Learning vector quantization for proximity data

Daniela Hofmann

A thesis presented for the degree of
Doctor of Natural Sciences

Cognitive Interaction Technology Center of Excellence
Theoretical Computer Science
Bielefeld University
Germany

Abstract

Prototype-based classifiers such as learning vector quantization (LVQ) often display intuitive and flexible classification and learning rules. However, classical techniques are restricted to vectorial data only, and hence not suited for more complex data structures. Therefore, a few extensions of diverse LVQ variants to more general data which are characterized based on pairwise similarities or dissimilarities only have been proposed recently in the literature.

In this contribution, we propose a novel extension of LVQ to similarity data which is based on the kernelization of an underlying probabilistic model: kernel robust soft LVQ (KRSLVQ). Relying on the notion of a pseudo-Euclidean embedding of proximity data, we put this specific approach as well as existing alternatives into a general framework which characterizes different fundamental possibilities how to extend LVQ towards proximity data: the main characteristics are given by the choice of the cost function, the interface to the data in terms of similarities or dissimilarities, and the way in which optimization takes place. In particular the latter strategy highlights the difference of popular kernel approaches versus so-called relational approaches.

While KRSLVQ and alternatives lead to state of the art results, these extensions have two drawbacks as compared to their vectorial counterparts: (i) a quadratic training complexity is encountered due to the dependency of the methods on the full proximity matrix; (ii) prototypes are no longer given by vectors but they are represented in terms of an implicit linear combination of data, i.e. interpretability of the prototypes is lost.

We investigate different techniques to deal with these challenges: We consider a speed-up of training by means of low rank approximations of the Gram matrix by its Nyström approximation. In benchmarks, this strategy is successful if the considered data are intrinsically low-dimensional. We propose a quick check to efficiently test this property prior to training.

We extend KRSLVQ by sparse approximations of the prototypes: instead of the full coefficient vectors, few exemplars which represent the prototypes can be directly inspected by practitioners in the same way as data. We

compare different paradigms based on which to infer a sparse approximation: sparsity priors while training, geometric approaches including orthogonal matching pursuit and core techniques, and heuristic approximations based on the coefficients or proximities.

We demonstrate the performance of these LVQ techniques for benchmark data, reaching state of the art results. We discuss the behavior of the methods to enhance performance and interpretability as concerns quality, sparsity, and representativity, and we propose different measures how to quantitatively evaluate the performance of the approaches.

We would like to point out that we had the possibility to present our findings in international publication organs including three journal articles [50, 53, 40], four conference papers [52, 49, 51, 33] and two workshop contributions [48, 47].

Contents

Chapter 1

Introduction

Motivation

Since electronic data sets increase rapidly with respect to size and complexity, humans have to rely on automated methods to access relevant information from such data. Apart from classical statistical tools, machine learning has become a major technique in the context of data processing since it offers a wide variety of inference methods. Today, a major part of applications is concerned with the inference of a function or classification prescription based on a given set of examples, accompanied by data mining tasks in unsupervised machine learning scenarios and more general settings as tackled for example in the frame of autonomous learning. Example applications are widespread, including network intrusion detection, image recognition, protein structure prediction, speech processing, robot navigation, and so forth. In this contribution, we focus on classification problems as one of the most widespread machine learning applications, meaning the task to classify data into a finite number of known classes based on given training examples.

There exist many different classification techniques in the context of machine learning ranging from symbolic methods such as decision trees to statistical methods such as Bayes classifiers. Because of its often excellent classification and generalization performance, the support vector machine (SVM) constitutes one of the current flagships in this context, having its roots in learning theoretical principles as introduced by Vapnik and colleagues [12]. Due to its inherent regularization of the result, it is particularly suited if high dimensional data are dealt with. Further, the interface to the data is given by a kernel matrix such that, rather than relying on vectorial representations, the availability of the Gram matrix is sufficient to apply this technique. Other top classifiers available today include random forests, neu-

ral networks, or boosting ensembles [25].

With machine learning techniques becoming more and more popular in diverse application domains and the tasks becoming more and more complex, there is an increasing need for models which can easily be interpreted by practitioners. For complex tasks, often, practitioners do not only apply a machine learning technique but also inspect and interpret the result such that a specification of the tackled problem or an improvement of the model becomes possible [94]. In this setting, a severe drawback of many state-of-the-art machine learning tools such as the SVM occurs. They act as black-boxes. In consequence, practitioners cannot easily inspect the results and it is hardly possible to change the functionality or assumptions of the model based on the result of the classifier. This is the case for all classifiers ranked best in the recent comparison [25].

Prototype-based methods enjoy a wide popularity in various application domains due to their very intuitive and simple behavior. They represent their decisions in terms of typical representatives contained in the input space and a classification is based on the distance of data as compared to these prototypes [61]. Thus, models can be directly inspected by experts since prototypes can be treated in the same way as data. Popular techniques in this context include standard learning vector quantization (LVQ) schemes and extensions to more powerful settings such as variants based on cost functions or metric learners such as generalized LVQ (GLVQ) or robust soft LVQ (RSLVQ), for example [81, 84, 88, 85]. These approaches are based on the notion of margin optimization similar to SVM in case of GLVQ [84], or based on a likelihood ratio maximization in case of RSLVQ, respectively [88]. For GLVQ and RSLVQ, learning rules which closely resemble standard LVQ2.1 result, whereby the performance is superior to this latter heuristics, in particular excellent generalization ability can be observed [85]. A few recent applications of LVQ technology can be found in the context of biomedical data analysis or life-long learning, as an example [22, 31, 59]. These applications crucially rely on the representation of the models in terms of representative prototypes which opens the way towards model interpretability and compact model representation, respectively.

With data sets becoming more and more complex, input data are often no longer given as simple Euclidean vectors, rather structured data or dedicated formats can be observed such as sequences, graphs, tree structures, time series data, functional data, relational data and so forth as occurs in bioinformatics, linguistics, or diverse heterogeneous databases. Several techniques extend statistical machine learning tools towards non-vectorial data. Kernel methods such as SVM using structure kernels, recursive and graph networks, functional methods, relational approaches, and similar [26, 82, 29, 78, 41].

Recently, popular prototype-based algorithms have also been extended to deal with more general data. Several techniques rely on a characterization of the data by means of a matrix of pairwise similarities or dissimilarities only rather than explicit feature vectors. In this setting, median clustering as provided by median self-organizing maps, median neural gas, or affinity propagation characterizes clusters in terms of typical exemplars [28, 62, 20, 70]. More general smooth adaptation is offered by relational extensions such as relational neural gas or relational learning vector quantization [39]. A further possibility is offered by kernelization such as proposed for neural gas, self-organizing maps, or different variants of learning vector quantization [75, 14, 76, 96]. By formalizing the interface to the data as a general similarity or dissimilarity matrix, complex structures can be easily dealt with. Structure kernels for graphs, trees, alignment distances, string distances, and so forth open the way towards these general data structures [73, 29].

Contribution of the thesis

This thesis will center around extensions of learning vector quantization technology towards general data structures by means of its representation in terms of proximities such as a kernel matrix or general distances. There exist a few successful attempts in this realm, such as relational GLVQ or kernel GLVQ, which extend the powerful GLVQ scheme towards dissimilarities or kernels, respectively [76, 43]. We will develop yet another approach in this realm which is based on a probabilistic treatment of prototype-based classification and extends this technique by means of the kernel trick towards similarity data, showing superior results which are comparable to the performance of an SVM.

Albeit these different techniques to extend LVQ towards proximities seem very similar, classification results are not always equivalent. Hence the question occurs what the differences and what the similarities of the techniques are. We answer this question by developing a general framework which summarizes these techniques under a common umbrella. The core observation consists in the fact that all type of proximity data represented by a symmetric proximity matrix can be associated to an implicit vector space embedding, the so-called pseudo-Euclidean embedding [73]. This vectorial counterpart opens the possibility of an LVQ scheme in this pseudo-Euclidean vector space. Since the computation of this embedding is costly, proximity-based learning methods perform it only implicitly. The techniques essentially differ in the way how this embedding is implicitly accessed and how it is integrated into the update rule for learning the classifier. Interestingly, there do not only

result different classification accuracies, but also different mathematical guarantees as concerns convergence of the learning rules and generalization ability of the classifier.

Besides this general view, we address two issues which are of great practical relevance in this thesis, in particular the efficiency of the model and model interpretability. Unlike their vectorial counterparts, proximity-based LVQ variants display a quadratic time complexity, hence the techniques become infeasible already for medium-sized data sets. We investigate the so-called Nyström technique which enables a low-rank matrix approximation of the proximity matrix resulting in a linear time scheme. While the Nyström approximation has been used earlier in a similar context [35, 83], it is not clear a priori in which cases it enables a nearly lossless result. We develop a quick check which can test the suitability of the Nyström approximation efficiently and prior to training. It essentially relies on an estimation of the variance which results when approximating proximities based on different landmarks. Hence a linear technique together with a reliable test about its suitability results.

Another problem consists in the fact that LVQ variants which are based on proximities represent prototypes in a distributed way as a virtual linear combination of data points. This not only slows down the classification time, but it also prohibits an intuitive interpretation and inspection of the resulting prototypes by experts. We address this issue by investigating different possibilities to approximate the prototypes by sparse counterparts which can be represented in terms of few data points only. We identify different principled possibilities how such an approximation can efficiently be realized, and we elucidate the suitability of the different techniques in an extensive comparison based on benchmarks.

We had the opportunity to present large parts of the results which are contained in this thesis in a number of international journals, conferences and workshops, as referenced in the following:

Journal articles

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

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

[53] D. Hofmann, F.-M. Schleif, B. Paaßen, and B. Hammer. Learning interpretable kernelized prototype-based models. Neurocomputing, 141: 84–96, 2014.

Conference articles

[27] B. Frenay, D. Hofmann, A. Schulz, M. Biehl, and B. Hammer. Valid interpretation of feature relevance for linear data mappings. Computational Intelligence and Data Mining, 149–156, 2014.

[33] A. Gisbrecht, D. Hofmann, and B. Hammer. Discriminative dimensionality reduction mappings. Advances in Intelligent Data Analysis, 7619: 126–138, 2012.

[49] D. Hofmann, A. Gisbrecht, and B. Hammer. Efficient approximations of kernel robust soft lvq. Workshop on Self-Organizing Maps, 198: 183–192, 2012.

[51] D. Hofmann and B. Hammer. Kernel robust soft learning vector quantization. Artificial Neural Networks in Pattern Recognition, 7477: 14–23, 2012.

[52] D. Hofmann and B. Hammer. Sparse approximations for kernel learning vector quantization. European Symposium on Artificial Neural Networks, 549–554, 2013.

Workshop contributions

[47] D. Hofmann. Sparse approximations for kernel robust soft lvq. Mittweida Workshop on Computational Intelligence, 2013.

[48] D. Hofmann, A. Gisbrecht, and B. Hammer. Discriminative probabilistic prototype based models in kernel space. New Challenges in Neural Computation, TR Machine Learning Reports, 2012.

Structure of the thesis

In this contribution, cost function based learning vector quantization (LVQ) [60] variants such as Robust Soft LVQ (RSLVQ) [88] or Generalized (GLVQ) [81] are introduced in Chapter 2. We consider the question how to extend these methods to similarity or dissimilarity data, respectively in Chapter 3.

Existing techniques such as kernel GLVQ [76], relational GLVQ [42], and relational RSLVQ [44] are covered and we investigate the novel possibility of kernel RSLVQ.

We propose a general way to extend LVQ methods based on an implicit pseudo-Euclidean embedding of the data, in Chapter 4 and discuss in how far instantiations of this framework differ from each other. Using it, we cover existing techniques, which offer valid classifiers and training methods for an arbitrary symmetric similarity or dissimilarity. Some mathematical properties, however, such as an interpretation via a likelihood ratio or interpretation of learning as exact gradient, are only guaranteed in the Euclidean case for some of the possible choices. In this context, we investigate the effect of corrections of the matrix to make data Euclidean. The effectivity of the novel technique kernel RSLVQ is demonstrated in a couple of benchmarks, where it will be extensively tested in comparison to popular alternatives such as k-nearest neighbor classifiers and the SVM.

Afterwards, we will focus on kernel RSLVQ which allows to priorly specify the model complexity, meaning the number of prototypes which represent the classifier. Unlike RSLVQ, it represents prototypes implicitly by means of a linear combination of data in kernel space, which has two drawbacks. On the one hand, an adaptation step does no longer scale linearly with the number of data points, rather, quadratic complexity is required. This makes the technique infeasible if large data sets are considered. Due to this we consider the Nyström approximation of Gram matrices in Chapter 5, which has been proposed in the context of SVMs in [97]. It constitutes a low rank approximation of the matrix based on a small subsample of the data. Assuming a fixed size of the subsample, a linear adaptation technique results. This approximation technique accounts for an efficient update and the effects on the accuracy are tested in a variety of benchmarks. Additionally we suggest a quick check for an a priori performance estimation of the Nyström approximation, based on the correlation.

On the other hand, prototypes are no longer directly interpretable, since the vector of linear coefficients is usually not sparse. Hence, in theory, all data points can contribute to the prototype. We propose a few possibilities to approximate prototypes in a relational LVQ scheme by sparse approximations in Chapter 6, thereby partially relying on classical solutions, but also taking into account simple heuristics which are motivated by the underlying geometrical background. Thereby, we propose one technique which emphasizes sparsity already while training, comparing this to two mathematical approximation schemes of the representation, namely classical orthogonal matching pursuit [15] and core techniques to approximately solve the minimum enclosing ball problem for the receptive fields of prototypes [4]. As an

alternative, we investigate two simple heuristics, namely an approximation of the prototypes by their closest exemplars, and a simple numerical rounding of the coefficient vector obtained by full training. We investigate the performance of these different techniques as concerns their classification accuracy and degree of sparsity. As one quantitative measure which can be related to the model interpretability, we use Rissanen's minimum description length principle in a supervised setting as well as the overall data entropy to judge the representativity of prototypes in an unsupervised perspective [77].

We conclude with a discussion in Chapter 7.

Funding acknowledgment

The following institutions and associated grants are gratefully acknowledged:

- The Cognitive Interaction Technology Center of Excellence (CITEC), funded by the German Science Foundation (DFG).
- The project Discriminative Dimensionality Reduction (DiDi) funded by the German Science Foundation (DFG) under grant number HA2719/7-1.

Chapter 2

Vectorial learning vector quantization

Chapter overview *In this chapter, we introduce the vectorial LVQ classifier, we have a short glimpse at traditional heuristic learning schemes, and we introduce two popular learning schemes. Generalized LVQ can be related to a maximization of the hypothesis margin, whereas robust soft LVQ relies on a likelihood ratio optimization. These methods, which are derived from explicit cost functions, will form the foundation for the extension of LVQ to proximity data. An abstract formalization of the form underlying this classification rule and learning schemes will pave the way towards these extensions.*

Learning vector quantization (LVQ) constitutes a very popular class of intuitive prototype based learning algorithms with successful applications ranging from telecommunications to robotics [61]. LVQ as originally proposed by Kohonen bases its decision on a winner-takes-all scheme and its learning rule on variants of Hebbian learning. Original LVQ1 is surprisingly good in typical model situations such as investigated for example in [9] and improvements such as LVQ2.1, LVQ3, or OLVQ aim at a higher convergence speed or better approximation of the Bayesian borders. These types of LVQ schemes have in common that their learning rule is essentially heuristically motivated and cannot be interpreted as direct optimization of a valid cost function [10, 11]. Against this background, researchers have proposed variants of LVQ which can directly be derived from an underlying cost function which is optimized during training for example by means of a stochastic gradient ascent or descent. One of the first attempts relates to large margin maximization [45, 84] and can be found in [81] with an exact computation of the validity at class boundaries in [85]. Later, a very elegant LVQ scheme

which is a statistical approach and which can be seen as a more robust probabilistic extension of LVQ2.1 has been proposed in [88].

2.1 Learning vector quantization

Assume data $\xi_i \in \mathbb{R}^n$ with $i = 1, \ldots, N$ are labeled y_i where labels stem from a finite number of different classes. A LVQ classifier is characterized by m prototypes $w_j \in \mathbb{R}^n$ with priorly fixed labels $c(w_j)$. Classification takes place by a winner takes all scheme

$$\xi \mapsto c(w_j) \text{ where } d(\xi, w_j) \text{ is minimum} \tag{2.1}$$

with squared Euclidean distance $d(\xi, w_j) = \|\xi - w_j\|^2$, breaking ties arbitrarily. We refer to the data ξ_i which are closest to a given prototype w_j as the receptive field R_j of the prototype.

LVQ learning aims at a positioning of prototypes such that the resulting classification error is minimized [60, 61]. Since optimization of the classification error itself is an NP-hard problem, the original learning rules rely on heuristics. Given a data point ξ_i, LVQ1 recursively modifies the winner w_j by the update

$$\Delta w_j = \begin{cases} \alpha(\xi_i - w_j) & \text{if } c(w_j) = y_i \\ -\alpha(\xi_i - w_j) & \text{if } c(w_j) \neq y_i \end{cases} \tag{2.2}$$

where $\alpha \in (0, 1)$ is the learning rate. This update can be interpreted as stochastic gradient descent on the cost function as analyzed in [81]

$$\text{Cost}_{\text{LVQ}} = \sum_i f_{\text{LVQ}}(d(\xi_i, w_+), d(\xi_i, w_-)) \tag{2.3}$$

where w_+ constitutes the closest prototype to ξ_i labeled with y_i and w_- denotes the closest prototype with a different label than ξ_i and where

$$f_{\text{LVQ}}(d(\xi_i, w_+), d(\xi_i, w_-)) = \begin{cases} (\xi_i - w_+)^2 & \text{if } (\xi_i - w_+)^2 \leq (\xi_i - w_-)^2 \\ (\xi_i - w_-)^2 & \text{if } (\xi_i - w_+)^2 > (\xi_i - w_-)^2 \end{cases} \tag{2.4}$$

Unfortunately, this cost function is discontinuous and has stability problems for overlapping data distributions. Further, it does not constitute a valid global cost function but a local one only, in the sense that its value is smaller (negative) if more data are misclassified due to the discontinuity.

$f_{\text{LVQ2.1}}(d(\xi_i, w_+), d(\xi_i, w_-)) = v\left((\xi_i - w_+)^2 - (\xi_i - w_-)^2\right)$ as choice of the term in the above sum yields the cost function of LVQ2.1 as explained

in [60], where v restricts updates to data points which fall into a window around the decision boundary. This produces an instable dynamic, meaning prototypes diverge because repelling forces might be larger than attracting forces. The window must be chosen carefully to prevent this behavior. w_+ and w_- are changed if the data point is far from the winners and they belong to different labels, meaning

$$\min\left(\frac{|\xi_i - w_+|}{|\xi_i - w_-|}, \frac{|\xi_i - w_-|}{|\xi_i - w_+|}\right) > \frac{1-v}{1+v} \tag{2.5}$$

with $0 < v \leq 1$. This leads to the following update rule

$$\Delta w_+ = \alpha\left(\xi_i - w_+\right), \ \Delta w_- = -\alpha\left(\xi_i - w_-\right) \tag{2.6}$$

For training, it is usually assumed that the number and classes of prototypes are fixed. In practice, these are often determined using cross-validation, or a further wrapper technique or incremental learning [99, 64] is added to obtain model flexibility. In the following, we will not consider the issue of model selection, but rely on standard techniques such as cross-validation.

We will rather focus on more recent alternatives which prevent these divergence problems and which are derived from suitable cost functions, instead of considering these heuristic LVQ learning schemes. Interestingly, all LVQ classification schemes can be accompanied by large margin generalization bounds [21], comparable to support vector machine (SVM) [19].

2.2 Generalized learning vector quantization

As before, generalized LVQ (GLVQ) as introduced in [81] relies on training data $\xi_i \in \mathbb{R}^n$ accompanied by labels y_i. A GLVQ network is determined by m prototypes $w_j \in \mathbb{R}^n$, where the labels of prototypes $c\left(w_j\right)$ are fixed.

Training is formalized in terms of a cost function which aims at finding positions of the prototypes such that the classification accuracy of the training set is optimized and, in addition, the generalization ability is taken into account

$$\sum_i F\left(\frac{d\left(\xi_i, w_+\right) - d\left(\xi_i, w_-\right)}{d\left(\xi_i, w_+\right) + d\left(\xi_i, w_-\right)}\right) \tag{2.7}$$

where w_+ constitutes the closest prototype to ξ_i labeled with y_i and w_- denotes the closest prototype with a different label than ξ_i and where F is taken as monotonic function such as the sigmoid function, the hyperbolic tangent function, or the identity function. In the following, we will only use the

identity function and, in consequence, drop the symbol. As recently shown, however [57, 56] a careful adjustment of F can be beneficial in practice. d refers to the squared Euclidean metric. The nominator is negative if and only if ξ_i is classified correctly, thus GLVQ tries to maximize the number of correct classifications. In addition, it aims at an optimization of the hypothesis margin $d(\xi_i, w_-) - d(\xi_i, w_+)$ which determines the generalization ability of the method [84].

Training takes place by a simple stochastic gradient descent, meaning given a data point ξ_i, adaptation takes place via the update rules

$$\begin{aligned} \Delta w_+ &\sim -\frac{2 \cdot d(\xi_i, w_-)}{\left(d(\xi_i, w_+) + d(\xi_i, w_-)\right)^2} \cdot \frac{\partial d(\xi_i, w_+)}{\partial w_+} \\ \Delta w_- &\sim \frac{2 \cdot d(\xi_i, w_+)}{\left(d(\xi_i, w_+) + d(\xi_i, w_-)\right)^2} \cdot \frac{\partial d(\xi_i, w_-)}{\partial w_-} \end{aligned} \tag{2.8}$$

2.3 Robust soft learning vector quantization

Robust soft LVQ (RSLVQ) as introduced in [88] models data by a mixture of Gaussians and derives learning rules as a maximization of the log likelihood ratio of the given data. In the limit of small bandwidth σ, a learning rule which is similar to LVQ2.1 but which performs adaptation in case of misclassification only is obtained.

Assume, again, that data $\xi_i \in \mathbb{R}^n$ are given accompanied by labels y_i. A RSLVQ network represents a mixture distribution, which is determined by m prototypes $w_j \in \mathbb{R}^n$, where the labels of prototypes $c(w_j)$ are fixed. In addition, a parameter σ_j denotes the bandwidth. Then mixture component j induces the probability

$$p(\xi|j) = \text{const}_j \cdot \exp\left(f\left(\xi, w_j, \sigma_j^2\right)\right) \tag{2.9}$$

with normalization constant const_j and function f

$$f\left(\xi, w_j, \sigma_j^2\right) = -\left\|\xi - w_j\right\|^2 / \sigma_j^2 \tag{2.10}$$

The probability of a data point ξ is given by the mixture

$$p(\xi|W) = \sum_j P(j) \cdot p(\xi|j) \tag{2.11}$$

with prior probability $P(j)$ of mixture j and parameters W of the model. The probability of a data point ξ and a given label y is

$$p(\xi, y|W) = \sum_{c(w_j)=y} P(j) \cdot p(\xi|j) \tag{2.12}$$

Learning aims at an optimization of the log likelihood ratio

$$L = \sum_i \log \frac{p\left(\xi_i, y_i | W\right)}{p\left(\xi_i | W\right)} \tag{2.13}$$

A stochastic gradient ascent yields the following update rules, given a data point (ξ_i, y_i)

$$\Delta w_j = \alpha \cdot \begin{cases} \left(P_y\left(j|\xi_i\right) - P\left(j|\xi_i\right)\right) \cdot \text{const}_j \\ \qquad \cdot \partial f\left(\xi_i, w_j, \sigma_j^2\right) / \partial w_j & \text{if } c\left(w_j\right) = y_i \\ -P\left(j|\xi_i\right) \cdot \text{const}_j \\ \qquad \cdot \partial f\left(\xi_i, w_j, \sigma_j^2\right) / \partial w_j & \text{if } c\left(w_j\right) \neq y_i \end{cases} \tag{2.14}$$

with the learning rate $\alpha > 0$. The probabilities are defined as

$$P_y\left(j|\xi_i\right) = \frac{P\left(j\right) \exp\left(f\left(\xi_i, w_j, \sigma_j^2\right)\right)}{\sum_{c(w_j)=y_j} P\left(j\right) \exp\left(f\left(\xi_i, w_j, \sigma_j^2\right)\right)} \tag{2.15}$$

and

$$P\left(j|\xi_i\right) = \frac{P\left(j\right) \exp\left(f\left(\xi_i, w_j, \sigma_j^2\right)\right)}{\sum_j P\left(j\right) \exp\left(f\left(\xi_i, w_j, \sigma_j^2\right)\right)} \tag{2.16}$$

If class priors are equal, and small bandwidth is present, a learning rule similar to LVQ2.1 results.

Given a novel data point ξ, its class label is the most likely label y corresponding to a maximum value $p\left(y|\xi, W\right) \sim p\left(\xi, y|W\right)$. For typical settings, this rule can be approximated by a simple winner takes all rule, meaning ξ is mapped to the label $c\left(w_j\right)$ of the closest prototype w_j. It has been shown in [88], 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 w_j.

2.4 Abstract formulation

From an abstract point of view, we can characterize LVQ as a classifier, which classification rule is based on the quantities

$$D\left(\xi, w\right) := \left(d\left(\xi_i, w_j\right)\right)_{i=1,\ldots,N, j=1,\ldots,m} \tag{2.17}$$

for example selecting the minimum of these terms. Training aims at an optimization of a cost function of the form

$$f\left(D\left(\xi, w\right)\right) \tag{2.18}$$

with suitable function f by means of the gradients

$$\frac{\partial f\left(D\left(\xi, w\right)\right)}{\partial w_j} = \sum_{i=1}^{N} \frac{\partial f\left(D\left(\xi, w\right)\right)}{\partial d\left(\xi_i, w_j\right)} \cdot \frac{\partial d\left(\xi_i, w_j\right)}{\partial w_j} \tag{2.19}$$

with respect to the prototypes w_j or the corresponding stochastic gradients for one point ξ_i. This observation will constitute the key to transfer LVQ variants towards general proximity data in Chapter 4. So far, the distance measure d is chosen as squared Euclidean distance, but extensions are possible. A very popular choice which has been published under the umbrella of relevance or matrix learning substitutes the standard Euclidean distance by an adaptive quadratic form, which can autonomously infer a suitable scaling and ranking of the feature dimensions and their correlation based on given data [84]. We will not consider this extension in the following, rather we will focus on settings where pairwise distances d are given in terms of a general proximity matrix.

We would like to point out that not only modern LVQ variants are characterized by the essential ingredients given by Equation 2.17 and Equation 2.18, but also many unsupervised prototype based techniques can be written in this form. Popular examples include, for example, neural gas (NG) or the self-organizing map (SOM) in the form proposed by Heskes, or probabilistic counterparts [67, 46]. Due to this observation, the general framework which we will develop is not restricted to supervised prototype-based methods but the arguments directly transfer to unsupervised prototype-based techniques provided the latter are derived from a suitable cost function, such that kernel or relational extensions of SOM and NG are covered [39]. For the sake of simplicity, however, we will not elucidate this link in the following.

2.5 Discussion

We have introduced the basics of modern LVQ variants which are derived on cost functions, notably GLVQ and RSLVQ, which will be used later on. Due to their intuitive learning and classification rule based on a winner-takes-all scheme, these techniques enjoy a great popularity in diverse application domains ranging from telecommunication and robotics up to bioinformatics and data mining [61, 8, 30]. Apart from an only linear training time and its suitability for online scenarios, such as demonstrated for example in [59, 23], one of its benefits is given by the fact that models are represented in terms of few prototypes which can be inspected by practitioners in the same way as data. Hence this inherent representation scheme lends itself as an intuitive interface to the model, unlike many black box alternatives in machine learning

which offer state-of-the-art results but, usually, do not provide a justification why a certain classification takes place [2]. In complex settings where the overall task is not necessarily clear a priori or in settings where the human has to take responsibility for a subsequent action, interpretability becomes crucial. Here, human insight is often the only way to further specify a priorly unclear training setting or to substantiate mere observations by causalities. Due to this reason, there is an increasing demand of interpretable models which provide a human understandable interface to their decisions besides excellent classification accuracy in areas such as biomedical data analysis or interactive data inspection [94].

Apart from prototype based data representations, quite a few approaches have addressed the interpretability of powerful machine learning algorithms, including, for example, intelligent approximation techniques and feature selection mechanisms for SVM, blind signal separation, enhanced score methods, or visualization techniques [80, 6, 13, 7, 38]. For LVQ, interpretability is guaranteed per the design of the model [11]. Interestingly, some LVQ techniques can be easily enhanced such that they additionally provide an inherent low dimensional visualization of their decisions [16], or an extension of the models by directly interpretable relevance terms is possible [84, 85]. Further, as already mentioned, strong learning theoretical guarantees substantiate LVQ algorithms as classification models with excellent generalization behavior [5, 9, 84].

These classical LVQ methods as introduced above are restricted to vectorial data. In recent years, data are often no longer vectorial in many application domains for example due to improved sensor technology or dedicated data formats. Rather, complex structures are dealt with for which a problem specific similarity or dissimilarity measure has been designed. This measure accounts for the structural form of the data such as alignment techniques for bioinformatics sequences, dedicated functional norms for mass spectra, the compression distance for texts, or metabolic networks, where complex alignment techniques, background information, or general information theoretical principles, for example, drive the comparison of data points [74, 66, 54]. In these settings, it is possible to compute pairwise similarities or dissimilarities of the data rather than to arrive at an explicit vectorial representation, which LVQ methods are limited to. In the following, we will therefore investigate how LVQ schemes can be extended to proximity data instead of standard Euclidean vectors.

Chapter 3

LVQ for proximities

Chapter overview *The goal of this chapter is to introduce a few extensions of LVQ versions towards more general proximity data rather than vectors, in particular summarizing the three existing techniques relational GLVQ, relational RSLVQ, and kernel GLVQ. In doing so, we also introduce the so-called pseudo-Euclidean embedding of proximity data, which is necessary for the derivation of the relational LVQ variants, and which will form the mathematical base for the general framework we will introduce in Chapter 4. In addition to this summary, we propose the novel technique kernel RSLVQ, which extends the probabilistic RSLVQ in a very clear way towards general kernels. Parts of this chapter are based on the publications [51, 47].*

As discussed in Chapter 2 prototype-based methods often display very intuitive classification and learning rules. However, the introduced LVQ variants are restricted to vectorial data only such that they cannot be applied if data are non-vectorial and represented in terms of pairwise similarities or dissimilarities. Examples for such settings include structured data such as graphs, trees, sequence data, extensible markup language, or the like [26, 29, 82]. Often, these data can be addressed by means of a dedicated similarity measure or kernel, including for example sequence alignment, the normalized compression distance, graph kernels, or similar [29, 18, 17, 74, 41, 54, 65, 68]. As such, the similarity or dissimilarity measure can serve as a canonical interface of the model towards the given data set, as is the case for example in popular kernel approaches. In the following we will discuss techniques how to extend LVQ algorithms to more general data characterized by pairwise similarities or dissimilarities only.

Two different principles have been proposed in the literature. Kernel GLVQ assumes a valid Gram matrix and extends GLVQ by means of kernelization, see [76]. In contrast, relational GLVQ assumes the more general

setting of possibly non-Euclidean dissimilarities, and extends GLVQ to this setting by an alternative expression of distances based on the given dissimilarity data [42]. Both techniques can analogously be applied to RSLVQ [51, 44]. We introduce these four techniques, including the novel kernel RSLVQ which has been proposed by us. In Chapter 4, we will argue that both instances can be unified as LVQ variants referring to the pseudo-Euclidean embedding of similarity or dissimilarity data, respectively. First, we will address kernel LVQ variants, before coming to relational extensions and its underlying pseudo-Euclidean embedding.

3.1 Kernel GLVQ

Based on the minimum error classification criterion [55], which is a discriminant training criterion that minimizes an overall expected loss function by using a gradient descent procedure, the GLVQ algorithm [81] as introduced in Section 2.2 has been proposed. This algorithm can yield accurate and stable classification results because the piecewise linear boundaries of the receptive fields of all prototypes try to approximate the optimal Bayesian boundaries. However, it is hard to specify a reasonable number of prototypes to approximate complex boundaries, when borders between classes are non-linear, especially when many substructures exist in each class.

The kernel GLVQ algorithm as introduced in [76] makes use of the same cost function as the original algorithm but with the distance calculations done in a higher dimensional feature space, the kernel space. For this purpose, the existence of a non-linear function Φ that maps data points ξ_i from the input space to a possibly high dimensional feature space is assumed. Without need of the knowledge about the specific form of Φ, the dot product of two points $\Phi(\xi_i)$ and $\Phi(\xi_l)$ can be implicitly computed by using the Mercer kernel function k_{il} [87] defined in the data space, characterized by the identity

$$k_{il} := k(\xi_i, \xi_l) = \Phi(\xi_i)^t \Phi(\xi_l) \tag{3.1}$$

for all data points ξ_i, ξ_l. Using this kernel function any computations in the feature space can be efficiently converted into operations in the data space [90].

Under this setting the prototypes cannot explicitly be expressed as vectors in the feature space due to lack of knowledge about the feature space. Instead, the feature space can be regarded as being spanned by all images $\Phi(\xi_i)$, thus inducing a description of a prototype vector by some linear combination of the feature space data samples $w_j = \sum_m \gamma_{jm} \Phi(\xi_m)$. This induces a formula

to compute the distance $d\left(\Phi\left(\xi_i\right), w_j\right)$ directly by means of k_{il}

$$\begin{aligned}\left\|\Phi\left(\xi_i\right)-w_j\right\|^2 &= \left\|\Phi\left(\xi_i\right)-\sum_m \gamma_{jm}\Phi\left(\xi_m\right)\right\|^2 \\ &= k_{ii}-2\cdot\sum_m \gamma_{jm}k_{im}+\sum_{s,t}\gamma_{js}\gamma_{jt}k_{st}\end{aligned} \tag{3.2}$$

where the norm in the feature space is referred to by $\|\cdot\|^2$.

This observation extends the classification rule. Given an input vector ξ_i the updating rule in Equation 2.8 of the original GLVQ algorithm can be generalized from the original data space into the feature space as follows

$$\begin{aligned}\Delta\sum_m \gamma_{+m}\Phi\left(\xi_m\right) \sim & -\frac{2\cdot d\left(\Phi\left(\xi_i\right), w_-\right)}{\left(d\left(\Phi\left(\xi_i\right), w_+\right)+d\left(\Phi\left(\xi_i\right), w_-\right)\right)^2} \\ & \cdot\left(\Phi\left(\xi_i\right)-\sum_m \gamma_{+m}\Phi\left(\xi_m\right)\right) \\ \Delta\sum_m \gamma_{-m}\Phi\left(\xi_m\right) \sim & \frac{2\cdot d\left(\Phi\left(\xi_i\right), w_+\right)}{\left(d\left(\Phi\left(\xi_i\right), w_+\right)+d\left(\Phi\left(\xi_i\right), w_-\right)\right)^2} \\ & \cdot\left(\Phi\left(\xi_i\right)-\sum_m \gamma_{-m}\Phi\left(\xi_m\right)\right)\end{aligned} \tag{3.3}$$

where γ_{+m} and γ_{-m} correspond to the best matching prototype vector w_+ of $\Phi\left(\xi_i\right)$ with the same class label y_i and the best matching prototype vector w_- of $\Phi\left(\xi_i\right)$ with a different class label y_i. This update rule for the prototype vector in the feature space is equivalent to the following update of the coefficients γ

$$\begin{aligned}\Delta\gamma_{+m} \sim & \begin{cases}\left(1-\dfrac{2\cdot d\left(\Phi\left(\xi_i\right), w_-\right)}{\left(d\left(\Phi\left(\xi_i\right), w_+\right)+d\left(\Phi\left(\xi_i\right), w_-\right)\right)^2}\right)\gamma_{+m} & \text{if } \xi_m\neq\xi_i \\ \left(1-\dfrac{2\cdot d\left(\Phi\left(\xi_i\right), w_-\right)}{\left(d\left(\Phi\left(\xi_i\right), w_+\right)+d\left(\Phi\left(\xi_i\right), w_-\right)\right)^2}\right)\gamma_{+m} & \\ \quad +\dfrac{2\cdot d\left(\Phi\left(\xi_i\right), w_-\right)}{\left(d\left(\Phi\left(\xi_i\right), w_+\right)+d\left(\Phi\left(\xi_i\right), w_-\right)\right)^2} & \text{if } \xi_m=\xi_i\end{cases} \\ \Delta\gamma_{-m} \sim & \begin{cases}\left(1+\dfrac{2\cdot d\left(\Phi\left(\xi_i\right), w_+\right)}{\left(d\left(\Phi\left(\xi_i\right), w_+\right)+d\left(\Phi\left(\xi_i\right), w_-\right)\right)^2}\right)\gamma_{-m} & \text{if } \xi_m\neq\xi_i \\ \left(1+\dfrac{2\cdot d\left(\Phi\left(\xi_i\right), w_+\right)}{\left(d\left(\Phi\left(\xi_i\right), w_+\right)+d\left(\Phi\left(\xi_i\right), w_-\right)\right)^2}\right)\gamma_{-m} & \\ \quad -\dfrac{2\cdot d\left(\Phi\left(\xi_i\right), w_+\right)}{\left(d\left(\Phi\left(\xi_i\right), w_+\right)+d\left(\Phi\left(\xi_i\right), w_-\right)\right)^2} & \text{if } \xi_m=\xi_i\end{cases}\end{aligned} \tag{3.4}$$

where the distance calculations can be based on the kernel function through the Equation 3.2. While retaining the merits of the original algorithm, this kernel GLVQ (KGLVQ) algorithm can more effectively cope with datasets with non-linear boundaries between classes and non-vectorial data by means of a sufficiently powerful kernel such as a structure kernel. It might be advisable to restrict prototype positions towards convex combinations of the data, which corresponds to the restriction that the coefficients γ_{jm} are non-negative and sum up to 1.

3.2 Kernel RSLVQ

Similar to GLVQ, RSLVQ [88] in its original form as introduced in Section 2.3 is restricted to Euclidean vectors. Here, we derive a novel kernel extension similar to kernel GLVQ which is suited for more general data structures. As before, we assume the existence of a feature map Φ which corresponds to a kernel k. Prototypes can be implicitly represented in terms of linear combinations of data $w_j = \sum_m \gamma_{jm} \Phi(\xi_m)$ with coefficients γ_{jm}. Again, if appropriate, we can restrict the coefficients γ_{jm} to non-negative values which sum up to 1. This corresponds to the assumption that prototypes are located in the convex hull of data, which is a reasonable assumption provided the LVQ scheme should yield representative prototypes.

Having made this assumption, it is possible to formalize the cost function of kernel RSLVQ

$$L = \sum_i \log \frac{\sum_{c(w_j)=y_i} P(j)\, p(\Phi(\xi_i)\,|j)}{\sum_j P(j)\, p(\Phi(\xi_i)\,|j)} \tag{3.5}$$

which relies on the Gaussian probabilities, implicitly in terms of the Gram matrix of data and coefficients of prototypes only. The Gaussian $p(\Phi(\xi_i)\,|j)$ constitutes an exponential function on the distance, which can be computed similarly to Equation 3.2 implicitly by means of the equality $\|\Phi(\xi_i) - w_j\|^2 = \|\Phi(\xi_i) - \sum_m \gamma_{jm}\Phi(\xi_m)\|^2 = k_{ii} - 2 \cdot \sum_m \gamma_{jm} k_{im} + \sum_{s,t} \gamma_{js}\gamma_{jt} k_{st}$ where the distance in the feature space is referred to by $\|\cdot\|^2$.

We assume equal bandwidth $\sigma^2 = \sigma_j^2$, for simplicity. More complex adjustment schemes based on the data have been investigated in [86], for example, usually leading to only a minor increase of accuracy. Note that the position of prototypes is not clear a priori, such that a prior adaptation of the bandwidth according to the data density is not possible. Further, we assume constant prior $P(j)$ and mixture components induced by normalized Gaussians.

There are two ways to optimize the cost function of kernel RSLVQ as we will see in Chapter 4 where we introduce a general framework for non-vectorial LVQ schemes for proximity data. The cost function L can be optimized directly with respect to the model parameters γ_{jm} by relying on some standard numeric optimization procedure such as gradient techniques. As an alternative, the cost function can be optimized with respect to the prototypes w_j, and the resulting update rules can be decomposed into contributions of the coefficient vectors γ_{jm}, resulting in update rules for the latter. Note that there is no guarantee that the gradient commutes with linear combinations of parameters such that the two update rules yield numerically different behavior, albeit the same local and global minima are present. Further, it is not clear a priori whether a decomposition of the update rule of w_j in terms of coefficients is possible. Whenever this is the case, kernelization is possible, such as for kernel GLVQ and, as we will see, kernel RSLVQ. We will later see that Euclideanity of the embedding space constitutes a crucial prerequisite for this fact.

The RSLVQ updates in Equation 2.14 can be rephrased as follows

$$\begin{aligned}\Delta w_j &= \Delta \sum_m \gamma_{jm} \Phi(\xi_m) \\ &\sim \begin{cases} \left(P_y(j|\Phi(\xi_i)) - P(j|\Phi(\xi_i))\right) & \\ \qquad \cdot \left(\Phi(\xi_i) - \sum_m \gamma_{jm}\Phi(\xi_m)\right) & \text{if } c(w_j) = y_i \\ -P(j|\Phi(\xi_i)) & \\ \qquad \cdot \left(\Phi(\xi_i) - \sum_m \gamma_{jm}\Phi(\xi_m)\right) & \text{if } c(w_j) \neq y_i \end{cases}\end{aligned} \tag{3.6}$$

which decomposes into the following adaptation rules for γ_{jm}

$$\Delta\gamma_{jm} \sim \begin{cases} -\left(P_y(j|\Phi(\xi_i)) - P(j|\Phi(\xi_i))\right) & \\ \qquad \cdot\gamma_{jm} & \text{if } \xi_m \neq \xi_i, c(w_j) = y_i \\ \left(P_y(j|\Phi(\xi_i)) - P(j|\Phi(\xi_i))\right) & \\ \qquad \cdot (1 - \gamma_{jm}) & \text{if } \xi_m = \xi_i, c(w_j) = y_i \\ P(j|\Phi(\xi_i)) & \\ \qquad \cdot\gamma_{jm} & \text{if } \xi_m \neq \xi_i, c(w_j) \neq y_i \\ -P(j|\Phi(\xi_i)) & \\ \qquad \cdot (1 - \gamma_{jm}) & \text{if } \xi_m = \xi_i, c(w_j) \neq y_i \end{cases} \tag{3.7}$$

with respectively $P(j|\Phi(\xi_i)) = \frac{P(j)\exp\left(f\left(\Phi(\xi_i), w_j, \sigma_j^2\right)\right)}{\sum_j P(j)\exp\left(f\left(\Phi(\xi_i), w_j, \sigma_j^2\right)\right)}$ and $P_y(j|\Phi(\xi_i)) =$

$\frac{P(j)\exp\left(f\left(\Phi(\xi_i),w_j,\sigma_j^2\right)\right)}{\sum_{c\left(w_j\right)=y_j} P(j)\exp\left(f\left(\Phi(\xi_i),w_j,\sigma_j^2\right)\right)}$. Note that these probabilities depend on distances of data and prototypes in the feature space only, such that they can be computed based on the given kernel. We refer to this learning scheme as kernel RSLVQ (KRSLVQ).

This scheme performs exactly the same updates as RSLVQ in the feature space if prototypes are in the linear span of the data. Often, a further restriction of the parameters to the convex hull takes place to ensure a representative location of the prototypes. We will follow this principle by applying a correction to guarantee non-negativity and normalization after every adaptation step to already boost the interpretability of the prototype coefficients while training. As an alternative, barrier techniques could be used, or the restrictions could be dropped entirely allowing more general linear combinations as solutions.

The derivative of kernel RSLVQ in this form can be used whenever a fixed kernel k is given together with the data, or the Gram matrix itself is given, implicitly representing the data [73]. Note that it can easily be checked whether a symmetric matrix constitutes a valid Gram matrix by referring to the eigenvalues, which should be non-negative. In this case, the adaptation rule as introduced above mimics the standard vectorial update of RSLVQ in the feature space, but without the necessity of explicitly computing this embedding.

Provided the similarity matrix of the data is not positive semidefinite, meaning we do not face a valid kernel, the validity of kernel RSLVQ and kernel GLVQ is not clear. We will deal with this issue in Chapter 4. Before, we introduce the so-called pseudo-Euclidean embedding [73], which enables a vectorial embedding of general similarity matrices and which forms the base for alternative, so-called relational extensions of LVQ variants.

3.3 Pseudo-Euclidean embedding

Kernels constitute a specific way to compare given data, and they have the benefit that an underlying embedding in a possibly high dimensional feature space is present. Here we consider the more general setting that data are characterized by pairwise similarities $s_{ij} = s\left(\xi_i, \xi_j\right)$ such as pairwise inner products for Euclidean data or dissimilarities $d_{ij} = d\left(\xi_i, \xi_j\right)$ such as pairwise squared Euclidean distances for Euclidean data only. As before, no explicit vectors are given. In addition, it is not clear whether these values stem from a kernel, hence whether a substantiating vectorial embedding exists. We refer to the corresponding matrices as S and D, respectively, its dimensionality

given by the number of observed objects. Since data are given by pairwise relations only rather than vectors or a kernel, corresponding approaches are often referred to as relational approaches. We always assume symmetry, meaning $S = S^t$ and $D = D^t$ as well as zero diagonal in D, meaning $d_{ii} = 0$. We do not assume Euclideanity, however. First we have a closer look at the data and its properties, ending up with a vectorial embedding which can be regarded as an extension of a kernel embedding, based on which a generalization of LVQ techniques to such data is possible.

Relation of S and D

The first question is how these two representations S and D are related. There exist classical methods to turn similarities to dissimilarities and vice versa, see for example [73]. Given a similarity, a dissimilarity is obtained by the transformation

$$X : S \to D, d_{ij} = s_{ii} - 2s_{ij} + s_{jj} \tag{3.8}$$

while the converse is obtained by double centering

$$\Psi : D \to S, s_{ij} = -\frac{1}{2}\left(d_{ij} - \frac{1}{N}\sum_i d_{ij} - \frac{1}{N}\sum_j d_{ij} + \frac{1}{N^2}\sum_{i,j} d_{ij}\right) \tag{3.9}$$

While it holds that the composition of these two transforms $\Psi \circ X = \mathbf{I}$, $\mathbf{I}$ being the identity, the converse, $X \circ \Psi$ yields the identity if and only if data are centered, since offsets of data which are characterized by dissimilarities are arbitrary and hence not reconstructable from D. That means, if S is generated from vectors via some quadratic form, the vectors should be centered in the origin. So essentially, for techniques which rely on dissimilarities of data, we can treat similarities or dissimilarities as identical via these transformations. The same holds for similarity based approaches only if data are centered. However, even if this transformation is possible it is usually costly, such that techniques which can directly be used for either similarities or dissimilarities are preferred.

Vectorial embedding

A crucial step to extend LVQ variants to non-vectorial data consists in the construction of an implicit embedding space, such as a kernel embedding for kernel variants. In that case we assumed a non-linear mapping Φ of data points to a high dimensional or infinite dimensional Hilbert space $\mathcal{H}$

equipped with the inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. Opposed to that a Krein space is an indefinite inner product space endowed with a Hilbertian topology. An inner product space $(\mathcal{K}, \langle \cdot, \cdot \rangle_{\mathcal{K}})$ with an indefinite inner product $\langle \cdot, \cdot \rangle_{\mathcal{K}}$ is a Krein space if two Hilbert spaces $\mathcal{H}_+$ and $\mathcal{H}_-$ exist spanning $\mathcal{K}$ such that $\forall g \in \mathcal{K}$, $g = g_+ + g_-$ with $g_+ \in \mathcal{H}_+$ and $g_- \in \mathcal{H}_-$ and $\forall g, h \in \mathcal{K}, \langle g, h \rangle_{\mathcal{K}} = \langle g_+, h_+ \rangle_{\mathcal{H}_+} - \langle g_-, h_- \rangle_{\mathcal{H}_-}$. A finite dimensional Krein space is a so-called pseudo-Euclidean space. For general data, the key observation is that every finite data set which is characterized by pairwise similarities or dissimilarities can be embedded in such a pseudo-Euclidean vector space [36]. Essentially, this is a finite dimensional real-vector space of dimensionality N, characterized by the signature $(p, q, N - p - q)$, which captures the degree up to which elements are Euclidean. N refers to the number of given data points. Distances along the first p dimensions are Euclidean whereas the next q dimensions serve as correction factors to account for the non-Euclidean elements of the dissimilarity d. We follow the presentation of pseudo-Euclidean spaces as derived in [73].

Assume a similarity matrix S or corresponding dissimilarity matrix D is given. Since S is symmetric, a decomposition

$$S = Q \Lambda Q^t = Q \left|\Lambda\right|^{1/2} I_{pq} \left|\Lambda\right|^{1/2} Q^t \tag{3.10}$$

with diagonal matrix Λ and orthonormal columns in the matrix Q can be found. I_{pq} denotes the diagonal matrix with the first p elements 1, the next q elements -1, and $N - p - q$ elements 0. By means of this representation, the number of positive and negative eigenvalues of S is made explicit as p and q, respectively. We set $\xi_i = \sqrt{|\Lambda_{ii}|} q_i$, q_i being column i of Q. Further, we define the quadratic form

$$\langle u, v \rangle_{pq} = u_1 v_1 + \ldots + u_p v_p - u_{p+1} v_{p+1} - \ldots - u_{p+q} v_{p+q} \tag{3.11}$$

Then we find

$$s_{ij} = \langle \xi_i, \xi_j \rangle_{p,q} \tag{3.12}$$

For a given dissimilarity matrix, we can consider the matrix $\Psi(D)$ obtained by double centering in Equation 3.9. This similarity matrix can be treated in the same way as S leading to vectors ξ_i such that

$$d_{ij} = \|\xi_i - \xi_j\|^2_{p,q} \tag{3.13}$$

where the symmetric bilinear form is associated to the quadratic form in Equation 3.11

$$\begin{aligned} \|u - v\|^2_{pq} = &|u_1 - v_1|^2 + \ldots + |u_p - v_p|^2 \\ &- |u_{p+1} - v_{p+1}|^2 - \ldots - |u_{p+q} - v_{p+q}|^2 \end{aligned} \tag{3.14}$$

Thus, in both cases, vectors in a vector space can be found which induce the similarity or dissimilarity, respectively. The quadratic form in this vector space, however, is not positive definite. Rather, the first p components can be considered as standard Euclidean contribution whereas the next q components serve as a correction. This vector space is referred to as pseudo-Euclidean space with its characteristic signature $(p, q, N-p-q)$.

Note that dissimilarities defined via $\|u-v\|_{pq}^2$ or similarities defined via $\langle u, v\rangle_{pq}$ can become negative, albeit, often, the negative part is not large in practical applications. Similarities or dissimilarities stem from a Euclidean vector space if and only if $q = 0$ holds. Exactly in this case, a kernel embedding of the data exists, meaning the similarities are in fact a kernel.

Distance computation in LVQ for (dis-)similarities

The pseudo-Euclidean embedding allows us to transfer LVQ based classifiers to similarity or dissimilarity data in a very generic way, which covers relational and kernel GLVQ and RSLVQ as a special case. Essentially, we embed data and prototypes in pseudo-Euclidean space and we instantiate the squared 'distance' $d(\xi_i, w_j)$ used in LVQ algorithms by the pseudo-Euclidean dissimilarity $\|\xi_i - w_j\|_{pq}^2$. Albeit this is no longer a 'distance' strictly speaking, we will address this quantity as such in the following. Before introducing relational GLVQ and relational RSLVQ, we elucidate how to compute distance terms as occur in the classification prescription and learning rules of LVQ variants provided data are given as proximities only.

In principle, we could explicitly embed data and perform vectorial LVQ variants in the embedding space. However, this has cubic complexity, so the question is whether this can be avoided. As before, we restrict the position of prototypes to the convex hull of the data. Thus, we assume

$$w_j = \sum_l \gamma_{jl} \xi_l \tag{3.15}$$

where $\gamma_{jl} \geq 0$. Then, we can compute for a given data point ξ_i:

$$\|\xi_i - w_j\|_{pq}^2 = s_{ii} - 2\sum_l \gamma_{jl} s_{il} + \sum_{ll'} \gamma_{jl}\gamma_{jl'} s_{ll'} \tag{3.16}$$

Hence we can compute distances of all data points and prototypes based on pairwise data similarities only in quadratic time. Further, we do not need to represent prototypes w_j explicitly, rather, the coefficients γ_{jl} are sufficient. Similarly, we find

$$\|\xi_i - w_j\|_{pq}^2 = \sum_l \gamma_{jl} d_{il} - 1/2 \cdot \sum_{ll'} \gamma_{jl}\gamma_{jl'} d_{ll'} \tag{3.17}$$

provided $\sum_l \gamma_{jl} = 1$ [39].

This way, it is possible to compute an LVQ classifier based on pairwise dissimilarities or similarities only, representing prototypes only implicitly in terms of the coefficients γ_{jl}.

To provide out-of-sample extensions for a novel data point ξ we assume that novel data points are represented in terms of their similarity or dissimilarity to the training points $s(\xi_i, \xi)$ or $d(\xi_i, \xi)$, respectively. Then, similarly, we obtain the distance

$$\|\xi - w_j\|^2_{pq} = s(\xi, \xi) - 2\sum_l \gamma_{jl} s(\xi, \xi_l) + \sum_{ll'} \gamma_{jl}\gamma_{jl'} s_{ll'} \tag{3.18}$$

which is based on known similarities and the coefficients only. Since the first term is a constant, we can simply drop it to compute the closest prototype for ξ. As an alternative, we find

$$\|\xi - w_j\|^2_{pq} = \sum_l \gamma_{jl} d(\xi, \xi_l) - 1/2 \cdot \sum_{ll'} \gamma_{jl}\gamma_{jl'} d_{ll'} \tag{3.19}$$

based on known dissimilarities and the coefficients of the prototypes.

We have just derived formulas which compute distances in terms of the similarities/dissimilarities only. Hence the result of the classification is entirely independent of the chosen embedding of prototypes and any other embedding which yields the same similarities/dissimilarities will give the same result. Further, we can even ensure that the training process is independent of the concrete embedding, provided that learning rules are expressed in a similar way in terms of similarities or dissimilarities only. We now turn to possible training algorithms for these classifiers.

3.4 Relational GLVQ

For GLVQ [81], a kernelized version has been proposed in Section 3.1. However, this refers to a kernel matrix only, meaning it requires Euclidean similarities instead of general symmetric dissimilarities. Here we assume that pairwise dissimilarities $d_{il} = d(\xi_i, \xi_l)$ are given which do not necessarily correspond to a Euclidean, but pseudo-Euclidean embedding only, meaning we assume symmetry $d_{il} = d_{li}$ and zero diagonal $d_{ii} = 0$. Based on the pseudo-Euclidean embedding, for training, we use the cost function in Equation 2.7 where we substitute the distance computations by Equation 3.17

$$\sum_i \frac{(D\gamma_+)_i - \frac{1}{2} \cdot \gamma_+^T D\gamma_+ - (D\gamma_-)_i + \frac{1}{2} \cdot \gamma_-^T D\gamma_-}{(D\gamma_+)_i - \frac{1}{2} \cdot \gamma_+^T D\gamma_+ + (D\gamma_-)_i - \frac{1}{2} \cdot \gamma_-^T D\gamma_-} \tag{3.20}$$

where the closest correct and wrong prototype are referred to, indicated by the superscript + and −, respectively. A stochastic gradient descent leads to adaptation rules for the coefficients γ_+ and γ_-. Component l of these vectors is adapted by the rules

$$\begin{aligned}\Delta\gamma_{+l} &\sim -\frac{2\cdot d\left(\xi_i, w_-\right)}{\left(d\left(\xi_i, w_+\right)+d\left(\xi_i, w_-\right)\right)^2}\cdot\frac{\partial\left(\left(D\gamma_+\right)_i-\frac{1}{2}\cdot\gamma_+^T D\gamma_+\right)}{\partial\gamma_{+l}}\\ \Delta\gamma_{-l} &\sim \frac{2\cdot d\left(\xi_i, w_+\right)}{\left(d\left(\xi_i, w_+\right)+d\left(\xi_i, w_-\right)\right)^2}\cdot\frac{\partial\left(\left(D\gamma_-\right)_i-\frac{1}{2}\cdot\gamma_-^T D\gamma_-\right)}{\partial\gamma_{-l}}\end{aligned} \tag{3.21}$$

where the later derivative can be computed easily as $d_{il}-\sum_{l'}\gamma_{\pm l}d_{ll'}$. This way, the relational GLVQ (RGLVQ) algorithm as introduced in [42, 43], which adapts prototypes in a supervised manner similar to GLVQ is given for general dissimilarity data, whereby prototypes are implicitly embedded in pseudo-Euclidean space. Clusters are represented in terms of prototypes for general dissimilarity data by the resulting classifier. These prototypes can usually not be inspected directly, although they correspond to vector positions in pseudo-Euclidean space, because the pseudo-Euclidean embedding is not computed directly.

3.5 Relational RSLVQ

In a similar way, RSLVQ can be extended to general dissimilarity data [44]: Prototype w_j is represented implicitly by means of the coefficient vectors γ_j. Then, the equivalent characterization of distances can be used in the RSVLQ cost function in Equation 2.13 leading to the costs of relational RSLVQ (RRSLVQ)

$$\sum_i \log\frac{\sum_{c(w_j)=y} P\left(j\right)\cdot\text{const}_j\cdot\exp\left(\left(D\cdot\gamma_j\right)_i-\frac{1}{2}\cdot\gamma_j^T D\gamma_j/\sigma_j^2\right)}{\sum_j P\left(j\right)\cdot\text{const}_j\cdot\exp\left(\left(D\cdot\gamma_j\right)_i-\frac{1}{2}\cdot\gamma_j^T D\gamma_j/\sigma_j^2\right)} \tag{3.22}$$

A stochastic gradient descent leads to the adaptation rule

$$\Delta\gamma_{jl}\sim\begin{cases}\left(P_y\left(j|\xi_i\right)-P\left(j|\xi_i\right)\right)\\ \qquad\cdot\dfrac{\partial\left(\left(D\gamma_j\right)_i-\frac{1}{2}\cdot\gamma_j^T D\gamma_j\right)}{\partial\gamma_{jl}} & \text{if } c\left(w_j\right)=y_i\\ -P\left(j|\xi_i\right)\\ \qquad\cdot\dfrac{\partial\left(\left(D\gamma_j\right)_i-\frac{1}{2}\cdot\gamma_j^T D\gamma_j\right)}{\partial\gamma_{jl}} & \text{if } c\left(w_j\right)\neq y_i\end{cases} \tag{3.23}$$

As before, the probabilities are defined as $P(j|\xi_i) = \frac{P(j)\exp\left(f\left(\xi_i,w_j,\sigma_j^2\right)\right)}{\sum_j P(j)\exp\left(f\left(\xi_i,w_j,\sigma_j^2\right)\right)}$ and $P_y(j|\xi_i) = \frac{P(j)\exp\left(f\left(\xi_i,w_j,\sigma_j^2\right)\right)}{\sum_{c\left(w_j\right)=y_j} P(j)\exp\left(f\left(\xi_i,w_j,\sigma_j^2\right)\right)}$ like for the Equation 2.14. Note however, that these terms do not necessarily have a valid counterpart as probabilities due to the fact that distances can become negative in pseudo-Euclidean space.

3.6 Discussion

We have introduced four extensions of prototype-based methods to general distances or kernels. Thereby, two of these approaches rely on a kernelization and can be used for valid kernels only, two alternatives focus on general dissimilarity data. Besides these approaches, there do exist further alternatives in the literature. Median approaches, as an example, restrict the prototype positions to exemplars, such that distances are always well defined. The challenge is to find efficient schemes which can locate suitable optima in this discrete space of possible solutions, example optimization relying on expectation-maximization schemes [70]. Alternatives further restrict the considered similarity measures, and focus on differentiable kernels or general divergences, for which smooth adaptation is possible by means of gradient techniques [96, 95].

We will not consider these alternatives in the following. Rather, the main focus is on two aspects, on the one hand what the main principle behind these four approaches is and in how far they are similar / different and on the other hand how these techniques can efficiently be used for applications. We will elucidate these questions by introducing a general framework which covers these approaches and points out their differences / similarities. Further, we will deliver extensive comparisons of the techniques for a variety of benchmarks. Afterwards, we will have a glimpse at questions which occur when applying the techniques, namely their efficiency and interpretability.

Chapter 4

General view

Chapter overview *The goal of this chapter is to stratify the proposed approaches for non-vectorial LVQ within a general framework, and to elucidate the differences of the different realizations. Further, the behavior of the techniques in practical applications will be considered and compared to alternative state-of-the-art techniques. Since the latter also includes SVMs, we discuss techniques how general proximities can be transferred to a valid kernel. Parts of this chapter rely on the publication [40].*

As discussed previously, original LVQ can be used for standard Euclidean vectors only, but kernel and relational variants of generalized LVQ or robust soft LVQ extend their scope towards data characterized by pairwise proximities. We discussed in Chapter 3 techniques how to extend GLVQ and RSLVQ, respectively, to deal with pairwise similarities or dissimilarities. Now the question occurs about the differences of these techniques. We will propose a general framework how the methods can be combined based on the background of a pseudo-Euclidean embedding of the data. This does not only cover the approaches kernel GLVQ, relational GLVQ, kernel RSLVQ, and relational RSLVQ, but also unsupervised prototype based techniques which are based on a cost function can be put into this framework including kernel and relational neural gas and kernel and relational self-organizing maps based on Heskes' cost function.

The principled way how to train such LVQ classifiers is essentially independent of the precise form of the cost function. For similarity or dissimilarity data, there exist two different possibilities to arrive at valid training rules for online learning, concrete instances of which are given by kernel variants or relational variants. Here, we give a more fundamental view on these two possibilities of the optimization of the cost function by stochastic gradient techniques and their mathematical background.

4.1 Optimization concerning the coefficients

The cost function of both, GLVQ in Equation 2.7 and RSLVQ in Equation 2.13 has the form $f\left(D\left(\xi, w\right)\right)$ with $D\left(\xi, w\right) = \left(d\left(\xi_i, w_j\right)\right)_{i=1,\ldots,N, j=1,\ldots,m}$ as already stated in Section 2.4. Provided prototypes are given by linear combinations of data in the underlying pseudo-Euclidean embedding $w_j = \sum \gamma_j \Phi(x_j)$, these costs become

$$f\left(\left(s_{ii} - 2\sum_l \gamma_{jl} s_{il} + \sum_{ll'} \gamma_{jl}\gamma_{jl'} s_{ll'}\right)_{i=1,\ldots,N, j=1,\ldots,m}\right) \tag{4.1}$$

for similarities or

$$f\left(\left(\sum_l \gamma_{jl} d_{il} - 1/2 \cdot \sum_{ll'} \gamma_{jl}\gamma_{jl'} d_{ll'}\right)_{i=1,\ldots,N, j=1,\ldots,m}\right) \tag{4.2}$$

for dissimilarities based on Equation 3.16 and Equation 3.17 respectively. We can smoothly vary prototypes w_j in pseudo-Euclidean space by adapting the coefficients γ_{jl}. The latter can be adapted by a standard gradient technique as proposed in relational RSLVQ [44] and relational GLVQ [42, 43]. In both cases, a gradient method with respect to γ_{jl} is driven by the term

$$\frac{\partial f}{\partial \gamma_{jl}} = \sum_i \frac{\partial f\left(D\left(\xi, w\right)\right)}{\partial d\left(\xi_i, w_j\right)} \cdot \left(-2 s_{il} + 2\sum_{l'} \gamma_{jl} s_{ll'}\right) \tag{4.3}$$

if similarities are considered or by the term

$$\frac{\partial f}{\partial \gamma_{jl}} = \sum_i \frac{\partial f\left(D\left(\xi, w\right)\right)}{\partial d\left(\xi_i, w_j\right)} \cdot \left(d_{il} - \sum_{l'} \gamma_{jl} d_{ll'}\right) \tag{4.4}$$

for dissimilarities, providing adaptation rules for both cost functions by means of a gradient descent or ascent, or corresponding single summands only in case of a stochastic gradient technique. In particular, in these rules, only pairwise similarities or dissimilarities of data are required, meaning it is not necessary to compute the pseudo-Euclidean embedding.

As an example the corresponding adaptation rule of RSLVQ in Equation 2.14 for dissimilarities, which we refer to as *relational RSLVQ* (RRSLVQ) as introduced in Section 3.5, yields by optimization of the cost function with

respect to γ_{jl} the update rule, given a data point ξ_i

$$\Delta\gamma_{jl} \sim \begin{cases} \left(P_y\left(j|\xi_i\right) - P\left(j|\xi_i\right)\right) \cdot \left(d_{il} - \sum_{l'} \gamma_{jl} d_{ll'}\right) & \text{if } c\left(w_j\right) = y_i \\ -P\left(j|\xi_i\right) \cdot \left(d_{il} - \sum_{l'} \gamma_{jl} d_{ll'}\right) & \text{if } c\left(w_j\right) \neq y_i \end{cases} \tag{4.5}$$

where the probabilities $P_y\left(j|\xi_i\right)$ and $P\left(j|\xi_i\right)$ are computed as before based on the dissimilarities $d\left(\xi_i, w_j\right)$ which are expressed via d_{ij}.

Analogously, the corresponding adaptation rule of GLVQ in Equation 2.8 for dissimilarities, which we refer to as *relational GLVQ* (RGLVQ) as introduced in Section 3.4, yields the update rule, given a data point ξ_i

$$\begin{aligned} \Delta\gamma_{+l} &\sim -\frac{2 \cdot d\left(\xi_i, w_-\right)}{\left(d\left(\xi_i, w_+\right) + d\left(\xi_i, w_-\right)\right)^2} \cdot \left(d_{il} - \sum_{l'} \gamma_{+l} d_{ll'}\right) \\ \Delta\gamma_{-l} &\sim \frac{2 \cdot d\left(\xi_i, w_+\right)}{\left(d\left(\xi_i, w_+\right) + d\left(\xi_i, w_-\right)\right)^2} \cdot \left(d_{il} - \sum_{l'} \gamma_{-l} d_{ll'}\right) \end{aligned} \tag{4.6}$$

Note that the parameters γ_{jl} are not yet normalized. This can be achieved in different ways, for example by explicit normalization after every adaptation step, or by the inclusion of corresponding barrier functions in the cost function, which yields additional regularizing terms of the adaptation. We will use an explicit normalization in the following, meaning after every adaptation step, we divide the vector of coefficients by its component-wise sum.

This principle gives an explanation of relational LVQ, and it opens a way to directly use LVQ variants provided similarities rather than dissimilarities are given, since the gradient scheme in Equation 4.3 can be used alternatively.

4.2 Optimization concerning the prototypes

Kernel variants follow a different principle as compared to these relational variants. We consider the more general case of a similarity or dissimilarity matrix, for the moment. The gradient of the cost function with respect to the prototype w_j yields

$$\sum_i \frac{\partial f\left(D\left(\xi, w\right)\right)}{\partial d\left(\xi_i, w_j\right)} \cdot \frac{\partial d\left(\xi_i, w_j\right)}{\partial w_j} \tag{4.7}$$

which is a computation which refers to the embedding space provided by a pseudo-Euclidean embedding. The dissimilarity d is defined as $d\left(\xi_i, w_j\right) =$

$(\xi - w_j)^t \cdot I_{pq} \cdot (\xi - w_j)$ in pseudo-Euclidean space, where I_{pq} is the diagonal matrix with p entries 1 and q entries -1 as before. Thus, we obtain $\partial d(\xi_i, w_j)/\partial w_j = -2 \cdot I_{pq}(\xi_i - w_j)$. This yields the stochastic gradient update, given one data point ξ_i

$$\Delta w_j \sim -\frac{\partial f\left(\left(d(\xi_i, w_j)\right)_{i,j}\right)}{\partial d(\xi_i, w_j)} \cdot I_{pq}\left(\xi_i - \sum_l \gamma_{jl}\xi_l\right) \tag{4.8}$$

The idea of the learning rule as proposed in kernel RSLVQ [51] and kernel GLVQ [76], respectively, is to decompose this update into the contributions of the coefficients γ_{jl}, such that updates can be computed without an explicit reference to the embedding space. This is possible if and only if the update rule decomposes into a sum of the form $\sum_l \Delta\gamma_{jl}\xi_l$. In this case, an update of the coefficients which is proportional to the terms $\Delta\gamma_{jl}$ of this decomposition mimics the effect of a stochastic gradient for the prototype w_j, and updates can be performed implicitly by updates of the coefficients only.

This decomposition, however, is usually not possible. While most components of the update in Equation 4.8 can be decomposed into contributions of the coefficients since they do not refer to components of the vector ξ_i, the ingredient I_{pq} refers to a vectorial operation which depends on the pseudo-Euclidean embedding. Thus, it is in general not possible to turn this adaptation rule into a rule which can be done implicitly without explicit reference to the pseudo-Euclidean embedding.

In one very relevant special case, however, a decomposition can be found. Assume data are Euclidean, meaning $q = 0$, in other words a valid kernel is present. In this case, we can assume without loss of generality that p equals the dimensionality of the vectors ξ_i, since components beyond p do not contribute to the distance measure in the embedding. Thus, the learning rule in Equation 4.8 becomes

$$\Delta w_j \sim \frac{\partial f\left(\left(d(\xi_i, w_j)\right)_{i,j}\right)}{\partial d(\xi_i, w_j)} \cdot \left(\sum_l (\gamma_{jl} - \delta_{il})\,\xi_l\right) \tag{4.9}$$

with Kronecker symbol δ_{il}. Hence we obtain the update

$$\Delta\gamma_{jl} \sim \begin{cases} \dfrac{\partial f\left(\left(d(\xi_i, w_j)\right)_{i,j}\right)}{\partial d(\xi_i, w_j)} \cdot \gamma_{jl} & \text{if } l \neq i \\ \dfrac{\partial f\left(\left(d(\xi_i, w_j)\right)_{i,j}\right)}{\partial d(\xi_i, w_j)} \cdot (\gamma_{jl} - 1) & \text{if } l = i \end{cases} \tag{4.10}$$

As an example the corresponding adaptation rule of RSLVQ in Equation 2.14 for Gram matrices, which we refer to as *kernel RSLVQ* (KRSLVQ) as introduced in Section 3.2, yields by optimization of the cost function with respect to w_j the update rule, given a data point ξ_i

$$\Delta\gamma_{jm} \sim \begin{cases} -\left(P_y\left(j|\Phi\left(\xi_i\right)\right) - P\left(j|\Phi\left(\xi_i\right)\right)\right) & \\ \qquad \cdot\gamma_{jm} & \text{if } \xi_m \neq \xi_i, c\left(w_j\right) = y_i \\ \left(P_y\left(j|\Phi\left(\xi_i\right)\right) - P\left(j|\Phi\left(\xi_i\right)\right)\right) & \\ \qquad \cdot\left(1 - \gamma_{jm}\right) & \text{if } \xi_m = \xi_i, c\left(w_j\right) = y_i \\ P\left(j|\Phi\left(\xi_i\right)\right) & \\ \qquad \cdot\gamma_{jm} & \text{if } \xi_m \neq \xi_i, c\left(w_j\right) \neq y_i \\ -P\left(j|\Phi\left(\xi_i\right)\right) & \\ \qquad \cdot\left(1 - \gamma_{jm}\right) & \text{if } \xi_m = \xi_i, c\left(w_j\right) \neq y_i \end{cases} \tag{4.11}$$

where the probabilities $P_y\left(j|\xi_i\right)$ and $P\left(j|\xi_i\right)$ are computed as before based on the kernel $k\left(\xi_i, w_j\right)$ which is expressed via k_{ij}.

Analogously, the corresponding adaptation rule of GLVQ in Equation 2.8 for Gram matrices, which we refer to as *kernel GLVQ* (KGLVQ) as introduced in Section 3.1, yields the update rule, given a data point ξ_i

$$\begin{aligned} \Delta\gamma_{+m} \sim & \begin{cases} \left(1 - \frac{2 \cdot d\left(\Phi\left(\xi_i\right), w_-\right)}{\left(d\left(\Phi\left(\xi_i\right), w_+\right) + d\left(\Phi\left(\xi_i\right), w_-\right)\right)^2}\right)\gamma_{+m} & \text{if } \xi_m \neq \xi_i \\ \left(1 - \frac{2 \cdot d\left(\Phi\left(\xi_i\right), w_-\right)}{\left(d\left(\Phi\left(\xi_i\right), w_+\right) + d\left(\Phi\left(\xi_i\right), w_-\right)\right)^2}\right)\gamma_{+m} & \\ \qquad + \frac{2 \cdot d\left(\Phi\left(\xi_i\right), w_-\right)}{\left(d\left(\Phi\left(\xi_i\right), w_+\right) + d\left(\Phi\left(\xi_i\right), w_-\right)\right)^2} & \text{if } \xi_m = \xi_i \end{cases} \\ \Delta\gamma_{-m} \sim & \begin{cases} \left(1 + \frac{2 \cdot d\left(\Phi\left(\xi_i\right), w_+\right)}{\left(d\left(\Phi\left(\xi_i\right), w_+\right) + d\left(\Phi\left(\xi_i\right), w_-\right)\right)^2}\right)\gamma_{-m} & \text{if } \xi_m \neq \xi_i \\ \left(1 + \frac{2 \cdot d\left(\Phi\left(\xi_i\right), w_+\right)}{\left(d\left(\Phi\left(\xi_i\right), w_+\right) + d\left(\Phi\left(\xi_i\right), w_-\right)\right)^2}\right)\gamma_{-m} & \\ \qquad - \frac{2 \cdot d\left(\Phi\left(\xi_i\right), w_+\right)}{\left(d\left(\Phi\left(\xi_i\right), w_+\right) + d\left(\Phi\left(\xi_i\right), w_-\right)\right)^2} & \text{if } \xi_m = \xi_i \end{cases} \end{aligned} \tag{4.12}$$

Note that this update constitutes a gradient technique only for Euclidean data, and it exactly resembles the underlying vectorial counterpart. One can nevertheless apply this update rule also for non-Euclidean settings, where the update step often at least improves the model since the positive parts of the pseudo-Euclidean space are usually dominant. However, it is not guaranteed that a valid gradient technique is present in this case. Note that, again,

normalization of the coefficients can take place in different ways, via direct normalization, which we will use in the following, or barrier functions, for example.

4.3 Characteristics of the methods

In Figure 4.1, we put these specific approaches into a general framework which characterizes different fundamental possibilities how to extend non-vectorial LVQ classifiers towards similarities or dissimilarities based on the background of a pseudo-Euclidean embedding of the data. The main characteristics are given by the choice of the cost function, the way in which optimization takes place, and the interface to the data in terms of similarities or dissimilarities.

The most obvious difference of these two ways to update the coefficients consists in the fact that updates with respect to the coefficients follow a gradient technique whereas updates with respect to the weights, if done implicitly without explicit reference to the embedding, constitute a valid gradient method only if data are Euclidean.

But also in the Euclidean case where both updates follow a gradient technique, differences of the two update rules are observed. Prototypes depend linearly on the coefficients. Hence every local optimum of the cost function with respect to the weights corresponds to a local optimum with respect to the coefficients and vice versa. As a consequence, the solutions which can be found by these two update rules coincide as regards the entire set of possible solutions provided the gradient techniques are designed in such a way that local optima are reached.

However, the single update steps of the two techniques are not identical, since taking the gradient does not commute with linear operations. Thus, it is possible that different local optima are reached in a single run even if the methods are started from the same initial condition.

Other differences of the techniques rely on the choice of the cost function, where besides GLVQ and RSLVQ, also unsupervised counterparts could be used. Further, the techniques differ in their data access which can take place via similarities or dissimilarities. In summary, when extending LVQ schemes to general similarities or dissimilarities, we have the choices as described in Table 4.1. Interestingly, quite a few further techniques could be designed, corresponding to alternative combinations of the basic constituents, for example relational LVQ variants which work on similarity rather than dissimilarity data.

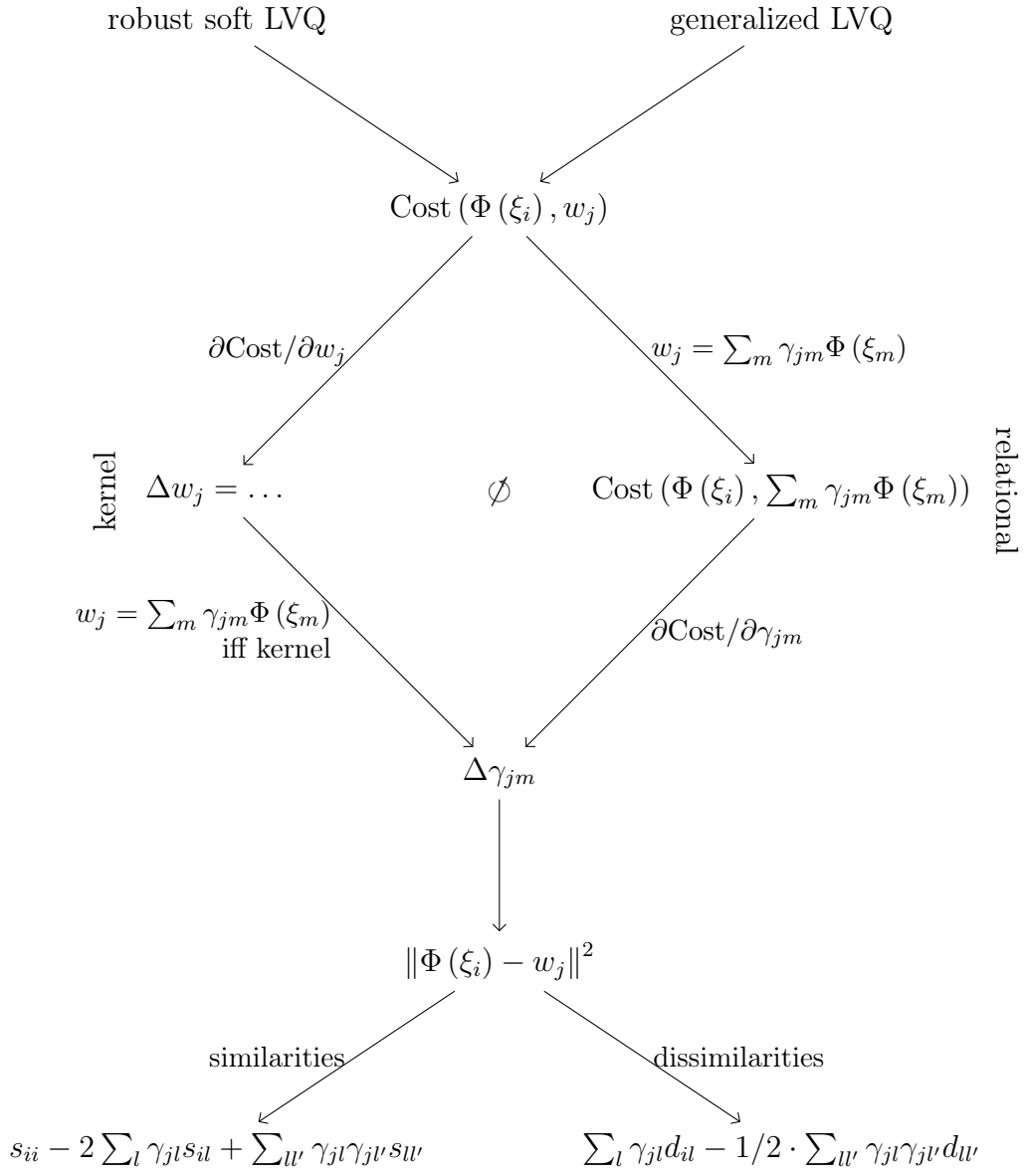


Figure 4.1: General framework for non-vectorial LVQ schemes for proximity data.

	data representation		training technique	
	dissimilarities	similarities	gradient concerning	
		or kernel	γ_{jl}	w_j
kernel GLVQ		x		x
kernel RSLVQ		x		x
relational GLVQ	x		x	
relational RSLVQ	x		x	

Table 4.1: Different possible choices when applying LVQ schemes for (dis-)similarity data.

4.4 Transferability of the mathematical background

We have already observed that one of the adaptation rules as proposed in the literature constitutes a valid gradient technique if and only if data are Euclidean. There are more severe reasons why it constitutes a desired property of the data to be Euclidean if referring to the theoretical motivation of the method in case of RSLVQ.

This algorithm is derived as a likelihood ratio optimization technique. Since distances in pseudo-Euclidean space can become negative, a Gaussian distribution based on these distances does no longer constitute a valid probability distribution. Thus, Euclidean data are required to preserve the mathematical motivation of RSLVQ schemes as a likelihood optimization for (dis-)similarities.

GLVQ, in contrast, relies on the idea to optimize the hypothesis margin, which is the distance that the classifier can travel without changing the way it labels any of the sample points. In comparison the distance between an instance and the decision boundary induced by the classification rule is the sample margin, which is maximized by the support vector machine (SVM). Generalization bounds which depend on this hypothesis margin can be found which are based on the so-called Rademacher complexity of the function class induced by prototype based methods. Essentially, wide parts of the argumentation as given in [84] can be directly transferred to the pseudo-Euclidean setting, although the situation might be more difficult if the rank of the space is not limited and Krein spaces come into the play. The article [84] considers the more general setting where, in addition to adaptive prototypes, the quadratic form can be learned. Here, we consider the more simple function associated to GLVQ networks. Thereby, we restrict to the classification

problems incorporating two classes 0 and 1 only, similar to [84].

In our case, the classification is based on the real-valued function

$$\xi \mapsto \left(\min_{w_i \,:\, c(w_i)=0} \|w_i - \xi\|_{pq}^2 - \min_{w_i \,:\, c(w_i)=1} \|w_i - \xi\|_{pq}^2 \right) \tag{4.13}$$

the sign of which determines the output class, and the size of which determines the hypothesis margin. This function class is equivalent to the form

$$\begin{aligned} \xi \mapsto \Bigg(& \min_{w_i \,:\, c(w_i)=0} \left(\|w_i\|_{pq}^2 - 2\langle w_i, \xi \rangle_{pq} \right) \\ & - \min_{w_i \,:\, c(w_i)=1} \left(\|w_i\|_{pq}^2 - 2\langle w_i, \xi \rangle_{pq} \right) \Bigg) \end{aligned} \tag{4.14}$$

It is necessary to estimate the so-called Rademacher complexity of this function class relying on techniques as introduced for example in [5], to bound the generalization error. Since we do not refer to specifics of the definition of the Rademacher complexity, rather we refer to well-known structural results only, we do not introduce a precise definition at this place. Essentially, the complexity measures the amount of surprise in LVQ networks by taking the worst case correlation to random vectors.

As in [84] a structural decomposition can take place. This function can be decomposed into the linear combination of a composition of a Lipschitz-continuous function (min) and the function $\|w_i\|_{pq}^2 - 2\langle w_i, \xi \rangle_{pq}$. We can realize the bias $\|w_i\|_{pq}^2$ as additional weight if we enlarge every input vector by a constant component 1. Further, the sign of the components of w_i can be arbitrary, thus the signs in this bilinear form are not relevant and can be simulated by appropriate weights. Thus, we need to consider a linear function in the standard Euclidean vector space. As shown in [5] its Rademacher complexity can be bounded by a term which depends on the maximum Euclidean length of ξ and w_i and the square root of the number of samples for the evaluation of Rademacher complexity. The Euclidean lengths ξ and w_i can be limited in terms of the a priorly given similarities or dissimilarities. The vectorial representation of ξ corresponds to a column of $Q\,|\Lambda|^{1/2}$ with unitary Q and diagonal matrix of eigenvalues Λ. Thus, the Euclidean length of this vector is limited in terms of the largest eigenvalue of the similarity matrix S or $\Psi(D)$. Since w_i is described as a convex combination, the same holds for w_i.

As a consequence, the argumentation of [84] can be transferred immediately to the given setting, meaning large margin generalization bounds hold for GLVQ networks also in the pseudo-Euclidean setting. Since only the form of the classifier is relevant for this argumentation, but not the training

technique itself, the same argumentation also holds for a classifier obtained using the RSLVQ cost function in pseudo-Euclidean space, and it holds for both training schemes as introduced above.

4.5 Techniques to enforce that data are Euclidean

Albeit large margin generalization bounds transfer to the pseudo-Euclidean setting, it might be beneficial for the training prescription to transform data to obtain a purely Euclidean similarity or dissimilarity.

There exist two prominent approaches to turn a given similarity matrix into a valid Gram matrix, see for example [17, 73]:

- *clip*: set all negative eigenvalues of the matrix Λ associated to the similarities to 0, meaning use only the positive dimensions of the pseudo-Euclidean embedding. The corresponding matrix is referred to as Λ_{clip}. This preprocessing corresponds to the linear transformation $Q\Lambda_{\text{clip}}Q^t$ of the data, where Q stems from the pseudo-Euclidean embedding, see Equation 3.10. The assumption underlying this transformation is that negative eigenvalues are caused by noise in the data, and the given matrix should be substituted by the nearest positive semidefinite one.

- *flip*: take $|\Lambda|$ instead of the matrix Λ, meaning use the standard Euclidean norm in pseudo-Euclidean space. This corresponds to the linear transform $Q\,|\Lambda|\,Q^t$ of the similarity matrix. The motivation behind this procedure is the assumption that the negative directions contain relevant information. Hence the simple Euclidean norm is used instead of the pseudo-Euclidean one.

Since both corrections correspond to linear transformations, their out-of-sample extension is immediate. It has already been tested in the context of SVM in [17] in how fare these preprocessing steps yield reasonable results, in some cases greatly enhancing the performance.

4.6 Experiments

We test the various LVQ variants from Chapter 3 in a couple of benchmark data sets. Relational RSLVQ is trained using dissimilarities and gradients with respect to γ_{jl} and kernel RSLVQ is trained based on similarities and gradients with respect to w_j. In the literature, the corresponding settings for

GLVQ can be found [76, 42]. We compare the methods to the SVM and a k-nearest neighbor classifier (k-NN) on a variety of benchmarks as introduced in [17].

Results for SVM and k-NN are recomputed using the setting as described in [17], leading to the same or better results. Thereby, data are preprocessed using clip or flip to guarantee positive definiteness for SVM, if necessary. The latter is used with the RBF kernel and optimized meta-parameters in [17]. For multi-class classification, the one versus one scheme has been used. For k-NN k $\in \{1, 3, 5\}$ is chosen. In comparison, we train kernel and relational GLVQ and RSLVQ networks using the preprocessing steps clip and flip in comparison to a direct application of the methods for the original data.

Results of a 20-fold cross-validation with the same partitioning as proposed in [17] are reported. Note that a decomposition of a data set characterized by a similarity matrix into training and test set corresponds to a selection of a set of indices I. The sub-matrix formed by $(k_{ij})_{i,j\in I}$ characterizes the training set and distances of prototypes to test points for a classification of the test set can be computed based on $(k_{ij})_{i\in I, j\notin I}$.

For training, prototypes are initialized by means of normalized random coefficients γ_{jm}. Class labels are taken into account, setting the coefficient m to zero if the label of point ξ_m does not coincide with the prototype label $c(w_j)$, which among others was used in [76] in order to stabilize the kernel GLVQ algorithm. Meta-parameters are optimized on the data sets using cross-validation. Further, while training, we guarantee that prototypes are contained in the convex hull of the data by setting negative coefficients to zero after every adaptation step and adding a normalization of the vector γ_i to 1 after every adaptation step.

Meta-parameters specifically for kernel RSLVQ such as the learning rate have only a minor influence on the final result, but on the speed of convergence only. As already discussed in [88], the bandwidth of the model for the RSLVQ variants influences the result and the prototype location, and strategies to also adapt the bandwidth in parallel to the prototype locations have been proposed in [86, 89], for example. Since the bandwidth should be adapted on a slower scale than the prototype positions, very time consuming algorithms result this way, because of which we simply optimize σ by cross-validation in the range between 0.05 and 1.0 with a step size of 0.05. The variance between the optimum parameters was mostly in a range of 10^{-5}.

Benchmark data sets

We compare the presented techniques with different methods on a variety of benchmarks. The data sets represent a variety of similarity matrices which

are, in general, non-Euclidean. It is standard to symmetrize the matrices by taking the average of the matrix and its transposed. Further, the substitution of a given similarity by its normalized variant constitutes a standard preprocessing step, arriving at diagonal entries 1. Even in symmetrized and normalized form, the matrices do not necessarily provide a valid kernel. There exist two prominent approaches to turn a given similarity matrix into a valid Gram matrix as discussed in Section 4.5.

We also report the signatures of the data whereby a cutoff at 0.0001 is made to account for numerical errors of the eigenvalue solver. Additionally, the number of used prototypes is reported, which is chosen as a small multiple of the number of classes. We use a fixed number of prototypes only, taking the values from previous experimental settings [51], noticing that the exact number of prototypes is not severely influencing the result since no overfitting takes place.

Six benchmark data sets were used as introduced in [17]:

- *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. This matrix is fairly sparse and mildly non-Euclidean with signature $(192, 1, 11)$. Class labeling of a book is given by the author. The number of prototypes which is chosen in all LVQ settings is 94.

- *Aural Sonar*: This data set consists of 100 wide band sonar signals corresponding to two classes, observations of interest versus clutter. Similarities are determined based on human perception, averaging over 2 random probands for each signal pair. The signature is $(61, 38, 1)$. Class labeling is given by the two classes target of interest versus clutter. The number of prototypes chosen in LVQ scenarios is 10.

- *Face Rec*: 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 signature is given by $(45, 0, 900)$. Labeling corresponds to the 139 different persons. The number of prototypes is 139.

- *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. The signature is $(54, 66, 121)$. Class labeling corresponds to the eight patrol units. The number of prototypes is 24.

- *Protein*: 213 proteins are compared based on evolutionary distances comprising four different classes according to different globin families. The signature is $(169, 38, 6)$. Labeling is given by four classes corresponding to globin families. The number of prototypes is 20.
- *Voting*: Voting contains 435 samples with categorical data compared by means of the value difference metric. The signature is $(16, 1, 418)$. Class labeling into two classes is present. The number of prototypes is 20.

Note that the rank of the Gram matrix is given by the number of positive eigenvalues if clip is used as preprocessing, and the sum of non-negative eigenvalues if the original data or flip are used. The eigenvalue spectra of the data sets are depicted in Figure 4.2. As can be seen from the graphs, the data sets FaceRec and Voting are almost Euclidean, while all others contain a considerable percentage of negative eigenvalues. Interestingly, the intrinsic dimensionality as mirrored by the number of eigenvalues which have a relevant absolute value is high for Amazon47 and Patrol.

Results

The results obtained by the kernelized and relationalized versions of GLVQ and RSLVQ, which were introduced in Chapter 3, in comparison to k-NN and SVM are reported in Table 4.2. Due to its almost Euclidean nature, preprocessing by clip and flip has hardly an effect for FaceRec and Voting. For the data sets Patrol and Protein, flip and clip change the similarity severely, as can be spotted by the change of the k-NN error. Albeit all other data sets also display a considerable non-Euclideanity as can be seen by the spectrum, flip or clip do have a minor effect on these data only, resulting in up to 3% change of the classification accuracy. Note that it depends very much on the data set and the used technique, which preprocessing yields best results. In general, SVM can show instabilities for non-positive definite data because some numeric schemes used for parameter optimization in SVM built on positive definite similarity matrices. Unless data are Euclidean, where preprocessing using clip or flip has no effect, it is not clear a priori which technique is best, and it can happen that the best preprocessing also depends on the different learning algorithms as can be seen for the Patrol data.

Interestingly, for all data sets, one or several of the kernel or relational LVQ techniques display a quality which is at least competitive to if not better than k-NN and SVM on the data set or an appropriate preprocessing. There is an interesting outlier when comparing the different LVQ techniques.

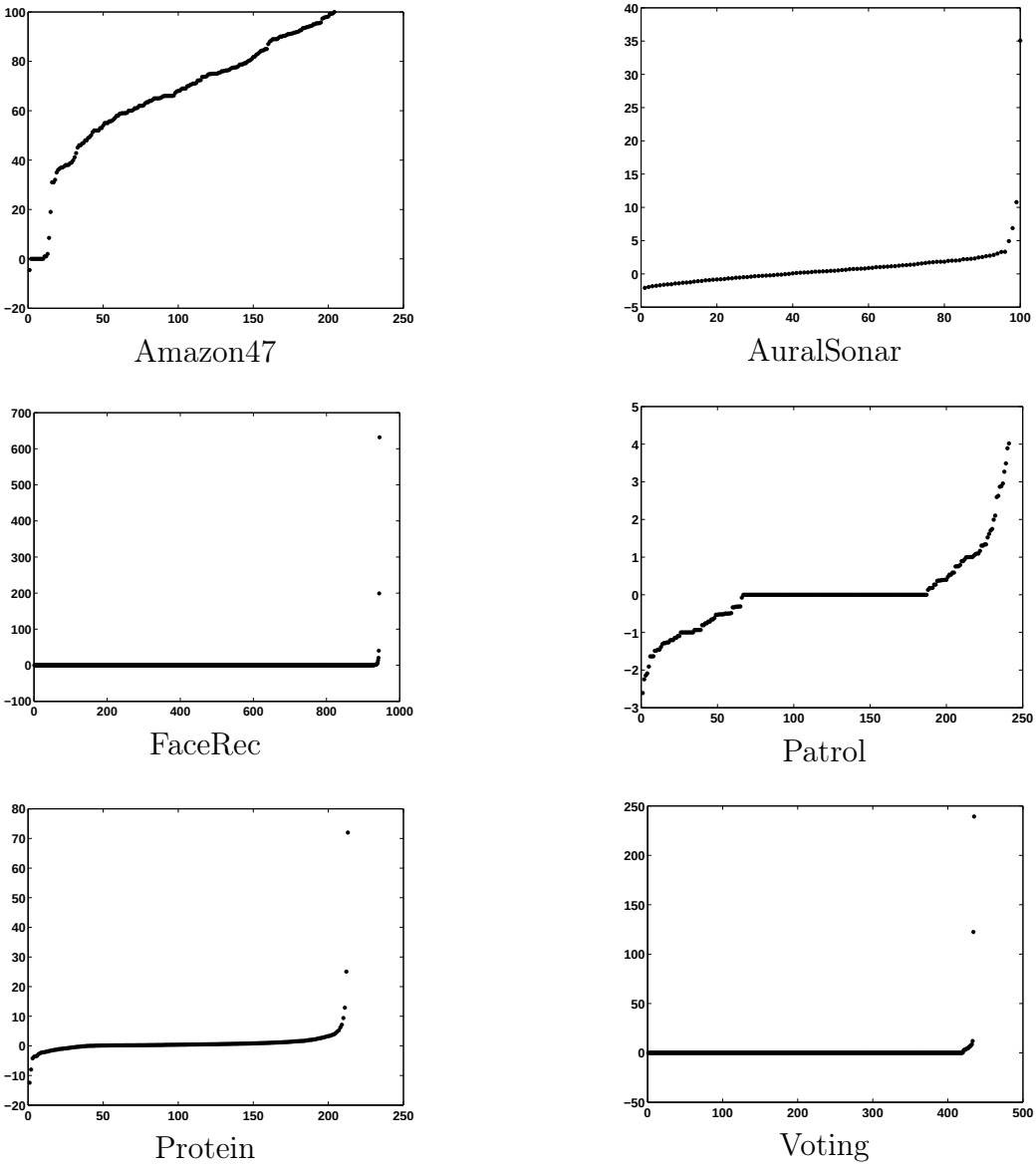


Figure 4.2: Characteristic spectrum of the considered similarities. The data sets differ as concerns negative eigenvalues corresponding to non-Euclideanity, and the number of eigenvalues which are different from zero, corresponding to a high dimensional feature space.

Kernel GLVQ yields more than 50% error for the FaceRec data, corresponding possibly to a local optimum in this case with large basin of attraction. Overall, both, relational GLVQ and kernel RSLVQ yield constantly good classification accuracy.

Interestingly, there are big differences of the different LVQ variants, pointing out the relevance of the different modeling schemes. In general, kernel variants seem to work better for RSLVQ than relational counterparts, possibly due to the unclear interpretation of the notion of probability for the latter. Differences of kernel and relational variants indicate that different basins of attraction are found also in case of GLVQ. Moreover, the different cost functions yield different results, which is a known effect from its vectorial counterpart already [85]. It is hard to determine which method is best over all data sets. Notably, kernel RSLVQ provides best results for half of the data sets, and SVM does not provide the best result for any data set.

Note that the computational complexity of LVQ for similarities or dissimilarities increases as compared to vectorial LVQ schemes. The space complexity for prototype storage becomes $\mathcal{O}(N)$, N being the number of data points, assuming a fixed number of prototypes m. The time complexity is dominated by a matrix multiplication in every adaptation step to compute the dissimilarity which is of order $\mathcal{O}(N^2)$. For SVM, depending on the implementation, space and time complexity are similar, the number of support vectors being usually a fraction of the training set, and training having worst case complexity $\mathcal{O}(N^3)$ unless speedup for example via sequential minimal optimization or core techniques are used.

Usually the storage of the distance matrix constitutes the main bottleneck concerning space – albeit the final classifier requires linear space only, the matrix required to represent the training data is quadratic. We will, however, discuss one speedup technology in Chapter 5, which severely reduces time and memory complexity of the technique.

4.7 Discussion

In summary, when extending LVQ schemes to general similarities or dissimilarities, we have the choices as described in Table 4.1. These yield to eight different possible combinations. In addition, we can further preprocess data into Euclidean form using for example clip or flip. Additional preprocessing steps have been summarized in [17], for example. Interestingly, the results of these different techniques can differ severely on given data sets, and the techniques yield state-of-the-art results in a variety of benchmarks comparable to SVM but easing the burden of preprocessing which is necessary for

	k-NN		SVM		KGLVQ		RGLVQ		KRSLVQ		RRSLVQ	
Amazon47	28.54	(0.83)	21.46	(5.74)	22.80	(5.38)	18.17	(5.39)	**15.37**	(0.36)	22.44	(5.16)
clip	28.78	(0.74)	21.22	(5.49)	21.95	(5.65)	23.78	(7.20)	**15.37**	(0.41)	25.98	(7.48)
flip	28.90	(0.68)	22.07	(6.25)	23.17	(6.10)	20.85	(4.58)	16.34	(0.42)	22.80	(4.96)
Aural Sonar	14.75	(0.49)	12.25	(7.16)	13.00	(7.70)	13.50	(5.87)	11.50	(0.37)	13.00	(7.50)
clip	17.00	(0.51)	12.00	(5.94)	14.50	(8.30)	13.00	(6.96)	**11.25**	(0.39)	13.25	(7.12)
flip	17.00	(0.93)	12.25	(6.97)	12.30	(5.50)	13.00	(6.96)	11.75	(0.35)	13.50	(7.63)
Face Rec	7.46	(0.04)	3.73	(1.32)	**3.35**	(1.29)	3.47	(1.33)	3.78	(0.02)	7.50	(1.49)
clip	7.35	(0.04)	3.84	(1.16)	3.70	(1.35)	3.81	(1.67)	3.84	(0.02)	7.08	(1.62)
flip	7.78	(0.04)	3.89	(1.19)	3.63	(1.16)	3.78	(1.48)	3.60	(0.02)	7.67	(2.21)
Patrol	22.71	(0.33)	15.52	(4.02)	11.67	(4.60)	18.02	(4.65)	17.50	(0.25)	17.71	(4.24)
clip	9.90	(0.16)	13.85	(4.39)	**8.96**	(3.90)	17.29	(3.45)	17.40	(0.29)	21.77	(7.10)
flip	10.31	(0.16)	12.92	(5.09)	9.74	(4.90)	18.23	(5.10)	19.48	(0.34)	20.94	(4.51)
Protein	51.28	(0.77)	30.93	(6.79)	27.79	(7.60)	28.72	(5.24)	26.98	(0.37)	5.58	(3.49)
clip	25.00	(0.74)	12.56	(5.46)	1.63	(2.10)	12.79	(5.36)	4.88	(0.17)	11.51	(5.03)
flip	7.79	(0.18)	1.98	(2.85)	12.33	(6.10)	3.49	(3.42)	**1.40**	(0.05)	4.42	(3.77)
Voting	5.00	(0.01)	5.06	(1.84)	6.55	(1.90)	9.14	(2.10)	5.46	(0.04)	11.26	(2.23)
clip	4.83	(0.02)	5.00	(1.84)	6.55	(1.90)	9.37	(2.02)	5.34	(0.04)	11.32	(2.31)
flip	**4.66**	(0.02)	4.89	(1.78)	6.49	(1.90)	9.14	(2.22)	5.34	(0.03)	11.26	(2.43)

Table 4.2: The mean classification error of different classifiers for benchmark data are reported. Standard deviations are given in parenthesis. The best results are shown in boldface.

SVM for invalid kernels.

The design decisions to arrive at LVQ for (dis-)similarities differ in the following sense, as explained above:

- The cost functions of RSLVQ in Equation 2.13 and GLVQ in Equation 2.7 obey different principles, relevant differences being observable already in the Euclidean setting [85]. The motivation of RSLVQ as likelihood transfers to the Euclidean setting only, while large margin bounds of GLVQ can be transferred to the pseudo-Euclidean case.

- When turning dissimilarities into similarities and backwards, the identity is reached. When starting at similarities, however, data are centered using this transformation.

- Training can take place as gradient with respect to the parameters γ_{jl} or the prototypes w_j. The latter constitutes a valid gradient only if data are Euclidean, while the former follows a gradient also in the pseudo-Euclidean setting. In the Euclidean setting, the same set of local optima is valid for both methods, but the numerical update steps can be different resulting in different local optima in single runs.

A common feature both the kernel and relational extensions share, is the squared training complexity as opposed to cubic complexity for an explicit embedding, which is encountered due to the dependency of the method on the full Gram matrix or a matrix of dissimilarities respectively in contrast to vectorial LVQ, which scales linearly with the number of data points. One approach which is taken in this context and will be discussed in Chapter 5 is the Nyström approximation, which can improve the complexity to linear time [50].

In the following, we will focus on only one approach, kernel robust soft LVQ as proposed in Section 3.2, since it offers an intuitive representation of data in terms of a mixture of labeled Gaussians, and it provides excellent overall results as compared to the investigated alternatives.

Chapter 5

Efficiency

Chapter overview *Within this chapter, we will deal with the Nyström technique which offers a low rank approximation of a given proximity matrix based on few landmarks. By integrating this method into kernel RSLVQ or, alternatively, kernel GLVQ or relational approaches, a linear time method results. We will introduce this technique and demonstrate how it can be integrated into kernel RSLVQ. Further, we elucidate in which cases this approximation is successful, and we develop a quick check which can test the suitability of this approximation prior to training. Parts of this chapter rely on the publications [49, 50].*

In Chapter 4, we have introduced relational and kernel extensions of GLVQ and RSLVQ, making it suitable for complex data sets described in terms of pairwise relations only. In the following, we will exemplarily address kernel RSLVQ which extends the applicability of vectorial LVQ to data which are described by a general Gram matrix as mentioned in Section 3.2, due to its superior performance in benchmark data. While leading to state of the art results, all these relational or kernel extensions have the drawback that quadratic training complexity is encountered due to the dependency of the method on the full Gram matrix or a matrix of dissimilarities respectively in contrast to vectorial LVQ. Even more severely, these techniques require the storage and computation of the full proximity matrix, which is infeasible for large data sets. We investigate the possibility of a speed-up of training by means of a low rank approximation of the Gram matrix. Thereby, we rely on insights from kernel techniques, where a low rank approximation dubbed Nyström method has been integrated in the approach [97]. It turns out that this efficient Nyström approximation can also be integrated into LVQ variants for proximity data, and it yields excellent results if data are intrinsically low dimensional. We show that this latter property can efficiently be checked

by sampling the variance of the approximation prior to training. We demonstrate the behavior of these approximations in a couple of benchmarks which were introduced in Section 4.6. Now, we introduce the Nyström technique and its motivation, first. Then we show how it can efficiently be embedded into the LVQ scheme which was introduced in Chapter 3. Finally, we propose a novel quick check based on sampling which can estimate the suitability of the Nyström technique prior to training.

5.1 Nyström approximation of the Gram matrix

The Nyström technique has been presented in [97] in the context of support vector machines. It allows to approximate a Gram matrix by a low rank approximation [34]. Interestingly, as demonstrated in the latter work [34], the use of the Nyström approximation is not restricted to kernel matrices, meaning positive semidefinite forms, rather it can also be used for more general symmetric matrices which are not necessarily valid Gram matrices. Here, for simplicity, we only consider a kernel as one example. Then this approximation can be integrated into the learning rules in such a way that updates with linear complexity result.We shortly review the main idea behind this approach in the following.

A valid kernel $k(\xi_j, \xi_l)$ can be expanded by orthonormal eigenfunctions ϕ_i and non-negative eigenvalues λ_i in the form

$$k(\xi_j, \xi_l) = \sum_{i=1}^{\infty} \lambda_i \phi_i(\xi_j) \phi_i(\xi_l) \tag{5.1}$$

The eigenfunctions and eigenvalues of a kernel are the solutions of an integral equation

$$\int k(\xi_j, \xi) \phi_i(\xi) p(\xi) d\xi = \lambda_i \phi_i(\xi_j) \tag{5.2}$$

which can be approximated based on the Nyström technique by sampling ξ independent and identically distributed according to p, denoting the sampled values as $\xi_1, \ldots, \xi_m$ after possible reenumeration

$$\frac{1}{m} \sum_{l=1}^{m} k(\xi_j, \xi_l) \phi_i(\xi_l) \approx \lambda_i \phi_i(\xi_j) \tag{5.3}$$

We denote the submatrix corresponding to the m sampled points of the Gram matrix by $\mathbf{K}_{m,m}$. The eigenvalues and eigenvectors of this matrix are denoted

by $\mathbf{U}^{(m)}$ and $\mathbf{\Lambda}^{(m)}$, respectively, characterized by the eigenvalue equation

$$\mathbf{K}_{m,m}\mathbf{U}^{(m)} = \mathbf{U}^{(m)}\mathbf{\Lambda}^{(m)} \tag{5.4}$$

These solutions enable an approximation of the eigenfunctions and eigenvalues

$$\lambda_i \approx \frac{\lambda_i^{(m)}}{m}, \qquad \phi_i\left(\xi_l\right) \approx \frac{\sqrt{m}}{\lambda_i^{(m)}}\mathbf{k}_{\xi_l}\mathbf{u}_i^{(m)} \tag{5.5}$$

where $\mathbf{u}_i^{(m)}$ is the ith column of $\mathbf{U}^{(m)}$ and we use the vector of kernel values

$$\mathbf{k}_{\xi_l} = \left(k\left(\xi_1, \xi_l\right), ..., k\left(\xi_m, \xi_l\right)\right)^T \tag{5.6}$$

This allows us to approximate a given full Gram matrix $\mathbf{K}$ by a low-rank counterpart, since we can use these approximations in the kernel expansion. Subsampling corresponds to a choice of m rows and columns of the matrix, the corresponding submatrix is denoted by $\mathbf{K}_{m,m}$ as before. The corresponding m rows and columns are denoted by $\mathbf{K}_{m,N}$ and $\mathbf{K}_{N,m}$, respectively. These are transposes of each other, since the matrix is symmetric. The approximation as introduced above leads to the following approximation of the kernel expansion by orthonormal eigenfunctions

$$\tilde{\mathbf{K}} = \sum_{i=1}^{m} 1/\lambda_i^{(m)} \cdot \mathbf{K}_{N,m}\mathbf{u}_i^{(m)}\left(\mathbf{u}_i^{(m)}\right)^T \mathbf{K}_{m,N} \tag{5.7}$$

where $\lambda_i^{(m)}$ and $\mathbf{u}_i^{(m)}$ correspond to the $m \times m$ eigenproblem as above. In the case that some $\lambda_i^{(m)}$ are zero, we replace the corresponding fractions with zero. Thus we get, $\mathbf{K}_{m,m}^{-1}$ denoting the Moore-Penrose pseudoinverse,

$$\tilde{\mathbf{K}} = \mathbf{K}_{N,m}\mathbf{K}_{m,m}^{-1}\mathbf{K}_{m,N} \tag{5.8}$$

For a given matrix $\mathbf{K}$ with rank m, this approximation is exact, if the m chosen m-dimensional points are linearly independent resulting in a low rank approximation problem.

5.2 Nyström approximation for LVQ

Hence we can approximate the full Gram matrix as used in kernel RSLVQ which was introduced in Section 3.2 by a low rank approximation. This equation for $\tilde{\mathbf{K}}$ can directly be integrated into the computation of the distances in Equation 3.16 which determine the Gaussians using the identity

$$\left\|\Phi\left(\xi_i\right) - w_j\right\|^2 = \mathbf{e}_i^t\mathbf{K}\mathbf{e}_i - 2 \cdot \mathbf{e}_i^t\mathbf{K}\gamma_j + \gamma_j^t\mathbf{K}\gamma_j \tag{5.9}$$

where $\mathbf{e}_i$ denotes the ith unit vector. Using $\tilde{\mathbf{K}}$ instead, linear complexity results if the matrix vector multiplications are computed first.

We can apply this result for kernel RSLVQ, using this approximation for the kernel matrix

$$\begin{aligned}\|\Phi(\xi_i) - w_j\|^2 \approx &\left(\mathbf{e}_i^t \mathbf{K}_{N,m}\right) \cdot \mathbf{K}_{m,m}^{-1} \cdot \left(\mathbf{K}_{m,N} \mathbf{e}_i\right) \\ &- 2 \left(\mathbf{e}_i^t \mathbf{K}_{N,m}\right) \cdot \mathbf{K}_{m,m}^{-1} \cdot \left(\mathbf{K}_{m,N} \gamma_j\right) \\ &+ \left(\gamma_j^T \mathbf{K}_{N,m}\right) \cdot \mathbf{K}_{m,m}^{-1} \cdot \left(\mathbf{K}_{m,N} \gamma_j\right)\end{aligned} \tag{5.10}$$

Since the full matrix $\tilde{\mathbf{K}}$ is never explicitly computed this way, using rather matrix vector operations in dimensionality m only, the complexity $\mathcal{O}(m^3 + Nm)$ results. Hence the computation is of complexity $\mathcal{O}(m^3 N)$ instead of $\mathcal{O}(N^2)$ for the original matrix, meaning that it is linear in the number of data points N, provided the sample size m is fixed. The Nyström approximation is exact if the number of samples m is chosen according to the rank of $\mathbf{K}$, meaning the m chosen points are linearly independent. Bounds on the quality of the approximation for a chosen subsample can be derived as presented for example in [98].

Originally the Nyström method was presented for positive semidefinite Gram matrices, but a direct transfer is possible for dissimilarity data [34]. A symmetric dissimilarity matrix $\mathbf{D}$ is a normal matrix and can be diagonalized $\mathbf{D} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$ according to the spectral theorem with an unitary matrix $\mathbf{U}$ whose column vectors are the orthonormal eigenvectors of $\mathbf{D}$ and a diagonal matrix $\mathbf{\Lambda}$ with the eigenvalues of $\mathbf{D}$. Therefore the dissimilarity matrix can be seen as an operator $d(\xi_j, \xi_l) = \sum_{i=1}^{\infty} \lambda_i \phi_i(\xi_j) \phi_i(\xi_l)$ where $\lambda_i \in \mathbb{R}$ correspond to the diagonal elements of $\mathbf{\Lambda}$ and ϕ_i denote the eigenfunctions. Thus, the same treatment as before is possible, the only difference to an expansion of a kernel being that the eigenvalues are allowed to be negative for non-Euclidean distances. All further mathematical manipulations can be applied in the same way.

Using the approximation in Equation 5.8 for the distance matrix allows to approximate the Equation 3.17 in the way

$$\begin{aligned}\|\Phi(\xi_i) - w_j\|^2 \approx &\left(\mathbf{e}_i^t \mathbf{D}_{N,m}\right) \cdot \mathbf{D}_{m,m}^{-1} \cdot \left(\mathbf{D}_{m,N} \gamma_j\right) \\ &- \frac{1}{2} \cdot \left(\gamma_j^T \mathbf{D}_{N,m}\right) \cdot \mathbf{D}_{m,m}^{-1} \cdot \left(\mathbf{D}_{m,N} \gamma_j\right)\end{aligned} \tag{5.11}$$

which is again linear in the number of data points N, assuming a fixed approximation m. Once more, the approximation is exact if m suits the rank of the matrix.

5.3 Quick check

As we will see in experiments, the Nyström approximation yields good results in a variety of benchmark problems, but it leads to a severe deterioration of the classification accuracy in others. Further, one crucial question is which number of landmarks m is sufficient for a good approximation. Therefore, it would be advisable to devise a test which can check the suitability of the Nyström approximation with a certain number of landmarks m prior to training without the necessity to compute the full matrix $\tilde{\mathbf{K}}$ on the one hand, or to train the classifier, on the other hand. Since LVQ relies on a winner-takes-all scheme, the exact values of the considered proximities are less important than their ordering. We take this observation as a starting point for a quick check whether a considered Nyström approximation is suitable. The basic idea is to test exemplarily regarding few samples whether the observed approximation preserves the ordering of proximities.

Thus, we first consider ways to computationally evaluate order preservation. Assume two vectors of identical dimensionality are given. Then, their principled shape is characterized by their correlation. There exist positive and negative correlations, though large correlation coefficients are no guarantee that there exists a causal relation between these variables. The Spearman rank correlation goes a step further by abstracting from the exact values. It is comparing rank orderings, which means that the entries of both vectors are examined for simultaneously monotonic relationships. This non-parametric correlation of statistical dependence yields numerical values between -1, suggesting a negative correlation of the induced ordering, and $+1$, revealing a positive correlation or the ordering, where values around zero indicate no association between the vectors.

Formally assume two vectors $x, y \in \mathbb{R}^n$ are given, and their ranks are denoted as $\mathrm{rnk}(c_k) = |\{c_i < c_k, i = 1 \ldots n\}|$, the measure is received by computing the normalized squared Euclidean distance

$$\rho(x, y) = 1 - \frac{6}{n \cdot (n^2 - 1)} \cdot \sum_{k=1}^{n} (\mathrm{rnk}(x_k) - \mathrm{rnk}(y_k))^2 \tag{5.12}$$

The computational complexity of this formula is $\mathcal{O}(n \cdot \log(n))$ due to the sorting. Spearman rank correlation can be interpreted in terms of the amount of mutual information between two variables and has many applications, for example the analysis of gene expression data [63]. It converts a non-linear ordinal data space into a rank based Euclidean space, where a compression of outliers and an enlargement of close values takes place due to the substitution of vector entries by their ranks. In the absence of ties this results in a uniform

distribution with unit spacing and invariant statistical moments of the data vectors.

For the Nyström approximation, we would like to check whether the approximation $\tilde{\mathbf{K}}$ preserves the ordering of distances in the original matrix $\mathbf{K}$. Therefore a natural measure for the suitability of the approximation is the Spearman correlation of exemplary rows of these matrices, meaning the average over the Spearman correlations of all rows i within a sample of indices I

$$\frac{1}{|I|} \cdot \sum_{i \in I} \rho_{\text{original}} := \frac{1}{|I|} \cdot \sum_{i \in I} \rho \left(\left[\mathbf{K} \right]_i, \left[\tilde{\mathbf{K}} \right]_i \right) \tag{5.13}$$

This indicates whether the ordering induced by the approximation would be consistent with the original ordering of the respective closest data points for every row. Since the Spearman correlation and the Nyström approximation values have to be computed for $|I|$ rows, the complexity of this computation is of order $\mathcal{O}\left(|I| \cdot N \log N + m^3 + |I| \cdot mN\right)$. It is possible to lower this complexity by referring to a random sample of the coefficients of the rows only of constant size, which leads to a scalable complexity depending on m, $|I|$ and the size of this sample rather than the number of data N.

Albeit this measure constitutes a good indicator whether the Nyström approximation is successful, it is often not suitable in practice since the full kernel matrix $\mathbf{K}$ is not available. One remedy is to substitute the measure in Equation 5.13 by a statistical estimation which relies on the variance of two approximations rather than the original matrix. The statistical argument underlying this rational relies on the fact that the Nyström approximation is derived as a sampling algorithm, and it converges towards the true values in the limit see for example [32] for a proof of this fact for general proximities.

Hence we can consider the following quantity

$$\begin{aligned} \frac{1}{|I|} \cdot \sum_{i \in I} \rho_{\text{pairwise}} := \frac{1}{|I|} \cdot \sum_{i \in I} \rho \Big(& \left[\mathbf{K}_{N,J_1} \mathbf{K}_{J_1,J_1}^{-1} \mathbf{K}_{J_1,N} \right]_i, \\ & \left[\mathbf{K}_{N,J_2} \mathbf{K}_{J_2,J_2}^{-1} \mathbf{K}_{J_2,N} \right]_i \Big) \end{aligned} \tag{5.14}$$

where different subsets $J_1, J_2 \subset \{1, \ldots, N\}$ of the data points of the same size m are taken for the Nyström approximation. This value computes the Spearman correlation which results from the Nyström approximation taking different samples of the reported size, and it is advisable to not only consider two such samples, but more choices for example stemming from a cross-validation. The computational complexity of this technique is the same as before, including the possibility of a further speedup by referring to a fixed

number of entries in the rows only. However, this method does not require the existence of the original Gram matrix. Therefore, we will use this measure in the following experiments as an efficient method, which gives some indication of the expected performance of the Nyström approximation prior to training.

5.4 Experiments

We compare kernel RSLVQ as one of the best performing variants of LVQ methods for proximity data and the quality for the Nyström approximation for different sizes of the approximation matrix maintaining the same experiment setup as described in Section 4.6 considering valid kernels by applying clip or flip preprocessing as introduced in Section 4.5.

The results for the Nyström approximation based on a subsample of 10% or 25%, respectively, are reported in Table 5.1. It preserves the excellent performance of kernel RSLVQ in four of the cases enabling a linear technique with excellent performance in these settings. For the two cases Amazon47 and Patrol, the Nyström approximation yields a degradation by more than 100%. As can be seen from the eigenvalue spectra as shown in Figure 4.2, a good performance of the Nyström approximation is directly correlated with the intrinsic dimensionality of the data set as measured by the number of eigenvalues with significant contribution. The two data sets Amazon47 and Patrol display eigenvalue profiles where a large number of values is very different from 0. Since the Nyström approximation is exact if the sampled points match the intrinsic rank of the given data, and it looses information of the remaining span, otherwise, it can be expected that the Nyström approximation fails in these two cases, which it does.

We can see that an intrinsically low dimensional matrix correlates to a good approximation of the Nyström approximation of the Gram matrix. Additionally, we report results of the Spearman correlation for the rows of the Nyström approximation and the original data matrix ρ_{original} and for pairwise different Nyström samples ρ_{pairwise}, as proposed in the Section 5.3. Interestingly, ρ_{original} displays particularly low values, smaller than .4, for the two data sets Amazon47 and Patrol. The result of ρ_{pairwise}, averaged over 10 different approximation sets, yields very low values, < 0.1, for the valid Gram matrices if and only if the Nyström approximation fails, otherwise resulting in values of at least 0.5. Hence, by sampling only a constant number of rows and computing their correlation this way, we obtain an efficient method to estimate whether the Nyström approximation can be successful prior to training.

	kernel RSLVQ		Nyström				ρ_{original}				ρ_{pairwise}			
			10%		25%		10%		25%		10%		25%	
Amazon47														
clip	15.37	(0.41)	**64.15**	(0.33)	77.93	(0.51)	0.22	(0.15)	0.35	(0.06)	0.02	(0.01)	0.05	(0.04)
flip	16.34	(0.42)	**65.73**	(0.30)	76.71	(0.62)	0.22	(0.19)	0.36	(0.08)	0.02	(0.02)	0.05	(0.03)
Aural Sonar														
clip	11.25	(0.39)	15.00	(0.63)	**13.00**	(0.43)	0.61	(0.25)	0.81	(0.02)	0.54	(0.53)	0.74	(0.03)
flip	11.75	(0.35)	16.25	(0.84)	**14.50**	(0.55)	0.56	(0.29)	0.75	(0.02)	0.53	(0.82)	0.71	(0.06)
FaceRec														
clip	3.84	(0.02)	**3.47**	(0.02)	3.49	(0.01)	1.00	(0.00)	1.00	(0.00)	1.00	(0.00)	1.00	(0.00)
flip	3.60	(0.02)	3.52	(0.02)	**3.47**	(0.01)	1.00	(0.00)	1.00	(0.00)	1.00	(0.00)	1.00	(0.00)
Patrol														
clip	17.40	(0.29)	47.50	(0.78)	**34.79**	(0.60)	0.18	(0.08)	0.31	(0.17)	0.04	(0.01)	0.08	(0.01)
flip	19.48	(0.34)	45.94	(0.66)	**35.10**	(0.39)	0.21	(0.02)	0.38	(0.02)	0.04	(0.01)	0.08	(0.02)
Protein														
clip	4.88	(0.17)	12.21	(0.36)	**7.44**	(0.23)	0.88	(0.12)	0.95	(0.01)	0.88	(0.07)	0.93	(0.03)
flip	1.40	(0.05)	8.02	(0.38)	**3.95**	(0.14)	0.88	(0.13)	0.94	(0.02)	0.87	(0.14)	0.92	(0.11)
Voting														
clip	5.34	(0.04)	**5.17**	(0.03)	5.69	(0.03)	1.00	(0.00)	1.00	(0.00)	1.00	(0.00)	1.00	(0.00)
flip	5.34	(0.03)	**5.34**	(0.04)	5.52	(0.03)	1.00	(0.00)	1.00	(0.00)	1.00	(0.00)	1.00	(0.00)

Table 5.1: Results of kernel RSLVQ compared to a Nyström approximation of the Gram matrix using 10% and 25% of the data are reported. Additionally Spearman correlation coefficients ρ_{original} and ρ_{pairwise} are given. Standard deviations are given in parenthesis. The best results of the Nyström approximations are shown in boldface.

5.5 Discussion

We have investigated the possibility to obtain efficient approximations of kernel RSLVQ by means of the Nyström approximation. This method aims at an advanced computational achievement of the technique, resulting in faster classification performance, thus addressing one of the most severe drawbacks of kernel RSLVQ.

We have shown that the excellent accuracy obtained by kernel RSLVQ can be preserved using the Nyström approximation, provided data have an intrinsically low dimensionality. The latter can efficiently be tested by referring to the correlation of different Nyström samples. Using this technique, we have taken a further step to bring kernel RSLVQ towards efficient methods with linear training time.

There exists another severe drawback of kernel or relational LVQ in comparison to its vectorial counterpart. Vectorial LVQ provides a direct interface for the applicant, who can directly inspect the prototypes in the same way as data. For proximity LVQ, this property is lost, since prototypes depend on all data points and are given only implicitly. This does not only have a crucial impact on the interpretability of the given models, but it also results in an increasing computational complexity and space complexity for the classification, which is linear instead of constant. In Chapter 6, we address this problem by means of sparse approximations of prototypes. In this case, prototypes are represented by one or few exemplars only, whereby the latter can be directly inspected by practitioners in the same way as data. At the same time, training benefits from the larger flexibility of a continuous adaptation space as provided by the full model.

Chapter 6

Interpretability

Chapter overview *Within this chapter, we address the fact that prototypes are no longer explicit within proximity extensions of LVQ but represented implicitly. We investigate different ways to substitute these models by sparse representations of the prototypes. For this purpose, we introduce different principles how to do so, and extensively evaluate the performance of these methods on benchmarks as well as one further illustrative data set with known semantical background. We do not only address the accuracy for model evaluation, but also investigate quantitative measures for the sparsity and representativity of the found solutions. Parts of this chapter rely on the publications [47, 48, 49, 50, 52, 53].*

As already stated, one of the benefits of vectorial LVQ techniques consists in the fact that solutions are represented by a small number of representative prototypes which constitute members of the input space. In consequence, prototypes can be inspected in the same way as data in the vectorial setting. Since the dimensionality of points ξ is typically high, this inspection is often problem dependent. Images, for example, lend itself to a direct visualization, oscillations can be addressed via sonification, spectra can be inspected as a graph which displays frequency versus intensity. Moreover, a low-dimensional projection of the data and prototypes by means of a non-linear dimensionality reduction technique offers the possibility to inspect the overall shape of the data set and classifier independent of the application domain.

Prototypes in relational or kernel settings correspond to positions in pseudo-Euclidean space which are representative for the classes if measured according to the given similarity/dissimilarity measure. Thus, prototype inspection faces two problems. On the one hand the pseudo-Euclidean embedding is usually only implicit, on the other hand it is not clear whether dimensions in this embedding carry any semantic information. Thus, al-

beit prototypes are represented as linear combinations of data also in the pseudo-Euclidean setting, it is not clear whether these linear combinations correspond to a semantic meaning.

One approach which we will follow is to approximate a prototype by one or several exemplars, meaning members of the data set, which are close by [52]. Thereby, the approximation can be improved if sparsity constraints for the prototypes are integrated while training. This way, every prototype is represented by a small number of exemplars which can be inspected in the same way as data. Another possibility is to visualize data and prototypes using some non-linear dimensionality reduction technique. We will very shortly address visualizations in one illustrative example. This enables an investigation of the overall shape of the classifier just as in the standard vectorial setting. However, since visualizations are usually non-linear, its semantic meaning is often not clear since the dimensions in the plane do not carry semantic information. Therefore, we will focus mostly on the first approach. Naturally, both techniques, a representation of prototypes by few exemplars as well as a projection to low dimensions incorporate errors depending on the dimensionality of the pseudo-Euclidean space and its deviation from the Euclidean norm.

As discussed for example in the seminal work [72], the principle of sparsity constitutes a common paradigm in nature-inspired learning. Interestingly, apart from an improved complexity, sparsity can often serve as a catalyzer for the extraction of semantically meaningful entities from data. In our case, the basic entities are represented by the data itself, and the task is to approximate given prototypes by sparse counterparts, thereby minimizing the loss of accuracy. It is well known that the problem of finding smallest subsets of coefficients such that a set of linear equations can still be fulfilled constitutes an NP-hard problem, being directly related to NP-complete subset selection. Because of this fact, approximation techniques have to be considered, one popular approach being for example a l_1-relaxation of the problem [24] such as used in LASSO.

Instead of the full coefficient vectors, few exemplars which represent the prototypes can be directly inspected by practitioners in the same way as data by applying sparse approximations to kernel RSLVQ. The validity of this assumption, however, strongly depends on the way in which prototypes are substituted by sparse approximations. We investigate different possibilities to approximate a prototype by a sparse counterpart during or after training relying on different heuristics or approximation algorithms, respectively, in particular sparsity constraints while training [72], geometric approaches, orthogonal matching pursuit [15], and core techniques for the minimum enclosing ball problem [4]. We discuss the behavior of these methods in several

benchmark problems as introduced in Section 4.6 as concerns quality, sparsity, and interpretability, and we propose different measures how to quantitatively evaluate the performance of the approaches.

6.1 Approximation of the prototypes

Kernel RSLVQ as well as other LVQ variants for proximities as introduced in Chapter 3 yields prototypes which are implicitly represented as linear combinations of data points

$$w_j = \sum_m \gamma_{jm} \Phi(\xi_m) \tag{6.1}$$

as discussed in Section 3.2, where Φ refers to the kernel embedding of the data or, more generally, the underlying pseudo-Euclidean embedding. Since γ_{jm} can be arbitrary, sparseness of the prototype is not given, but its location usually depends on all data ξ_m.

Here we propose different ways to arrive at sparse prototype representations, meaning counterparts where γ_{jm} equals zero for most coefficients m. If only few coefficients γ_{jm} are non-vanishing, a direct inspection of the corresponding exemplars ξ_m allows practitioners to judge the characteristics of the correlated prototype and its receptive field by a direct inspection of the exemplars. Formally, a sparse representation of a given prototype w_j refers to a set of one or more prototypes w_j^i of the form

$$\tilde{w}_j^i = \sum_m \tilde{\gamma}_{jm}^i \Phi(\xi_m) \tag{6.2}$$

such that

- the size of this set is small, ideally, only one approximating prototype $\tilde{w}_j^1$ for w_j is necessary,
- these vectors are sparse, meaning $\left|\tilde{\gamma}_j^i\right|_0$ is as small as possible,
- the set approximates w_j in the sense that the receptive field of w_j as compared to the union of the receptive fields of its approximations $\tilde{w}_j^i$ contains approximately the same set of data points.

One possibility to ensure that the last condition holds is to enforce $\tilde{w}_j^i \approx w_j$ as measured by the distance in the feature space.

This formulation includes as a subproblem the task to find a vector $\tilde{w}_j = \sum \tilde{\gamma}_{jm} \Phi(\xi_m) = w_j$ such that $|\tilde{\gamma}_j|_0$ is minimum, if possible. This problem is NP-hard, such that we have to rely on approximations [69]. In the following, we introduce a variety of possible schemes.

6.2 Sparse training

A classical way to enforce sparsity constraints consists in the addition of a regularization term while training. This technique has been proposed, among others, in the pioneering work of Olshausen and Field based on a probabilistic model, for example [72]. Thus, we substitute the cost function L in Equation 3.5 by the sum

$$L - \text{Const} \cdot S(\gamma) \tag{6.3}$$

where $S(\gamma)$ constitutes a constraint which emphasizes sparse solutions such as

$$S(\gamma) = \sum_{ji} \left|\gamma_j^i\right|_1 \tag{6.4}$$

as approximation of the 0-norm and $\text{Const} > 0$ is a priorly chosen constant which weighs the two objectives of the combination. Optimization of these costs can be done by a subgradient method [91], which reduces to a standard gradient ascent for most of the regions. For $\gamma_j^i = 0$, the subgradient is set to the constant 0 to emphasize sparse solutions. Note that, this way, sparse prototypes are chosen already while training, which has usually the effect that the final location of the resulting prototypes can be different from the prototypes obtained by standard kernel RSLVQ without sparsity constraint. This technique is the only one among the ones proposed here, which changes the shape of the prototypes already while training. All other techniques start from a trained set of prototypes and try to exchange the linear combinations by a sparse variant. Therefore, we refer to this method as **sparse training** in the following.

6.3 Simple heuristic approximations of the prototypes

As a simple alternative, we propose two intuitive heuristic approximation schemes which substitute trained prototypes by sparse approximations.

Geometric heuristic

The first approach, relies on the geometry of LVQ. For kernels, the distance of prototypes to points in their receptive field is changed to a small amount only, if we approximate the prototype by the closest exemplar. As a generalization

thereof, in particular to meet settings where the feature space is not densely populated, we can use the K_{approx} closest exemplars for some fixed number K_{approx}. Note that this method, which we refer to as K_{approx}**-approximation** in the following, represents a prototype by a set of K_{approx} new sparse ones with l_0-norm equal to one.

Numerical heuristic

As an alternative, we can consider the coefficient vector γ_j and take the size of the coefficients as an indicator for the importance of the underlying exemplar. For the K_{hull}**-convex hull**, we select the K_{hull} largest coefficients γ_{jm} and we delete all but these coefficients in the vector γ_j. This is then normalized to 1 by applying $\sum_m \tilde{\gamma}_{jm} = 1$. Thereby, we neglect the upper index since only one prototype is used for the approximation.

6.4 Approximate representations of the prototypes

As an alternative to these simple heuristic approximation schemes, we can use more fundamental optimization techniques which try to represent a given prototype as accurately as possible regarding some explicit mathematical objective, based on which an optimization can be performed.

Numeric approximation

We can formalize the task to approximate a given prototype as the mathematical objective to approximate a prototype by a sparse linear combination of data such that the residual error of this approximation and the original prototype is as small as possible. This corresponds to the following mathematical problem, where again, we use only one prototype for the approximation and, in consequence, neglect the corresponding index

$$\begin{array}{ll} \min & |\tilde{\gamma}_j|_0 \\ \text{s.t.} & |\sum_m \tilde{\gamma}_{jm} \Phi\left(\xi_m\right) - w_j| \leq \epsilon \end{array} \tag{6.5}$$

for a given $\epsilon > 0$. It is well-known that this problem is NP-hard [69]. Hence a variety of approximate solution strategies exist in the literature. Here, we rely on a popular and very efficient approximation offered by **orthogonal matching pursuit** (OMP) [15]. Given an acceptable error $\epsilon > 0$ of the approximation, a greedy approach is taken. The algorithm iteratively determines the most relevant direction and the optimum coefficient for this axis

Algorithm 6.1 kernelized orthogonal matching pursuit

$I := \emptyset$
$\tilde{\gamma}_j := 0$
while $(\gamma_j - \tilde{\gamma}_j)^t K (\gamma_j - \tilde{\gamma}_j) > \epsilon^2$ **do**
    $r := \gamma_j - \tilde{\gamma}_j$
    $l_0 := \text{argmax}_l |[Kr]_l|$
    $I := I \cup \{l_0\}$
    $\tilde{\gamma}_{jm} := (K_{II})^{-1} K_{Im}$ with $K_{II} :=$ restriction of K to index set I
end while
return $\tilde{\gamma}_j$

to minimize the remaining residual error. The algorithm can be easily kernelized, such that it can directly be used in our setting, where we assume a normalized kernel $k_{mm} = 1$ corresponding to a fixed length $\Phi(\xi_i)$ or alternatively, the normalization could be added to the greedy selection step. The pseudocode is given in Algorithm 6.1.

Geometric approximation

An alternative mathematical approximation can be derived based on a geometric view. We assume that a prototype is located at a central position of its receptive field, since it represents the center of the corresponding Gaussian mode. We denote the latter receptive field of w_j by R_j. Under the assumption of spherical classes, we can characterize a prototype as the center of a ball which encloses all data assigned to it. To achieve uniqueness, we choose the smallest ball. The following geometric optimization problem referred to as **minimum enclosing ball (MEB)** results

$$\begin{array}{ll} \min_{R^2,C} & R^2 \\ \text{s.t.} & \|C - \Phi(\xi_i)\|^2 \leq R^2, \, \forall \xi_i \in R_j \end{array} \tag{6.6}$$

Here, C is the center and R the radius of the MEB. We expect that $C \approx w_j$. The key observation of a sparse approximation technique starting from this characterization consists in the fact that the MEB can be approximately solved with a sparse vector C where the degree of sparsity is independent of the size of R_j. Further, a linear time approximation algorithm is available, see [4]. We shortly outline the idea of this sparse approximation, typically referred to as core approximation.

First, the dual problem of MEB can be phrased as follows

$$\begin{array}{ll} \min_{\alpha_i \geq 0} & \sum_{ij} \alpha_i \alpha_j k_{ij} - \sum_i \alpha_i k_{ii}^2 \\ \text{s.t.} & \sum_i \alpha_i = 1 \end{array} \tag{6.7}$$

Algorithm 6.2 minimum enclosing ball

```
1 S := {ξ_i, ξ_m} for a pair of largest distance ‖Φ(ξ_i) − Φ(ξ_m)‖² in R_j
2 repeat
3     solve MEB(S) → C, R
4     if exists ξ_l ∈ R_j where ‖Φ(ξ_l) − C‖² > R²(1 + ε)² then
5         S := S ∪ {ξ_l}
6     end if
7 until all ξ_l are covered by the R(1 + ε) ball in the feature space
8 return w̃_j := C
```

Any solution of the dual problem gives rise to a primal solution in terms of $C = \sum \alpha_i \Phi(\xi_i)$. This dual is a convex problem with a unique solution, but worst case effort $\mathcal{O}\left(|R_j|^3\right)$ and no bound on the sparsity of the resulting solution. Therefore, this problem is not solved for the entire receptive field R_j, rather, starting from the empty set, a core set of points is built for which this dual problem is solved. A surprising fact proved for example in [4] is that a fixed finite number of such points is sufficient to form a core set which represents the entirety of R_j. The size is thereby independent of the size of R_j and the dimensionality of points.

This iterative algorithm to determine a core set uses the dual MEB as a subroutine. It terminates with a core set of limited size as a subset of R_j, for which the dual variables α_i induce a center of the MEB for the entirety of R_j. We refer to this sparse center as $\tilde{w}_j$. The pseudocode is given in Algorithm 6.2.

It has been proved in [4] that the number of loops of this algorithm is limited by a constant of order $\mathcal{O}(1/\epsilon^2)$ independent of R_j. In each loop, the dual MEB problem is solved for a small subset S of constant size, such that each loop has linear complexity only. An approximation of w_j as center of an approximate MEB is given by the dual variables of the found core set $C_j = \sum_{i \in S} \alpha_i \Phi(\xi_i)$ hence a sparse approximation of w results by setting $\tilde{\gamma}_{ji}$ to α_i if and only if the coefficient i corresponds to a core point. We arrive at a sparse solution, whereby the quality of the approximation ϵ determines the resulting sparsity. Since data are used in the form of dot products only, all computations can be kernelized. Note that similar tricks have been used to speed up for example support vector machine training, see [92].

6.5 Characteristics of the techniques

Note that the proposed techniques differ in several characteristics, regarding

Method	control of sparsity	coefficients	location of exemplars
RSLVQ	no sparsity	convex	central
sparse training	soft Const	convex	often central
K_{approx}-approx.	fixed K_{approx}	set of exemplars	central
K_{hull}-convex hull	fixed K_{hull}	convex	not clear
OMP	soft ϵ	possibly negative	subject to variance
MEB	soft ϵ	convex	extremal

Table 6.1: Characteristics of different sparse approximations of prototype based models.

- their motivation being heuristics for the K_{approx}-approximation and the K_{hull}-convex hull are grounded in an explicit mathematical objective to approximate the prototypes,

- their application during or after training. Only the sparse approximation changes the representation of prototypes already while training,

- the way in which the degree of sparsity can be controlled,

- the way in which prototypes are represented in a sparse approximation. These correspond to one exemplar for a heuristic approximation using $K = 1$, a set of exemplars for the K_{approx}-approximation, or a sparsely populated element of the kernel space for all other techniques. In consequence, classification takes place by computing the distance to the new exemplar, or the minimum distance to all exemplars in the set representing a prototype in case of the K_{approx}-approximation,

- the sign and size of the coefficients. For RSLVQ, the coefficients are convex to increase interpretability, and we would like to maintain this fact also for the approximations. While OMP restricts to convex combinations, MEB does not allow this in an easy way,

- the location of the non-vanishing index set, which can be central as for the K_{approx}-approximation, induced by dimensionality characteristics like OMP, at boundaries as for MEB, which focuses on extremal points.

We summarize the characteristics of the methods in Table 6.1 and examine them on benchmark data sets in Section 6.6.

A remark on direct exemplar based approaches

The question occurs whether it is possible to directly learn a sparse prototype model instead of a posteriori approximation only, when considering sparse prototype approximations. Techniques which represent solutions in terms of prototypical exemplars only, meaning prototypes $\vec{w}_j = \xi_i$ which equal exactly a given data point, have been proposed in prototype-based research under the umbrella of median techniques, see for example [20] and references therein. Essentially, this corresponds to the case of a sparse model where the number of exemplars used to represent prototypes is reduced to $K = 1$. Recently, a median approach for supervised LVQ has also been proposed [71]. Essentially, median techniques try to devise efficient methods which optimize the given cost function but restricting prototypes to the discrete space formed by the given data.

One problem of such median approaches consists in the fact that their optimization is essentially discrete. Hence optimization is either costly, when relying on meta-heuristics for cost function optimization such as simulated annealing or similar, or optimization is prone to local optima due to the very restricted representation abilities in the discrete data space. This effect has been observed in unsupervised median prototype-based methods such as median neural gas in comparison to its continuous relational counterparts, such as relational neural gas, see [39]. Albeit median approaches of this form have quadratic costs only comparable to kernel methods, their performance is often inferior as compared to kernel or relational approaches.

One notable exception of this observation is offered by affinity propagation [28] which rephrases an exemplar based prototype-based clustering scheme in terms of a factor graph representing the data likelihood, for which efficient continuous optimization is possible using message passing algorithms. Hence this technique combines the efficiency of kernel approaches with a direct interpretability of the result by restricting prototypes to exemplars. Still, it is restricted to an unsupervised optimization of the quantization error, such that the obtained classification accuracy is inferior to supervised kernel LVQ approaches. The recent median LVQ variant proposed in [71] relies on an expectation-maximization scheme, which is also often prone to local optima.

6.6 Experiments

We compare kernel RSLVQ as one of the best performing variants of LVQ methods for proximity data and its sparse approximations maintain-

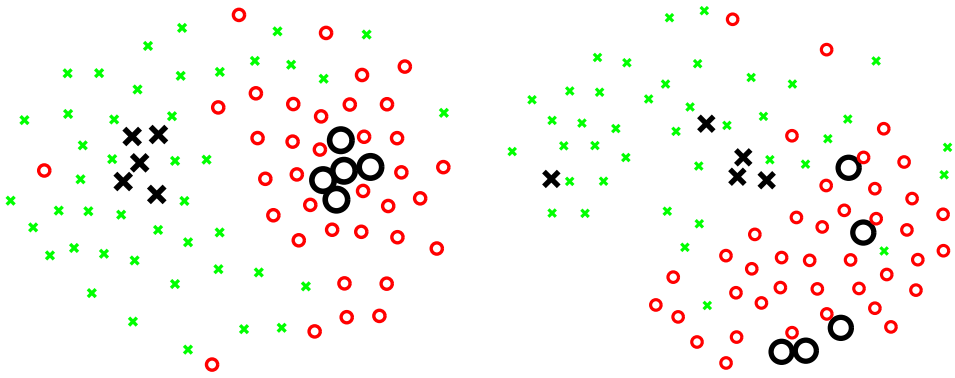


Figure 6.1: AuralSonar with spectrum flip visualized by t-stochastic neighbor embedding [93]. The left figure shows the results of sparse training and the right of OMP. In both settings, the location of the prototypes, not the corresponding exemplars, is shown. Obviously, very different prototype locations are obtained.

ing the same experiment setup as described in Section 4.6 considering valid kernels by applying clip or flip preprocessing as introduced in Section 4.5. Additionally the two illustrative datasets Artificial data and VBB Midi are investigated, which will be introduced later. Thereby, we particularly want to check whether characteristics of the data allow to infer which approximation is best suited for the given task.

We approximate the solutions of kernel RSLVQ by sparse approximations using the methods as specified above. Thereby, we set the sparsity to $K_{approx}, K_{hull} \in \{1, 10\}$. If training with sparsity constraint is used, an appropriate weighting parameter Const is determined by binary search such that a desired sparsity is obtained. The parameter Const can be very sensitive depending on the data, leading to non-trivial results in a small range only. For the approximations using OMP and MEB, the quality ϵ of the approximation is determined such that a sparsity in the range of 1 to 10 is obtained.

We demonstrate the effect of the different characteristics of the sparse approximations as introduced in Table 6.1 exemplarily in the two following figures. In Figure 6.1, the result of sparse training is compared to the result of OMP. Obviously, the location of the prototypes is very different which can be attributed to the fact that sparse training influences the prototype locations already while training.

In Figure 6.2, the location of the exemplars underlying the MEB approximation versus the K_{approx}-approximation is shown in a benchmark. The K_{approx}-approximation tends to locate the exemplars closer to the class cen-

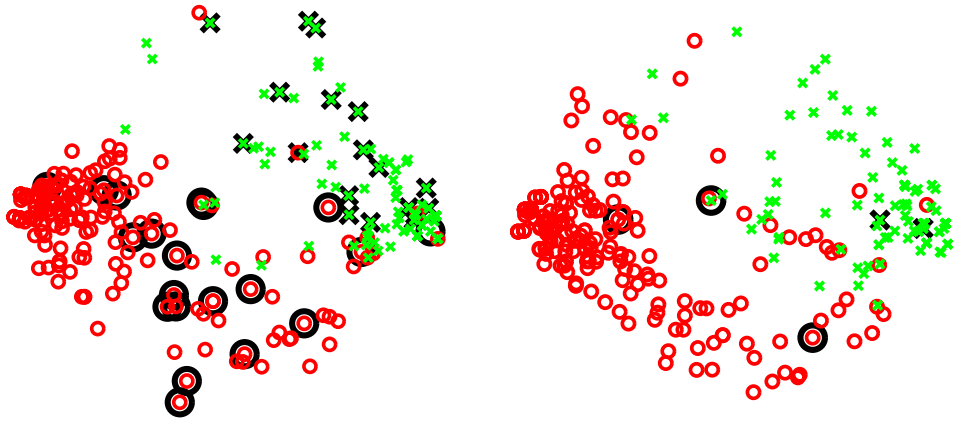


Figure 6.2: Voting with spectrum clip visualized by multidimensional scaling. The left figure shows the results of MEB and the right the results of the 1-approximation. In both cases, the exemplars corresponding to coefficients larger than zero are shown. Obviously, the 1-approximation puts exemplars close to the centers, while MEB also selects boundary positions due to its grounding in an MEB problem.

ters, while MEB also puts some of the exemplars on extremal positions.

Results as regards sparsity and accuracy

The classification accuracy is shown in Table 6.2. Interestingly, the obtained classification results when considering sparse approximations differ depending on the data set and the used technique. For the intrinsically low-dimensional data sets Protein, Voting and Face Rec, different sparse approximations give results comparable to full prototypes, while the situation seems more difficult for the other data sets. For Amazon47, none of the sparse approximations reaches the accuracy of the full model, which can be attributed to a high dimensionality of the data with few data points and a large number of classes. This is a situation where we would possibly expect that the full information of the data set is necessary to obtain a good classification accuracy. For Aural Sonar and Patrol, some sparse techniques yield results comparable to the full models.

It seems that there exists no universally suited method to enforce sparsity. Sparse approximation already while training yields best results in three of the cases. However, the choice of the parameter Const is crucial and a high degree of sparsity is not easy to achieve for this setting, as can be seen from the variance of the sparsity as reported in Table 6.3. In many cases a simple

	kernel RSLVQ	K_{approx}-approximation		K_{hull}-convex hull		OMP	MEB	sparse training
		K_{approx}=1	K_{approx}=10	K_{hull}=1	K_{hull}=10			
Amazon47								
clip	15.37	**32.26**	43.82	33.09	55.85	70.12	87.79	39.92
flip	16.34	**32.32**	46.06	34.18	54.51	68.66	88.54	43.18
Aural Sonar								
clip	11.25	25.75	14.50	58.50	23.25	15.00	13.50	**10.75**
flip	11.75	22.75	15.12	61.50	19.75	26.00	**14.75**	15.50
Face Rec								
clip	3.84	3.76	37.04	3.92	3.84	**3.65**	3.81	4.13
flip	3.60	**3.31**	37.00	4.21	3.60	3.60	3.62	4.07
Patrol								
clip	17.40	39.84	**19.90**	39.17	24.58	29.79	25.42	40.00
flip	19.48	38.91	**21.03**	40.16	25.52	33.33	24.17	41.56
Protein								
clip	4.88	18.49	26.94	36.28	27.44	52.09	14.59	**13.84**
flip	1.40	23.84	24.48	25.35	3.95	49.07	3.72	**2.21**
Voting								
clip	5.34	8.82	11.39	86.44	82.76	**5.34**	17.70	**5.34**
flip	5.34	7.99	9.91	86.95	82.53	**5.46**	17.18	5.80

Table 6.2: Results of kernel RSLVQ and diverse sparse approximations on the investigated benchmark data. The best results given as percentage misclassifications of the approximation methods are shown in boldface.

K_{approx}-approximation yields surprisingly good results, indicating that the location of the prototypes can often be well preserved by a simple substitution with its closest exemplar. Besides these observations, one can also detect two cases where the mathematical approximations OMP and MEB yield best results with respect to alternative posterior regularizations, whereby the degree of sparsity is easier to handle as compared to sparse training.

We exemplarily report the dependency of the approximation quality from the sparseness for the geometric methods and OMP in Figure 6.3. A more systematic comparison of the accuracy for different degrees of sparsity is there exemplarily shown. Since OMP does not allow to explicitly influence the sparsity, but the approximation quality only, these curves cannot be obtained for the full range displayed in the graphs. Clearly in all settings a simple geometric approach approximates the accuracy obtained by OMP and it is even better in a fraction of the graphs, and it shows that it varies depending on the data for which sparsity and for which techniques best results can be obtained. This can be attributed to the quite diverse geometric setting and learning scenario. However, since posterior geometric approximation techniques are rather fast, it is no problem to simply test different degrees of sparsity for both methods and take the best one, afterwards.

A sparse representation of the classifier in terms of few exemplars of the data set opens the way towards fast classification models and, in particular, interpretable models, provided a single data point can be inspected by applicants in a natural way. Note that several data sets allow classification schemes which rely on only one exemplar per class, meaning an efficient inspection of these representing data is particularly efficient.

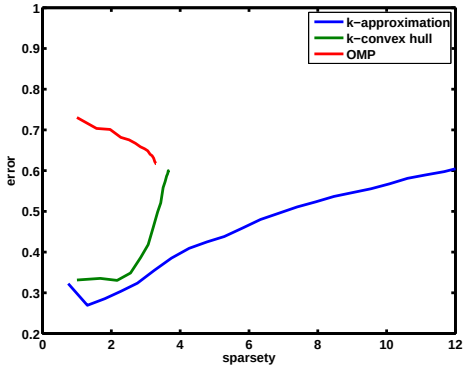
Results as regards representativity

The problem occurs how we can evaluate the representativity of the obtained prototypes for the given data. Eventually, this question has to be answered by practitioners in the field who inspect the found exemplars. Naturally, the degree of sparsity as reported in Table 6.3 is a first indicator about the complexity of the resulting model. However, a sparse model does not necessarily correlate with a good classification accuracy, nor the representativity of the found exemplars. Here, we investigate two principled ways to access the representativity of the models as a first try to quantitatively measure in how far models could be seen as interpretable.

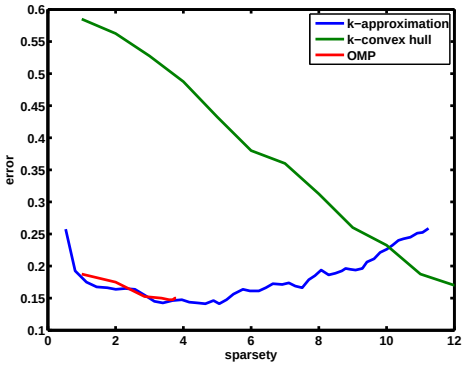
As a first measure which takes supervised labeling into account, we evaluate Rissanen's minimum description length as introduced in [37]. The minimum description length estimates the number of information it takes to represent the prototypes on the one hand and the errors induced by the pro-

	kernel RSLVQ	K_{approx}-approximation		K_{hull}-convex hull		OMP	MEB	sparse training
		K_{approx}=1	K_{approx}=10	K_{hull}=1	K_{hull}=10			
Amazon47								
clip	3.67	**0.75**	5.28	1.00	3.51	1.96	1.61	1.00
flip	3.67	**0.75**	5.31	1.00	3.51	1.95	1.60	1.00
Aural Sonar								
clip	40.00	**0.53**	3.15	1.00	10.00	3.79	5.30	12.75
flip	40.00	**0.47**	3.07	1.00	10.00	1.28	5.72	12.73
Face Rec								
clip	5.52	**1.00**	10.00	**1.00**	5.49	4.37	2.51	**1.00**
flip	5.52	**1.00**	10.00	**1.00**	5.49	4.22	2.58	**1.00**
Patrol								
clip	24.12	**0.68**	4.85	1.00	9.95	6.66	6.93	6.71
flip	24.12	**0.68**	4.43	1.00	9.95	3.55	6.98	6.69
Protein								
clip	42.50	**0.47**	3.25	1.00	10.00	1.84	4.89	13.37
flip	42.50	**0.43**	2.75	1.00	10.00	8.43	4.97	13.52
Voting								
clip	174.00	**0.29**	2.42	1.00	10.00	11.71	2.16	68.68
flip	174.00	**0.30**	2.31	1.00	10.00	8.82	1.99	59.92

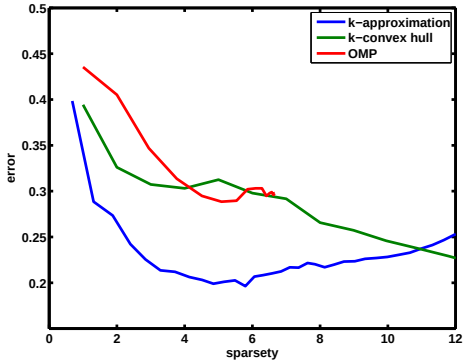
Table 6.3: Sparsity as the number of non-negative coefficients per prototype and label of kernel RSLVQ and diverse sparse approximations on the investigated benchmark data. Due to exemplars becoming identical, a sparsity smaller than 1 is possible.



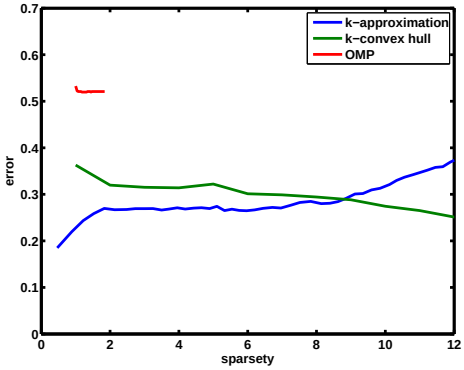
Amazon47 clip



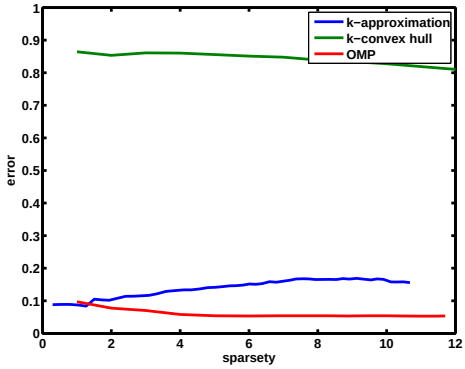
AuralSonar clip



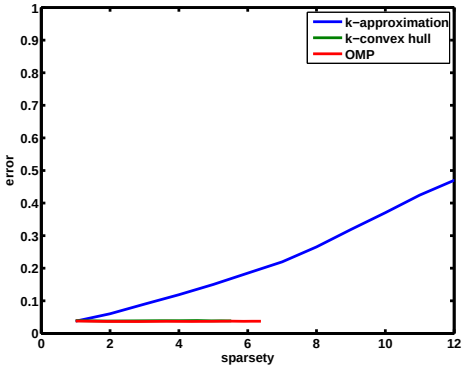
Patrol clip



Protein clip



Voting clip



FaceRec clip

Figure 6.3: For exemplary data sets, the obtained accuracy versus the degree of sparsity is depicted for the three techniques OMP, the convex hull, and the approximation by the nearest neighbors. For OMP only a small range of sparsity can be covered by reasonable choices of the control parameter.

totypes on the data on the other hand. The resulting quantity is depicted in Table 6.4 for the different sparse approximations. In all cases sparsity clearly yields a more compact representation of the available information as shown by the results reported in Table 6.4. Further, this measure highlights that simple techniques such as the K_{approx}-approximation seem a good compromise of accuracy and sparsity of the models.

As an unsupervised evaluation measure, we evaluate the entropy of the probability distribution which assigns data to prototypes. To account for different numbers of prototypes, a normalization by its logarithm takes place. Results are depicted in Table 6.5. The intuition is that a small entropy allows for clearly separated clusters, meaning representative exemplars, while a large entropy is an indicator for a more uniform distribution. Naturally, the result depends on the cluster structure of the underlying data, indicating for example that Voting does not seem to be easily separable into classes with gaps in between the classes. But also within data sets, differences of the different techniques can be found, indicating that the K_{approx}-approximation for $K_{approx} = 1$, for example, surprisingly is not able to separate the clusters as well as alternatives.

Figure 6.4 shows the approximations for extremal values of the entropy in an example data set. The smallest entropy is found in the K_{approx}-approximation setting, whereas most information can be found with K_{hull}-convex hull. Since data points at the border of the data set carry the most information about the location of the whole class it is not surprising, that these points get a larger value in the linear combination and give indeed most information about the data set, since they define the borders well. On the other hand the approximated location of the prototypes give more interpretable results, but can not specify the borders as well, ending in a lower entropy overall.

Two illustrative examples

The examples as introduced above allow already some insight into the behavior of the techniques, indicating, that

- it is not always possible to find sparse solutions of the same quality in particular when data dimensionality is large, but it is possible in many cases,

- for sparse approximations a simple K-nearest neighbor heuristic seems as appropriate as more fundamental approaches,

	kernel RSLVQ	K_{approx}-approximation		K_{hull}-convex hull		OMP	MEB	sparse training
		K_{approx}=1	K_{approx}=10	K_{hull}=1	K_{hull}=10			
Amazon47								
clip	151.82	**42.17**	43.39	43.59	44.44	246.80	367.90	252.33
flip	147.47	**39.49**	43.26	42.66	45.69	416.98	389.68	253.24
Aural Sonar								
clip	23.30	**4.74**	5.35	15.20	13.84	24.63	24.42	18.98
flip	21.94	5.21	**4.52**	12.60	13.42	31.31	23.08	16.87
Face Rec								
clip	2561.53	511.81	516.05	**499.01**	502.67	2531.63	2443.87	2484.54
flip	2561.53	511.81	516.05	**499.00**	502.68	2527.69	2443.86	2486.24
Patrol								
clip	235.12	35.65	**33.41**	53.63	56.99	274.79	226.68	174.57
flip	232.20	45.57	**36.71**	56.67	53.40	268.95	229.75	172.31
Protein								
clip	74.59	**14.83**	16.86	23.25	33.41	208.16	75.35	60.40
flip	51.42	20.34	20.41	22.91	**18.12**	339.72	49.56	38.34
Voting								
clip	190.86	**12.25**	12.38	200.83	199.01	75.94	174.90	103.37
flip	190.89	**15.84**	18.32	181.60	136.44	72.86	183.62	103.16

Table 6.4: Rissanen's minimum description length of kernel RSLVQ and diverse sparse approximations on the investigated benchmark data.

	kernel RSLVQ	K_{approx}-approximation		K_{hull}-convex hull		OMP	MEB	sparse training
		K_{approx}=1	K_{approx}=10	K_{hull}=1	K_{hull}=10			
Amazon47								
clip	3.18	3.99	**0.81**	4.37	3.16	3.25	3.93	3.98
flip	2.90	3.66	**0.74**	4.23	2.90	3.09	3.71	3.86
Aural Sonar								
clip	3.43	6.03	**1.41**	1.97	2.85	2.49	2.45	2.30
flip	1.10	2.23	**0.43**	1.88	0.82	0.73	0.76	0.73
Face Rec								
clip	231.87	232.25	**54.76**	232.21	232.25	232.25	232.09	231.83
flip	231.87	232.24	**54.76**	232.20	232.24	232.24	232.18	231.82
Patrol								
clip	3.31	4.81	**0.90**	3.16	2.93	2.68	2.36	2.28
flip	2.48	3.62	**0.72**	3.04	2.17	2.30	1.67	1.95
Protein								
clip	8.05	13.53	3.20	3.22	6.28	**1.94**	5.78	7.08
flip	6.58	11.36	**2.98**	3.14	5.39	4.71	4.80	5.47
Voting								
clip	89.86	76.23	56.71	**50.06**	77.84	80.68	72.14	75.16
flip	88.40	82.74	57.72	**51.37**	77.22	84.08	71.23	71.71

Table 6.5: Entropy of kernel RSLVQ and diverse sparse approximations on the investigated benchmark data.

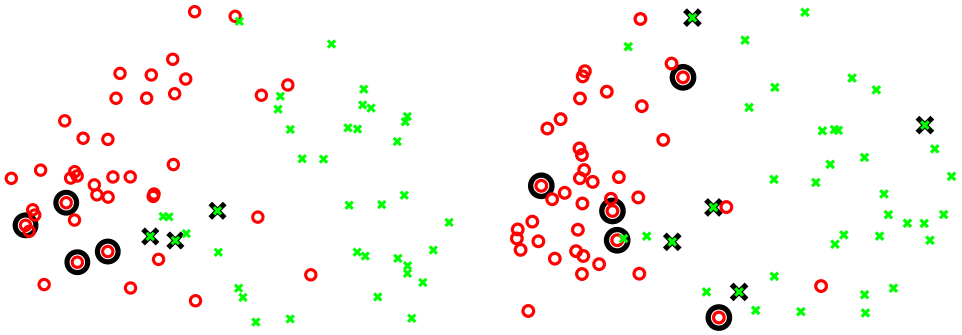


Figure 6.4: AuralSonar with spectrum clip visualized by multidimensional scaling. The left figure shows the results of 1-approximation and the right of 1-convex hull.

- the approximation methods differ in the final location of the exemplars, focusing partially on boundary points rather than central representatives,
- these effects are partially mirrored in measures such as the minimum description length or the entropy.

However, the experiments are in some way preliminary since the involved data are only implicitly given by their pairwise dissimilarities only. A direct inspection of the underlying data and its interpretability is problematic. Because of this fact we investigate two further data sets which can directly be inspected, in particular an artificial two dimensional Euclidean set, and a data set stemming from a transportation system:

- *Artificial data*: Data are randomly generated in two dimensions with ten data points for each of three classes, see Figure 6.6. Since data are Euclidean, we can also directly inspect the prototypes, its approximations, and the exemplars used for the approximation. Note that the approximation is identical to the prototypes for OMP due to the dimensionality of the data.
- *VBB Midi*: This dataset is based on openly accessible public transportation time-tables provided by the Verkehrsverbund Berlin Brandenburg (VBB)[1]. As data points we used a subset of 352 train and metro stops in Berlin and defined the distance of two stops as the shortest possible trip between them using the Berlin public transportation system including bus, train, or metro. The supervised learning

[1] `http://daten.berlin.de/datensaetze/vbb-fahrplan-2013`

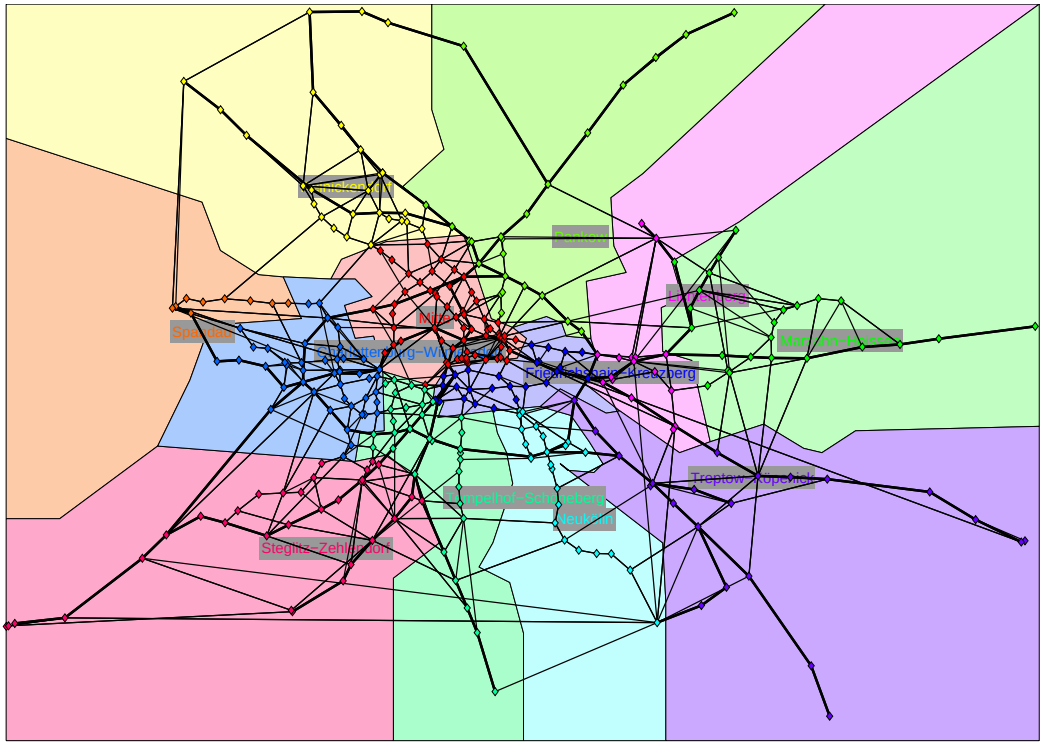


Figure 6.5: VBB Midi data set with classes meaning districts marked with different colors. The train, tram, and bus connections are shown and stations correspond to diamonds.

task is generated by using the 12 administrative districts of Berlin as class labels. Data are non-Euclidean and the distances are preprocessed using clip. See Figure 6.5 for the train, metro, and bus lines for the whole area.

Training takes place using one prototype per class and all data points in the training set.

In Table 6.6 the classification results, sparsity, Rissanen's minimum description length, and entropy are displayed. Interestingly, the classification accuracy is excellent for both data sets provided original kernel RSLVQ is used, while the accuracy deteriorates quite a lot for approximations for the VBB Midi data set due to its high intrinsic dimensionality. In contrast, the artificial data set allows a good approximation of the prototypes, with a drop in accuracy only for the two heuristic approximations. This indicates that more fundamental mathematical methods are better suited to find a close approximation of the prototypes, as can be expected due to the explicit

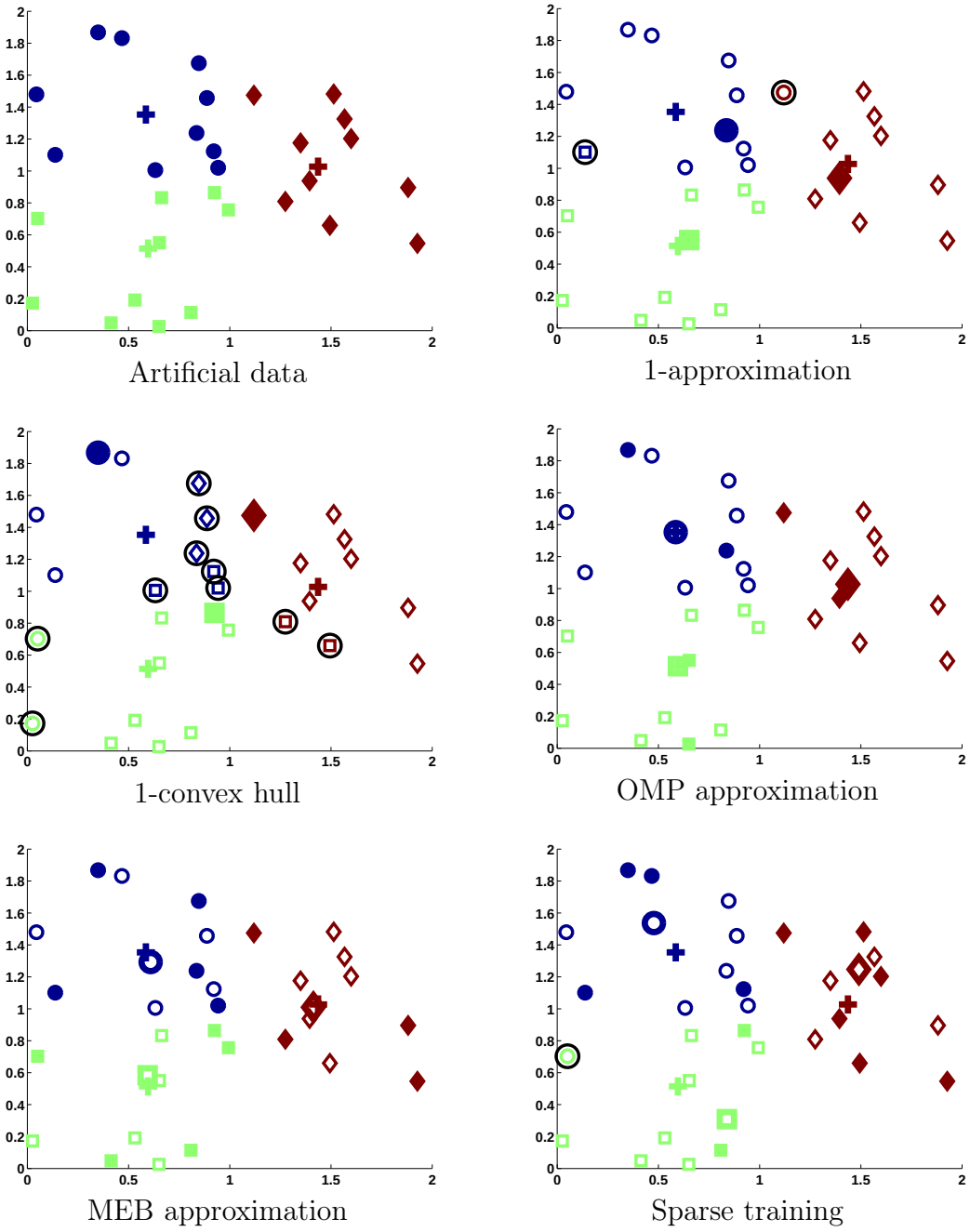


Figure 6.6: Two dimensional artificial data set with prototype locations (crosses) and the respective approximation (big symbols). The exemplars used to represent the approximated prototypes are shown via filled symbols. In addition, some prototype approximations cause errors, highlighted by black circles around the misclassified points.

mathematical modeling of the objective. Still, the K_{approx}-approximation gives reasonable results in both cases.

Interestingly, the exemplars which are used to represent the prototypes are qualitatively very dissimilar for the different approximation methods. For the artificial data set, only the 1-approximation searches exemplars from the class centers. All other approximations select exemplars which are located more at the class boundaries. Further, the number of exemplars which are necessary to obtain a good approximation is higher than for the 1-approximation. A similar conclusion can be drawn in the VBB Midi data set, see Figure 6.7, where the central part of the transportation map is displayed. For the 1-approximation, the prototypes and exemplars are located in the center, but distortions are observed for the other techniques. In particular the two techniques based on mathematical optimization, OMP and MEB, put exemplars at the boundaries of the receptive fields, as indicated by the encircled points. Interestingly, the prototypes itself which are displayed as closest exemplar due to the non-Euclideanity of the data set are often located at central positions of the traffic map, hence we would expect those to be representative as concerns centrality of the traffic stops. Note that bus lines are not displayed since these are too many. Nevertheless, bus lines often account for short distances of stations in particular at class boundaries, such that misclassifications can easily occur.

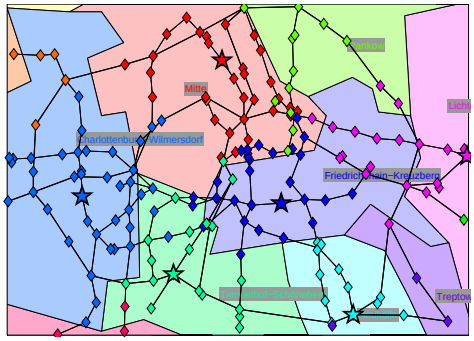
6.7 Discussion

We have investigated kernel robust soft LVQ and the possibility to obtain sparse solutions, by means of different approximation schemes applicable while or after training. These methods aim at an improved sparsity of the classifier, resulting in an enhanced interpretability of the results, thus addressing one of the most severe drawbacks of kernel RSLVQ.

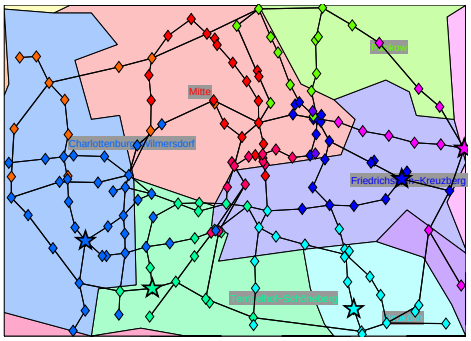
Interestingly, it is indeed possible to obtain sparse representations of high accuracy for all but one data set within a benchmark suite, however, the optimum method varies. Very simple techniques such as an approximation by the closest exemplars seem to work as well as more complex optimization approaches such as provided by OMP or MEB. The accuracy of MEB and OMP can be better due to their explicit mathematical minimization of the representation error, but they use exemplars located at class boundaries due to the used mathematical formalism. Hence it is not clear whether they are more interpretable. A higher number of exemplars is necessary to describe the class boundaries, while simple heuristics use exemplars at central positions of the classes. We have proposed first quantitative measures to evaluate

	kernel RSLVQ	K_{approx}-approximation		K_{hull}-convex hull		OMP	MEB	sparse training
		K_{approx}=1	K_{approx}=10	K_{hull}=1	K_{hull}=10			
Misclassifications								
VBB Midi								
clip	0.00	22.73	21.45	43.75	**14.77**	15.62	17.33	18.18
flip	0.00	29.55	20.45	38.35	18.47	21.31	17.05	**12.50**
Artificial data	0.00	6.67	0.00	33.33	0.00	0.00	0.00	3.33
Sparsity								
VBB Midi								
clip	29.33	**1.00**	10.00	**1.00**	9.92	4.08	7.00	14.42
flip	29.33	**1.00**	10.00	**1.00**	9.92	1.75	7.25	13.42
Artificial data	10.00	1.00	10.00	1.00	10.00	2.00	4.33	4.00
Rissanen's minimum description length								
VBB Midi								
clip	18.22	25.01	21.68	45.43	**18.35**	**18.35**	23.24	20.63
flip	18.21	29.60	22.60	42.41	**20.41**	24.84	23.91	20.46
Artificial data	1.47	5.15	1.47	29.83	1.47	3.50	1.78	2.88
Entropy								
VBB Midi								
clip	9.63	5.64	7.51	**5.10**	8.90	8.07	6.96	9.02
flip	5.54	4.34	6.35	3.72	5.04	3.61	**3.29**	4.48
Artificial data	2.22	1.53	2.22	1.70	2.22	2.00	1.57	2.45

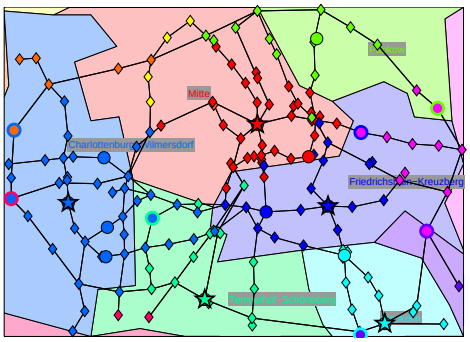
Table 6.6: Results of kernel RSLVQ and diverse sparse approximations on two illustrative examples.



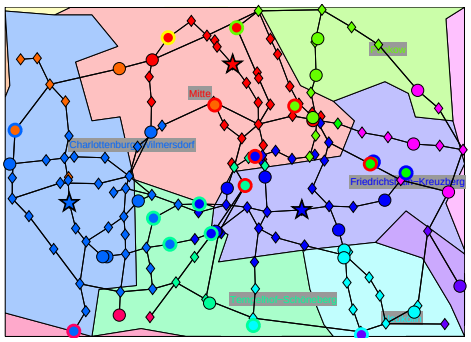
1-approximation



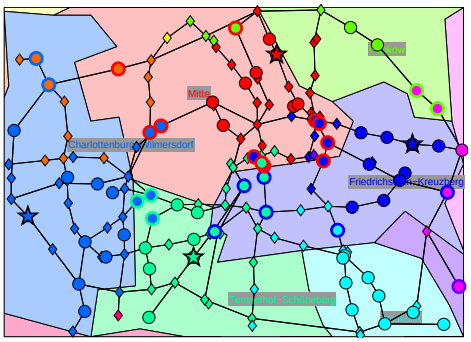
1-convex hull



OMP approximation



MEB approximation



Sparse training

Figure 6.7: Central part of the VBB Midi data set with classes meaning districts marked with different colors. Prototypes are represented by their closest exemplar, the data being non-Euclidean, displayed as a star. Further, the exemplars which are used to represent the prototypes, are marked with big circles. Points correspond to diamonds. In addition, train and tram connections are shown, but no bus connections. Misclassifications are indicated by color codes of the stations.

the usefulness of the results as regards interpretability, relying on Rissanen's minimum description length and the entropy.

Using these techniques, we have taken a further step to bring kernel RSLVQ towards efficient methods, which preserve the interpretability of their vectorial counterparts.

Chapter 7

Conclusions

Learning vector quantization (LVQ) as proposed by Kohonen [61] more than 20 years ago still constitutes a popular and widely used classification scheme, in particular due to its intuitive training algorithm and classification behavior. The fact that the classifier represents its classification prescription in a compact way in terms of a small number of prototypical representatives enables its applicability in particular in the medical domain, where human insight is often crucial, or in online learning scenarios such as online vision systems where a compact representation of the already gathered information is required for further adaptation [1, 8, 59, 23, 58]. While original LVQ has been proposed on heuristic grounds, mimicking learning paradigms in biological systems, quite a few variants have been proposed in the last years which can be derived based on mathematical cost functions.

In this thesis, we have focussed on two variants. Generalized LVQ (GLVQ) [81] relies on a cost function which can be linked to large margin classifiers [85], enabling a particularly robust classification scheme. As an alternative, robust soft LVQ (RSLVQ) models the data in terms of a mixture of Gaussians in a probabilistic framework. Training can be derived thereof as likelihood ratio optimization [88]. The formulation as cost function optimization allows to easily integrate a larger flexibility into the prescriptions such as the concept of metric learning [85, 88]. We have used this flexibility to extend the techniques towards even more general forms, in particular LVQ variants which can deal with arbitrary proximity data. This way, we followed the lines of a few approaches which have been developed in the last years to extend LVQ schemes or, more generally, prototype based approaches beyond the vectorial setting, see for example [62, 20, 28, 39, 14, 76, 71].

Starting from these approaches, within this thesis, we particularly tackled the following central questions in this realm:

- How to devise a LVQ technique which stems from a clear probabilistic

model and which can be used for general proximity data? In Chapter 3 we proposed kernel RSLVQ as a solution to this problem.

- In how far do the diverse methods, which have been proposed in the literature, differ? Do they have a common ground? In Chapter 4, we formalized a general framework based on the underlying pseudo-Euclidean embedding, which enables a clear description of the differences and the similarities of kernel and relational methods which integrate a smooth prototype adaptation into LVQ for proximities. In particular, we clarified a crucial difference of kernel and relational approaches, which does not only consist in the interface to the data like dissimilarities versus similarities, but, more severely, in different numeric optimization methods, namely gradients with respect to the prototypes or their coefficients, respectively.

- How to avoid the squared complexity of training of these techniques? In Chapter 5, we elucidated the Nyström technique, which has been used already before in this context, and which can be directly transferred to kernel RSLVQ. We substantiated this approximation technique with a method which enables to test prior to training whether the Nyström approximation is likely to work. Since the full data and learning capacity is often not available before training, or full training is costly provided large data sets are dealt with, this question can be of crucial relevance for the choice of the used method.

- How to maintain sparse, interpretable models? In Chapter 6 we discussed, that this question constitutes a core issue in LVQ schemes, which are often picked due to their intuitive and compact model in the vectorial setting. Albeit a few sparse approximation schemes have been proposed before in particular in the context of learning for big data [39], an extensive investigation how to provide and evaluate approximations has been missing in the literature. We addressed the problem of sparse approximations elucidating the properties and behavior of a variety of different techniques.

Together, these findings form a large step towards efficient and robust LVQ technology for general proximity data.

Note that LVQ schemes are in some sense complementary to popular classification schemes as provided for example using support vector machines (SVM). While both techniques constitute large margin approaches thus providing excellent generalization ability, one of the strengths of SVM is its very robust behavior due to a convex cost function with unique solutions.

LVQ, on the contrary, typically possesses local optima, and optimization using gradient techniques is usually necessary. However, while SVM represents models in terms of support vectors, which constitute points at the boundary, the number of which typically scales with the size of the training set, LVQ represents solutions in terms of few typically prototypes only, resulting in an improved interpretability and classification time. On the down-side, SVM can often represent the boundaries in more detail because of its focus on the boundaries, while LVQ classifiers stay with more simple models. Because of the need of interpretable models in domains such as biomedical applications where the ultimate responsibility lies with the human applicant, however, sparse interpretable models such as LVQ classifiers enjoy an increasing popularity among practitioners.

SVM has one severe benefit as compared to classical vectorial LVQ. Data are addressed in terms of kernel values only, such that the kernel constitutes a canonic interface based on which more general data structures can be treated. Based on this observation, structure kernels have been designed with great success for application areas involving complex structures such as biomedical data analysis or text processing [29, 26]. By extending LVQ to proximities, as investigated in this thesis, this gap is closed also for LVQ, since it becomes suitable not only for kernels, but also for more general proximity data. However, the question of efficiency and interpretability are crucial in this context, since they address two of the benefits because of which practitioners choose prototype-based variants in the first place instead of alternatives such as SVM. The theoretical as well as experimental findings demonstrate that LVQ for proximities provides an efficient classification technology for general data structures which is competitive to SVM and which can maintain the benefits of original vectorial LVQ such as sparsity in many cases.

The work as conducted in this thesis also opens the way towards a number of future perspectives. Large parts of this project have been conducted under the umbrella of the DFG research grant DiDi – Discriminative Dimensionality Reduction. Roughly speaking, this topic deals with the question how to devise mechanisms which enable the visualization of data guided by certain discriminative criteria for example visualization of medical cases as concerns a certain disease which might be present or not. One open problem in this context is how to visualize non-vectorial data in a discriminative way. LVQ variants provide one possible remedy for this problem. These methods enable a choice of representative prototypes which are particularly discriminative for a given task. Hence one can represent data by means of their relation to these prototypes, ending up in a discriminative vector of distances to these prototypes, which can easily be displayed using standard techniques.

Another topic which has been addressed in this thesis and which is of

great practical relevance concerns the evaluation of classifiers or, more generally, machine learning techniques. For decades, the classification accuracy has been the almost only criterion based on which classifiers have been compared – whereby the way in which the accuracy is evaluated can differ, referring to the simple classification error, a receiver operating characteristic curve, the F-measure, and so forth. However, this accuracy is partially an academic measure, since machine learning tools are always used within a greater context. Here, not only the performance for a very specific task, but also the classifier robustness, its provision of auxiliary information and interpretability, its ability of lifelong adaptation, its easy maintainability, its communicability, and so forth constitute important aspects based on which the technology is judged in the long run [79]. These properties, are however, often very hard to quantify, such that their integration into machine learning tools is difficult. We have made an attempt to quantify in how far the observed models provide representative prototypes and hence interpretability of the results. This is along the lines of other recent attempts [3], opening up new ideas for this important question.

Another point, which has only been touched in this thesis, lies at the ground of an open issue for both, theory and practice. With RSLVQ, we have considered a probabilistic model, and extended this model towards a kernel space. A pseudo-Euclidean embedding even enables its relationalization, meaning it can be technically applied to every symmetric proximity matrix, even if this is not a valid kernel, meaning not Euclidean. This is of great practical relevance since many concrete proximities or even distances are non-Euclidean such as alignment distances in bioinformatics or dynamic time warping for time series processing. This opens the question about what a valid probability model for such data is, since the pseudo-Euclidean space does not provide such an interpretation. We have avoided this problem by referring to kernels only and suitable kernel corrections for this setting, however, a more fundamental solution which enables a generic probabilistic interpretation would be desirable. Note that, in parts, a restriction to discrete values only within median variants can solve this dilemma [70], but discrete methods usually pay the price of a reduced representation capability and complex numeric optimization.

Bibliography

[1] W. Arlt, M. Biehl, A. E. Taylor, S. Hahner, R. Libe, B. A. Hughes, P. Schneider, D. J. Smith, H. Stiekema, N. Krone, E. Porfiri, G. Opocher, J. Bertherat, F. Mantero, B. Allolio, M. Terzolo, P. Nightingale, C. H. L. Shackleton, X. Bertagna, M. Fassnacht, and P. M. Stewart. Urine steroid metabolomics as a biomarker tool for detecting malignancy in adrenal tumors. Journal of Clinical Endocrinology and Metabolism, 96: 3775–3784, 2011.

[2] A. Backhaus and U. Seiffert. Quantitative measurements of model interpretability for the analysis of spectral data. Proceedings of IEEE Symposium Series on Computational Intelligence, 18–25, 2013.

[3] A. Backhaus and U. Seiffert. Classification in high-dimensional spectral data: Accuracy vs. interpretability vs. model size. Neurocomputing, 131: 15–22, 2014.

[4] M. Badoiu and K. L. Clarkson. Optimal core sets for balls. DIMACS Workshop on Computational Geometry, 40(1): 14–22, 2002.

[5] P. L. Bartlett and S. Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. Journal of Machine Learning Research, 3: 463–4982, 2002.

[6] V. Van Belle, B. Van Calster, D. Timmerman, T. Bourne, C. Bottomley, L. Valentin, P. Neven, S. Van Huffel, J. Suykens, and S. Boyd. A mathematical model for interpretable clinical decision support with applications in gynecology. PLoS One, 7(3), 2012.

[7] V. Van Belle and P. Lisboa. Automated selection of interaction effects in sparse kernel methods to predict pregnancy viability. Proceedings IEEE Symposium on Computational Intelligence and Data Mining, 26–31, 2013.

[8] M. Biehl, K. Bunte, and P. Schneider. Analysis of flow cytometry data by matrix relevance learning vector quantization. PLOS ONE, 8(3), 2013.

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

[10] M. Biehl, B. Hammer, P. Schneider, and T. Villmann. Metric learning for prototype-based classification. Bianchini M, Maggini M, Scarselli F, editors, Innovations in Neural Information Paradigms and Applications. Studies in Computational Intelligence, 247: 183–199, Springer, 2009.

[11] M. Biehl, B. Hammer, M. Verleysen, T. Villmann, and editors. Similarity based clustering. Springer Lecture Notes Artificial Intelligence, 5400. Springer, 2009.

[12] C. Bishop. Pattern recognition and machine learning. Springer, 2006.

[13] C. Bottomley, V. Van Belle, E. Kirk, S. Van Huffel, D. Timmerman, and T. Bourne. Accurate prediction of pregnancy viability by means of a simple scoring system. Human Reproduction, 28(1): 68–76, 2013.

[14] R. Boulet, B. Jouve, F. Rossi, and N. Villa. Batch kernel som and related laplacian methods for social network analysis. Neurocomputing, 71(7–9): 1257–1273, 2008.

[15] A. M. Bruckstein, D. L. Donoho, and M. Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. SIAM Review, 51(1): 34–81, 2009.

[16] K. Bunte, P. Schneider, B. Hammer, F.-M. Schleif, T. Villmann, and M. Biehl. Limited rank matrix learning, discriminative dimension reduction and visualization. Neural Networks, 26: 159–173, 2012.

[17] 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.

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

[19] C. Cortes and V. Vapnik. Support vector machine. Machine Learning, 20: 273–297, 1995.

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

[21] K. Crammer, R. Gilad-bachrach, A. Navot, and N. Tishby. Margin analysis of the lvq algorithm. Advances in Neural Information Processing Systems, 462–469, 2002.

[22] J. J. G. de Vries, S. C. Pauws, and M. Biehl. Insightful stress detection from physiology modalities using learning vector quantization. Neurocomputing, 151: 873–882, 2015.

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

[24] D. L. Donoho. For most large underdetermined systems of linear equations the minimal l1-norm solution is also the sparsest solution. Communications on pure and applied mathematics, 56(6): 797–829, 2006.

[25] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. G. Amorim. Do we need hundreds of classifiers to solve real world classification problems? Journal of Machine Learning Research, 15(1): 3133–3181, 2014.

[26] P. Frasconi, M. Gori, and A. Sperduti. A general framework for adaptive processing of data structures. IEEE Transactions on Neural Networks, 9(5): 768–786, 1998.

[27] B. Frenay, D. Hofmann, A. Schulz, M. Biehl, and B. Hammer. Valid interpretation of feature relevance for linear data mappings. Computational Intelligence and Data Mining, 149–156, 2014.

[28] B. J. Frey and D. Dueck. Clustering by passing messages between data points. Science, 315(5814): 972–976, 2007.

[29] T. Gärtner. Kernels for structured data. PhD thesis, University of Bonn, 2005.

[30] I. Giotis, K. Bunte, N. Petkov, and M. Biehl. Adaptive matrices and filters for color texture classification. Journal of Mathematical Imaging and Vision, 2012.

[31] I. Giotis, N. Molders, S. Land, M. Biehl, M. F. Jonkman, and N. Petkov. MED-NODE: a computer-assisted melanoma diagnosis system using non-dermoscopic images. Expert systems with applications, 42(19): 6578–6585, 2015.

[32] A. Gisbrecht. Advances in dissimilarity-based data visualisation. Universität Bielefeld, 2015.

[33] A. Gisbrecht, D. Hofmann, and B. Hammer. Discriminative dimensionality reduction mappings. Advances in Intelligent Data Analysis, 7619: 126–138, 2012.

[34] A. Gisbrecht, B. Mokbel, and B. Hammer. The nyström approximation for relational generative topographic mappings. Neural Information Processing Systems workshop on challenges of Data Visualization, 2010.

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

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

[37] P. Grünwald. The minimum description length principle. MIT Press, 2007.

[38] B. Hammer, A. Gisbrecht, and A. Schulz. Applications of discriminative dimensionality reduction. Proceedings of International Conference on Pattern Recognition Applications and Methods, 2013.

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

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

[41] B. Hammer, A. Micheli, and A. Sperduti. Universal approximation capability of cascade correlation for structures. Neural Computation, 17: 1109–1159, 2005.

[42] B. Hammer, B. Mokbel, F.-M. Schleif, and X. Zhu. Prototype based classification of dissimilarity data. Advances in Intelligent Data Analysis X, 7014: 185–197, 2011.

[43] B. Hammer, B. Mokbel, F.-M. Schleif, and X. Zhu. White box classification of dissimilarity data. Lecture Notes in Computer Science, 7208: 309–321, 2012.

[44] B. Hammer, F.-M. Schleif, and X. Zhu. Relational extensions of learning vector quantization. Bao-Liang Lu, Liqing Zhang, and James Kwok, editors, Neural Information Processing, 7063: 481–489. Springer, 2011.

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

[46] T. Heskes. Self-organizing maps, vector quantization, and mixture modeling. IEEE Transactions on Neural Networks, 12(6): 1299–1305, 2001.

[47] D. Hofmann. Sparse approximations for kernel robust soft lvq. Mittweida Workshop on Computational Intelligence, 2013.

[48] D. Hofmann, A. Gisbrecht, and B. Hammer. Discriminative probabilistic prototype based models in kernel space. New Challenges in Neural Computation, TR Machine Learning Reports, 2012.

[49] D. Hofmann, A. Gisbrecht, and B. Hammer. Efficient approximations of kernel robust soft lvq. Workshop on Self-Organizing Maps, 198: 183–192, 2012.

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

[51] D. Hofmann and B. Hammer. Kernel robust soft learning vector quantization. Artificial Neural Networks in Pattern Recognition, 7477: 14–23, 2012.

[52] D. Hofmann and B. Hammer. Sparse approximations for kernel learning vector quantization. European Symposium on Artificial Neural Networks, 549–554, 2013.

[53] D. Hofmann, F.-M. Schleif, B. Paaßen, and B. Hammer. Learning interpretable kernelized prototype-based models. Neurocomputing, 141: 84–96, 2014.

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

[55] B.-H. Juang and S. Katagiri. Discriminative learning for minimum error classification [pattern recognition]. IEEE Transactions on Signal Processing, 40(12): 3043–3054, 1992.

[56] M. Kaden, M. Riedel, W. Hermann, and T. Villmann. Border-sensitive learning in generalized learning vector quantization: an alternative to support vector machines. Soft Computing, 19(9): 2423–2434, 2015.

[57] M. Kästner, M. Riedel, M. Strickert, W. Hermann, and T. Villmann. Border-sensitive learning in kernelized learning vector quantization. Advances in Computational Intelligence, 7902: 357–366, 2013.

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

[59] S. Kirstein, H. Wersing, H.-M. Gross, and E. Körner. A life-long learning vector quantization approach for interactive learning of multiple categories. Neural Networks, 28: 90–105, 2012.

[60] T. Kohonen. Learning vector quantization. The handbook of brain theory and neural networks, 537–540, 1997.

[61] T. Kohonen. Self-oganizing maps. Springer, 3rd edition, 2000.

[62] T. Kohonen and P. Somervuo. How to make large self-organizing maps for nonvectorial data. Neural Networks, 15(8–9): 945–952, 2002.

[63] M. Kotlyar, S. Fuhrman, A. Ableson, and R. Somogyi. Spearman correlation identifies statistically significant gene expression clusters in spinal cord development and injury. Neurochemical Research, 27(10): 1133–40, 2002.

[64] V. Losing, B. Hammer, and H. Wersing. Interactive online learning for obstacle classification on a mobile robot. International Joint Conference on Neural Networks, 2015.

[65] C. Lundsteen, J. Phillip, and E. Granum. Quantitative analysis of 6985 digitized trypsin g-banded human metaphase chromosomes. Clinical Genetics, 18(5): 355–370, 1980.

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

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

[68] B. Mokbel, A. Hasenfuss, and B. Hammer. Graph-based representation of symbolic musical data. Graph-Based Representations in Pattern Recognition, 5534: 42–51, 2009.

[69] B. K. Natarajan. Sparse approximate solutions to linear systems. SIAM Journal on Computing, 24(2): 227–234, 1995.

[70] D. Nebel, B. Hammer, K. Frohberg, and T. Villmann. Median variants of learning vector quantization for learning of dissimilarity data. Neurocomputing, 169: 295–305, 2015.

[71] D. Nebel, B. Hammer, and T. Villmann. A median variant of generalized learning vector quantization. International Conference on Neural Information Processing, 8227: 19–26, 2013.

[72] B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature, 381: 607–609, 1996.

[73] E. Pekalska and R. P. Duin. The dissimilarity representation for pattern recognition. foundations and applications. World Scientific, 2005.

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

[75] A. K. Qin and P. N. Suganthan. Kernel neural gas algorithms with application to cluster analysis. Proceedings of the 17th International Conference on Pattern Recognition, 617–620, 2004.

[76] A. K. Qin and P. N. Suganthan. A novel kernel prototype-based learning algorithm. Proceedings of the 17th International Conference on Pattern Recognition, 621–624, 2004.

[77] J. Rissanen. Modeling by the shortest data description. Automatica, 14: 465–471, 1978.

[78] F. Rossi and N. Villa-Vialaneix. Consistency of functional learning methods based on derivatives. Pattern Recognition Letters, 32(8): 1197–1209, 2011.

[79] C. Rudin and K. L. Wagstaff. Machine learning for science and society. Machine Learning, 95(1): 1–9, 2014.

[80] H. Ruiz, I. H. Jarman, P. J. G. Lisboa, S. Ortega-Martorell, A. Vellido, E. Romero, and J. D. Martin. Towards interpretable classifiers with blind signal separation. Proceedings of the International Joint Conference on Neural Networks, 1–7, 2012.

[81] A. Sato and K. Yamada. Generalized learning vector quantization. Advances in Neural Information Processing Systems, MIT Press, 7: 423–429, 1995.

[82] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. Computational capabilities of graph neural networks. IEEE Transactions on Neural Networks, 20(1): 81–102, 2009.

[83] F.-M. Schleif, T. Villmann, B. Hammer, and P. Schneider. Efficient kernelized prototype based classification. International Journal of Neural Systems, 21(6): 443–457, 2011.

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

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

[86] P. Schneider, M. Biehl, and B. Hammer. Hyperparameter learning in probabilistic prototype-based models. Neuromputing, 73(7–9): 1117–1124, 2009.

[87] B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K.-R. Muller, G. Rätsch, and A. J. Smola. Input space versus feature space in kernel-based methods. IEEE Transactions on Neural Networks, 10(5): 1000–1017, 1999.

[88] S. Seo and K. Obermayer. Soft learning vector quantization. Neural Computation, 15: 1589–1604, 2003.

[89] S. Seo and K. Obermayer. Dynamic hyperparameter scaling method for lvq algorithms. International Joint Conference on Neural Networks, 3196–3203, 2006.

[90] J. Shawe-Taylor and N. Cristianini. Kernel methods for pattern analysis. Cambridge University Press, 2004.

[91] N. Z. Shor. Minimization methods for non-differentiable functions. Springer Series in Computational Mathematics. Springer, 1985.

[92] I. W. Tsang, J. T. Kwok, and P.-M. Cheung. Core vector machines: Fast svm training on very large data sets. Journal of Machine Learning Research, 6: 363–3920, 2005.

[93] L. van der Maaten and G. Hinton. Visualizing high-dimensional data using t-sne. Journal of Machine Learning Research, 9: 2579–2605, 2008.

[94] A. Vellido, J. D. Martin-Guerroro, and P. Lisboa. Making machine learning models interpretable. European Symposium on Artificial Neural Networks, 163–172, 2012.

[95] T. Villmann and S. Haase. Divergence-based vector quantization. Neural Computation, 23(5): 1343–1392, 2011.

[96] T. Villmann, S. Haase, and M. Kaden. Kernelized vector quantization in gradient-descent learning. Neurocomputing, 147: 83–95, 2015.

[97] C. K. I. Williams and M. Seeger. Using the nyström method to speed up kernel machines. Advances in Neural Information Processing Systems, 13: 682–688, 2001.

[98] K. Zhang, I. W. Tsang, and J. T. Kwok. Improved nyström low-rank approximation and error analysis. Proceedings of the 25th International Conference on Machine Learning, 1232–1239, 2008.

[99] X. Zhu, F.-M. Schleif, and B. Hammer. Adaptive conformal semi-supervised vector quantization for dissimilarity data. Pattern Recognition Letters, 49: 138–145, 2014.