter setting, we also report the results obtained for streaming data.

We can observe that the results are uniformly best for RGTM. As expected, the quality of all evaluation measures drops down if approximations are used. The overall trend, however, is diverse. This can be traced back to the following phenomena: on the one hand, some approximation results display a uniform decrease in accuracy such as the

Chromosomes	accuracy	QE	dualQE	CPUtime
AP	0.899(0.001)	70902.250 (0.000)	47112.667 (9.406)	380(73)
PAP	0.864(0.001)	76739.950 (928.209)	48518.834 (412.783)	2752(11)
RNG	0.902(0.009)	45751.008 (81.324)	45751.008 (81.324)	148(3)
RNG (Ny 10%)	0.759(0.132)	57620.918 (11936.192)	51801.754 (5479.342)	123(7)
RNG (Ny 2%)	0.810(0.098)	52415.133 (6287.455)	49776.813 (4657.611)	49(2)
RNG (Ny 1%)	0.884(0.006)	46580.623 (232.367)	46216.928 (104.868)	48(4)
PRNG ($k = 1$)	0.858(0.007)	76761.575 (520.301)	49118.806 (318.913)	63(1)
PRNG ($k = 3$)	0.879(0.008)	61582.019 (432.681)	47875.753 (248.223)	83(2)
PRNG ($k = 5$)	0.884(0.008)	58155.238 (383.595)	47432.292 (148.811)	99(4)
RGTM	0.941(0)	41407.211 (0.000)	37737.074 (0.000)	3262(49)
RGTM (Ny 10%)	0.409(0.072)	428041.085 (1030835.396)	66159.509 (3041.827)	512(113)
RGTM (Ny 2%)	0.689(0.190)	60274.937 (1449.945)	50668.785 (7935.304)	424(34)
RGTM (Ny 1%)	0.885(0.012)	44763.112 (1929.780)	41277.147 (419.516)	416(7)
PRGTM ($k = 1$)	0.856(0.010)	61176.709 (676.005)	43915.486 (723.462)	359(18)
PRGTM ($k = 3$)	0.869(0.011)	51346.674 (197.432)	40622.258 (443.058)	620(25)
PRGTM ($k = 5$)	0.886(0.006)	48464.344 (308.555)	39935.687 (361.247)	1089(63)

Table 5.1: Results on the Chromosome data set: The different methods are trained for the full data set and the classification accuracy (accuracy), the quantization error (QE), and the dual quantization error (dual QE) are reported. In addition, we include the CPU time. The standard deviation is given in parentheses.

Nyström approximation with 10% of the data. This can be explained by a bad quality of the resulting dissimilarity matrix. We will specify this issue in a separate paragraph shortly. Hence, in this setting, the Nyström approximation fails.

On the other hand, we can observe a drop down of the quantization error by more

than 20% as compared to the best result, but the dual quantization error and the classification error do not follow the trend. This holds in particular for AP and for patch approximations. The reason behind this effect is that prototypes are located at restricted positions in such settings because they are represented by exemplars (a small number of k exemplars for patch processing). This causes the quantization error to increase, albeit the resulting cluster separation is still acceptable as measured by the dual quantization error. For this reason, we suggest that the dual quantization error is more suited to compare the results.

When focussing on the dual quantization error, we can observe that the approximation techniques yield results which are mostly comparable to the original techniques for AP and RNG. More precisely, the increase is less than 3% for PAP as compared to AP, less than 9% for PRNG and RNG (Ny) as compared to RNG except for Nyström approximation using 10%. For RGTM, the increase is less than 8% for the Nyström approximation using 1% of the data, and for patch processing with $k \in \{3, 5\}$. Thus, it seems that the approximation techniques are well suited if appropriate parameters concerning the approximation are taken.

This overall picture is confirmed if the generalization ability of a classifier by posterior labeling is addressed as displayed in Tab. 5.2. Interestingly, the sensitivity of the approximation techniques to the ordering of the data is quite diverse. While the classification accuracy of patch processing is still more than 84% for streaming data, the Nyström approximation cannot deal with such settings. The letter can be accounted for by the fact that the approximate matrix has a small rank if only a subspace of the full data space is isolated by means of a specific ordering.

Due to the comparably small size of the data set, the speed-up of the approximation techniques is only partially observable. As displayed in Tab. 5.1, a speed-up up to 3 can be gained for RNG, and more than 7 for RGTM. Surprisingly, PAP needs longer run time than AP. This can be explained by the fact that we also include necessary restarts to obtain a correct number of clusters by binary search.

In summary, RGTM and a patch approximation with sufficient k seems best suited in this setting to arrive at a good cluster structure or classification. The Nyström approximation seems problematic due to very different results for different parameter choices in this setting.

The Chromosomes data set has also been addressed in the approach [51]. In this approach, k-nearest neighbor classification is based on the edit distance to compare the data. Results report a classification error of only about 5%, alternative classifiers such as Hidden Markov Models or feedforward networks account for an error of about 9%. Our experimental setting is slightly different. Still, the classification error is in the same order of magnitude for RNG, albeit the latter does not take into account the class labels during training.

Results for the Vibrio data

For the Vibrio data set, the overall picture is similar as summarized in Tab. 5.3 and Tab. 5.4. RGTM displays the best quantization error and dual quantization error, while AP leads to the best classification accuracy. This is an indicator that, in this setting, the

Chromosomes	accuracy	Streaming data
AP	0.895 (0.006)	
PAP	0.838 (0.005)	0.783 (0.022)
RNG	0.902 (0.003)	
RNG (Ny 10%)	0.497 (0.057)	0.101 (0.053)
RNG (Ny 2%)	0.700(0.039)	0.088(0.034)
RNG (Ny 1%)	0.871 (0.009)	0.088 (0.031)
PRNG ($k = 1$)	0.854 (0.006)	0.621 (0.021)
PRNG ($k = 3$)	0.877 (0.004)	0.747 (0.016)
PRNG ($k = 5$)	0.880 (0.006)	0.763 (0.012)
RGTM	0.881 (0.007)	
RGTM (Ny 10%)	0.516 (0.066)	0.431 (0.082)
RGTM (Ny 2%)	0.638 (0.081)	0.582 (0.107)
RGTM (Ny 1%)	0.878 (0.027)	0.756 (0.099)
PRGTM ($k = 1$)	0.840 (0.005)	0.843 (0.007)
PRGTM ($k = 3$)	0.858 (0.007)	0.857 (0.006)
PRGTM ($k = 5$)	0.873 (0.005)	0.871 (0.005)

Table 5.2: Results on the Chromosomes data set in a repeated ten-fold cross-validation evaluated by posterior labeling, the standard deviation is shown in parenthesis

priorly known class structures are only partially mirrored in the cluster structure found in the data set. As for Chromosomes, exemplar based techniques lead to an increased quantization error, while its dual is competitive to the results of RNG and RGTM. In this setting, the approximation techniques are uniformly good, leading to an increase of the dual quantization error of less than 6% in all settings but patch RGTM with $k = 1$. In Tab. 5.4, the classification accuracy obtained for streaming data is displayed in comparison to a random permutation. Again, the Nyström approximation cannot deal with this setting in the same way as patch processing, the latter yielding an accuracy of more than 80% for all but the 1-approximation.

Again, due to the size of the data set, the speed-up of the approximation techniques is only partially visible. It accounts for a factor close to 4 for PAP as compared to AP, up to 6 for approximations of RNG, and more than 3 for RGTM approximations. Generally, the Nyström approximation seems more efficient with respect to the time complexity as compared to patch processing due to the smaller overhead of the technique.

In summary, AP and PAP are preferred if the prior classification task is addressed this way. If the dual quantization error is in the focus, all methods and their approximations seem universally suited. For comparison, the result of an SVM classification (standard Matlab implementation in the bioinformatics toolbox with default parameters, the SVM directly works on the given Gram matrix) leads to 100% classification accuracy in a cross-validation. As reported above, AP yields almost the same accuracy albeit not taking class labels into account during training.

Vibrio	accuracy	QE	dualQE	CPUtime
AP	0.999(0.000)	594.480(0.000)	303.755(0.000)	136(1)
PAP	0.992(0.013)	604.993(2.546)	304.702(1.218)	35(3)
RNG	0.796(0.025)	327.899(4.332)	327.889(4.332)	18(1)
RNG (Ny 10%)	0.895(0.024)	316.945(2.464)	316.904(2.497)	13(1)
RNG (Ny 2%)	0.841(0.015)	333.296(3.892)	330.847(3.589)	3(0)
RNG (Ny 1%)	0.764(0.029)	358.924(8.236)	344.848(5.423)	3(0)
PRNG ($k = 1$)	0.732(0.034)	661.101(9.277)	343.007(7.624)	16(0)
PRNG ($k = 3$)	0.864(0.035)	422.829(8.383)	319.353(4.389)	23(0)
PRNG ($k = 5$)	0.859(0.019)	385.893(5.979)	321.858(2.501)	34(7)
RGTM	0.960(0.000)	305.637(0.000)	285.757(0.000)	111(9)
RGTM (Ny 10%)	0.925(0.022)	308.804(6.096)	301.493(5.427)	48(2)
RGTM (Ny 2%)	0.946(0.019)	309.885(5.501)	293.813(5.332)	38(1)
RGTM (Ny 1%)	0.806(0.025)	357.398(4.175)	296.326(8.282)	35(2)
PRGTM ($k = 1$)	0.702(0.103)	451.467(6.351)	328.479(20.430)	166(7)
PRGTM ($k = 3$)	0.876(0.027)	375.971(2.109)	293.201(2.491)	234(11)
PRGTM ($k = 5$)	0.897(0.030)	341.489(4.233)	291.986(3.756)	309(14)

Table 5.3: Results on the Vibrio data set when trained for the full data set. The standard deviation is given in parentheses.

Results for the SwissProt data

For the SwissProt data set, results are reported in Tab. 5.5. The dual quantization error is best for AP, but the dual quantization error for PAP and RNG and its approximation uniformly less than 10% away. Thereby, no big difference can be observed for the different approximation technique. The RGTM seems universally less suited for the data set, probably because of too strong topological constraints which prevent a good representation of the data in this case. Albeit the dual quantization error is competitive, the classification accuracy varies in the diverse cases. It displays more than 90% accuracy for AP and its approximation, but less than 70% for some of the approximations of RNG. Thus, the link between the priorly given classes and the cluster structure is not clear.

Because of the size of the data set, the speed-up of the approximation techniques can clearly be observed in all cases. It accounts for a factor 2.5 for PAP as compared to AP, up to 6 for approximations of RNG, and up to 10 for approximations of RGTM. The speed-up depends on the parameter choice, smaller values of k for patch processing and a smaller percentage of data points for the Nyström approximation accounting for considerably reduced CPU time.

In summary, any approximation of AP and RNG seems suited in this case if the goal is a good clustering structure (as measured by the dual quantization error), while AP and PAP lead to the best classification accuracy in this setting. For comparison, an SVM has been trained in a 2-fold cross-validation for the given Gram matrix (standard Matlab implementation in bioinformatics toolbox). It yields 98% classification accuracy. Hence AP and PAP are in the same order of magnitude, albeit they do not take into account

Vibrio	accuracy	streaming data
AP	0.999 (0.000)	
PAP	0.999 (0.000)	0.993 (0.004)
RNG	0.798(0.012)	
RNG (Ny 10%)	0.885 (0.010)	0.392 (0.097)
RNG (Ny 2%)	0.823 (0.012)	0.179 (0.117)
RNG (Ny 1%)	0.715 (0.014)	0.052 (0.034)
PRNG ($k = 1$)	0.729 (0.011)	0.658 (0.022)
PRNG ($k = 3$)	0.842 (0.012)	0.819 (0.013)
PRNG ($k = 5$)	0.858 (0.005)	0.820 (0.014)
RGTM	0.947 (0.005)	
RGTM (Ny 10%)	0.939 (0.007)	0.848 (0.023)
RGTM (Ny 2%)	0.781 (0.021)	0.697 (0.040)
RGTM (Ny 1%)	0.547 (0.031)	0.462 (0.058)
PRGTM ($k = 1$)	0.692 (0.029)	0.645 (0.034)
PRGTM ($k = 3$)	0.867 (0.013)	0.852 (0.017)
PRGTM ($k = 5$)	0.903 (0.011)	0.898 (0.011)

Table 5.4: Results on the Vibrio data set when evaluated in a repeated ten-fold cross-validation for randomly permuted data sets (accuracy) and trained for the streaming setting, i.e. data sets presented according to the class labels, the standard deviation is shown in parenthesis

the class labels during training. Interestingly, SVM training is quite affected when taking the Nyström approximation for speed-up. The classification accuracy drops down to 86% for 1% data for approximation, and to 63% for 10% data for approximation. These results are in the same order of magnitude as the classification results of the Nyström approximation of RNG.

5.4.1 Quality of the Nyström approximation

In the experiments, we generally observe the fact that patch processing leads to better results the larger the approximation parameter k, which is expected. In contrast, the Nyström approximation does not alway lead to better results if a larger fraction of the data is used for the approximation. In this paragraph, we investigate in how far this observation can be traced back to the quality of the Nyström approximation when using different fractions of the data. For this purpose, we sample a fraction ranging from 1% up to 50% (10% for SwissProt) of the data based on which the Nyström approximation is computed. For the clustering techniques, the absolute value of the dissimilarities is generally less relevant as compared to the induced order of the data. Therefore, we compare the resulting Nyström approximation with the original dissimilarity matrix taking Spearman correlation of the rows. We average over ten random selections of the data.

The results are displayed in Fig. 5.1 and Fig. 5.2 for all three data sets. All values

SwissProt	accuracy	crossVal	dualQE	CPUtime
AP	0.931(0.001)	0.925 (0.001)	3370.656(0.000)	14162(37)
PAP	0.925(0.000)	0.919 (0.001)	3451.163(13.659)	5644(316)
RNG	0.883 (0.008)	0.873 (0.004)	3476.074(6.427)	2769(16)
RNG(Ny 10%)	0.639(0.168)	0.660(0.057)	3742.972(466.172)	2767(45)
RNG(Ny 2%)	0.781(0.010)	0.840(0.004)	3582.827(13.728)	801(22)
RNG(Ny 1%)	0.761(0.016)	0.825 (0.009)	3712.956(55.872)	437(27)
PRNG($k = 1$)	0.857 (0.006)	0.858 (0.003)	3738.329(16.182)	863(5)
PRNG($k = 3$)	0.628 (0.029)	0.633 (0.010)	3628.645(31.275)	1285(12)
PRNG($k = 5$)	0.639 (0.018)	0.635 (0.010)	3588.068(17.277)	1712(16)
RGTM	0.700(0.000)	0.702 (0.015)	3943.099(0.000)	78518(635)
RGTM(Ny 10%)	0.579(0.172)	0.620 (0.076)	3937.313(454.717)	14582(160)
RGTM(Ny 2%)	0.841(0.035)	0.769 (0.010)	3582.751(89.055)	8045(57)
RGTM(Ny 1%)	0.831(0.041)	0.630 (0.017)	3651.053(89.256)	7247(171)
PRGTM($k = 1$)	0.423(0.039)	0.398 (0.023)	4177.034(116.670)	5059(372)
PRGTM($k = 3$)	0.451(0.034)	0.412 (0.016)	4069.374(45.699)	6755(289)
PRGTM($k = 5$)	0.465(0.049)	0.388 (0.006)	4003.668(78.533)	8917(1852)

Table 5.5: Results on the SwissProt data set, the different methods are trained for the full data set and evaluated with the classification accuracy (column accuracy) and the dual quantization error (dualQE), and they are trained in a repeated cross-validation and evaluated by the classification error (column crossVal). The CPU time is measured for one run for the full data set.

are significant corresponding to p-values smaller than 0.1. Interestingly, the graphs are not monotonously increasing. That means, the Nyström approximation in terms of the induced ordering of data points does not necessarily become better the larger the fraction of the data used for approximation. In particular for Chromosomes, approximations with only 1% of the data yield an almost perfect correlation, while this drops down for larger sizes. In contrast, the approximation quality of the Vibrio data set increases for a percentage $\leq 15\%$. This observation can be one reason why the accuracy of the clustering results is also not monotonic with respect to the percentage of data points as detailed above. For SwissProt, the approximation quality seems universally good for a range from 1% to 5% resulting in a high correlation.

5.4.2 Computational complexity

As already mentioned, the computational complexity of all clustering techniques AP, RGTM, and RNG is $\mathcal{O}(n^2)$. Further, the full dissimilarity matrix D has to be computed, which also has time and space complexity $\mathcal{O}(n^2)$. Where are the current limits for the exact methods? Assuming double precision and standard memory of 12 Gigabytes, a dissimilarity matrix of up to 30,000 objects would currently fit into main memory. Hence the SwissProt data set is already in the order of data sets which only just fit into main memory – it amounts to about 500 Megabytes. Probably the larger bottleneck is given by

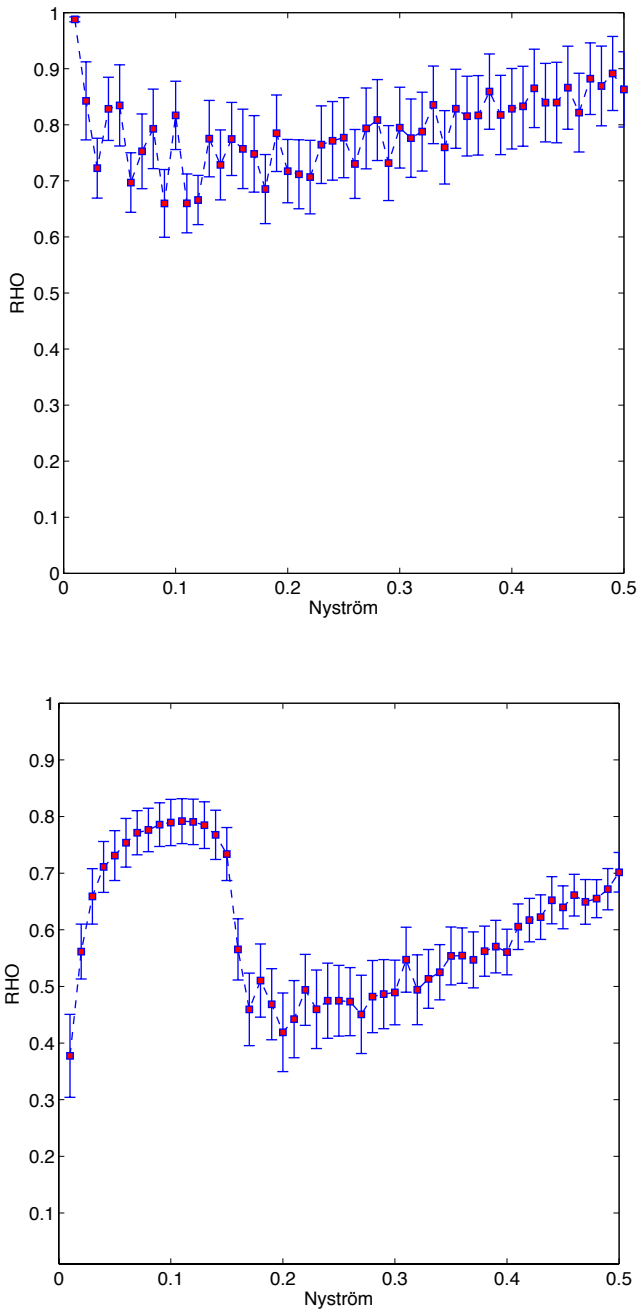


Figure 5.1: Quality of the Nyström approximation as evaluated by the Spearman correlation of the rows of the approximated matrix and the original one. The approximation is based on a different fraction of the data set as indicated by the x-achses. The graphs show the result for the Chromosomes (top) and Vibrio (bottom).

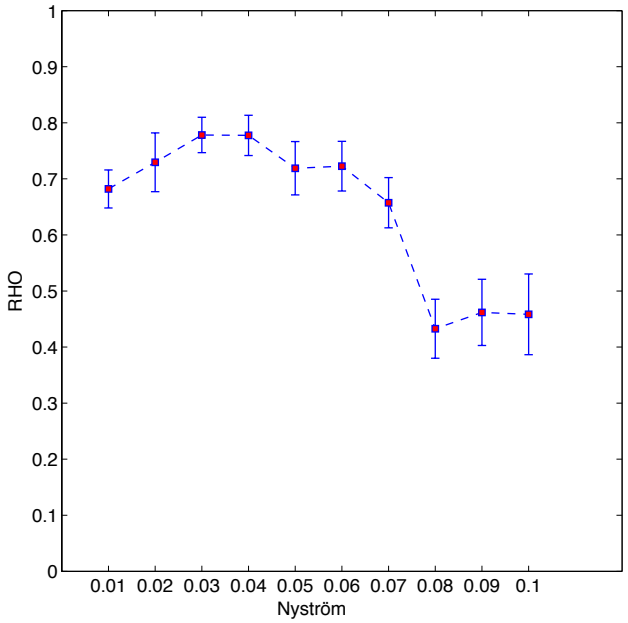


Figure 5.2: Quality of the Nyström approximation as evaluated by the Spearman correlation for SwissProt.

the computation of the dissimilarity matrix. For the SwissProt data set, its computation took about 8 days (using the Bioinformatics toolbox of Matlab for Smith Waterman).

If either approximation is used, the computational complexity as well as the size of the matrix which is required reduces to a linear part with respect to n, assuming fixed approximation parameters (i.e. a fixed percentage of data for the Nyström approximation and a fixed patch size, respectively. Further, naturally, initialization of RGTM (Ny) has to rely on landmark MDS to transfer this linear complexity to the initialization.) For an approximation with 100 points for the Nyström technique or patch sizes of 100 points, the required space would reduce to about 4.5 Megabytes and a computation time of less than 2 hours for the required dissimilarities.

5.5 Patch and Nyström RGLVQ

In previous sections we have integrated two approximations techniques to relational prototype-based clustering techniques. In the same way these techniques can be directly transferred to supervised learning techniques introduced in Section 4.3, since they have the same problem: for supervised techniques such as RGLVQ, RRSLVQ, they rely on the full dissimilarity matrix D and represent prototypes implicitly by means of coefficients α_{ji} (4.1) referring to the contribution of data point $\mathbf{x}_i$ to prototype $\mathbf{w}_j$, i.e. the algorithm has squared time complexity and linear space complexity. Here we extend patch processing to RGLVQ as an example, extending it to relational RSLVQ is analog.

The workflow of patch processing for RGLVQ is the same as for clustering techniques : A fixed size of the patches M is chosen, and data are separated into patches. Then the patches of data are processed consecutively using RGLVQ. To apply RGLVQ, the dissimilarities of data and these prototypes need to be available. In patch processing, these are retrieved on the fly. To compute these dissimilarities efficiently, after processing a patch, we approximate a prototype by its k-approximation for fixed k. This way, the dissimilarity matrix considered in step p corresponds to a fixed size $M + kK$ matrix, K being the number of prototypes.

Exemplars/k-approximated prototypes representing the previous training results represent a subset of data. Thus, to weight their relevance correspondingly, we again assign a multiplicity to these prototypes which corresponds to the size of its receptive field divided by k. This means, we assume that the corresponding prototypes are contained in the data set not only once but multiple times. Note that RGLVQ can easily be extended to deal with sets where data points are equipped with multiplicities. For a points $\mathbf{x}_i$ with multiplicity m_i its contribution to the costs is simply multiplied by m_i. Hence the corresponding step width of a gradient descent algorithm is simply multiplied with m_i. The resulting algorithm, Patch RGLVQ, is depicted in Algorithm 2.

Algorithm 2 Patch RGLVQ

init: $E := \emptyset$; ▷ stores exemplars/k-approximated prototypes
  $p := 0$; ▷ patch number
repeat
  $P_{M,M} := \{d(\mathbf{x}_i, \mathbf{x}_j)\} \mid i, j \in \{p \cdot M + 1, \ldots, (p+1) \cdot M\}\}$; ▷ patch size M
  $P_{M,|E|} := \{d(\mathbf{x}_i, \mathbf{x}_j) \mid p \cdot M < i \leq (p+1) \cdot M, \mathbf{x}_j \in E\}$;
    ▷ dissimilarities of patch and exemplars
  $P_{|E|,|E|} := \{d(\mathbf{x}_i, \mathbf{x}_j) \mid \mathbf{x}_i, \mathbf{x}_j \in E\}$; ▷ dissimilarities of exemplars
  $P := \begin{pmatrix} P_{M,M} & P_{M,|E|} \\ P_{M,|E|}^T & P_{|E|,|E|} \end{pmatrix}$; ▷ full matrix for loop
  $m_i := 1$ for x_i in $P_{M,M}$; ▷ standard points in this patch
  perform RGLVQ for P
  approximate prototypes by k closest exemplars; ▷ k-approximation
    $E :=$ set of exemplars obtained this way; ▷ new exemplars
    $m_i :=$ size of receptive field/k counted as multiplicities for $\mathbf{x}_i \in E$;
  p:=p+1; ▷ next patch
until all dissimilarities are considered

Using (5.4) for distance calculation we can directly integrate the Nyström approximation in RGLVQ or RRSLVQ by taking a random subsample of M points to approximate the dissimilarity matrix. The sample size M is differed during training showing the effect of the approximation on the percentage used for the approximation.

	RGLVQ	RGLVQ (Ny 10%)	PRGLVQ		
			$k = 1$	$k = 3$	$k = 5$
Vibrio	1.000(0.000)	0.992(0.001)	0.999(0.000)	1(0)	1(0)
Chromosomes	0.927(0.002)	0.782(0.004)	0.867(0.002)	0.840(0.006)	0.828(0.003)
SwissProt	0.823 (0.000)	0.834(0.002)	0.833(0.008)	0.824(0.007)	0.822(0.014)
speed-up factor	1	7.6	26.2	20	13.2

Table 5.6: Results on three data sets: RGLVQ, RGLVQ with Nyström, and Patch RGLVQ (PRGLVQ) are evaluated in a repeated cross-validation. The classification accuracy, and the speedup factor for SwissProt according to the CPU time are reported.

Experiments

We evaluate the Patch and Nystöm RGLVQ on the same benchmark data sets as used for the unsupervised case in Section 5.4. We compare the results of RGLVQ for the full dissimilarity matrix, patch processing and a Nyström approximation (10%). The setting is as follows in the experiments:

- *Evaluation*: We evaluate the result by means of the classification accuracy obtained in a ten-fold cross-validation with 10 repeats (Chromosomes, Vibrio), or a 2-fold cross-validation with 10 repeats (SwissProt).

- *RGLVQ:* The prototypes are initialized randomly, and training takes place for 5 epochs. We use 49 (Vibrio), 63 (Chromosomes), and 64 (SwissProt) prototypes evenly distributed among the classes.

- *Patch processing*: For patch processing, ten patches are chosen. The value k for the k-approximation for patch processing is taken in $\{1, 3, 5\}$.

- *Nyström approximation*: A fraction of 10% of the data is used.

- *Implementation:* For all data sets, we use a 12 Intel(R) Xeon X5690 machine with 3.47GHz processors and 48 GB DDR3 1333MHz memory.

For *Vibrio* and *SwissProt*, the classification accuracy obtained with a linear time approximation is the same as for full RGLVQ. For Chromosomes, it decreases by 6% using patch approximation as compared to almost 25% for the Nystöm approximation. Interestingly, all patch approximations already yield a high quality when approximating the prototypes by its closest exemplar ($k = 1$). This approximation has the side effect that classes can directly be inspected in terms of this representative exemplar, i.e. interpretable models result. We measure the speed-up of the technique for the SwissProt data set which deals with close to $11,000^2$ entries. Original RGLVQ takes 24481 seconds CPU time (i.e. almost seven hours), which can be accelerated by a factor 26 to 15 minutes using patch processing – the Nyström approximation requires considerably more time.

Chapter		*Supervised*		*Unsupervised*		
		GLVQ	*RSLVQ*	*NG*	*AP*	*GTM*
2	*cost function*	margin	likelihood ratio	topographical QE	QE	dual likelihood
	complexity	$\mathcal{O}(n)$	$\mathcal{O}(n)$	online: $\mathcal{O}(n)$ batch: $\mathcal{O}(1)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
	data type	vectorial	vectorial	vectorial	vectorial	vectorial
	working style	online	online	batch/online	online	batch
4	*relational data*	$\surd$	$\surd$	[37]	similarity	[29]
5	***patch***	$\surd$	analog	[37]	$\surd$	$\surd$
	nyström	$\surd$	analog	$\surd$	-	[28]

Table 5.7: Summary of chapter 5: Patch/Nyström relational prototype-based learning methods

5.6 Conclusions

We have presented two different ways to speed-up prototype-based clustering and classification of dissimilarity data: the Nyström approximation and patch processing. We showed whether and how these techniques can be included into three representative clustering techniques, affinity propagation, relational neural gas, and relational generative topographic mapping, respectively, and one supervised technique, relational generalized learning vector quantization, summarized in Table 5.7. Assuming a fixed quality of the approximation, this way, linear instead of squared complexity can be achieved. The corresponding speed up has been presented in a large scale setting, a fraction of the popular SwissProt data set.

As demonstrated in several experiments, the approximation techniques lead to a decrease of accuracy which is, depending on the setting, only in the range of a few percentage. However, the result depends on the chosen evaluation measure, the chosen approximation technique, and the data set. It seems that patch approximation is often suited if k is chosen sufficiently large. The Nyström approximation gives good results if the quality of the approximation of the dissimilarity matrix is sufficient. This can depend on the data set and, surprisingly, it is not necessarily monotonic with respect to the fraction of the data taken into account.

Note that the two approximation techniques presented in this chapter are suited for different settings. For the Nyström approximation, it can be specified a priori which parts of the dissimilarity matrix are required for training. To be applicable, however, a representative subsample of the data need to be available a priori, thus it is not suited if streaming data are dealt with. Further, it can only be integrated into algorithms which include the dissimilarity matrix in full matrix form. Affinity propagation, which deals with these values in a distributed way, cannot be accelerated using this technique.

In contrast, patch processing requires a (right from the beginning fixed) linear part of the dissimilarity matrix located along the diagonal, and, in addition, dissimilarities of data and exemplars which are determined during training only. Hence it can be applied only in settings where additional dissimilarities can be retrieved on demand. However,

due to its way in which information is compressed, it can also deal with streaming data. Hence it seems particularly suited for life-long adaptation. Further, it constitutes a very simple meta-heuristic which is directly applicable for every clustering scheme which represents clusters in terms of exemplars, and which can deal with multiple data points in an efficient way.

Since both techniques rely on a linear subpart of the full dissimilarity matrix, they can lead to a severe information loss: in theory, it is possible that the most relevant information is located at those parts of the dissimilarity matrix which are not even computed in the approximations. Thus, there cannot exist a non-trivial upper bound on the information loss if no further assumptions on the dissimilarity matrix are assumed. However, in practice, it seems that the approximation schemes are well suited to preserve the relevant information. This is mirrored in an information loss of only a few percentage provided an appropriate approximation technique is chosen.

The best approximation technique as well as the meta-parameters such as the size of the subsamples and the number of exemplars used to approximate the prototypes are not clear a priori and the quality can differ a lot depending on the data setting. Hence it would be desirable to derive characteristics of the dissimilarity matrix which guarantee that a certain approximation technique is well suited in the given setting. This question could be tackled empirically as well as from a theoretical side. It is the subject of ongoing research.

Moreover, another more general drawback of prototype-based learning is the number of prototypes, as in the experiments we have to specify the number of prototypes for each data set beforehand. It crucially depends on the data, maybe one general rule for setting the right number is that at least each class should have one prototype. But if the data are multimodal, one prototype per class can not be sufficient. In the next chapter we will tackle this problem: by means of a statistical measure, the sufficiency of prototypes can be evaluated. If necessary, new prototypes can be added into the model so that sufficient model complexity can be obtained to more appropriately represent the data distribution.

Chapter 6

Adaptive conformal learning vector quantization

Chapter overview

Existing prototype-based algorithms have two general problems. The first one is the model complexity, i.e. the number of prototypes, which has to be predefined before training. Although this problem can be partially solved by cross-validation for an evaluation of different sizes of the model, it is unfeasible for large data sets and not suitable for online learning. The second problem is the lack of reliability about the classification decision. In this chapter using concepts from conformal prediction we tackle these two problems and propose an adaptive extension of Relational GLVQ with point wise confidence values evaluating the corresponding predictions. This chapter is partially based on the publication [85].

6.1 Introduction

Prototype-based learning enjoys a wide popularity due to its intuitive training process and interpretable model inspecting the data. Extensions such as kernel variants [77, 78], or metric learning [86] make classical methods also suitable for more complex or heterogeneous data sets. Furthermore, recent research on extending their ability to proximity data even makes this possible that they can directly deal with arbitrary dissimilarity data without necessity of an explicit embedding into some space. As elaborated in the previous chapters, this opens a more broad application area for prototype-based learning, especially in bioinformatics [39, 41, 37].

However, existing prototype-based algorithms have two general problems. The first one is the model complexity, i.e. the number of prototypes which has to be defined in advance. Second, most of them provide only crisp classification without any measure of reliability, similar to p- or q-values from statistics, which is always expected and beneficial especially in life sciences or medical applications. There are some extensions investigated to automatically adjust the number of prototypes by adding new prototypes or deleting redundant ones [33, 54, 52], but most of them are restricted to vector spaces and based on heuristics, but not in a statistical sense. Especially, they can not be directly transferred

to dissimilarity data. Also, only few attempts exist to give reliability estimates for these techniques or reject options meaning that for a given reject level and a rejection measure if the corresponding values for some data points are under this level, their predictions will not be considered [94, 23, 100], most of them are cost function based or probabilistic, not in a statistical sense.

In this chapter we keep the focus on classification tasks in the context of dissimilarity learning and solve these two open problems together by using concepts from *conformal prediction* (CP) [105] which is a statistically well founded theory and can be used for evaluating classifiers, and it has been widely used in different areas [4]. In the end, a relational prototype learner is proposed extended by conformal prediction concepts, referred to as Adaptive Conformal Relational GLVQ (AC-RGLVQ). AC-RGLVQ can be *directly* applied to dissimilarity data, providing sparse interpretable models where each classification is supported by a measure of confidence. In addition, the confidence is used for a dynamic adaptation of the model complexity during learning, growing the model complexity as required by the resulting conformal regions.

First, we introduce the basic concepts of conformal prediction, then show how to combine both together. Actually, this kind of combination can be considered as a general framework which can be straightforwardly transferred to other prototype-based supervised techniques.

6.2 Conformal prediction

Conformal prediction (CP) is a very general theory about prediction and also a new approach to obtain confidence values. Conformal prediction can be built on top of traditional algorithms, while besides predictions it also provides two measures "confidence" and "credibility" which indicate how suitable are the training data for classification of new examples. Conformal prediction has been widely used by combining with other methods such as SVM, nearest-neighbor method, ridge regression, etc. [105], also in many applications such as outlier detection, feature selection, quality assessment, etc. [4]. Because of the theoretically proved validity of conformal prediction, it constitutes an very attractive tool for many real-world problems such as face recognition, biomedical applications, etc. More applications can be found in [4]. Originally, conformal prediction is focused on online settings. For our purposes, we follow the batch version. In the following we will outline the main idea of conformal prediction. A more comprehensive treatment of this topic is provided in the book [105].

Generally, we follow the notation of conformal prediction as in [105, 91]. Denote the labeled training data $\mathbf{z}_i := (\mathbf{x}_i, y_i) \in \mathbb{Z} = \mathbb{X} \times \mathbb{L}$. Furthermore let $\mathbf{x}_{n+1}$ be a new data point with unknown label y_{n+1}, i.e. $\mathbf{z}_{n+1} := (\mathbf{x}_{n+1}, y_{n+1})$. For given training data $(\mathbf{z}_i)_{i=1,\ldots,n}$, an observed data point $\mathbf{x}_{n+1}$, and a chosen *significance level* ϵ, the *conformal prediction* (CP) computes an $(1-\epsilon)$-*prediction region*

$$\Gamma^{\epsilon}(\mathbf{z}_1, \ldots, \mathbf{z}_n, \mathbf{x}_{n+1}) \subseteq \mathbb{L}$$

consisting of a number of possible label assignments with probability. It ensures that

if the data $\mathbf{z}_i$ are *exchangeable*[1], i.e. the distribution of a data sequence $\mathbf{z}_1, \ldots, \mathbf{z}_n$ is invariant under permutations, then

$$P(y_{n+1} \notin \Gamma^\epsilon(\mathbf{z}_1, \ldots, \mathbf{z}_n, \mathbf{x}_{n+1})) \leq \epsilon \tag{6.1}$$

holds for each distribution of $\mathbb{Z}$, which means that the set Γ^ϵ that contains y_{n+1} with probability at least $1-\epsilon$. One says that the predictor is *valid*. It is important to mention that the probability is unconditional, such that if we repeat the process of drawing data $(\mathbf{z}_1, ..., \mathbf{z}_n, \mathbf{x}_{n+1})$ and generating Γ^ϵ a number of s times we will find that in at most $\epsilon \cdot s$ cases the real label y_{n+1} is not among the predicted labels of Γ^ϵ, if statistical fluctuations are ignored.

6.2.1 Prediction region and non-conformity measure

To compute the conformal prediction region Γ^ϵ, a *non-conformity measure* $A(\mathcal{D}, \mathbf{z})$ is fixed, where $\mathcal{D}=\{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$ denotes the data set and $\mathbf{z} = (\mathbf{x}, y)$ denotes an observation. It is used to calculate a *non-conformity value* (denoted as a_x^y) that estimates how this observation $\mathbf{z}$ fits to given representative data $\mathcal{D}$. In theory, any measure could be used, providing a nontrivial result for suitable choices only. It is the part of the method that can incorporate detailed knowledge about the data distribution. In [105] a number of non-conformity measures with respect to SVM, nearest neighbors, neural networks, etc. has been discussed. Given a non-conformity measure A, a significance level ϵ, a set of examples $\mathcal{D}$, a new object $\mathbf{x}_{n+1}$ and a possible label y, it is decided whether y is contained in $\Gamma^\epsilon(\mathbf{z}_1, \ldots, \mathbf{z}_n, \mathbf{x}_{n+1})$ according to Algorithm 3.

As an example of non-conformity measure, since we focus on prototype-based methods, for a given labeled data point $\mathbf{z}$ and a trained GLVQ model W based on the training data $\mathcal{D}$, we choose the following function based on the model W as non-conformity measure throughout this thesis

$$A(\mathcal{D}, \mathbf{z}) := A(W, \mathbf{z}) = a_{\mathbf{x}}^y = \frac{d_+(\mathbf{x})}{d_-(\mathbf{x})}\,. \tag{6.2}$$

with $d_+(\mathbf{x})$ being the distance between $\mathbf{x}$ and the closest prototype labeled y, and $d_-(\mathbf{x})$ being the distance between $\mathbf{x}$ and the closest prototype labeled differently than y where distances are computed according to Eq. (4.2) for relational setting. For vectorial setting, the Euclidean distance can be directly used. This choice of non-conformity measure is very similar to the case for nearest neighbors as in [91] in which the ratio of the distances to two nearest neighbors with different labels (in the sense that one has the same label as the data and another has a different one) is as non-conformity measure. We expect that values $a_{\mathbf{x}}^y$ are small for data $\mathbf{z}$ for which the prediction has high confidence, but it is large if the label does not comply with data.

Depend on the non-conformity values, the p-value for each possible label (row 8 in Algorithm 3) can be used in two different ways: First, as shown in Algorithm 3 row 10, for given significance level ϵ (typically a small constant, e.g. 0.05), the prediction region

[1] *Exchangeability* is a weaker condition than data being i.i.d. which is readily applicable to the online setting as well, see [105] for more details.

Algorithm 3 Conformal Prediction (CP)

function CP($\mathcal{D}$, $\mathbf{x}_{n+1}$, ϵ)
  for all $y \in \mathbb{L}$ **do**
    $\mathbf{z}_{n+1} := (\mathbf{x}_{n+1}, y)$
    for $i = 1, \ldots, n+1$ **do**
      $\mathcal{D}_i := \{\mathbf{z}_1, \ldots, \mathbf{z}_n, \mathbf{z}_{n+1}\} \backslash \{\mathbf{z}_i\}$
      $a_i^y = A(\mathcal{D}_i, \mathbf{z}_i)$ ▷ non conformity of $\mathbf{z}_i$ against $\mathcal{D}_i$ wrt. label y
    end for
    $p_{n+1}^y = \frac{|\{i=1,\ldots,n+1 \mid a_i^y \geq a_{n+1}^y\}|}{n+1}$ ▷ p-value of label y
  end for
  return $\Gamma^\epsilon := \{y : p_{n+1}^y > \epsilon\}$
end function

can be obtained which contains the labels whose p-values are larger than ϵ such that one can be $1-\epsilon$ confidence that the true label will be included. Another way is that for each new data point, the predicted label together with a *confidence* and a *credibility* measure about this classification can be obtained, which we will discuss in the following.

6.2.2 Confidence and credibility

The prediction region $\Gamma^\epsilon(\mathbf{z}_1, \ldots, \mathbf{z}_n, \mathbf{x}_{n+1})$ stands in the center of conformal prediction. For a given significance level ϵ, it contains the possible labels of $\mathbb{L}$ that ensure (6.1). But how can we use it for prediction?

Suppose we use a meaningful non conformity measure A such as (6.2). If the value ϵ is approaching 0, a conformal prediction with almost no errors is required, which can only be satisfied if the prediction region contains all possible labels. If we raise ϵ, we allow errors to occur and as a benefit the conformal prediction algorithm excludes unlikely labels from the prediction region, increasing its information content. In detail those labels are discarded for which the p-values are less than or equal to ϵ. Hence only a few $\mathbf{z}_i$ are as non conformal as $\mathbf{z}_{n+1} = (\mathbf{x}_{n+1}, y)$. This is a strong indicator that $\mathbf{z}_{n+1}$ does not belong to the distribution $\mathbb{Z}$ and so y seems not to be the right label. If one further raises ϵ, only those labels y remain in the conformal region that can produce a high p-value meaning that the corresponding $\mathbf{z}_{n+1}$ is rated as very typical by A.

So one can trade significance level ϵ against information content. The most useful prediction is those containing exactly one label. Therefore, given an input $\mathbf{x}_i$ two error rates are of particular interest, ϵ_1^i being the smallest ϵ and ϵ_2^i being the largest ϵ so that $|\Gamma^\epsilon| = 1$. ϵ_2^i is the p-value of the best label (i.e. the largest p-value) and ϵ_1^i is the p-value of the second best label (i.e. the second largest p-value). Thus, typically, a conformal predictor outputs the label y which describes the prediction region for such choices ϵ, i.e. $\Gamma^\epsilon = \{y\}$, and the classification is accompanied by the two measures

$$\textit{confidence} : cf_i = 1 - \epsilon_1^i = 1 - p^{y_{2\text{nd}}} \tag{6.3}$$

$$\textit{credibility} : cr_i = \epsilon_2^i = p^{y_{1\text{st}}} \tag{6.4}$$

Confidence says something about being sure that the second best label and all worse ones are wrong, i.e. the higher the confidence value for a label the less likely for any other label being the true label. *Credibility* says something about to be sure that the best label is right such that the data point is typical with respect to the given data and not an outlier.

An example is shown in Figure 6.1: The data consist of two well-separated clusters. The data points around the centers (in the dashed circles) have higher credibility and higher confidence than the data farther from the centers. The data points that are a bit farther from the centers but not outliers (in the dashed ellipses) have higher credibility but lower confidence (because they are nearer to the other cluster than the data around the centers). Furthermore there are two types of outliers: (i) the data points are far away from the centers but nearer to the other cluster than other data points in the same cluster, so they have lower credibility and lower confidence. (ii) the data points are far away from the centers and even farther away from the other cluster than other data points in the same cluster, so they have lower credibility and higher confidence.

The non-conformity measure has a direct impact on the efficiency of the prediction region. A good, informative measure will exclude wrong labels for low significance levels and will reject typical data only for high significance levels, meaning that $\epsilon_2^i - \epsilon_1^i$ is large for typical data $\mathbf{x}_i$. That means, that a good measure can give useful information already for low significance level ϵ_1^i and on the other hand one would have to face up a high average significance level ϵ_2^i to exclude the right label from the prediction region.

We would like to point out that the concept of conformal prediction permits pointwise measures of confidence which change if the training data is adapted, also if the decision boundaries remain the same. This means, that similar as in classical statistics, more densely populated training regions permit a better confidence in a decision.

6.2.3 Inductive conformal prediction

Because the original conformal prediction has to be done for all leave-one-out multi-sets for each of the test objects with all possible labels $(\mathbf{x}_{n+1}, y)$ as shown in Algorithm 3, it would entail high computational costs, especially for large data sets, To overcome this problem, extensions of conformal prediction have been published, i.e. *Inductive Conformal Prediction* (ICP) [105, 102] and *Cross Conformal Prediction* (CCP) [103].

Inductive conformal prediction (Algorithm 4) divides the training data into two subsets: *proper training set* and *calibration set*. The model is trained on the proper training set and then used to calculate the non-conformity values of the calibration set. For new data points, classification takes place only based on the non-conformity of the calibration set. In Algorithm 4, given a model trained using $\mathcal{D}_{tr}$, for each entry in the calibration set $\mathcal{D}_{cal}$ a non-conformity value is calculated (line 4-6). Based on these non-conformity values a p-value is estimated for each possible label and a test point (line 7-10). For classification using the conformal classifier, the label of a test item will be finally predicted as the label with the largest p-value. This refers to the label set provided by the conformal predictor which contains only one label. More complex schemes, by analyzing for example label sets with more than one label would be possible as well, but are not further considered here. The confidence value (cf) is given as one minus the

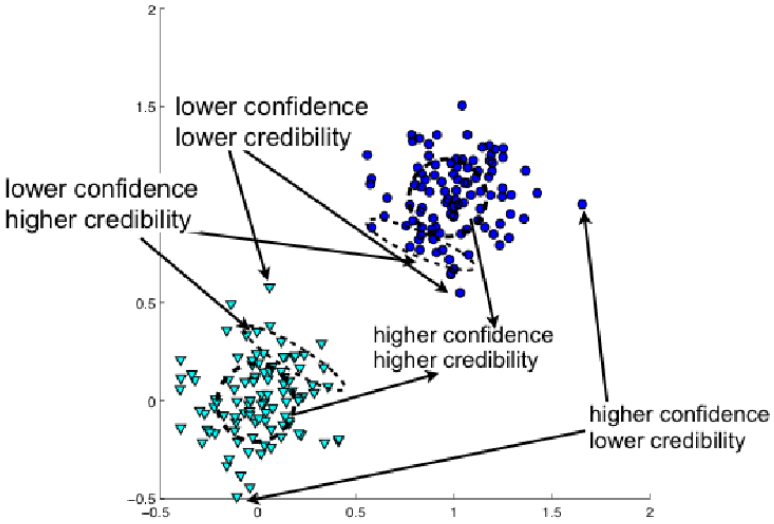


Figure 6.1: An example about confidence and credibility: two Gaussians indicating two clusters for two classes

second largest p-value (6.3) and the credibility (cr) is the largest p-value of this item (6.4), as described in section 6.2.2. As pointed out by [102] the size of the calibration set should be reasonably large to cover the data statistic.

Although ICP is computationally more efficient, since the training process only has to be done once, it is predictively less efficient in comparison to the original conformal prediction, in which the training set serves as proper training set and also as calibration set. To avoid this problem another approach, cross-conformal prediction has been proposed, which combines cross-validation with inductive conformal prediction. During the cross-validation process (by taking one fold as calibration set and the remaining folds as proper training set) the data statistic of the whole training set is accumulatively considered, finally the non-conformity of each calibration is merged to classify new data, see [103] for more details.

6.2.4 Validity of conformal predictors

It has been proved that conformal predictors are (unconditionally) *valid* (Proposition 4.1 in [105]) in the sense that in a long run the probability that an error occurs - the underlying label is not in the prediction region (6.1) - dose not exceed ϵ at each chosen confidence level $1-\epsilon$. Non-conformity measures do not effect the validity of the conformal predictor, but its *efficiency*, in the sense that the prediction regions should be as small as possible. This can be achieved only by choosing meaningful non-conformity measures such as (6.2). Inductive conformal predictors as a computationally efficient version also satisfy the same property of validity [105]. Additionally, the paper [102] studied various versions of conditional validity of inductive conformal predictors and their modifications.

Algorithm 4 Inductive Conformal Prediction (ICP)

function ICP($\mathcal{D}$, $\mathbf{x}_{n+1}$, ϵ)
  $\mathcal{D}_{tr} \cup \mathcal{D}_{cal} := \mathcal{D}$ ▷ split $\mathcal{D}$ into proper training set $\mathcal{D}_{tr}$ and calibration set $\mathcal{D}_{cal}$
  $W :=$ the model trained on $\mathcal{D}_{tr}$ ▷ train the model on $\mathcal{D}_{tr}$
  for all $\mathbf{z}_i \in \mathcal{D}_{cal}, i = 1, \ldots, |\mathcal{D}_{cal}|$ **do**
    $a_i^{y_i} = A(W, \mathbf{z}_i)$ ▷ non conformity of the calibration set based on W: e.g. (6.2)
  end for
  for all $y \in \mathbb{L}$ **do**
    $\mathbf{z}_{n+1} := (\mathbf{x}_{n+1}, y)$
    $a_{n+1}^{y} = A(W, \mathbf{z}_i)$ ▷ non conformity of $\mathbf{z}_{n+1}$ based on W: e.g. (6.2)
    $p_{n+1}^{y} := \frac{|\{i=1,\ldots,n \mid a_i^{y_i} \geq a_{n+1}^{y}\}|}{n+1}$ ▷ p-value w.r.t label y using the non conformity of $\mathcal{D}_{cal}$
  end for
  return $\Gamma^{\epsilon} := \{y : p_{n+1}^{y} > \epsilon\}$
end function

A note on the literature of conformal prediction

The basic idea of conformal prediction was first published in [104, 82] in 1999. Later on, this approach was compared to some similar frameworks such as Bayesian framework [64], at that time it was called "typicalness framework". Additionally, some applications for classification and regression tasks in combinations with other learning methods such as SVM were published, e.g. [67, 76]. At the same time, duo the computational inefficiency of the original approach, inductive conformal prediction was proposed in [69]. Few years later, in 2005, the first book of conformal prediction [105] came up, which constitutes a comprehensive treatment of conformal prediction and summarizes previous published work by the authors. It first explains conformal prediction under the assumption of i.i.d. data, and then points out that this assumption can be relaxed to exchangeable data. It discusses variants from online setting to offline setting and various non-conformity measures in combination with a number of popular machine learning algorithms, and focuses more on theoretical details and proofs such as the validity. In 2008, a tutorial [91] about this topic was published, which emphasizes the validity of conformal prediction for online setting, the meaning of exchangeability and the generalization of other online compression models. Recently published work such as [103] proposed the cross conformal prediction which tackles the predictive inefficiency issue of inductive CP, and the work [102] discussed conditional validity of inductive CP. The most recently published book [4] accumulated a broad range of applications of conformal prediction such as for feature selection, outlier detection, etc., as well as for a number of real-world problems.

6.3 Adaptive conformal relational GLVQ

By traditional prototype methods the model complexity, i.e. number of prototypes, has to be defined beforehand, and it highly depends on the data distribution. It is difficult

Algorithm 5 Adaptive conformal relational GLVQ

init: $W :=$ randomly initialized
  Define a proper training set $\mathcal{D}_{tr}$, a calibration set $\mathcal{D}_{cal}$ and a test set T_{te}
  $T_{tr} :=$ randomly selected 80% of $\mathcal{D}_{tr}$
  $T_{com} :=$ the remaining proper training data as complexity set
  $W_{new} := \emptyset$; $W_{best} := W$; $\mathcal{B} = \emptyset$; $itr = 0$; $itr_{best} = 0$; $max_{itr} = 100$; $win_{max\_itr} = 10$; $acc_{best} = 0$
repeat
  $W := W \cup W_{new}$ ▷ retraining step
  $W :=$ retrain W using RGLVQ on T_{tr};
  $acc :=$ evaluation of W on T_{tr}
  if $acc - acc_{best} \geq 1\%$ **then**
    $W_{best} = W$; $acc_{best} = acc$; $itr_{best} = 0$;
  else
    $itr_{best} = itr_{best} + 1$;
  end if
  ▷ adaptation of the model complexity step
  $\mathcal{A}_{T_{tr}} := \{a_i^{y_i} \mid i \in T_{tr}\}$, ▷ non-conformities of T_{tr}
  $\mathcal{A}^{\mathbb{L}}_{T_{com}} := \{a_i^y \mid i \in T_{com}, y \in \mathbb{L}\}$ ▷ non-conformity values of T_{com} $\forall y \in \mathbb{L}$
  $\mathcal{P}^{\mathbb{L}}_{T_{com}} := \{p_i^y \mid i \in T_{com}, y \in \mathbb{L}\}$ ▷ p-values of T_{com} for all possible labels based on $\mathcal{A}_{T_{tr}}$ and $\mathcal{A}^{\mathbb{L}}_{T_{com}}$
  $\mathcal{CF}_{T_{com}} := \{cf_i \mid i \in T_{com}\}$, $\mathcal{CR}_{T_{com}} := \{cr_i \mid i \in T_{com}\}$
  generate $\mathcal{B}$
  generate new prototypes W_{new} from $\mathcal{B}$
until $itr = max_{itr}$ or $itr_{best} = win_{max\_itr}$ or $|\mathcal{B}| < 1\% \cdot |T_{com}|$
$\mathcal{A}_{\mathcal{D}_{cal}} := \{a_i^{y_i} \mid i \in \mathcal{D}_{cal}\}$ ▷ inductive conformal prediction process
$\mathcal{A}^{\mathbb{L}}_{T_{te}} := \{a_i^y \mid i \in T_{te},\ y \in \mathbb{L}\}$,
$\mathcal{P}^{\mathbb{L}}_{T_{te}} := \{p_i^y \mid i \in T_{te},\ y \in \mathbb{L}\}$ ▷ p-values of T_{te} for all possible labels based on $\mathcal{A}_{\mathcal{D}_{cal}}$ and $\mathcal{A}^{\mathbb{L}}_{T_{te}}$
return labels with largest p_i^y for each $i \in T_{te}$

to find the appropriate number of prototypes for different data sets.

Here, we use the additional information provided by conformal prediction to automatically adapt the structural complexity of the model. We generally follow the concept of inductive conformal prediction, but with a small modification for adaptation of the model complexity. We split the data set into multiple subsets, where we assume that each of them should be reasonably large to cover the data statistic. The *proper training set* consists of two subsets, $\mathcal{D}_{tr} := T_{tr} \cup T_{com}$, where T_{tr} is used to train the classifier in a standard manner and T_{com}, called *complexity set*, is used to adapt the complexity of the trained model on T_{tr}. The calibration set is $\mathcal{D}_{cal}$ used during the prediction to calibrate the p-values. Additionally, we will refer to the test set as T_{te} and assume that the labels of T_{te} are unknown and have to be predicted by the classifier. In classical inductive conformal prediction the model is trained only *once* based on $\mathcal{D}_{tr}$, providing a

general classification rule, and the data of $\mathcal{D}_{cal}$ are used to calculate the p-values which are taken to calculate the confidence and credibility measures for unknown data T_{te}. In the original scheme T_{com} does not exist and is subsumed by $\mathcal{D}_{tr}$.

Algorithm 5 consists of three steps. Step one, covering lines 1-5, is the initialization phase where the data are divided into four subsets as described before, T_{tr}, T_{com}, $\mathcal{D}_{cal}$ and T_{te}, and some basic variables are initialized. Step two (lines 6-22) covers the training of RGLVQ, which is repeated each time the model complexity is adapted. First a RGLVQ model is learned, based on the current prototype representations given in W and using the training data T_{tr}. The optimized prototype representation is kept when there is a substantial improvement in the prediction accuracy on the training data T_{tr}. Further we test for data regions not well covered by the model using T_{tr} and T_{com}, (lines 15-20). This triggers a model complexity adaptation. The algorithm iterates until the stopping criteria are met: line 22. The representation of the prototypes summarized in W is the matrix of the α coefficients used in (4.2) and is based on T_{tr} and T_{com}. The size of this matrix (number of columns) is adapted in each complexity modeling step. Eventually, in step three, the obtained optimized prototype representation W_{best} is used to predict the label and confidence or credibility values of the points from set T_{te} using $\mathcal{D}_{cal}$ in accordance to the schema in Algorithm 4 and by using (6.2) as the non-conformity measure.

Depending on the size of available data we can either use a full inductive conformal setting in the complexity adaptation and model prediction phase or limit inductive conformal prediction only to the model prediction. The last one means that the relational prototype classifier is not using inductive conformal prediction during the complexity adaptation but only in the prediction of the items from test set T_{te} using the calibration set $\mathcal{D}_{cal}$. Accordingly, for model complexity adaptation based on the complexity set T_{com} we would not use a calibration set.

Alternatively we can use inductive conformal prediction also during the complexity adaptation of the relational prototype classifier. This however requires an additional calibration set in Algorithm 5, line 19. In the following we discuss the simplified case where the model complexity adaptation is based on T_{tr} and T_{com} only.

As discussed before the available data are divided into multiple subsets, for training (T_{tr}), complexity (T_{com}), calibration ($\mathcal{D}_{cal}$) and test (T_{te}). We use 80% of the proper training data $\mathcal{D}_{tr}$ as T_{tr} to train the model and 20% as T_{com} to estimate the suitability of the current model, or the model complexity, by means of conformal prediction. Note that in this case we use a simplified version of conformal prediction in which we ignore all leave-one-out multisets and train the model on the whole T_{tr}. That means training only has to be performed once. The reasons thereof are: First, the locations of the prototypes depend on the whole data distribution, and will not be widely affected by a single data point. Secondly, the information loss will be minimal if the size of training data is sufficiently large, in this case adding a data point but leaving out another data point will not really affect the learning results. The calibration set $\mathcal{D}_{cal}$ and the test set T_{te} are left out and used only in the prediction phase of the final trained model and not during the model complexity adaptation.

Adaptation of model complexity

For T_{tr} and T_{com}, we compute non-conformity values according to (6.2). These values are used to calculate p-values for T_{com} (here an alternative calibration set can be used to get unbiased p-value estimates for T_{com} given a large data set). This provides point estimates for confidence and credibility of the classifier. We collect the set of points $\mathcal{B}$ with low credibility and/or confidence.

A low confidence is given if $(1 - \epsilon_1^i) \leq \zeta_1$, where ζ_1 is a user defined threshold, for example above the upper quartile of confidence values for the second best label. A low credibility is observed for $\epsilon_2^i \leq \zeta_2$, where ζ_2 is another threshold, e.g below the first quartile of confidence values for the best label. Hence we define the so-called low confidence/credibility region $\mathcal{B}$

$$\mathcal{B} = \{\mathbf{x}_i : cf_i \leq \zeta_1 \vee cr_i \leq \zeta_2\} \tag{6.5}$$

If $|\mathcal{B}|$ is large (in our case we take a threshold of $\geq 1\% \cdot |T_2|$), the complexity of the classifier is not yet sufficient. Hence, this parameter and ζ_1, ζ_2, control the sparsity of the model. For the data considered in the experimental section the threshold of $|\mathcal{B}|$ is in the range of 4 to 10. A new prototype is created and set to the representative data point (median) of $\mathcal{B}$. Here the notion "median" refers to the values cf_i and cr_i in this context. For both, we determine the (one dimensional) median cf_m and $cr_{m'}$, respectively, and represent the set $\mathcal{B}$ by one (if $m = m'$) or two (if $m \neq m'$) exemplars which cause these values. This step automatically adapts the model complexity. In the retraining step the new prototype will be trained on T_{tr}. We refer to this method as *Adaptive Conformal Relational GLVQ* (AC-RGLVQ).

Validity

The validity of inductive conformal prediction can be directly transferred to this approach, since generally we follow the concept of inductive conformal prediction using the same proper training set and calibration set as no such a modification was made. In this approach we split an additional subset T_{com} from the proper training set $\mathcal{D}_{tr}$ for adjusting the model complexity, so our model is directly based on T_{tr} (for retraining) and indirectly on T_{com} (for finding new prototypes), in an iterative manner.

Sparse approximation of prototypes

Since relational LVQs represent prototypes indirectly by means of coefficients α_{ij} (4.1) which depend on all data points in the pseudo-Euclidean space, they can not be directly interpreted. In Section 4.3.2 and 5.2, an approximation technique called *k-approximation* is discussed which approximates prototypes in pseudo-Euclidean space by its k closest data points. Unlike prototypes, these data points can be directly inspected. We will see in experiments that the resulting classification accuracy is still quite good for the approximated models with k=1 and we present an example showing the interpretability of the result.

Relational LVQ (just as SVM) depends on the full proximity matrix and thus displays quadratic time and space complexity. Depending on the chosen dissimilarity, the main

computational bottleneck is given by the computation 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), such that a computation of the full dissimilarities for some thousand points as in the subsequent examples, would already lead to a computation time of more than some days (Intel Xeon QuadCore 2.5 GHz, alignment done by Smith-Waterman or FASTA) and a storage requirement of some 100 Megabyte. As mentioned in Sections 5.3 and 5.5 efficient approximation strategies based on the Nyström approximation can be used. Here, we use the k-approximation to obtain interpretable models and consider full similarity and dissimilarity matrices during training. The k-approximation is also extremely helpful in the test case because (dis-)similarities of the test point need only be calculated to very few training samples.

6.4 Experiments

We first evaluate AC-RGLVQ on a artificial data set *checkerboard data*, then on four biomedical data sets: *ProDom*, *Protein*, *SwissProt10* and *Bacteria*.

Checkerboard data

Initial experiments were done for the simulated *checkerboard data*, with known vector representation. It consists of two classes with 1250 points, in two dimensions and 5×5 clusters (Figure 6.2 (left)). The dissimilarity matrix D was obtained using the Euclidean distance. RGLVQ can learn this data only if the prototypes are initialized near the centers of the multi-modal distributions, provided a sufficient number of prototypes. The AC-RGLVQ on the other hand automatically adapts its model complexity according to the introduced schema, leading to an effective model with a minimum initialization of one prototype per class only. We observe that the number of prototypes is slightly above the true number of 25 clusters, but the clusters are slightly overlapping and a number of 34 prototypes is considered a good result. The runtime behavior of the confidence and credibility measure during learning is shown in Figure 6.3.

We observe that at the initial point of learning, with only two prototypes, the number of points with a low confidence is very high but the credibility is in average quite good. This is an indicator that a large number of points is wrongly assigned, since the second label maybe similar likely. Due to the small number of prototypes a reasonable number of assignments are however considered to be correct, or the credibility is rather high, which is a natural consequence, because by chance $\approx 50\%$ got the correct label. To modify the model complexity, AC-RGLVQ continuously analyzes the behavior of confidence and credibility. If one or both measures drop below the threshold an adaptation of the model complexity takes place. In the experiment above, the number of prototypes increased step wise, leading to a higher confidence on average. The credibility on the other hand suffers, because there are more similar prototypes (actually, those with the same label), which are alternative clusters for the considered point. Figure 6.3 also shows that the approach shows a convergent behavior which is also caused by limiting the minimal cluster size (if $|\mathcal{B}|<1\% \cdot |T_{com}|$ terminates the program (Algorithm 5 line 22)). As the final prediction accuracy for AC-RGLVQ$_1$ we get $96.48\% \pm 3.56$ which is a very good result.

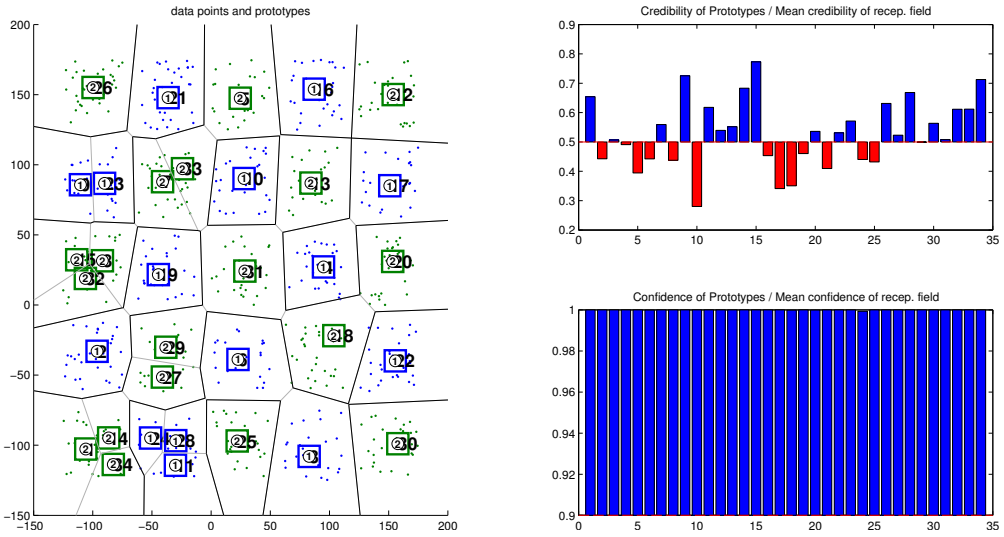


Figure 6.2: Typical result of the AC-RGLVQ for the two class checkerboard data, with 25 clusters. The initial model contained only one prototype per class, using the described conformal prediction schema the number of prototypes are auto-adjusted to 34 with almost perfect (96.48%(3.56)) separation of the true clusters evaluated on the test. A standard RGLVQ model with the same number of prototypes, as finally obtained by AC-RGLVQ, was not able to learn the data and we got 50.72%(2.59) accuracy on the test data. The class labels are given in the circled numbers. Right: mean of credibility values and confidence values of the prototypes in the final model. Left: the final model of AC-RGLVQ.

Biomedical data sets

For further comparison, we test the algorithm on four biomedical data sets: *ProDom*, *Protein*, *SwissProt10* and *Bacteria* as described in Section 3.5. We compare our results with the reference method for dissimilarity learning, the kNN-Dissimilarity classifier (kNN-Diss) [72] (the kNN rule is applied to dissimilarities) and a support vector machine (SVM) implementation [101]. For SVM results for different preprocessing of the similarity-matrix are reported, as detailed in Section 3.3.3. The crossvalidation scheme, the kNN-Diss algorithm and the SVM have been implemented using prtools and distools [20]. The parameter C for the SVM was estimated in an internal cross validation on the training data, with a grid search $C \in [0.25, 2.5]$ with a step size of 0.25 using a linear kernel[2]. The k in kNN-Diss was auto-optimized by the distools-Toolbox, typically resulting in $k = 5$. The initial prototypes for RGLVQ and AC-RGLVQ were initialized within the class centers using random samples from the classes and optimized in the pre-described training procedure with up to 10 cycles (full training data sweeps). The initial number of prototypes is chosen according to the priorly known number of classes. We used 53

[2]For the considered data we did not observe relevant improvements using an RBF kernel or similar, in particular since, in most cases, the Gram matrix is dealt with directly.

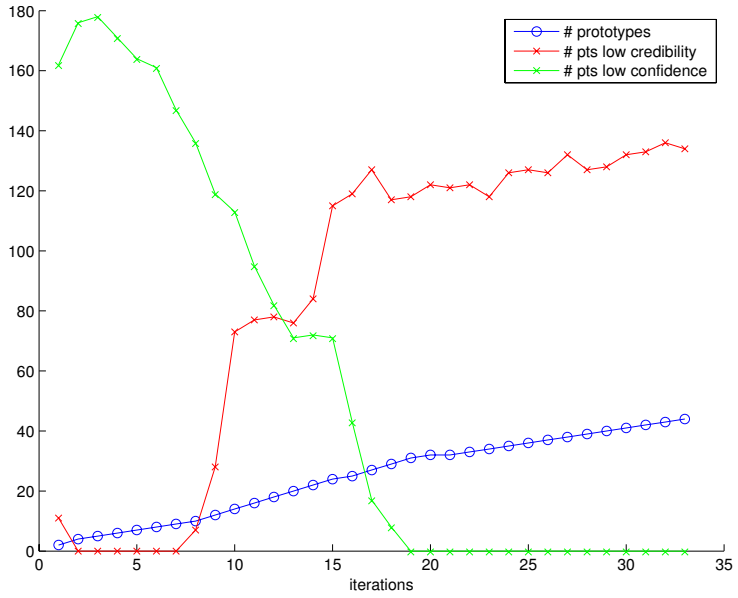


Figure 6.3: Change of the number of low confidence/credibility data points during learning as well as the number of prototypes

for ProDom, 10 for SwissProt, 4 for Protein and 30 for Bacteria.

Experiments are done within a 10−fold cross validation and with 10 repeats. We report the mean and standard deviation of the error on the test sets. For AC-RGLVQ label prediction is based on the label with the highest p-value. Further we provide values for the model complexity, by means of the number of points used to represent the prototypes or, in case of SVM, the number of support vectors in the full-class model (see Table 6.1). For SVM we provide results where the proximity matrices have been processed as mentioned before to obtain metric similarities using clipping or flipping. This procedure has a complexity of $O(n^3)$ but is necessary for kernel machines. For comparison we also tried to obtain models without a costly eigenvalue correction (indicated by **no**) but failed for SVM due to convergence issues. Instead we provide some obtained results using a core vector machine (CVM) [99]. Theoretically CVM also can be used only for psd matrices but is less sensitive with respect to non psd matrices as long as the negative eigenvalues are small or not so relevant. For the ProDom data the negative eigenvalues are a substantial part of the data space, with a similar scale as the positive eigenvalues and it was not possible to run a kernel machine on the unprocessed proximity data.

In Figure 6.4, as two examples, we show the important advantage of AC-RGLVQ by providing the confidence measure. After the model adaptation process one can analyze the test data with respect to their high/low confidence and credibility values. Instead of providing only a predicted label the pointwise measures for confidence and credibility also permit to identify the safety of this prediction and the consideration of alternative class label prediction (for example if a larger predicted label set, not only containing a single label, is similar likely). For the bacteria data set it is common see [61] to support

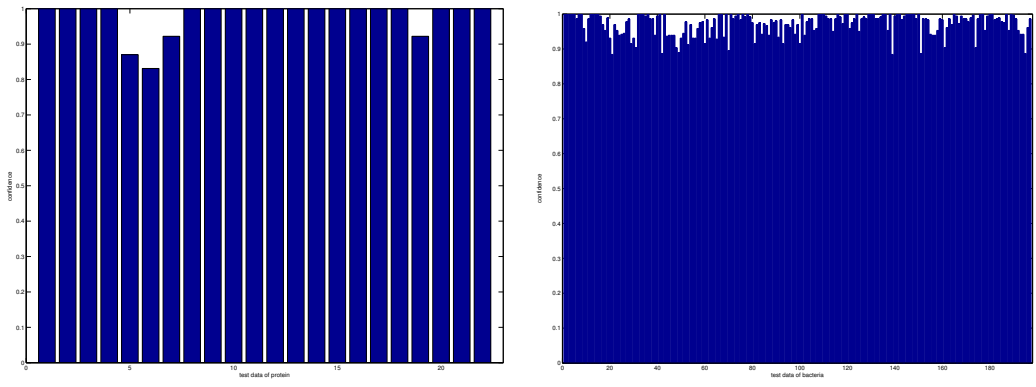
	ProDom	SwissProt10	Protein	Bacteria
RGLVQ	95.00 (1.44-Full)	93.33 (0.96 - Full)	97.91 (2.83 - Full)	**91.96** (0.25 - Full)
RGLVQ ($k = 1$)	67.24 (4.73 - 53)	94.37 (0.83 - 10)	88.73 (3.22 - 4)	42.43 (1.05 - 30)
AC-RGLVQ	86.65 (1.84 - Full)	93.74 (0.98 -Full)	**98.18** (0.41 - Full)	88.71 (0.38 - Full)
AC-RGLVQ ($k = 1$)	85.83 (2.31- 88)	94.59 (1.12 -12)	88.77 (1.14 - 4)	59.72 (1.79 - 50)
kNN-Diss	**99.44** (0.00 - Full)	98.08 (0.10 - Full)	79.48 (0.45—Full)	91.85 (0.19 - Full)
CVM-no	-	97.27 (0.74 - 33)	76.73 (8.94 - 18.8)	72.54 (2.24 - 67)
SVM-flip	97.73 (1.02 - 782)	99.43 (0.36 - 712)	98.10 (3.33 - 140)	90.48 (2.24 - 1807)
SVM-clip	98.00 (1.05 - 779)	**99.52** (0.25 - 699)	94.78 (5.70 - 165)	90.38 (2.53 - 1807)

Table 6.1: Mean test set accuracies for different dissimilarity data using the kNN-Dissimilarity classifier, SVM with clipping or flipping and AC-RGLVQ. The standard deviations are given in parenthesis, together with the (mean) number of distinctive sample points or support vectors, rounded to whole numbers, used in the models. Full - indicates that roughly all training points belong to the model.

the identification by a so called score measure. While this score is based on a simple non-metric measure of the similarity between the test item and a reference sample a conformal prediction is based on sound mathematical foundations. It would for example possible to identify regions of weak support or strong overlap in such databases.

Considering the different experiments we could not identify one single best method, with respect to the prediction accuracy. For Protein, AC-RGLVQ performed best with 20% better prediction compared to kNN-Diss and slightly better compared to SVM. For the SwissProt data the best prediction result was obtained by SVM with 99.5% compared to 94.37% using RGLVQ and 98.08% with kNN-Diss. The ProDom data have been best predicted by kNN-Diss with 99.44% which is 1.5% better than with SVM and 4% better compared with RGLVQ. As expected the Bacteria data are effectively modeled by all methods. Using k-approximation the results remain often quite good. Considering only $k = 1$ we obtain for AC-RGLVQ ($k = 1$) 86.65% (ProDom), 94.59% (SwissProt) and 88.77% (Protein) which is not as good as the best reported results, but with a significantly less number of sample points in the model. For ProDom only 3% of the points build the model, compared to $\approx 30\%$ using SVM. This effect is even more pronounced for SwissProt with 0.2% of the points used by AC-RGLVQ($k = 1$) and 12% by SVM and similar for Protein $\approx 2\%$ with AC-RGLVQ and 65% using SVM. The kNN-Diss classifier keeps roughly all points in the training data.

The reason why k-approximation for bacteria data set was found to be less effective is mainly due to the intrinsic dimensionality of bacteria data, which is very high. The



(a) confidence of protein (b) confidence of bacteria

Figure 6.4: Confidence values after training for two exemplary data sets: (a) the confidence values of the test data of protein (b) the confidence values of the test data of bacteria. Point wise confidence and credibility values can be used to identify items which are not well classified, although the proposed label is correct.

intrinsic dimensionality can be estimated by looking at the ratio of the number of absolute eigenvalues of corresponding similarity matrix above a predefined threshold to the size of the matrix. In this case we transformed the dissimilarity to similarity matrix by using double centering and took 10% of the maximal eigenvalue as the threshold. For ProDom we obtained an intrinsic dimensionality 14.59%, for SwissProt 4.35%, for Protein 7.47%, and for Bacteria 90.43%. For high intrinsic dimensionality the prototypes can not be approximated well by using small k, they may depend on all data points. But still by means of conformal prediction we got some improvement compared to the standard approach.

The number of sample points in the model is often very relevant for dissimilarity data. As mentioned before the calculation of the scores, for example by the Smith-Waterman algorithm, is very costly. To map a new training point into the models, the (dis-)similarities to all points in the training data have to be calculated, hence a small number of sample points or sparse model is very desirable.

Interpretation of AC-RGLVQ models

Considering the SwissProt data set and (AC-)RGLVQ and a k-approximation of $k = 1$ we obtain a prediction accuracy of $\approx 94\%$. This provides direct access to a very small number of associated data points, for which meta-information can be inspected.

Selecting the point associated with the k=1 approximation of the *Zinc-finger* class we can track back the original swiss/uniprot reference number. Here, we get the ID 'O13124' as most representative for the group *Zinc-finger*. This leads directly to all associated meta information in the swiss-prot database. The expert can now consider the items represented by this prototype as very similar to the 'O13124' entry, revealing potential similar chemical properties within the group *Zinc-finger* modeled by (C)RGLVQ.

For some dissimilarity data like mass-spectrometry scores in the context of bacteria

identification [61] the k-approximated prototypes can be directly linked to median representations of the underlying data, here spectra. These databases are quite new and rapidly growing, requesting for inspection tools and interpretable classifiers to ensure validity of the stored results and data.

In contrast, the kNN-Diss classifier model is quite complex and an inspection is ineffective. In case of SVM the model parameters are the support vectors, which are close to the decision boundary, and hence, in general, atypical – limiting their usefulness for an interpretation.

For new points the model now also provides pointwise estimates of the confidence and credibility of the classification according (6.2). The classification is therefore accompanied by two values indicating the safety of the classification. Points which are probably assigned to the wrong class can be identified by, most often low confidence and low credibility values. But also cases where points are equally similar to two classes, for example, can be detected and appropriate analysis of the meta-data, more specific sub-models or reject operations can be applied.

Theoretical complexity analysis

With respect to the runtime complexity, kernel methods, for example SVM needs $\mathcal{O}(n^2) - \mathcal{O}(n^3)$ operations to transfer the (dis-)similarity matrix into a valid kernel, as discussed before, and SVM training scales with $\approx \mathcal{O}(n^2)$ (using Sequential Minimal Optimization (SMO) [75]). Taking both operations together we still have a runtime complexity of $\mathcal{O}(n^2)$, for non-psd matrices.

The size of the model, given by the number of support-vector depends on the data sets. Often support vector models are large and may cover the whole training set. The relational prototype method on the other hand is trained on non-psd matrices directly and scales quadratic with the number of examples for the training and the size of prototype representations is linear with respect to the number of examples.

For AC-RGLVQ, due to the model adaptation, it has to retrain the model several times (denoted as s), normally $s \ll n$, so the retraining process of AC-RGLVQ remains $\mathcal{O}(n^2)$. Additionally, the complexity of conformal prediction can be considered as linear $\mathcal{O}(n)$, since after each retraining the non-conformity values with respect to all possible labels have to be calculated, i.e. $\mathcal{O}(s \cdot n \cdot |\mathbb{L}|)$, and usually $|\mathbb{L}| \ll n$, so the complexity of conformal prediction step is $\mathcal{O}(n)$.

The training time of kNN-Diss is $\mathcal{O}(n^2)$ with maximum model complexity. Again we would like to point out that the transformation to a valid psd matrix is not only costly, but also can degenerate the results as pointed out in [71]. The complexity of all methods is at least $\mathcal{O}(n^2)$, either due to the psd-correction or the training procedure. Our objective is not to obtain a faster training time, nor to achieve higher prediction accuracy. Instead we focus on *sparse, interpretable* models which can be trained in reasonable time and keep good generalization and query time for the test set, permitting pointwise measures of confidence.

Chapter		***Supervised***		*Unsupervised*		
		GLVQ	*RSLVQ*	*NG*	*AP*	*GTM*
2	*cost function*	margin	likelihood ratio	topographical QE	QE	dual likelihood
	complexity	$\mathcal{O}(n)$	$\mathcal{O}(n)$	online: $\mathcal{O}(n)$ batch: $\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
	data type	vectorial	vectorial	vectorial	vectorial	vectorial
	working style	online	online	batch/online	online	batch
4	*relational data*	$\surd$	$\surd$	[37]	similarity	[29]
5	*patch*	$\surd$	analog	[37]	$\surd$	$\surd$
	nyström	$\surd$	analog	$\surd$	-	[28]
6	***adaptive***	$\surd$	analog	-	-	-

Table 6.2: Summary of chapter 6: Adaptive conformal RGLVQ

6.5 Conclusions

We have defined the sparse conformal relational prototype classifier, an efficient classifier for dissimilarity data based on the relational prototype classifier RGLVQ and the conformal prediction concept. In addition to the good prediction accuracy, AC-RGLVQ automatically adapts the model complexity and outputs reliability measures of its accuracy by means of point wise confidence and credibility values, with a clear probabilistic interpretation. The experimental results show good performance compared to standard techniques but with easier access to much sparser models by means of k-approximation. In future work we will in more detail address the interpretability of the obtained models and how this can be linked to the supervised modeling of sequence databases and other application fields.

To summarize, we update the Table 6.2 accordingly and denote this kind of extension as adaptive variant of LVQs. Since conformal prediction can be considered as a wrapper method beyond the regular LVQ methods, it can be easily integrated to other LVQ methods, not only for relational data but also for vectorial setting. Because in conformal prediction concept the label information is essential, it is impossible to directly transfer this idea into unsupervised techniques.

Chapter 7

Adaptive conformal semi-supervised LVQ

Chapter overview

Most of existing semi-supervised learning (SSL) algorithms focus on vectorial data given in Euclidean space or representations by means of valid kernel matrices. A lot of real life data, especially in bioinformatics domain, are non-vectorial and non-metric given in a form of (dis-)similarities. Those data are not widely addressed in the SSL domain. In this chapter we extend a prototype-based classifier for dissimilarity data to semi-supervised tasks employing conformal prediction providing point-wise confidence measures about the classification. Parts of this chapter are based on the publication [113].

7.1 Introduction

Big data is getting more and more challenging regarding storage and analysis requirements. Besides the amount of data, only few of these data are completely labeled, and labeling of all these data is indeed very costly and time consuming. Accordingly many data sets, in life sciences for example, are only partially labeled. Techniques of data mining, visualization, and machine learning are necessary to help people to analyze those data. Especially Semi-Supervised Learning (SSL) techniques are widely used for this setting. The idea of semi-supervised learning is to learn the model not only from the labeled training data, but also to incorporate structural and statistical information in additionally available unlabeled data. A variety of SSL methods has been published [11, 111]. Most of them focus on vectorial data given in Euclidean space or representations by means of positive semidefinite (psd) kernel matrices.

A lot of real world data, like biological sequences, are non-vectorial, often non-Euclidean and given in the form of pairwise proximities, which are based on pairwise comparisons of objects providing some score-value of the (dis-)similarity of the objects. Those data are also referred to as *proximity* or *relational data*, for which a vector space is not necessarily available and there is no guarantee of metric conditions (see also Section 3.2 for more details).

Methods based on partially labelled similarity data, where the similarities are defined on a metric space discussed in [70], can be effectively handled by the semi-supervised extensions of kernel methods or other recently proposed, effective strategies [95]. However, in case of non-metric (dis-)similarity data without an explicit underlying vector representation and without requesting a metric space only few methods have proposed so far in the literature of SSL [98] and kernel techniques can be applied using some costly, potentially degenerating, transformations on the proximity data only, as described in Section 3.3.3 or [71]. In this chapter we also focus on this kind of partially labelled data.

First, let us take a glance at SSL methods. One way to categorize SSL methods is to divide the field into generative models, low-density separation methods, and graph-based techniques typically used for a classification objective. A recent introduction to SSL is given in [111]. In generative models, the most basic technique is given by *self-training* approach. A classifier is first trained on the labeled instances and is then applied to unlabeled instances. Usually, some subset of those newly labeled instances are then used together with the original labeled data, to retrain the model. The major advantages of self-training are its simplicity and the fact that it is a wrapper method. It can 'wrap' the learner without changing its inner workings. In this chapter we adopt this approach.

A more advanced approach employs expectation maximization (EM). It can be used to support the supervised learning by estimating parameters also on unsupervised data by EM [96]. In graph-based methods, the nodes of a graph represent labeled and unlabeled data, while some weights are assigned to its edges, represent the similarities of two nodes. Now one may assume that similar points share common labels, which can be propagated according to some heuristics as shown in [109, 111]. Thus, in this way labels are propagated from labeled data through the unlabeled data region. Different variations of this principle has been proposed, recently also for prototype based learning methods [17, 3] and on large scale problems [62].

Furthermore, probably the most popular semi-supervised learner in low-density separation methods is the *Transductive Support Vector Machine* (TSVM) or variants [11] thereof as the recently proposed S4VM. The semi-supervised SVM (S3VM) aims at approaching one optimal low-density separator employing unlabeled data, whereas *Safe* S3VM (S4VM) [58] tries to exploit multiple candidate low-density separators simultaneously to reduce the risk of identifying a poor separator with unlabeled data. Besides multi-kernel approaches have been recently analyzed for S3VM to incorporate additional meta-knowledge in the semi-supervised optimization [97]. While most of these methods are defined for two-class problems, employing e.g. one-vs-rest wrappers for the multi-class case, native multi-class semi-supervised learning are analyzed less intensively. A multi-class S3VM approach was proposed in [107], using a boosting strategy in [93] and employing sparse Newton-optimization [27]. Moreover, probabilistic models for semi-supervised learning based on nearest neighbor classifiers have been proposed recently [26] which allow multi-class learning. Some of these approaches are transductive like [26] and out of sample extensions are not naturally available limiting the applicability of the approaches in practice for novel data.

In this chapter we extend RGLVQ, by a self-training approach for semi-supervised tasks employing conformal prediction technique introduced in Section 6.2, which provides a confidence measure of the classification. By means of the confidence values in the self-

training process a so-called *secure region* of unlabeled data can be identified and used in the retraining, which can potentially enhance the performance of the training, and at the same time conformal prediction estimates a so-called *insecure region* of labeled data which helps to adapt the model complexity.

7.2 Semi-supervised conformal relational GLVQ

RGLVQ opens a way to directly deal with dissimilarity data [39, 41]. However, as mentioned in Chapter 6 it has two major limitations: (i) It is a crisp classifier without any additional information about the confidence of the prediction and (ii) the number of prototypes has to be defined in advance. In supervised case, these problems have been already addressed by transferring prototype-based classifier to a conformal predictor as shown in Chapter 6. In this chapter we focus on the semi-supervised case by means of self-training and propose a relational prototype-based conformal classifier with self-adaptation of model complexity based on the data with low confidence and credibility values provided by conformal prediction.

The concepts of conformal prediction has been introduced in the previous chapter, see Section 6.2 for more technical details. Note that since in this work we focus on semi-supervised problems, the size of the training set (i.e. labeled data) is normally not quite large, we can not really use the idea of inductive conformal prediction (ICP) or cross conformal prediction (CCP) as mentioned in Section 6.2.3 for our purpose. We decided to modify the original conformal prediction (Algorithm 3) in our own way: we ignore matching exactly against each data set $\mathcal{D}_i := \{\mathbf{z}_1, \ldots, \mathbf{z}_n, \mathbf{z}_{n+1}\} \backslash \{\mathbf{z}_i\}$ but instead use the whole training data (i.e. $\mathcal{D} := \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$, excluding $\mathbf{z}_{n+1}$). In this way learning must be performed only once on $\mathcal{D}$. This procedure is motivated by two facts: (1) since we intend to use prototype-based method to train the model, the positions of prototypes depend on the whole data distribution and are in general not widely affected by a single data point, (2) the information loss will be small if the number of training data is reasonably large, so that adding $\mathbf{z}_i$ but leaving out $\mathbf{z}_{N+1}$ will not affect the learning results. As in the previous chapter we use again (6.2) as non-conformity measure throughout this chapter.

First, we denote T_{lab} as labeled data and T_{unlab} as unlabeled data. Generally, in semi-supervised learning unlabeled data are used to improve the trained model based on labeled data in some way. *Self-training* [111] is a very simple approach, which takes iteratively a part of the unlabeled data with predicted labels as new training data into the retraining process to optimize the model, as shown in Algorithm 6. After the first training of model f on labeled data, the model f is then used to predict the labels of unlabeled data. A subset S of the unlabeled data together their predicted labels are selected and added to the labeled data, which builds a new larger set of labeled data. The model f is retrained on the new unlabeled data, and the procedure repeats. As pointed out by [111], the key assumption of self-training is that the predictions, at least the high confidence ones, tend to be correct. S should consist of the few unlabeled data with the most confident predictions.

In this work we combine conformal prediction with self-training to find the most confidence unlabeled data (see Algorithm 7.2). We first train the model W on labeled

Algorithm 6 Self-training

1: T_{lab}:= labeled data, T_{unlab}:= unlabeled data
2: **repeat**
3: Train model f based on T_{lab} using supervised learning
4: Apply f to T_{unlab}
5: remove a subsect S from T_{unlab} and add $\{(x, f(x))|x \in S\}$ to T_{lab}

data (T_{lab}) using RGLVQ, based on W we proceed then the conformal prediction step (line 12-17): For T_{lab} and T_{unlab}, we compute non-conformity values according to (6.2) (line 12, 13). Based on these non-conformity values a p-value is estimated for each possible label and each unlabeled point from T_{unlab} (line 14,15). For classification using the conformal classifier, the label of a unlabeled item will be finally predicted as the label with the largest p-value. This refers to the label set provided by the conformal predictor which contains only one label. More complex schemes, by analyzing for example label sets with more than one label would be possible as well, but are not further considered here. The confidence value (cf_i) is given as one minus the second largest p-value (6.3) and the credibility (cr_i) is the largest p-value of this item (6.4) (line 25-26) (for more detail see Section 6.2.2).

Data used for self-training

In oder to identify unlabeled items with high confidence predictions we define a measure cc as the product of confidence and credibility values: For a given data point $\mathbf{x}_i \in T_{unlab}$,

$$cc_i := cf_i \cdot cr_i \tag{7.1}$$

Actually, any other reasonable indicator can be used here which can detect the high confidence and high credibility values at the same time. In this case the sum of both values is not appropriate, since one of them can dominate the sum. A high cc-value of a unlabeled item indicates that with high probability its predicted label (that with the highest p-value) is the true underlying label. That means for self-training only the unlabeled data with predicted labels of high probability can be taken into the next retraining. The region which consists of these unlabeled items is referred to as '*secure region*' (denoted as $\mathcal{SR}$). To identify $\mathcal{SR}$ we take a fraction (prc) of the top cc-values of the unlabeled data[1].

Adaptation of model complexity

On the other hand we also collect a set of points of the 'labeled' data (i.e. original labeled items and the items with high cc-values labeled by previous iterations) with low credibility and confidence values, which builds a so-called '*insecure region*' ($\mathcal{ISR}$) of the training data,

$$\mathcal{ISR} := \{\mathbf{x}_i \in T_{\text{lab}} : cf_i \leq \zeta_1 \vee cr_i \leq \zeta_2\}\,. \tag{7.2}$$

[1] prc is customizable and in our experiments we set $prc = 5\%$ which is a good compromise between learning performance and efficiency.

Algorithm 7 Secure semi-supervised conformal RGLVQ

init: W : randomly initialized, $W_{\text{new}} := \emptyset$, $W_{\text{best}} := W$, $\mathcal{ISR} := \emptyset$; $\mathcal{SR} := \emptyset$; EvalSet $= T_{\text{lab}}$; $itr_{\text{best}} = 0$; $max_{\text{itr}} = 100$; $win_{\text{max\_itr}} = 10$; $acc_{\text{best}} = 0$
repeat ▷ self-training process
  $W := W \bigcup W_{\text{new}}$
  $T_{\text{lab}} := T_{\text{lab}} \cup \mathcal{SR}$, $T_{\text{unlab}} := T_{\text{unlab}} \backslash \mathcal{SR}$
  $W :=$ train T_{lab} by RGLVQ given W
  $acc :=$ evaluation of W on $EvalSet$;
  if $acc - acc_{\text{best}} \geq 1\%$ **then**
    $W_{\text{best}} = W$, $acc_{\text{best}} = acc$, $itr_{\text{best}} = 0$
  else
    $itr_{\text{best}} = itr_{\text{best}} + 1$
  end if
  $\mathcal{A}_{T_{\text{lab}}} := \{a_i^{y_i} \mid i \in T_{\text{lab}}\}$ ▷ conformal prediction step
  $\mathcal{A}^{\mathbb{L}}_{T_{\text{unlab}}} := \{a_i^{y} \mid i \in T_{\text{unlab}},\ y \in \mathbb{L}\}$
  $\mathcal{P}_{T_{\text{lab}}} := \{p_i \mid i \in T_{\text{lab}}\}$
  $\mathcal{P}^{\mathbb{L}}_{T_{\text{unlab}}} := \{p_i^{y} \mid i \in T_{\text{unlab}},\ y \in \mathbb{L}\}$
  $\mathcal{CF}_{T_{\text{lab}}} := \{cf_i \mid i \in T_{\text{lab}}\}$; $\mathcal{CR}_{T_{\text{lab}}} := \{cr_i \mid i \in T_{\text{lab}}\}$;
  $\mathcal{CF}_{T_{\text{unlab}}} := \{cf_i \mid i \in T_{\text{unlab}}\}$; $\mathcal{CR}_{T_{\text{unlab}}} := \{cr_i \mid i \in T_{\text{unlab}}\}$;
  generate $\mathcal{SR}$ of T_{unlab} based on $\mathcal{CF}_{T_{\text{unlab}}}$ and $\mathcal{CR}_{T_{\text{unlab}}}$
  generate $\mathcal{ISR}$ of T_{lab} based on $\mathcal{CF}_{T_{\text{lab}}}$ and $\mathcal{CR}_{T_{\text{lab}}}$
  generate W_{new} from $\mathcal{ISR}$ ▷ new prototype(s)
  $itr = itr + 1$
until $itr = max_{itr}$ or $itr_{\text{best}} = win_{\text{max\_itr}}$ or $T_{\text{unlab}} = \emptyset$ or $|\mathcal{ISR}| < 1\% \cdot |T_{\text{lab}}|$
return W_{best};

A low confidence value is given if the confidence value cf_i or the credibility cr_i below a user defined threshold ζ_i or ζ_2, respectively. Defined values for ζ_1 or ζ_2 can be derived from the quantiles of confidence/credibilty values as observed in the data.

The $\mathcal{ISR}$ will be represented by a new prototype as the median of $\mathcal{ISR}$ which is the same as finding new prototype(s) from $\mathcal{B}$ in Section 6.3. This step automatically adapts the complexity of the model, i.e. the number of prototypes. In the retraining this new prototype will be also trained on the new training data.

During the self-training process the training set T_{lab} is iteratively augmented by adding the secure region of the unlabeled data $\mathcal{SR}$ to itself while the unlabeled data T_{unlab} is shrunk by discarding the secure region. The performance of the retaining is evaluated based on original labeled data only. The method terminates if the improvement of the performance is not significant (less than 1%) after a certain number of iterations ($win_{\text{max\_itr}}$) or the maximal number of iterations are reached (max_{itr}) or the insecure region ($\mathcal{ISR}$) is too small or the unlabeled set T_{unlab} is empty, i.e. all unlabeled data have been considered in the retraining. Since the size of $\mathcal{ISR}$ (or ζ_1 and ζ_2) controls the complexity of the model, if $|\mathcal{ISR}|$ is sufficient large, in our case we take the boundary $\geq 1\% \cdot |T_{lab}|$, the complexity of the classifier is not yet sufficient. For the data considered in the experimental section the boundary of $\mathcal{ISR}$ is in the range of 4 to 10. The proposed

method is referred to as *Secure Semi-Supervised Conformal RGLVQ* (S3-C-RGLVQ).

Validity

Theoretically, the validity of this approach can not be guaranteed, since we neither follow the concept of original conformal prediction, nor the inductive conformal prediction. For inductive conformal prediction, due to the limited size of labeled data, it is unreasonable to divide the data into subsets whose sizes should be sufficiently large such that the data statistic can be somehow covered as pointed out in [105]. For original conformal prediction, we ignore all leave-one-out data sets and perform learning on the original training data without the new example, so the theoretical validity of conformal prediction can not be transferred to this approach. However, we empirically studied that no significant difference can be observed, because of the nature of prototype-based learning that the positions of prototypes in the data space are not sensitive to a tiny change of the training data.

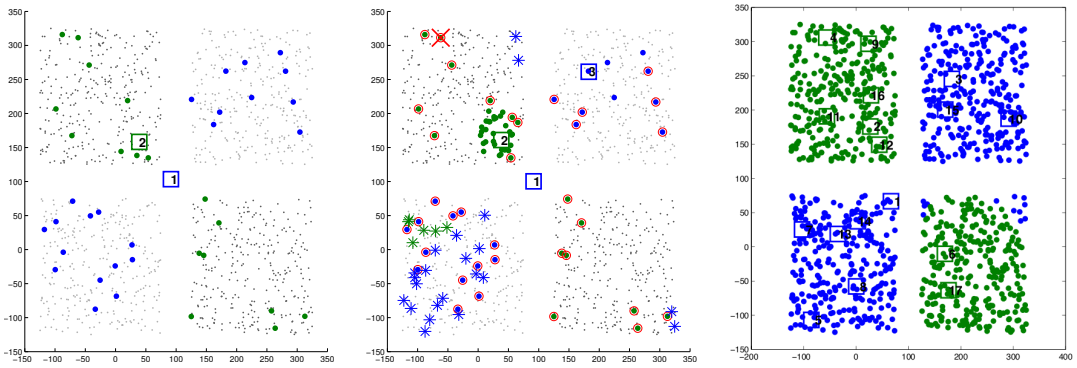
7.3 Experiments

We evaluate S3-C-RGLVQ on a large range of tasks. First, we demonstrate two artificial data sets: checkerboard data and banana-shaped data, with known vector representation to show the ability of dealing with partially labeled data, especially non i.i.d labeled data. Then we compare S3-C-RGLVQ with state-of-the-art semi-supervised SVMs on SSL binary-class benchmarks. For vectorial data the dissimilarity matrices D are obtained using the squared Euclidean distance. Additionally, five real life non-vectorial multi-class data sets from bioinformatic domain are used to compare with original RGLVQ (trained only on labeled data). For all experiments, prototypes are randomly initialized based on labeled data and one prototype per class.

Two artificial data sets

Checkboard data

The checkerboard data set consists of two classes with 1200 data points, in two dimensions and $2 \cdot 2$ clusters. The clusters with different classes distribute along each axis. We randomly select about 3% as labeled data and the remaining data as unlabeled data. RGLVQ can learn this data only if the prototypes are initialized near the centers of the multi-modal distributions, provided a sufficient number of prototypes. The S3-C-RGLVQ on the other hand automatically adapts its model complexity according to introduced schema, leading to an effective model with minimum initialization of one prototype per class only. As an example, Figure 7.1 shows some intermediate results up to convergence. We randomly initialized two prototypes only on labeled data. Figure 7.1(a) shows that after the initial training two prototypes (marked by squares) are located in the center of the labeled data. Obviously, in this case one prototype per each class is not sufficient to model the whole data space. In Figure 7.1(b) after the conformal prediction process, the secure region of unlabeled data (marked by stars) and the insecure region of labeled data



(a) Initial training (b) 3. iteration (c) the final S3-C-GRLVQ

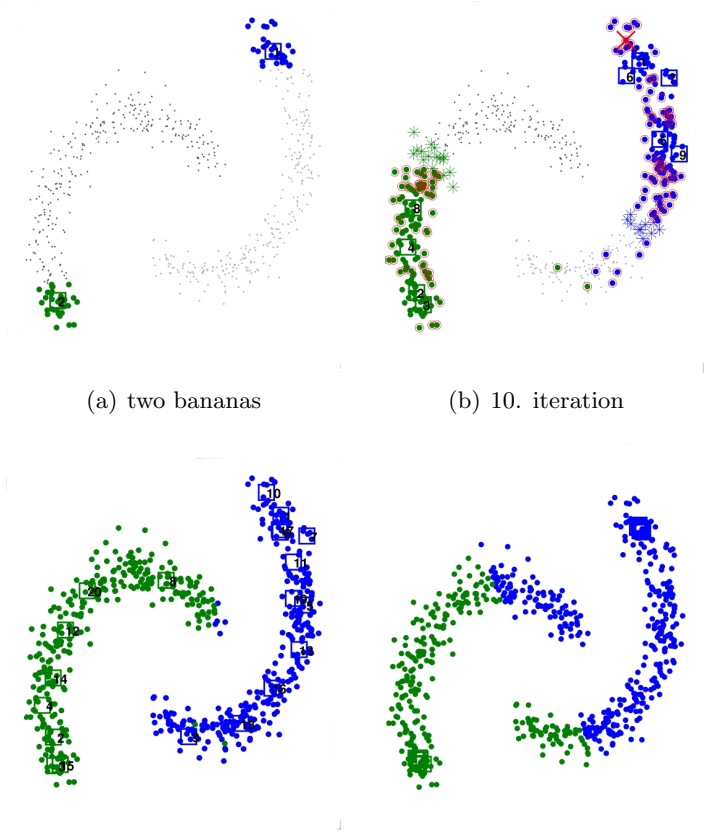
Figure 7.1: Checkerboard data set consists of green/blue labeled data and gray unlabeled data.

(marked by red circles) can be identified. To 'cover' the insecure region a new prototype (marked by red cross) is added right there.

Moreover, there are some unlabeled data misclassified by CP, which will be taken into the current retraining process. The reason thereof is that duo to the smaller number of prototypes at the early stage which are not well distributed into the multi-modal clusters, a reasonable number of points with relatively lower confidence/credibility values (i.e. lower *cc*-value) exists, which is a natural consequence, because by chance 50% got the correct label. By a larger value of the parameter '*prc*' some of these points might be considered in the next training. In this case those points can also be considered as outliers. Duo to the fact that the prototype-based method is very stable against outliers, i.e. the positions of prototypes depend on the whole data distribution and are not widely affected by a single point, the movement of the prototypes is mainly dominated by the correctly classified points and the labeled data. As shown in Figure 7.1(c), once the algorithm converges those points can be correctly assigned to their closest prototypes.

Banana-shaped data

Another simulated data set consists two banana-shaped data clouds indicating two classes. Each banana consists of 300 two dimensional data points in Figure 7.2. We select randomly non i.i.d. a small fraction (ca. 5%) of each banana as labeled data, the remaining as unlabeled data. The dissimilarity matrix D thereof is obtained by Euclidean distance again. With the same setting for checkerboard data we start with one prototype per class and train the initial model on the labeled data as shown in Fig. 7.2(a). In Fig. 7.2(b), the number of prototypes increased step-wise during the retraining process by adding new prototype in the insecure region, while by means of secure region the unlabeled data are iteratively considered. Thereby at the end the data manifold can be well studied as shown in Fig. 7.2(c). RGLVQ is trained only on labeled data with the same number of prototype for each class which S3-C-RGLVQ finally outcomes and can not learn the whole data space very well (Fig. 7.2(d)). The average accuracy



(a) two bananas

(b) 10. iteration

(c) the final model of S3-C-RGLVQ

(d) the final model of RGLVQ

Figure 7.2: Banana-shaped data

(on unlabeled data) of 10 times randomly non i.i.d. selected labeled data is reported: SSC-RPC: **94.55%**(8.38), RPC: 77.29%(13.13).

UCI two-class data sets

Furthermore, we evaluate the proposed method on different widely used benchmarks for semi-supervised learning from the UCI repository[2] and compare it with the best semi-supervised SVM with RBF-kernel taken from [58][3]. To keep the same experimental setting, we randomly select 100 examples of the data to be used as labeled examples, and use the remaining data as unlabeled data. The experiments are repeated for 12 times and the average test-set accuracy (on the unlabeled data), standard deviation and the average number of prototypes of the final model are reported in Table 7.1. Except voting

[2]http://archive.ics.uci.edu/ml/datasets.html

[3]In this paper the authors made a comprehensive comparison between different semi-supervised SVMs, e.g. TSVM, S3VM, S4SVM, etc. with linear and rbf kernels. For our experiments we pick the best result of rbf-kernel among them as reference for each data.

Two-class UCI data	S3-C-RLVQ	#Protos.	Semi-SVM^{best}(rbf)
diabetes	70.17 ± 2.32	2.3 ± 0.5	70.3 ± 2.1
german	71.61 ± 1.14	2.2 ± 0.4	71.0 ± 1.1
haberman	**73.30 ± 5.02**	3.8 ± 2.2	68.3 ± 2.8
voting	89.20 ± 0.89	2.2 ± 0.4	**92.6 ± 1.6**
wdbc	92.34 ± 1.19	2.2 ± 0.4	**93.6 ± 1.7**
austrailian	**83.22 ± 1.51**	2.1 ± 0.3	81.8 ± 1.9
breast-cancer	**96.20 ± 0.51**	2.0 ± 0.0	95.5 ± 1.0

Table 7.1: Classification accuracy (% ± std) of UCI Benchmarks for two classes problems for SSL

data, the proposed method provides comparable results for all remaining data sets.

Real life multi-class data sets

Moreover, we also evaluate the methods on five real life relational data sets from bioinformatic domain, where no direct vector embedding exists and the data are given as (dis-)similarities. They are *SwissProt10*, *Chromosomes*, *Sonatas*, *zinger* and *Vibrio*, see Section 3.5 for more details. These data sets constitute typical examples of non-Euclidean data which occur in complex systems, such as medical image analysis, mass spectrometry, and symbolic domains. In all cases, dedicated preprocessing steps and dissimilarity measures for structures are used. The dissimilarity measures are inherently non-Euclidean and cannot be embedded isometrically in a Euclidean vector space.

In general, we use the same experimental setting of UCI data, i.e. randomly select 100 examples as labeled data, the remaining as unlabeled data (with 10 repeats), prototypes are randomly initialized based on labeled data and one prototype per class. For comparison, we report the results of RGLVQ trained only on labeled data to investigate another problem for SSL, i.e. the degeneration issue as discussed by [92, 111]. In order to keep the comparisons fair the number of prototypes for each class for RGLVQ is set to the number of prototypes for each class of the final S3-C-RGLVQ model. The mean classification accuracies are reported in Table 7.3.

In all cases but one, a better classification accuracy can be obtained using conformal prediction compared to original RGLVQ only based on labeled data without consideration of additional information about unlabeled data. The chromosome is a perfectly balanced data set, it leads to the fact that the initial model based only on the labeled data is almost perfectly trained by RGLVQ, so that the potential to improve the model by considering unlabeled information in this case is very limited.

In all cases, the incorporation of information about unlabeled data into the classifier leads to an increased, at least equal, classification accuracy of the resulting model, since the additional available information can better be taken into account to optimize the class boundaries. Thus, S3-C-RGLVQ constitutes a very promising method to infer a high quality semi-supervised prototype-based classifier for general dissimilarity data sets which offers point-wise measures for confidence and credibility about the classification.

Data	S3-C-RGLVQ	RGLVQ
SwissProt	**81.06 ± 5.53**	79.37 ± 4.78
Chromosome	78.88 ± 3.28	78.78 ± 3.70
Sonatas	**77.98 ± 3.94**	71.99 ± 2.92
Zongker	**87.93 ± 0.84**	86.48 ± 1.50
Vibrio	**98.76 ± 0.47**	97.40 ± 0.84

Table 7.2: Classification accuracy (% ± std) for real life data.

Chapter		***Supervised***		*Unsupervised*		
		GLVQ	*RSLVQ*	*NG*	*AP*	*GTM*
2	*cost function*	margin	likelihood ratio	topographical QE	QE	dual likelihood
	complexity	$\mathcal{O}(n)$	$\mathcal{O}(n)$	online: $\mathcal{O}(n)$ batch: $\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
	data type	vectorial	vectorial	vectorial	vectorial	vectorial
	working style	online	online	batch/online	online	batch
4	*relational data*	√	√	[37]	similarity	[29]
5	*patch*	√	analog	[37]	√	√
	nyström	√	analog	√	-	[28]
6	*adaptive*	√	analog	-	-	-
7	***semi-super.***	√	analog	-	-	-

Table 7.3: Summary of chapter 7: Adaptive semi-supervised RGLVQ

7.4 Conclusions

In this chapter, we have developed an efficient semi-supervised classification technique for general dissimilarity data, which represents the decisions in the form of prototypes, based on conformal prediction concept and relational prototype-based classifier. It naturally inherits the merits from both techniques. Due to prototypical representation, unlike many alternative black-box techniques, it offers the possibility of a direct inspection of the classifier by humans. This technique does not require that data are embeddable into Euclidean space, rather, a general symmetric dissimilarity matrix is sufficient. Due to the properties of conformal prediction, instead of providing only a predicted label, it also permits to identify the safety of the prediction by means of point-wise measures for confidence and credibility. Thereby the 'secure' unlabeled data can be exploited and used to optimize the trained model, at the same time the 'insecure' training data can be identified and accordingly the complexity of the model is adapted.

We demonstrated the quality of the technique on different SSL data sets. As a result, a powerful semi-supervised learning algorithm can be derived, which in most cases achieves comparable results to semi-supervised SVM and with direct interpretability of the classification in term of the prototypes and works especially well for non i.i.d labeled data. Due to the multi-class capability of prototype-based method, it can directly deal with multi-class data sets. Furthermore, it does not degenerate the learning performance

by incorporating additional information of unlabeled data which is still a crucial issue in the semi-supervised learning [92, 58, 111].

One central problem of this technique as introduced above has not yet been considered in this chapter: we used a global *prc* to identify the secure region of the training data in every iteration. It may cause some uncertainty issues at the earlier stages of retraining as we have seen in the checkerboard data, if the number of prototypes is not sufficient high and the prototypes are not well distributed in the data space. In spite of the fact that this potential issue can be partially solved by the nature of prototype-based method, i.e. its stability against outliers, it should be more seriously studied, e.g. using a local *prc* for each iteration to more precisely identify the high confidence items which is a matter of ongoing research. Future work will also address the model sparsity for large scale problem and linear approximation techniques as introduced in [110].

Finally we update the Table 7.3 by adding a row representing the 'extendibility for semi-supervised setting' of the methods introduced in this thesis. For Relational RSLVQ the corresponding extension is also analog as for RGLVQ.

Chapter 8

Conclusions

In this thesis we addressed some problems of prototype-based learning:

- the applicability for dissimilarity data,
- the quadratic complexity issue for dealing with dissimilarity data due to the need of full dissimilarity matrix,
- the reliability of prototype-based classifiers,
- the predefined model complexity issue,
- the extendability of prototype-based classifiers for semi-supervised problems.

In Chapter 2 we briefly reviewed three unsupervised prototype-based techniques (NG, GTM, and AP) and two supervised techniques (GLVQ and RSLVQ) which serve as basic methods in this thesis. Those methods have some commonalities in some degree, and they also have their own characteristics. To summarize, GLVQ tries to maximize the margin between the two nearest prototypes with respect to observed data point, RSLVQ optimizes the likelihood ratio between class-specific probability density of data generated by the mixture model and probability density of the full data, NG is based on quantization error while taking the local topology into account, AP reformulates the quantization error in a another way that it can be modeled as factor graph and solved by the max-sum algorithm, and GTM is based on constrained mixture of Gaussians whose parameters can be trained by EM algorithm. Except AP, they are all online methods, which means the time complexity scales with the number of data points. AP has a quadratic complexity.

After introducing the basic methods, in Chapter 3 we talked about the data representation. Except AP which is based on similarities, all other methods work on vectorial data. Dissimilarity representation of data brings some challenges to those methods, for which we shortly reviewed the meta properties of dissimilarity data and different ways to deal with them. In Chapter 4, based on the idea of directly dealing with dissimilarities without explicit embedding from the unsupervised case, we extended it to two supervised methods, GLVQ and RSLVQ. However, the consequential problem appears in the foreground, i.e. the quadratic complexity due to the need of the full dissimilarity matrix. It

makes these methods infeasible for large data sets. In Chapter 5 we tackled this problem by means of two acceleration techniques: patch processing and Nyström approximation, which turn the quadratic effort to linear problems.

Further, we focussed on the reliability of prototype-based classifiers in Chapter 6. By means of combining the well-founded statistical concept, conformal prediction, we proposed an extension of RGLVQ which inherits advantages from both sites: From the prototype-based point of view, it can directly deal with any arbitrary symmetric dissimilarity matrix by implicit embedding of the data into a pseudo-Euclidean space. From the conformal prediction point of view, it provides also point wise confidence measure about the prediction. This allow us to get more insights about the classification, and depending on this additional information the model complexity can be accordingly adjusted.

Moreover, by given a meaningful non-conformity measure, conformal prediction can be also considered as a classifier by taking the label with largest p-value for each data point as the prediction. It allows us to very easily extend prototype-based classifiers to semi-supervised setting by using self-training approach: iteratively, a part of the unlabeled data with high confidence predicted labels are taken for retraining. As an example, we realized this approach with RGLVQ resulting in a prototype-based semi-supervised learner which can directly with dissimilarity data, automatically adjust the model complexity and enjoys a reliability measure for each prediction.

Altogether, these techniques offer a powerful framework to handle complex learning tasks, including non-vectorial data, big data sets, and partial labeling. Still, it is some way to go towards fully autonomous learning systems: as an example, all methods, although automatically adjusting the number of prototypes, are still based on crucial hyper-parameters, such as the parameter *prc* for growing strategy which defines the full size of the resulting model. Further the techniques require a homogeneous task, such as classification which extends to partial labeling, being only a first preliminary step towards full model autonomy.

Bibliography

[1] N. Alex, A. Hasenfuss, and B. Hammer. Patch clustering for massive data sets. *Neurocomputing*, 72(7-9):1455–1469, 2009.

[2] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, Oct. 1990.

[3] G. P. Amis and G. A. Carpenter. Self-supervised artmap. *Neural Networks*, 23(2):265–282, 2010.

[4] V. Balasubramanian, S.-S. Ho, and V. Vovk, editors. *Conformal Prediction for Reliable Machine Learning: Theory, Adaptations and Applications*. Morgan Kaufmann, May 2014.

[5] S. B. Barbuddhe, T. Maier, G. Schwarz, M. Kostrzewa, H. Hof, E. Domann, T. Chakraborty, and T. Hain. Rapid identification and typing of listeria species by matrix-assisted laser desorption ionization-time of flight mass spectrometry. *Applied and Environmental Microbiology*, 74(17):5402–5407, 2008.

[6] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, April 2002.

[7] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 1981.

[8] M. Biehl, A. Ghosh, and B. Hammer. Dynamics and generalization ability of LVQ algorithms. *Journal of Machine Learning Research*, 8:323–360, 2007.

[9] C. M. Bishop and C. K. I. Williams. Gtm: The generative topographic mapping. *Neural Computation*, 10:215–234, 1998.

[10] 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*, 31:365–370, 2003.

[11] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

[12] Y. Chen, E. K. Garcia, M. R. Gupta, A. Rahimi, and L. Cazzanti. Similarity-based classification: Concepts and algorithms. *Journal of Machine Learning Research*, 10:747–776, 2009.

[13] R. Cilibrasi and P. M. B. Vitányi. Clustering by compression. *IEEE Transactions on Information Theory*, 51(4):1523–1545, 2005.

[14] L. P. Cordella, P. Foggia, C. Sansone, F. Tortorella, and M. Vento. Reliability parameters to improve combination strategies in multi-expert systems. *Pattern Analysis and Applications*, 2(3):205–214, 1999.

[15] M. Cottrell, B. Hammer, A. Hasenfuss, and T. Villmann. Batch and median neural gas. *Neural Networks*, 19:762–771, 2006.

[16] K. Crammer, R. Gilad-bachrach, A. Navot, and N. Tishby. Margin analysis of the lvq algorithm. In *In: Advances in Neural Information Processing Systems 2002*, pages 462–469. MIT press, 2002.

[17] R. Cruz-Barbosa and A. Vellido. Semi-supervised geodesic generative topographic mapping. *Pattern Recognition Letters*, 31(3):202–209, 2010.

[18] A. Denecke, H. Wersing, J. J. Steil, and E. Körner. Online figure-ground segmentation with adaptive metrics in generalized lvq. *Neurocomput.*, 72(7-9):1470–1482, 2009.

[19] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, New York, 2nd edition, 2001.

[20] R. P. Duin. PRTools, march 2012.

[21] R. P. W. Duin, M. Loog, E. Pekalska, and D. M. J. Tax. Feature-based dissimilarity space classification. In D. Ünay, Z. Çataltepe, and S. Aksoy, editors, *ICPR Contests*, volume 6388 of *Lecture Notes in Computer Science*, pages 46–55. Springer, 2010.

[22] F. Farnstrom, J. Lewis, and C. Elkan. Scalability for clustering algorithms revisited. *SIGKDD Exploration*, 2(1):51–57, 2000.

[23] L. Fischer, B. Hammer, and H. Wersing. Rejection strategies for learning vector quantization. In *22th European Symposium on Artificial Neural Networks, ESANN 2014, Bruges, Belgium, April 23-25, 2014*, 2014.

[24] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315:972–976, 2007.

[25] 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 Res*, 31(3784-3788), 2003.

[26] A. K. Ghosh. A probabilistic approach for semi-supervised nearest neighbor classification. *Pattern Recognition Letters*, 33(9):1127–1133, 2012.

[27] F. Gieseke, A. Airola, T. Pahikkala, and O. Kramer. Sparse quasi-newton optimization for semi-supervised support vector machines. In *ICPRAM 2012 - Proceedings of the 1st International Conference on Pattern Recognition Applications and Methods*, volume 1, pages 45–54, 2012.

[28] A. Gisbrecht, B. Mokbel, and B. Hammer. The nystrom approximation for relational generative topographic mappings. In *NIPS workshop on challenges of Data Visualization*, 2010.

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

[30] L. Goldfarb. A unified approach to pattern recognition. *Pattern Recognition*, 17(5):575 – 582, 1984.

[31] L. Goldfarb. A new approach to pattern recognition. *Progress in Pattern Recognition*, 2:241–402, 1985.

[32] T. Graepel, R. Herbrich, P. Bollmann-Sdorra, and K. Obermayer. Classification on pairwise proximity data. In *Advances in Neural Information Processing Systems 11*, pages 438–444. MIT Press, January 1999.

[33] M. Grbovic and S. Vucetic. Learning vector quantization with adaptive prototype addition and removal. In *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*, pages 994–1001, 2009.

[34] D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 1997.

[35] B. Haasdonk and C. Bahlmann. Learning with distance substitution kernels. In C. E. Rasmussen, H. H. Bülthoff, B. Schölkopf, and M. A. Giese, editors, *DAGM-Symposium*, volume 3175 of *Lecture Notes in Computer Science*, pages 220–227. Springer, 2004.

[36] B. Hammer. *Learning with Recurrent Neural Networks*, volume 254 of *Lecture Notes in Control and Information Sciences*. Springer, 2000.

[37] B. Hammer and A. Hasenfuss. Topographic mapping of large dissimilarity datasets. *Neural Computation*, 22(9):2229–2284, 2010.

[38] B. Hammer, A. Hasenfuss, F. Rossi, and M. Strickert. Topographic processing of relational data. In *Proceedings of 6th International Workshop on Self-Organizing Maps (WSOM 2007), Bielefeld, Germany, September 3-6, 2007*, 2007.

[39] B. Hammer, D. Hofmann, F.-M. Schleif, and X. Zhu. Learning vector quantization for (dis-)similarities. *Neurocomputing*, 131(0):43–51, 2014.

[40] B. Hammer, F.-M. Schleif, and X. Zhu. Relational extensions of learning vector quantization. In B.-L. Lu, L. Zhang, and J. Kwok, editors, *Neural Information Processing*, volume 7063 of *Lecture Notes in Computer Science*, pages 481–489. Springer, 2011.

[41] B. Hammer, F.-M. Schleif, and X. Zhu. Relational extensions of learning vector quantization. In B.-L. Lu, L. Zhang, and J. T. Kwok, editors, *ICONIP (2)*, volume 7063 of *Lecture Notes in Computer Science*, pages 481–489. Springer, 2011.

[42] B. Hammer and T. Villmann. Generalized relevance learning vector quantization. *Neural Networks*, 15(8-9):1059–1068, 2002.

[43] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[44] R. J. Hathaway and J. C. Bezdek. Nerf c-means: Non-euclidean relational fuzzy clustering. *Pattern Recognition*, 27(3):429 – 437, 1994.

[45] D. Hebb. *The Organization of Behavior*. Wiley: New York, 1949.

[46] N. J. Higham. Computing a nearest symmetric positive semifefinite matrix. *Linear Algrbra and its Applications*, 103:103–118, May 1988.

[47] D. Hofmann, A. Gisbrecht, and B. Hammer. Efficient approximations of robust soft learning vector quantization for non-vectorial data. *Neurocomputing*, 147(0):96 – 106, 2015.

[48] D. Hofmann, A. Gisbrecht, and B. Hammer. Efficient approximations of robust soft learning vector quantization for non-vectorial data. *Neurocomputing*, pages 96–106, 2015.

[49] P. J. Ingram, M. P. Stumpf, and J. Stark. Network motifs: structure does not determine function. *BMC Genomics*, 7(108), 2006.

[50] A. K. Jain and D. Zongker. Representation and recognition of handwritten digits using deformable templates. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(12):1386–1391, Dec. 1997.

[51] 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 *ICPR*, pages 2676–2679, 2000.

[52] T. C. Kietzmann, S. Lange, and M. Riedmiller. Incremental grlvq: Learning relevant features for 3d object recognition. *Neurocomputing*, 71 (13-15):2868–2879, 2008.

[53] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag New York, Inc., 3rd edition, 2001.

[54] T. Kohonen, J. Kangas, J. Laaksonen, and K. Torkkola. Lvq pak: A program package for the correct application of learning vector quantization algorithms. pages 725–730. IEEE, 1992.

[55] T. Kohonen and P. Somervuo. How to make large self-organizing maps for nonvectorial data. *Neural Networks*, 15:945–952, 2002.

[56] J. Laub and K.-R. Müller. Feature discovery in non-metric pairwise data. *Journal of Machine Learning Research*, 5:801–818, 2004.

[57] J. Laub, V. Roth, J. M. Buhmann, and K.-R. Müller. On the information and representation of non-euclidean pairwise data. *Pattern Recognition*, 39(10):1815–1826, 2006.

[58] Y.-F. Li and Z.-H. Zhou. Towards making unlabeled data never hurt. In L. Getoor and T. Scheffer, editors, *ICML*, pages 1081–1088. Omnipress, 2011.

[59] D. J. Lipman and W. R. Pearson. Repid and sensitive protein similarity searches. *Science*, 227(4693):1435–1441, 1985.

[60] C. Lundsteen, J. Phillip, and E. Granum. Quantitative analysis of 6985 digitized trypsin G-banded human metaphase chromosomes. *Clinical Genetics*, 18:355–370, 1980.

[61] T. Maier, S. Klebel, U. Renner, and M. Kostrzewa. Fast and reliable maldi-tof ms–based microorganism identification. *Nature Methods*, (3), 2006.

[62] A. Mantrach, N. van Zeebroeck, P. Francq, M. Shimbo, H. Bersini, and M. Saerens. Semi-supervised classification and betweenness computation on large, sparse, directed graphs. *Pattern Recognition*, 44(6):1212–1224, 2011.

[63] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten. 'neural-gas' network for vector quantization and its application to time-series prediction. *IEEE-Transactions on Neural Networks*, 4(4):558–569, 1993.

[64] T. Melluish, C. Saunders, I. Nouretdinov, and V. Vovk. Comparing the bayes and typicalness frameworks. In L. De Raedt and P. Flach, editors, *Machine Learning: ECML 2001*, volume 2167 of *Lecture Notes in Computer Science*, pages 360–371. Springer Berlin Heidelberg, 2001.

[65] B. Mokbel, A. Hasenfuss, and B. Hammer. Graph-based representation of symbolic musical data. In A. Torsello, F. Escolano, and L. Brun, editors, *Graph-Based Representation in Pattern Recognition (GbRPR 2009)*, volume 5534 of *Lecture Notes in Computer Science*, pages 42–51, Berlin, 2009. Springer.

[66] M. Neuhaus and H. Bunke. Edit distance-based kernel functions for structural pattern classification. *Pattern Recognition*, 39(10):1852–1863, 2006.

[67] I. Nouretdinov, T. Melluish, and V. Vovk. Ridge regression confidence machine. In *In Proceedings of the Eighteenth International Conference on Machine Learning*, pages 385–392. Morgan Kaufmann, 2001.

[68] I. Olier, A. Vellido, and J. Giraldo. Kernel generative topographic mapping. In *ESANN*, 2010.

[69] H. Papadopoulos, K. Proedrou, V. Vovk, and A. Gammerman. Inductive confidence machines for regression. In T. Elomaa, H. Mannila, and H. Toivonen, editors, *ECML*, volume 2430 of *Lecture Notes in Computer Science*, pages 345–356, 2002.

[70] E. Pekalska and R. P. Duin. *The Dissimilarity Representation for Pattern Recognition. Foundations and Applications*. World Scientific, 2005.

[71] E. Pekalska, R. P. W. Duin, S. Günter, and H. Bunke. On not making dissimilarities euclidean. In A. L. N. Fred, T. Caelli, R. P. W. Duin, A. C. Campilho, and D. de Ridder, editors, *SSPR/SPR*, volume 3138 of *Lecture Notes in Computer Science*, pages 1145–1154. Springer, 2004.

[72] E. Pekalska, R. P. W. Duin, and P. Paclík. Prototype selection for dissimilarity-based classifiers. *Pattern Recognition*, 39(2):189–208, 2006.

[73] E. Pekalska, P. Paclik, and R. P. W. Duin. A generalized kernel approach to dissimilarity-based classification. *J. Mach. Learn. Res.*, 2:175–211, Mar. 2002.

[74] O. Penner, P. Grassberger, and M. Paczuski. Sequence alignment, mutual information, and dissimilarity measures for constructing phylogenies. *PLoS ONE*, 6(1):e14373, 01 2011.

[75] J. C. Platt. Fast training of support vector machines using sequential minimal optimization. pages 185–208. MIT Press, Cambridge, MA, USA, 1999.

[76] K. Proedrou, I. Nouretdinov, V. Vovk, and A. Gammerman. Transductive confidence machines for pattern recognition. In T. Elomaa, H. Mannila, and H. Toivonen, editors, *ECML*, volume 2430 of *Lecture Notes in Computer Science*, pages 381–390. Springer, 2002.

[77] A. K. Qin and P. N. Suganthan. Kernel neural gas algorithms with application to cluster analysis. In *ICPR (4)*, pages 617–620, 2004.

[78] A. K. Qin and P. N. Suganthan. A novel kernel prototype-based learning algorithm. In *ICPR (4)*, pages 621–624, 2004.

[79] V. Roth, J. Laub, J. M. Buhmann, and K.-R. Müller. Going metric: Denoising pairwise data. In S. Becker, S. Thrun, and K. Obermayer, editors, *NIPS*, pages 817–824. MIT Press, 2002.

[80] V. Roth, J. Laub, M. Kwanabe, and J. M. Buhmann. Optimal cluster preserving embedding of nonmetric proximity data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(12):1540–1551, Dec 2003.

[81] 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.

[82] C. Saunders, A. Gammerman, and V. Vovk. Transduction with confidence and credibility. In *In Proceedings of the International Joint Conference on Artificial Intelligence*, volume 2, pages 722–726, 1999.

[83] 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.

[84] F.-M. Schleif, T. Villmann, B. Hammer, and P. Schneider. Efficient kernelized prototype based classification. *Int. J. Neural Syst.*, 21(6):443–457, 2011.

[85] F.-M. Schleif, X. Zhu, and B. Hammer. Sparse conformal prediction for dissimilarity data. *Annals of Mathematics and Artificial Intelligence (AMAI)*, 2014.

[86] P. Schneider, M. Biehl, and B. Hammer. Adaptive relevance matrices in learning vector quantization. *Neural Computation*, 21(12):3532–3561, 2009.

[87] P. Schneider, M. Biehl, and B. Hammer. Distance learning in discriminative vector quantization. *Neural Computation*, 21:2942–2969, 2009.

[88] P. Schneider, M. Biehl, and B. Hammer. Hyperparameter learning in robust soft lvq. In *ESANN 2009, 17th European Symposium on Artificial Neural Networks, Bruges, Belgium, April 22-24, 2009, Proceedings*, 2009.

[89] P. Schneider, M. Biehl, and B. Hammer. Hyperparameter learning in probabilistic prototype-based models. *Neurocomputing*, 73(7-9):1117–1124, 2010.

[90] S. Seo and K. Obermayer. Soft learning vector quantization. *Neural Computation*, 15(7):1589–1604, 2003.

[91] G. Shafer and V. Vovk. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9:371–421, 2008.

[92] A. Singh, R. D. Nowak, and X. Zhu. Unlabeled data: Now it helps, now it doesn't. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *NIPS*, pages 1513–1520. Curran Associates, Inc., 2008.

[93] E. Song, D. Huang, G. Ma, and C.-C. Hung. Semi-supervised multi-class adaboost by exploiting unlabeled data. *Expert Systems with Applications*, 38(6):6720–6726, 2011.

[94] C. D. Stefano, C. Sansone, and M. Vento. To reject or not to reject: that is the question: an answer in case of neural classifiers. *IEEE Transactions on Systems, Man and Cybernetics Part C*, 30(1):84–93, 2000.

[95] A. Subramanya and J. Bilmes. Semi-supervised learning with measure propagation. *Journal of Machine Learning Research*, 12:3311–3370, 2011. cited By (since 1996) 0.

[96] J. Suzuki, A. Fujino, and H. Isozaki. Semi-supervised structured output learning based on a hybrid generative and discriminative approach. In *EMNLP-CoNLL*, pages 791–800. ACL, 2007.

[97] X. Tian, G. Gasso, and S. Canu. A multiple kernel framework for inductive semi-supervised svm learning. *Neurocomputing*, 90:46–58, 2012.

[98] M. W. Trosset, C. E. Priebe, Y. Park, and M. I. Miller. Semisupervised learning from dissimilarity data. *Computational Statistics and Data Analysis*, 52(10):4643 – 4657, 2008.

[99] I. W. Tsang, A. Kocsor, and J. T. Kwok. Simpler core vector machines with enclosing balls. In Z. Ghahramani, editor, *ICML*, volume 227 of *ACM International Conference Proceeding Series*, pages 911–918. ACM, 2007.

[100] A. Vailaya and A. Jain. Reject option for vq-based bayesian classification. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 2, pages 48–51 vol.2, 2000.

[101] V. N. Vapnik. *The nature of statistical learning theory*. Statistics for engineering and information science. Springer, 2000.

[102] V. Vovk. Conditional validity of inductive conformal predictors. *Journal of Machine Learning Research - Proceedings Track*, 25:475–490, 2012.

[103] V. Vovk. Cross-conformal predictors. *CoRR*, abs/1208.0806, 2012.

[104] V. Vovk, A. Gammerman, and C. Saunders. Machine-learning applications of algorithmic randomness. In *In Proceedings of the Sixteenth International Conference on Machine Learning*, pages 444–453. Morgan Kaufmann, 1999.

[105] V. Vovk, A. Gammerman, and G. Shafer. *Algorithmic Learning in a Random World*. Springer, New York, 2005.

[106] C. Williams and M. Seeger. Using the nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press, 2001.

[107] L. Xu and D. Schuurmans. Unsupervised and semi-supervised multi-class support vector machines. In M. M. Veloso and S. Kambhampati, editors, *AAAI*, pages 904–910. AAAI Press / The MIT Press, 2005.

[108] K. Zhang, I. W. Tsang, and J. T. Kwok. Improved Nystrom low-rank approximation and error analysis. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 1232–1239, New York, NY, USA, 2008. ACM.

[109] X. Zhu, Z. Ghahramani, and J. D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In T. Fawcett and N. Mishra, editors, *ICML*, pages 912–919. AAAI Press, 2003.

[110] X. Zhu, A. Gisbrecht, F.-M. Schleif, and B. Hammer. Approximation techniques for clustering dissimilarity data. *Neurocomputing*, 90:72–84, 2012.

[111] X. Zhu and A. B. Goldberg. Introduction to semi-supervised learning. *Synthesis Lectures on Artif. Intell. and Machine Learning*, 3(1):1–130, 2009.

[112] X. Zhu, F.-M. Schleif, and B. Hammer. Patch processing for relational learning vector quantization. In J. Wang, G. G. Yen, and M. M. Polycarpou, editors, *Advances in Neural Networks – ISNN 2012*, volume 7367 of *Lecture Notes in Computer Science*, pages 55–63. Springer, 2012.

[113] X. Zhu, F.-M. Schleif, and B. Hammer. Adaptive conformal semi-supervised learning vector quantization for dissimilarity data. *Pattern Recognition Letters*, 49(138-145), 2014.