Rejection and Online Learning with Prototype-based Classifiers in Adaptive Metrical Spaces

Lydia Fischer

Dissertation

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

Disputation am 23. September 2016

Universität Bielefeld, Technische Fakultät

First Examiner: Prof. Dr. Barbara Hammer, Bielefeld University, Germany
Second Examiner: Dr. Heiko Wersing, Honda Research Institute Europe, Germany
Third Examiner: Prof. Dr. Thomas Martinetz, Lübeck University, Germany

Printed on non-ageing paper according to ISO 9706.

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

Lydia Fischer
CorLab – Research Institute for Cognition and Robotics
Theoretical Computer Science Research Group
Honda Research Institute Europe

Abstract

The rising amount of digital data, which is available in almost every domain, causes the need for intelligent, automated data processing. Classification models constitute particularly popular techniques from the machine learning domain with applications ranging from fraud detection up to advanced image classification tasks. Within this thesis, we will focus on so-called prototype-based classifiers as one prominent family of classifiers, since they offer a simple classification scheme, interpretability of the model in terms of prototypes, and good generalisation performance. We will face a few crucial questions which arise whenever such classifiers are used in real-life scenarios which require robustness and reliability of classification and the ability to deal with complex and possibly streaming data sets. Particularly, we will address the following problems:

- Deterministic prototype-based classifiers deliver a class label, but no confidence of the classification. The latter is particularly relevant whenever the costs of an error are higher than the costs to reject an example, e. g., in a safety critical system. We investigate ways to enhance prototype-based classifiers by a certainty measure which can efficiently be computed based on the given classifier only and which can be used to reject an unclear classification.

- For an efficient rejection, the choice of a suitable threshold is crucial. We investigate in which situations the performance of local rejection can surpass the choice of only a global one, and we propose efficient schemes how to optimally compute local thresholds on a given training set.

- For complex data and lifelong learning, the required classifier complexity can be unknown a priori. We propose an efficient, incremental scheme which adjusts the model complexity of a prototype-based classifier based on the certainty of the classification. Thereby, we put particular emphasis on the question how to adjust prototype locations and metric parameters, and how to insert and/or delete prototypes in an efficient way.

- As an alternative to the previous solution, we investigate a hybrid architecture which combines an offline classifier with an online classifier based on their certainty values, thus directly addressing the stability/plasticity dilemma. While this is straightforward for classical prototype-based schemes, it poses

some challenges as soon as metric learning is integrated into the scheme due to the different inherent data representations.

- Finally, we investigate the performance of the proposed hybrid prototype-based classifier within a realistic visual road-terrain-detection scenario.

Thanks . . .

. . . to my encouraging supervisors who had a main impact on my development as PhD student and the work I did.

. . . to Dr. Heiko Wersing.

. . . to Prof. Dr. Barbara Hammer.

. . . to my proofreaders of the thesis.

. . . to the Honda Research Institute (HRI-EU) for funding the PhD project giving me this great opportunity.

. . . to my friends at the Honda Research Institute for a great time, fruitful discussions, and your support.

. . . to my friends at the Technical Computer Science Group at Bielefeld University for great conference trips, dinners, and jam sessions . . . we rocked ;-)

. . . to my friends at the Computational Intelligence Group at Mittweida University of applied sciences.

. . . to my family.

Contents

List of Tables

List of Figures

Abbreviations and Symbols

Abbreviations

k-NN	k-nearest neighbour classifier
ARC	accuracy-reject-curve
CFE	catastrophic forgetting effect
DP	dynamic programming
GLVQ	generalised learning vector quantisation
GMLVQ	generalised matrix learning vector quantisation
GMM	Gaussian mixture model
ioLVQ	incremental online learning vector quantisation
iSVM	incremental support vector machine
LGMLVQ	localised generalised matrix learning vector quantisation
LVQ	learning vector quantisation
NNC	nearest neighbour classifier
OOL	combined offline and online learning
RSLVQ	robust soft learning vector quantisation
RTDS	road terrain detection system
SVM	support vector machine

Classifier variables

α_j	a leaf of a decision tree
$\eta(\mathbf{w})$	accumulated certainty values of classifications of points in the Voronoi cell represented by $\mathbf{w}$
Γ	the decision border induced by the decision tree
g_{max}	maximum storage capacity of S
$\hat{p}(\cdot)$	estimated probability
Λ	global matrix of d_Λ
Ω	square root of Λ
$\Omega(l,k)$	single element in the l-th row and the k-th column of Ω
$\Phi(\cdot)$	a monotone increasing function

r_{num}	In periods of r_{num} seen training data points, it is checked if prototypes should be removed.
σ	variance of a Gaussian
ε	learning rate
$\lvert \cdot \rvert$	cardinality of a set
$\mathbf{w}$	a prototype
$\mathbf{w}^{\pm}$	closest prototype of the same class/a different class with respect to a data point
$\mathbf{x}$	a data point
$\mathbf{x}(l), \mathbf{w}(l), [\cdot]_l$	l-th element of a vector
ξ	number of prototypes
ζ	number of the partitions of the input space
ζ	number of the partitions of the input space
c_j	label of prototype $\mathbf{w}_j$
$d(\cdot)$	dissimilarity measure
$d^{\pm}$	dissimilarity of a data point to the closest prototype of the same/of a different class
$d_{\Lambda_j}(\cdot)$	dissimilarity measure of LGMLVQ
$d_{\Lambda}(\cdot)$	dissimilarity measure of GMLVQ
$E_{\ldots}$	cost function of ...
M	dimension of data, prototypes
$p(\cdot)$	a probability
S	set of wrongly classified data points
V_j	Voronoi cell of prototype $\mathbf{w}_j$
W	set of prototypes
X	a set of data points
y_i	class label of data point $\mathbf{x}_i$
Z	number of classes

Rejection

$\mathcal{E}_{\theta_j}$	true rejects in Υ_j
$\mathcal{E}_{\theta}$	true rejects: rejected data points which would be classified wrongly
$\mathcal{L}_{\theta_j}$	false rejects in Υ_j
$\mathcal{L}_{\theta}$	false rejects: rejected data points which would be classified correctly
$\mathcal{X}_{\theta_j}$	rejected points in Υ_j
$\mathcal{X}_{\theta}$	rejected data points with a lower certainty value than θ

$\mathrm{opt}(n, j, i)$	It measures the maximum number of true rejects obtained with n false rejects, and a restricted threshold vector (see 59).
$\boldsymbol{\theta}$	threshold vector (local thresholds)
$\boldsymbol{\theta}$	threshold vector (local thresholds)
Θ	the set of thresholds corresponding to correctly classified points
θ	threshold for rejection
Θ_j	the set of thresholds corresponding to correctly classified points with respect to Υ_j
θ_j	local threshold, responsible for Υ_j
Υ_j	The j-th partition of the input space, e. g., a Voronoi cell.
E	set of wrongly classified data points, errors
E_j	set of wrongly classified data points, errors in Υ_j
L	set of correctly classified data points
L_j	set of correctly classified data points in Υ_j
$r(\cdot)$	certainty measure
X_{θ_j}	accepted points in Υ_j
X_θ	accepted data points with a higher certainty value than θ

1. Introduction

Chapter overview *In this chapter, we first motivate the tackled topics of this thesis. Thereafter we give a structural overview of the single chapters.*

1.1. Motivation

Digitalisation is progressing in many domains like in industrial applications with regard to industry 4.0, in biomedicine, personalisation, and driver assistance systems. This leads to a huge amount of data. Inspection and handling of gathered data make automated data analysis more and more important, since an analysis of the data could hardly be done by humans in an acceptable time frame. Machine learning and related research areas[1] provide techniques to extract structures or information automatically from available data.

These techniques are either supervised or unsupervised[2]. If information is available which groups data into a finite number of classes, a supervised technique can be applied. Unsupervised techniques deal with data without such information. For supervised machine learning, one aim is to extract characteristics of the classes and to identify where classes differ and what they have in common, respectively. One of the most popular principles to reach this aim is classification.

The learning techniques used for classification have a basic principle in common: they want to minimise the number of errors in classification since they cause costs. Depending on the application these costs range between low (not critical) and huge (very critical, e. g., in the biomedical domain or safety related applications). One realisation of this principle which puts a particular emphasis on its generalisation ability, is based on maximising the margin between classes. Popular representatives are support vector machines (SVM, Boser et al., 1992) and learning vector quantisation (LVQ, Kohonen, 1989). LVQ constitutes a prototype-based technique, and it will play a central role in this thesis as classification scheme. Another basic idea is to combine many weak classifiers into one powerful classifier,

[1] These areas are for instance data analysis, computational intelligence, neural computation, pattern recognition, and statistical modelling. They aim at automatically extracting information from given data.

[2] We are aware of finer grained subgroups as for instance semi-supervised techniques. Since we only use supervised techniques, we only distinguish these two groups.

to a so-called ensemble classifier. Random forests belong to ensemble classifier and they consist of several decision trees. This classifier is suited for large data sets due to its short training time and its very efficient classification scheme. Neural networks are classifiers which are inspired by the human brain. They try to mimic the functionality of the brain. Lately they got popular with deep neural networks which are a powerful tool for classification, but the classifier is complex and the internal mechanisms are hard to interpret. There are classifiers based on probabilistic modelling as well, e. g., the Bayes classifier. They aim at estimating the class densities, e. g., with Gaussian distributions. Later, we will deal mainly with LVQ schemes as a typical classification techniques in this thesis because of their simple classification scheme and interpretable classification models. In chapter 3, we will explain the used LVQ schemes within this thesis as well as the concept of metric learning.

Since wrong classifications can have severe effects, a certainty information can be at least as important as the classification itself. Depending on the application, it can be better to reject a classification instead of making a mistake. Hence a reject can have lower costs than a false classification. Supervised techniques based on a probabilistic data modelling include such certainty information. The estimated probabilities of a data point belonging to a specific class can be used for this purpose. A low probability value can lead to rejection since the classification is uncertain. In chapter 4 we will discuss rejection based on probabilities. Techniques lacking such information can be enhanced with a statistical modelling on top or with certainty information based on heuristics, e. g., statistics of the neighbourhood or on distances. Using rejection comes along with the issue of choosing a suited certainty measure and a threshold judging an uncertain decision. More sophisticated rejection strategies use multiple thresholds, e. g., class-wise or even more fine-grained partitioning of the data. In chapter 4 we discuss several certainty measures suited for prototype-based classifiers and it also analyses rejection with one global threshold. The analysis of strategies with multiple thresholds can be found in chapter 5. There we also provide techniques which determine the best possible thresholds for a given setting.

Besides the hot topic of rejection, the area of lifelong machine learning gets more and more important (Polikar and Alippi, 2014). A recent definition states: "*Lifelong Machine Learning, or LML, considers systems that can learn many tasks over a lifetime from one or more domains. They efficiently and effectively retain the knowledge they have learned and use that knowledge to more efficiently and effectively learn new tasks (Silver et al., 2013).*" Hence, approaches usable for lifelong learning have to deal with various issues. They have to be flexible such that including new content is possible (plasticity). Contrarily, already known information has to be maintained for later use (stability). The trade-off between those two

principles is known as *stability plasticity dilemma* (Grossberg, 1980). To our knowledge there exists no satisfying solution for this dilemma in machine learning. However, the human brain seems to have a very efficient solution. Some of its working principles are explored and serve as inspiration for machine learning techniques with promising results (see, e. g., Kirstein et al., 2005, 2009, 2012). Ongoing research studies how these mechanisms operate and how to use them in lifelong machine learning. In chapter 6 and 7, we will extend and merge existing lifelong learning approaches in order to combine their advantages and to improve their behaviour regarding the stability plasticity dilemma. With our approaches, we focus on lifelong adaptability of the classifier complexity for LVQ schemes and on guaranteeing a minimal performance of a system by the combination of an offline and an online classifier.

1.2. Contribution of this Thesis

This thesis contains novel insights into the following questions:

- What are good deterministic certainty measures suitable for prototype-based classifiers and which properties do they have?

  We investigate several deterministic certainty measures, e. g., the distance to the decision border. The analysed measures work well in almost all settings and they show a similar performance as probabilistic counterparts.

- Rejection can be based on a global threshold or on local thresholds which are valid in single partitions of the data space only. Which strategy should be applied in which setting and how can we choose best thresholds (global or local)?

  Local thresholds are beneficial in settings where the data have local or class specific characteristics which are not captured by the used classifier. For settings with no local characteristics or with classifiers dealing with these characteristics, a global threshold suffices. We provide an optimal dynamic programming scheme and a fast greedy algorithm which determine the best thresholds on a given setting.

- Is it useful to integrate certainty information in lifelong learning architectures and is the powerful concept of metric adaptation (Bellet et al., 2013) compatible with this learning concept?

  We show that the integration of certainty information in lifelong learning architectures is useful and that it can be combined with metric adaptation.

- Is it beneficial to transfer the concept of the discussed lifelong learning architecture to a real word application to improve its performance?

  We exemplary test this for the road terrain detection system (RTDS, Fritsch et al., 2014). The system distinguishes between road and non-road in real traffic scenes. We show that a simple on-the-fly trained approach compared to the advanced RTDS provides reasonable results with some limitations.

The next section contains the structure of this thesis and shortly recaps each chapter.

1.3. Structural Overview of this Thesis

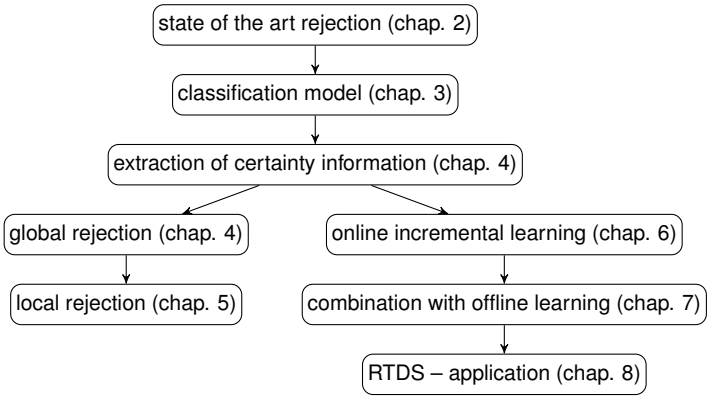


Figure 1.1.: Structural overview of this thesis (RTDS – road terrain detection system). The certainty information can either be used for rejection or for incremental learning.

The content of this thesis is visualised in Fig. 1.1. Each chapter of this thesis discusses one part of the figure in general.

Rejection strategies are the main focus of this thesis. We start with the introduction of basic rejection concepts and a literature overview in chapter 2. A reject option relaxes the constraint of a classifier to provide a class label in case of an uncertain decision. This way the classification of a data point is postponed and marked for further treatment. We establish the notation of global and local rejection and the theoretical framework for understanding and we discuss several existing rejection strategies. The literature survey enables us to point to relevant research topics which we tackle in chapter 4 and 5.

We present prototype-based classifiers and the later used LVQ schemes in chapter 3. We chose these classification technique due to its simple classification scheme, the interpretability of the classifiers and the lifelong learning suitability. But there are only few rejection strategies available for prototype-based classifiers such that we carry on the research in this important topic.

The focus of chapter 4 is global rejection, especially in the context of prototype-based classifiers. We present several dissimilarity-based certainty measures which are suited for this classifier type and we discuss their properties. Subsequently, we present results for global rejection based on those measures for artificial toy data and public benchmarks and we compare the results with probabilistic counterparts and a state of the art approach. Note that global rejection implicitly assumes an equally scaled certainty measure in the whole input space.

In chapter 5 we move on to the topic of local rejection which relaxes the implicit assumption of global rejection, since it relies on a partitioning of the input space only assuming a homogeneous scaling in each partition. We describe an optimal dynamic programming scheme and an efficient greedy scheme to determine the best parameters for local rejection. Since lots of classifiers provide a partitioning of the input space, we compare local and global rejection for several types of classifiers on artificial toy data and public benchmarks. Experiments on a medical data set highlight the usefulness in this domain.

Certainty measures indicate areas where a classifier tends to do mistakes. In chapter 6 we analyse weather certainty information can be used not only for rejection, but also in other machine learning areas, e. g., in lifelong learning. We use certainty information as criterion for inserting or deleting prototypes in a new online, incremental learning vector quantisation approach for lifelong learning. Additionally, this chapter contains an analysis of the parameters of the incremental classifier and an investigation of integrating different metric learning schemes. We report results for public benchmarks and one real-life robotic data set.

Chapter 7 contains the analysis of a different approach for lifelong learning which is an architecture combining the classifier from chapter 6 with a static offline trained one to avoid the so-called catastrophic forgetting effect. A dynamic classifier selection based on certainty values provided by the two classifiers defines the final output class label for a given input. We provide results of this architecture on artificial toy data, public benchmarks, and for real-life robotic data and we compare them to other lifelong learning methods.

So far we analysed lifelong learning approaches without a concrete application. A possible area of application could be the field of road terrain detection. Thus, we analyse in chapter 8 how good an online, trained prototype-based classifier can get in comparison to the offline trained road terrain detection system. Subsequently, we sum up the achieved knowledge and we point to future work in chapter 9.

1.4. Publications and Funding Related to this Thesis

The following articles have been published in the context of this thesis:
(More detailed references are provided in the Appendix Section A.1, on page 121.)

Journal articles

[J16] L. Fischer, B. Hammer, and H. Wersing. Optimal local rejection for classifiers. *Neurocomputing*, http://dx.doi.org/10.1016/j.neucom.2016.06.038, 2016.

[J15] L. Fischer, B. Hammer, and H. Wersing. Efficient rejection strategies for prototype-based classification. *Neurocomputing*, 169 (2015) 334-342.

Conference articles

[C16] L. Fischer, B. Hammer, and H. Wersing. Online Metric Learning for an Adaptation to Confidence Drift. In *IJCNN*, pages 748–755, 2016.

[C15b] L. Fischer, B. Hammer, and H. Wersing. Combining offline and online classifiers for life-long learning. In *IJCNN*, pages 2808–2815, 2015.

[C15a] L. Fischer, B. Hammer, and H. Wersing. Certainty-based prototype insertion/deletion for classification with metric adaptation. In *ESANN*, pages 7–12, 2015.

[C14c] L. Fischer, B. Hammer, and H. Wersing. Local rejection strategies for learning vector quantization. In *ICANN*, pages 563–570, 2014.

[C14b] L. Fischer, D. Nebel, T. Villmann, B. Hammer, and H. Wersing. Rejection strategies for learning vector quantization – A comparison of probabilistic and deterministic approaches.[3] In *WSOM*, pages 109–118, 2014.

[C14a] L. Fischer, B. Hammer, and H. Wersing. Rejection strategies for learning vector quantization. In *ESANN*, pages 41–46, 2014.

Non-refereed publications

[TR16] L. Fischer, and T. Villmann. A Probabilistic Model with Adaptive Rejection. *Machine Learning Reports*, MLR-01-2016:1–19, 2016.

[TR15] L. Fischer, B. Hammer, and H. Wersing. Optimum Reject Options for Prototype-based Classification. *CoRR*, abs/1503.06549, 2015.

Funding acknowledgments

The following institutions and associated grants are gratefully acknowledged:

- The *Cor-Lab Research Institute for Cognition and Robotics* in cooperation with the *Honda Research Institute Europe*.

[3] Winner of the *best student paper* award at WSOM 2014.

2. Principles of Rejection

Chapter overview *We introduce the basic concept of global and local rejection strategies and we give an overview on existing state of the art approaches which can be applied to several types of classifiers, e. g., probabilistic classifiers and support vector machines. It turns out that the literature is lacking adequate certainty measures for prototype-based classifiers and an analysis/comparison of them. Furthermore, there does not exist a comparison between global and local rejection strategies for different classifier types. Regarding local rejection, there is no approach for determining appropriate local thresholds for a given setting. Those weak points motivated our research which is described in the subsequent chapters.*

Parts of this chapter are based on:

[J16**]** L. Fischer, B. Hammer, and H. Wersing. Optimal Local Rejection for Classifiers. *Neurocomputing*, submitted.

[J15**]** L. Fischer, B. Hammer, and H. Wersing. Efficient Rejection Strategies for Prototype-based Classification. *Neurocomputing*, 169 (2015) 334–342.

When creating a classifier, the main target is to achieve the best possible accuracy. In case of errors, one is interested in a mechanism indicating whether the decision of a classifier can be trusted (certain classification) or if the classifier is probably erring (uncertain classification) for a given input. Such a mechanism is especially important in applications where a wrong classification can have severe effects, e. g., in driver assistance systems or in the biomedical domain. Vailaya and Jain (2000) discuss two main reasons for uncertain classification (Fig. 2.1):

- Ambiguity: The classification of the data point is unclear, e. g., the point is close to a decision border, or it lies in a region with overlapping classes.
- Outliers: The data point is dissimilar to any already seen data point, e. g., it is caused by noise or it is an instance of a yet unseen class or cluster.

A reject option can be used in order to decide if a classification is certain enough (Fig. 2.2). Rejected inputs can be marked for further tests or they can be passed to domain experts who will do a manual classification.

Based on such considerations, quite a few heuristic rejection strategies have been proposed (see e. g., Cordella et al., 1995; Vailaya and Jain, 2000; Fumera et al., 2000; Stefano et al., 2000; Fumera and Roli, 2002; Suutala et al., 2004). In the next section we explain the general setting and we give a mathematical formulation for rejection strategies.

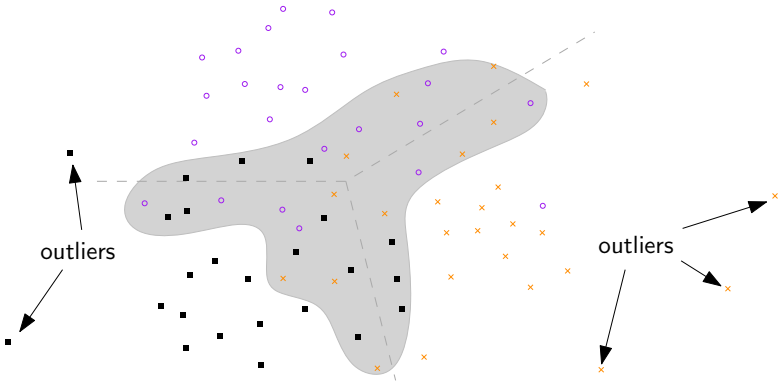


Figure 2.1.: Challenges in classification – A three class setting shows two main challenges in classification: overlapping classes and outliers (different symbols and colour indicate the class of the data points; the dashed lines indicate class borders and the grey area shows overlapping classes).

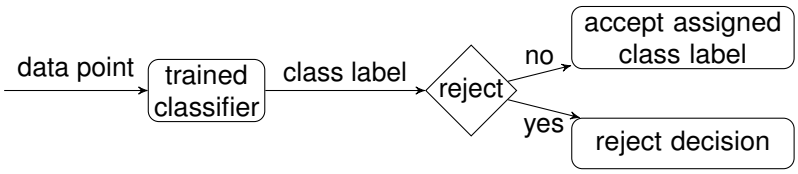


Figure 2.2.: Rejection process: A trained classifier provides a class label for the given data point. If the criteria for rejecting the decision is fulfilled, the decision is rejected and otherwise the assigned class label is accepted.

2.1. General Setting

As stated previously, we consider multi-class classification problems with training data $X = \{(\mathbf{x}_i, y_i) \in \mathbb{R}^M \times \{1, \dots, Z\}\}_{i=1}^N$, whereby data are drawn according to some unknown probability distribution P on $\mathbb{R}^M \times \{1, \dots, Z\}$. A classifier implements a function $c : \mathbb{R}^M \to \{1, \dots, Z\}$. For typical settings, a classifier aims at minimising the classification error

$$\mathbb{E}(c) := \int \mathbb{L}(c(\mathbf{x}), y) dP(\mathbf{x}, y) \ .$$

With $\mathbb{L}(c(\mathbf{x}), y)$ denoting the 0-1-loss function

$$\mathbb{L}(c(\mathbf{x}), y) := \begin{cases} 0 & \text{if } c(\mathbf{x}) = y \\ 1 & \text{otherwise} \ . \end{cases}$$

Since P is typically unknown, standard classification schemes often optimise the empirical error (2.1) instead, or a related, numerically simpler (e. g., convex)

surrogate loss.

$$\hat{\mathbb{E}}(c, X) := \frac{1}{N} \sum_{1 \leq i \leq N} \mathbb{L}(c(\mathbf{x}_i), y_i) \tag{2.1}$$

For popular classifiers, results from computational learning theory guarantee that the empirical error allows us to uniformly bound the true error for i. i. d. data and a proper regularisation (Tewari and Bartlett, 2007).

A multi-class classifier can be extended by a reject option after training (Fig. 2.2). Hence, we assume a trained classifier is given. In addition to the output class, many classifiers provide a certainty measure of its classification like the class probability or the distance to the decision border. This fact is used whenever classification is extended by a reject option. Points with a bad certainty value are rejected since we consider reject options based on certainty measures where a higher value indicates higher certainty. Formally, a reject option extends the classifier to a mapping (denoted with the same symbol) $c : \mathbb{R}^M \to \{1, \ldots, Z, ⓡ\}$, where the symbol ⓡ denotes the rejection of the classification of input $\mathbf{x}$ which is typically defined by an extended 0-1-loss function

$$\mathbb{L}(c(\mathbf{x}), y) := \begin{cases} 0 & \text{if } c(\mathbf{x}) = y \\ b & \text{if } c(\mathbf{x}) = ⓡ \\ 1 & \text{if } c(\mathbf{x}) \neq y,\ c(\mathbf{x}) \neq ⓡ\,, \end{cases} \tag{2.2}$$

where costs $b \in (0, 1)$ are assigned to a reject ⓡ (Bartlett and Wegkamp, 2008). As example, for driver assistance systems it is better to reject an uncertain decision instead of a classification which initiates a wrong interaction of the system possibly causing severe effects.

The definition (2.2) is similar to the early definition used by Chow (1970) who proposed an optimal rejection strategy in the sense of error reject trade-off when the true class probabilities are known. For values $b < 1$, it is beneficial to reject a wrong classification rather than to provide a false output, but rejects always come at the risk of rejecting correctly classified points as well. Hence, the threshold (or threshold vector, discussed later on) starting from which rejection is done, is a crucial parameter. For settings with known class probabilities the optimal threshold can be obtained according to Chow (1970), but in general this information is unavailable. It is crucial to find an answer to the question how to choose thresholds which optimise the modified classification error. While threshold optimisation is straightforward in the case of one global threshold, the optimisation problem is more difficult for local rejection strategies as proposed in Fumera et al. (2000) and it will be discussed in chapter 5.

In the following, the basic concept and differences between global and local rejection are explained.

Global Rejection

Given a certainty measure:

$$r : \mathbb{R}^M \to \mathbb{R}, \mathbf{x} \mapsto r(\mathbf{x}) \ ,$$

a data point $\mathbf{x}$ and a threshold $\theta \in \mathbb{R}$, a simple reject option is to reject $\mathbf{x}$ iff

$$r(\mathbf{x}) < \theta \ .$$

If a data point is rejected, no classification takes place and the decision is postponed. The data point is marked for further treatment. In most classical rejection strategies one global threshold value is taken, and an optimal value depends on the respective costs of misclassification versus reject. Since the threshold θ is chosen uniformly for the whole input space, we denote such a reject option as *global reject option*. This implicitly assumes an equally scaled measure across the input space. For a true probability two conditions hold: (i) it is normalised, and (ii) equal probability values lead to an identical uncertainty of the classification, independently of the location of the related data points. We say a certainty measure is equally scaled when both conditions hold. Though we will see promising results in section 4.4, we can construct situations where certainty measures violate these conditions (see section 5.4.2). Local threshold strategies relax this assumption.

As Vailaya and Jain (2000) mentioned, uncertainty can have two different reasons: data points being outliers, or data points being located in ambiguous regions. An optimal reject option would reject exclusively wrongly classified data points (Fig. 2.3). Of course in practice no certainty measure can fulfil this.

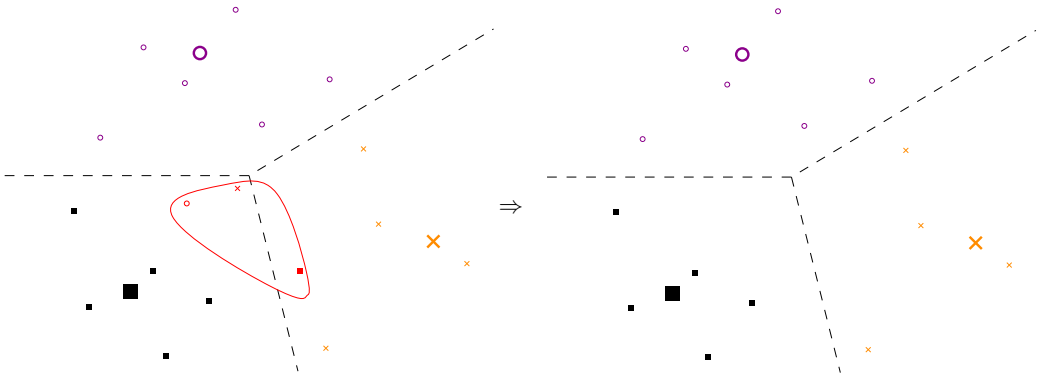


Figure 2.3.: Optimal reject option – Sketch of an artificial three-class setting (different symbols). The bigger symbols are the prototypes of the classifier. Left: no rejection; An optimal reject option rejects the red encircled errors only. Right: reject option is applied.

Local rejection strategies offer a finer grained control of rejection and the basic concept is explained in the following.

Local Rejection

A local threshold strategy relies on a given partition of the input space $\mathbb{R}^M$ into ζ disjunct, non-empty sets Υ_j such that

$$\mathbb{R}^M = \bigcup_{1 \leq j \leq \zeta} \Upsilon_j \, .$$

Using a different threshold θ_j in every set Υ_j enables a finer control of rejection (Vailaya and Jain, 2000). A separate threshold $\theta_j \in \mathbb{R}$ is chosen for every set Υ_j, and the reject option is given by a threshold vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_\zeta)$ of dimension ζ equal to the number of sets in the partition. A data point $\mathbf{x}$ is rejected iff

$$r(\mathbf{x}) < \theta_j \quad \text{where } \mathbf{x} \in \Upsilon_j \, .$$

Hence, θ_j determines the behaviour for the set Υ_j only. In the special case of one region Υ_j per classifier output class j, local thresholds realise class-wise rejection.

Figure 2.4 shows an instance of this local rejection strategy performing optimally if only labelling errors are rejected. In general this is not the case for local/global rejection strategies as mentioned before.

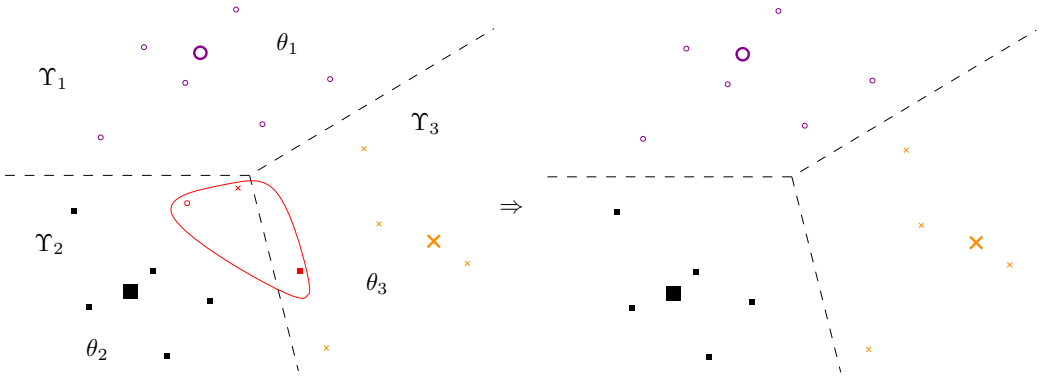


Figure 2.4.: Optimal local reject option – Sketch of an artificial three-class setting (different symbols) with an prototype-based model. The bigger symbols are the prototypes of the classifier. It is the same plot as Fig. 2.3. New: There are three partitions Υ_j and each one is associated with a single threshold θ_j. Left: Model without rejection, three encircled points are errors. Right: Model with optimal rejection since the errors are rejected only.

Comparing global and local rejection strategies, the choice of the global threshold θ or threshold vector $\boldsymbol{\theta}$ is crucial.

2.2. Evaluation of Reject Options

Nadeem et al. (2010) introduced accuracy-reject-curves (ARC) for evaluating

rejection strategies. An ARC is defined as follows: For a given global threshold θ the data X decompose into two sets $X = X_\theta \cup \mathcal{X}_\theta$. The set $\mathcal{X}_\theta$ contains rejected data points and X_θ contains accepted data points. For an increasing threshold θ starting from no reject (original model: $\theta = \min_i\{r(\mathbf{x}_i)\}$) to full reject ($\theta = \max_i\{r(\mathbf{x}_i)\}$, no data point is classified) the cardinality of $\mathcal{X}_\theta$ increases whereas the cardinality of X_θ decreases. In the ARC (Fig. 2.5), the relative size of $|X_\theta|/|X|$ ($t_a(\theta)$) versus the accuracy on X_θ ($t_c(\theta)$) is reported by means of a variation of the threshold θ in the interval $[\min_i\{r(\mathbf{x}_i)\}, \max_i\{r(\mathbf{x}_i)\}]$. Hence, an ARC consists of pairs $(t_a(\theta), t_c(\theta))$ which report the ratio of classified points (starting from a ratio 1 down to 0) versus the obtained accuracy for the classified points. An ARC can behave in three ways when increasing the threshold θ' to θ:

- increasing: rejection of more errors than correct classified data;
- constant: rejection of the same number of errors and correct classified data;
- decreasing: rejection of more correct classified data than errors.

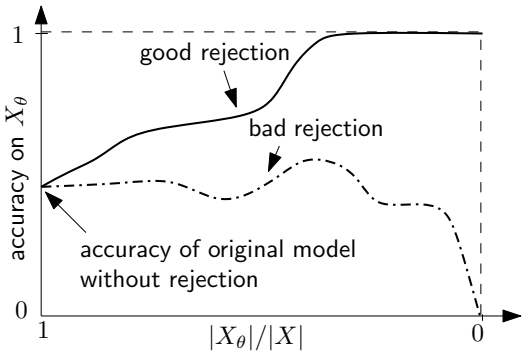


Figure 2.5.: Examples of possible accuracy-reject-curves. The shape of the solid black line (increasing, as fast as possible) is desired while the dash dotted shape is not desired.

Finding an optimal threshold (vector) for rejection refers to two contradicting objectives: A threshold θ or threshold vector $\boldsymbol{\theta}$ should be chosen such that the rejection of errors (*true rejects*) is maximised, while the rejection of correctly classified points (*false rejects*) is minimised. All optimal solutions/thresholds define a Pareto front, which can be formalised in the following way.

Pareto Front

Assume a labelled data set X ($|X| = N$) is given to determine the optimal local thresholds. A classifier decomposes X into a set of correctly classified data points

L and a set of wrongly classified data points (errors) E, i. e., $X = L \cup E$. These sets split with respect to the partition[1] Υ_j of the space into

$$L_j := L \cap \Upsilon_j \text{ and } E_j := E \cap \Upsilon_j, \quad j = 1, \ldots, \zeta \ .$$

An optimal reject option would reject all points in E, while classifying all points in L. This is usually impossible using local or global rejection. Applying a global threshold θ, the data set X decomposes into a set of rejected data points $\mathcal{X}_\theta$ and a set of accepted data points X_θ, i. e. $X = \mathcal{X}_\theta \cup X_\theta$. We refer to *false rejects* as

$$\mathcal{L}_\theta = \mathcal{X}_\theta \cap L$$

and to *true rejects* as

$$\mathcal{E}_\theta = \mathcal{X}_\theta \cap E \ ,$$

i. e., $\mathcal{X}_\theta = \mathcal{L}_\theta \cup \mathcal{E}_\theta$. Note that, when increasing θ, the cardinality of $|\mathcal{X}_\theta|$, $|\mathcal{L}_\theta|$, and $|\mathcal{E}_\theta|$ is monotonically increasing.

Similarly, for a threshold vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_\zeta)$, we denote the points in Υ_j which are rejected as $\mathcal{X}_{\theta_j}$, and the accepted points in Υ_j as X_{θ_j}. This relates to false rejects

$$\mathcal{L}_{\theta_j} = \mathcal{X}_{\theta_j} \cap L_j \text{ for } \Upsilon_j, \quad j = 1, \ldots, \zeta$$

and true rejects

$$\mathcal{E}_{\theta_j} = \mathcal{X}_{\theta_j} \cap E_j \text{ for } \Upsilon_j, \quad j = 1, \ldots, \zeta \ .$$

The false and true rejects are obtained as a union of these sets:

$$\mathcal{L}_{\boldsymbol{\theta}} = \bigcup_{1 \le j \le \zeta} \mathcal{L}_{\theta_j}, \quad \mathcal{E}_{\boldsymbol{\theta}} = \bigcup_{1 \le j \le \zeta} \mathcal{E}_{\theta_j} \quad \text{and} \quad X_{\boldsymbol{\theta}} = \bigcup_{1 \le j \le \zeta} X_{\theta_j} \ .$$

As for global rejection, monotonicity holds for the size of $\mathcal{X}_{\theta_j}$, $\mathcal{E}_{\theta_j}$, and $\mathcal{L}_{\theta_j}$ when raising the local threshold θ_j for partition Υ_j.

As mentioned before, the performance measure is an ARC (Nadeem et al., 2010). A given threshold θ leads to the accuracy of the classified points

$$t_a(\theta) := \frac{|L \backslash \mathcal{L}_\theta|}{|X_\theta|}$$

[1] In case of global rejection, there exists only one partition which equals the input space.

versus the ratio of the classified points

$$t_c(\theta) := \frac{|X_\theta|}{|X|} \ .$$

These two measures quantify conflicting objectives with limits $t_a(\theta)=1$ and $t_c(\theta)=0$ for large θ (all points are rejected) and $t_a(\theta)=|L|/|X|$ and $t_c(\theta)=1$ for small θ (all points are classified, the accuracy equals the accuracy of the given classifier for the full data set). The same quantities can be defined for a threshold vector $\boldsymbol{\theta}$. Note that a large number of optimised thresholds can lead to over-fitting. For simplicity, we refer to a single threshold or a threshold vector as threshold θ in the following. The aim of an optimisation of θ is maximising the value t_a, and minimising t_c. Hence, not all possible thresholds and correlated pairs $(t_a(\theta), t_c(\theta))$ are of interest, only optimal choices related to the so-called Pareto front. Note that pairs $(|\mathcal{L}_\theta|, |\mathcal{E}_\theta|)$ uniquely correspond to pairs $(t_a(\theta), t_c(\theta))$ and vice versa.

Every threshold uniquely induces a pair $(|\mathcal{L}_\theta|, |\mathcal{E}_\theta|)$ and a pair $(t_a(\theta), t_c(\theta))$. We say that θ' *dominates* the choice θ if $|\mathcal{L}_{\theta'}| \leq |\mathcal{L}_\theta|$ and $|\mathcal{E}_{\theta'}| \geq |\mathcal{E}_\theta|$ for at least one term, inequality holds. We aim at optimal rejection thresholds at the *Pareto front*

$$\mathcal{P}_\theta := \{(|\mathcal{L}_\theta|, |\mathcal{E}_\theta|) |\ \theta \text{ is not dominated by any } \theta'\} \ .$$

A dominated threshold (threshold vector) corresponds to a sub optimal choice: One can increase the number of true rejects without increasing the number of false rejects, or, conversely, false rejects can be lowered without lowering true rejects.

Optimal Thresholds for Given Rejection Costs

In the preceding section, we defined the Pareto front rather than a single threshold which is optimised according to the extended empirical risk $\hat{\mathbb{E}}(c, X)$ (2.1) for given rejection costs b. This has the benefit that, for *any* $b \in (0, 1)$, an optimal threshold can be extracted from the Pareto front due to the following relation: assume a threshold $\theta \in \mathcal{P}_\theta$ is chosen; using the notation from above, we can restate

$$\hat{\mathbb{E}}(c, X) = \frac{1}{N} \cdot (|E| - (1 - b) \cdot |\mathcal{E}_\theta| + b \cdot |\mathcal{L}_\theta|) \ . \tag{2.3}$$

Hence, the optimal threshold θ for rejection costs b is given by

$$\theta_{\text{opt}}(b) = \arg\max_\theta \left(|\mathcal{E}_\theta| - \frac{b}{1-b} \cdot |\mathcal{L}_\theta| \right) \ , \tag{2.4}$$

what can be extracted from the Pareto front. This enables a user to pick the optimal thresholds according to emerging rejection costs without a new optimisation.

The basic concepts and formalisations of global and local rejection are completed and in the following we give an overview on state of the art approaches.

2.3. State of the Art Approaches

This section summarises the state of the art for rejection strategies and accompanying certainty measures in supervised learning. The topic of reject options is also known as selective classification (El-Yaniv and Wiener, 2010). Vailaya and Jain (2000) highlight two main reasons for rejection: ambiguity and outliers. There exist several approaches explicitly addressing one of these reasons or a combination of both. Mostly, reject options are based on a measure which provides a certainty value about whether a given data point is correctly classified or not. In the following, we distinguish probabilistic and deterministic approaches.

Probabilistic approaches: Common certainty measures are based on probabilities. As already mentioned, Chow (1970) proposed optimal reject options, given the true probability density function is known. In this case, global rejection is an adequate strategy. A local strategy offers no benefit compared to a global one in such a setting. Chow's rule can therefore serve as a baseline provided that this ground truth is available. In general this is not the case and there are many approaches which use estimated class probabilities for rejection instead. There are two main ways to get those. Either one uses a probabilistic classifier providing an internal estimation of the probabilities, e. g., Bayes classifier, or the estimation is done in addition to a non-probabilistic classifier.

Hansen et al. (1994) prove that in the limit case, the rejection strategy (Chow, 1970) provides a bound for any other measure in the sense of the error-reject trade-off and they provide illustrative examples. They also extend Chows's rule to near optimal classifiers on finite data sets. The authors introduce a general scaling approach to compare error-reject curves of several independent experiments even with different classifiers or data sets. Herbei and Wegkamp (2006) link the work of Chow to a regression function and they provide bounds for the performance of rejection depending on the quality of the probability estimates. They further extend the formal framework of Chow towards the two possible errors in binary classification which is particularly important in medical studies where classifying an ill patient as healthy is worse than vice versa. Santos-Pereira and Pires (2005) propose a generalisation of Chows's rule towards different class conditional costs of wrong classifications, too. They also link rejection with the so-called receiver operating characteristic. Fumera et al. (2000) directly builds on Chow (1970) and they state that class-related rejection thresholds work better than a global one in case of estimated class probabilities. This effect is caused by the difference

between the original and the estimated probabilities which leads to shifted class borders. Class-related thresholds can be used in order to balance this effect.

Due to this theoretical background outlined above, many approaches follow the concept to empirically estimate the data distribution first. Often, Gaussian mixture models (GMM) are used for this purpose (Devarakota et al., 2006; Vailaya and Jain, 2000). Devarakota et al. (2006) extend a GMM to estimate the insecurity of a particular class membership for novel, previously unseen patterns of a new class; this estimation can lead to a reliable outlier reject option. Vailaya and Jain (2000) investigate the suitability of GMMs for both, rejection of outliers and ambiguous data for prototype-based models. In particular, they propose an efficient strategy on how to determine suitable rejection thresholds in these cases. The reliable estimation of GMMs is particularly problematic for high dimensional data. Therefore, Ishidera et al. (2004) propose a suitable approximation of the probability density function for high dimensionality, which is based on a low-dimensional projection of the data.

In case of non-probabilistic classifiers, there are two main options. Either one uses a probabilistic counterpart of the desired algorithm (e. g., the Bayes point machine (Herbrich et al., 2001) instead of a SVM or the robust soft LVQ (Seo and Obermayer, 2003) instead of distance-based LVQ variants) or one does a probabilistic modelling of the data in a post-processing step. Both ways provide estimated class probabilities which are usable for rejection. A third option is, to turn deterministic measures which are available in deterministic classifiers, e. g., distances, into probability estimates.

Turning deterministic measures into probabilities: The following methods turn deterministic measures into estimated probabilities such that rejection on these estimates is possible. Platt (1999) proposed a nowadays popular approach to turn the activity of a binary SVM into an approximation of a classification confidence. The certainty measure is based on the distance of a data point to the decision border, i. e., the activation of an SVM classifier. By means of a sigmoid function, the distance is transformed into a confidence value. The parameters of the sigmoid are fitted on the given training data. A transfer of this method for multi-class tasks is provided by Wu et al. (2004) and it is implemented in the LIBSVM toolbox (Chang and Lin, 2011). A newer approach (Langford and Zadrozny, 2005) is based on the so-called probing reduction which is more general than Platt's approach. The authors also guarantee that a low error rate in the binary classification task implies accurate probability estimates. A similar approach by Gao and Tan (2006) turns output scores from outlier detection algorithms into probabilities. The benefit of this method is that it is unsupervised and that the estimated probabilities perform better than the pure scores.

There exist settings where estimated class probabilities are unavailable or where their estimation is undesired. Focussing on such settings there are many heuristic approaches, e. g., distance-based measures.

Heuristic approaches: As an alternative, deterministic reject options have been proposed, directly addressing extensions of the 0-1-classification loss towards a reject option (2.2). Many rejection strategies base the reject option on a geometric alternative such as the distance to the decision border (e. g., Alvarez et al., 2007; Hu et al., 2009). For k-nearest neighbour classifiers (k-NN, Cover and Hart, 1967) a variety of simple certainty measures exist using the neighbourhood of a given data point (Delany et al., 2005; Hu et al., 2009). These measures rely on the correlation of the label of the data point and its neighbours (Fig. 2.6). In these approaches, several different realisations and combinations of neighbourhood statistics have been compared, resulting in an ensemble measure largely raising the stability of the single measures. Sugiyama and Borgwardt (2013) focus on effective outlier detection, relying on the distances of a new data point from elements of a randomly chosen subset of the given data. An outlier score is then given by the smallest distance. The resulting approach outperforms state of the art approaches such as proposed by Ramaswamy et al. (2000) in efficiency and accuracy. Zhang (2013) provides a survey with a large collection of outlier detection approaches. Sousa and Cardoso (2013) introduce a reject option identifying ambiguous regions in binary classifications. Their approach is based on a data replication method with the advantage that no rejection threshold has to be set externally, rather the approach itself provides a suitable cut-off. Stefano et al. (2000) address different neural network architectures including multi-layer perceptrons, learning vector quantisation, and probabilistic neural networks. Here an effectiveness function is introduced taking different costs for rejection and classification errors into account, very similar to the loss function as considered in Chow (1970) and Herbei and Wegkamp (2006). Also, different certainty measures based on the activation of the output neurons are studied.

These approaches deal often with a single rejection threshold and mostly with

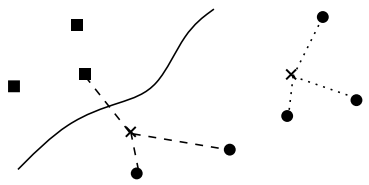


Figure 2.6.: Sketch of a 3-NN rejection strategy. Different symbols indicate different classes. Classification of the left data point ($\times$) is more uncertain than that of the right one ($\times$) since all neighbours are of the same class for the latter.

two-class classification only. Lately, extensions to more general settings such as multi-label classification (Pillai et al., 2013a,b) and multiple classes (Tax and Duin, 2008; Ramaswamy et al., 2015; Capitaine, 2014) have been considered.

Integrated rejection strategies: If the costs for wrong classifications and rejects are identified beforehand, this information and the reject option can be integrated in the classification model itself. Such models are based on the loss function (2.2) or approximations thereof. E. g. for dissimilarity-based options, rejection can be added a posteriori to the classifier, or it can already be taken into account while training (Villmann et al., 2015, 2016). Their approaches also provide an adapted threshold for rejection. The cost function of both classifiers integrates costs for rejected and wrongly classified data. While Villmann et al. (2016) focus on outlier rejection, Villmann et al. (2015) focus on rejection due to ambiguity. Fumera and Roli (2002) proposed one of the first embedded rejection strategies for SVM. Alternative formulations which approximate the 0-1-loss by a convex surrogate and which also prove the validity of this approach have been suggested (Grandvalet et al., 2008; Bartlett and Wegkamp, 2008; Yuan and Wegkamp, 2010). We are not focussing on such approaches because we assume settings without a priori information about costs for errors and rejects, and settings with dynamic costs due to, e. g., user interaction or concept drift.

A taxonomy for rejection (Fig. 2.7): In order to provide an overview of the aforementioned approaches, we grouped some of them into a rough taxonomy. We distinguish between probabilistic and deterministic approaches. The probabilistic rejection approaches are based on known probabilities or estimated ones. To obtain estimated probabilities one can simply use a probabilistic classifier which provides the probability internally or one can estimate the probabilities in a post-processing step independently of the used classifier. For deterministic approaches, we distinguish between embedded rejection, which is integrated in the classifier itself and optimised during training and approaches which work as a post-processing step. The latter approaches rely also on the classifier but parameters of the reject option are not optimised while training the classifier itself. In our taxonomy only the approaches of Vailaya and Jain (2000) and Fumera et al. (2000) provide local rejection.

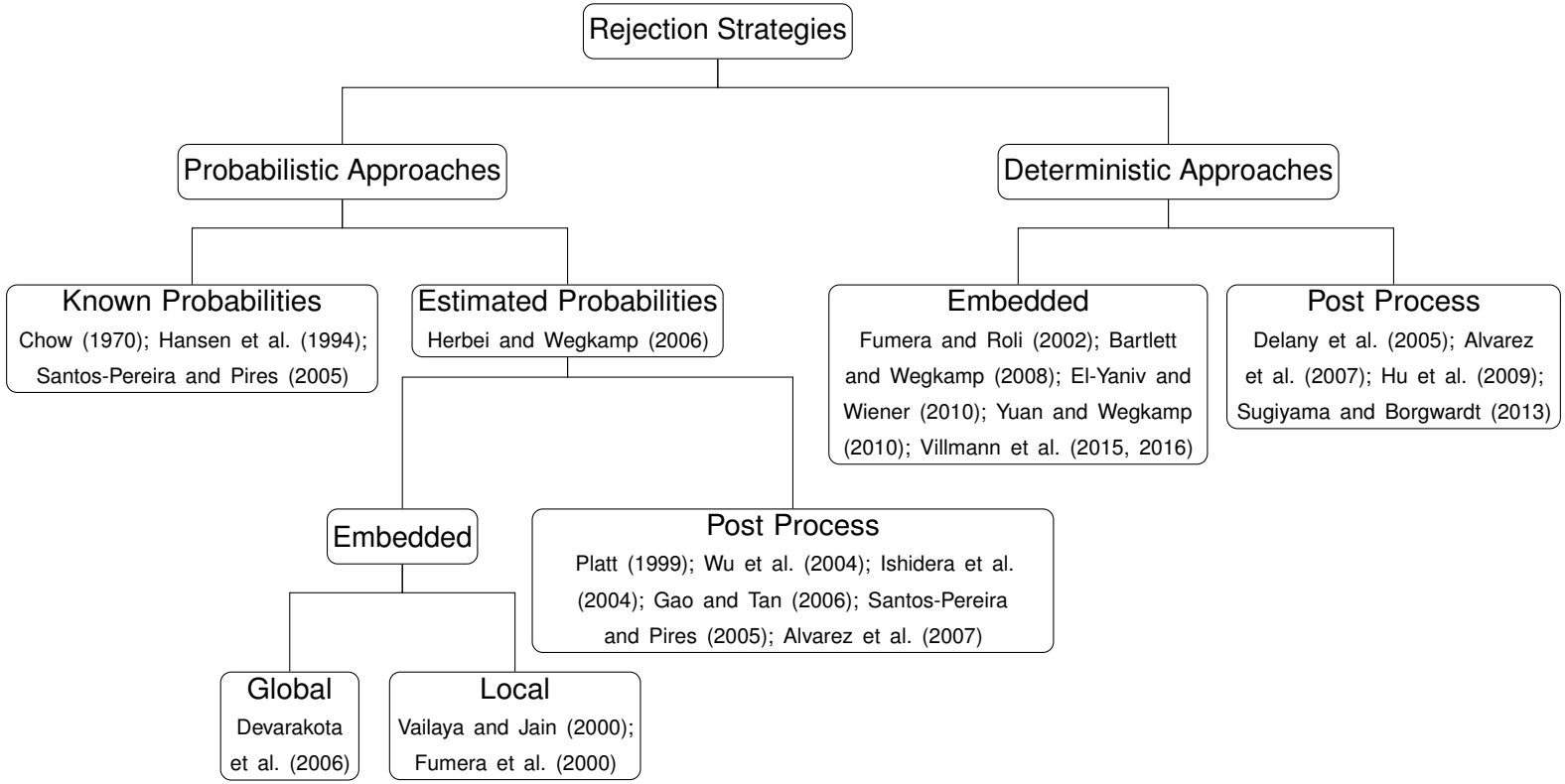


Figure 2.7.: Our taxonomy for rejection

2.4. Conclusion

Summing up the state of the art leads to the following findings: In settings with known class probabilities the problem of finding the right certainty measure and the optimal global threshold is solved by Chow (1970). Many approaches estimate the class probabilities in settings where they are unavailable. Those estimations are defective to varying degrees. The main shortcoming is, that it is difficult to determine the precision of the estimated class probabilities. It turns out that local thresholds are beneficial in scenarios with defective estimations (Fumera et al., 2000). The other large group of approaches operates with deterministic certainty measures, e. g., distance-based ones. There are well studied measures available, e. g., for SVM, but there are only a few simple approaches suited for prototype-based classifiers. Often it is difficult to set an appropriate threshold for rejection since the deterministic measures, e. g., as the distance to the decision border, may not be normalised. It is even more complicated if local thresholds have to be chosen. The challenge is to determine the best possible thresholds in order to optimise the error reject trade-off.

In the next chapter, we introduce the prototype-based classifiers first, which we use later on. Afterwards, we report our research results on rejection strategies.

3. Prototype-based Classification

Chapter overview *In this chapter we first motivate why we focus on prototype-based classifiers. A main advantage of those classifiers is their sparsity and that they allow a meaningful interpretation. Second we describe the used classifiers which belong to the popular family of LVQ approaches and we give a brief overview of their development.*

A prototype-based classifier stores prototypes for different classes. Hence such a classifier consists of a set W of ξ prototypes $\mathbf{w}_j \in \mathbb{R}^M$ where every prototype $\mathbf{w}_j$ has a class label $c_j \in \{1, \ldots, Z\}$. A data point $\mathbf{x} \in \mathbb{R}^M$ gets the same label c_l as its closest prototype (nearest neighbour, NN) $\mathbf{w}_l$ with

$$l = \arg \min_{j=1,\ldots,\xi} d(\mathbf{w}_j, \mathbf{x}) \tag{3.1}$$

where $d(\cdot)$ is a dissimilarity measure; e. g., the Euclidean distance. The closest prototype $\mathbf{w}_l$ is called the best matching unit. By means of the rule (3.1), a prototype-based classifier partitions the data into *Voronoi cells* or *receptive fields*

$$V_j = \{\mathbf{x} \,|\, d(\mathbf{w}_j, \mathbf{x}) \leq d(\mathbf{w}_k, \mathbf{x}), \forall\, k \neq j\},\ j = 1, \ldots, \xi\ .$$

The classification in a Voronoi cell V_j is constant and defined by the representing prototype $\mathbf{w}_j$.

A very basic approach is the nearest neighbour classifier (NNC) which uses the training data points as prototypes. The benefits of such a prototype-based classifier are:

- Prototypes $\mathbf{w}_j$ are elements of the same space as the data (for NNC they equal the data points) which make them interpretable and understandable.
- The classification scheme is simple and understandable.

But there are disadvantages as well since the number of prototypes in a NNC equals the number of training data points, hence the model is very complex. Kohonen (1989) proposed learning vector quantisation (LVQ) which aims at a sparser prototype-based model (Biehl et al., 2009). There the number of prototypes is predefined and the LVQ relies on the Hebbian learning paradigm (Kohonen, 1989). Although LVQ is a heuristic only, it achieves good results (Biehl et al., 2007). Due to its simple strategy and its low computational effort together with good performance,

LVQ is quite popular. In order to overcome instabilities and to achieve convergence guarantees, Sato and Yamada (1995) introduced the generalised LVQ (GLVQ) which is based on a suitable cost function. A probabilistic counterpart is the robust soft LVQ (RSLVQ) (Seo and Obermayer, 2003). Since the prototype models often focus on classification, the prototypes are discriminative. This means that they may not represent data in a generative way. Hammer et al. (2014) propose an approach enforcing generative prototypes. Due to the representation of models in terms of prototypes, LVQ schemes are suited for online scenarios (Denecke et al., 2009) or lifelong learning (Kirstein et al., 2012) as we will discuss in chapter 6.

The used dissimilarity measure of the prototype-based classifiers is a key aspect and the use of more general dissimilarity measures like e. g., divergences or functional metrics (Villmann and Haase, 2011) is easily possible. Hence a suitable dissimilarity measure for the data can be chosen, or the dissimilarity measure can be learned in addition to the prototypes because prototype-based classifiers provide a particularly efficient framework for integrating the powerful concept of metric learning (Bellet and Habrard, 2015; Bellet et al., 2013; Biehl et al., 2013b). For instance they offer efficient metric parametrisation strategies by their decomposition of the input space into Voronoi cells (see Schneider et al., 2009a,b). Recent LVQ schemes rely on a cost function and allow an extension to a global adaptive matrix: generalised matrix LVQ (GMLVQ) (Schneider et al., 2009a) and its local version (LGMLVQ) (Schneider et al., 2009a) with local adaptive matrices.

Prototype locations are usually learned on a given offline data set X with N data points $(\mathbf{x}_i, y_i) \in \mathbb{R}^M \times \{1, \ldots, Z\}$ of Z different classes. The aim is to find prototypes such that the induced classification of the data is as accurate as possible. A quality criterion is the *accuracy* which is the percentage of correctly classified data points of the given data. For the cost function based LVQ approaches there are guarantees on the generalisation performance and learning convergence of the related classifier (Biehl et al., 2007; Schneider et al., 2009a). The family of LVQ approaches has gained much attention recently in the biomedical domain (Arlt et al., 2011; Biehl et al., 2012) and in the context of big data and interpretable models due to its flexibility and intuitive classification scheme (see e. g., Kirstein et al., 2008, 2009, 2012; Bunte et al., 2012; Giotis et al., 2013; Biehl et al., 2013a, 2015; Nova and Estévez, 2014; Zhu et al., 2014; de Vries et al., 2015).

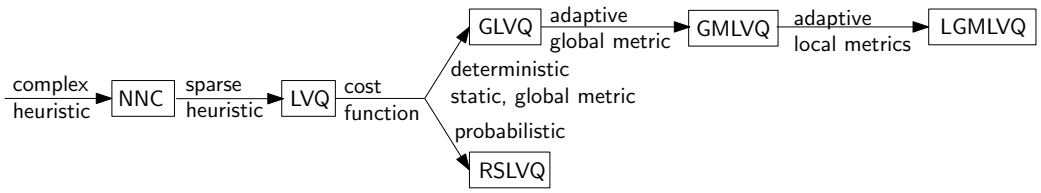


Figure 3.1.: Development of the learning vector quantisation methods used in this thesis

The sketch (Fig. 3.1) contains the relations between GLVQ, GMLVQ, LGMLVQ, and the RSLVQ which we mainly use and which we describe below.

Remark: Note that we refer to d as metric, although it may not be a metric in the mathematical sense. For instance, often the squared Euclidean distance is used which is not a metric in the mathematical sense since the triangular inequality does not hold for every given set of data points.

3.1. Generalised LVQ – GLVQ

A robust LVQ approach is the GLVQ (Sato and Yamada, 1995) based on a differentiable cost function approximating the classification error (3.2). It minimises:

$$E_{\text{GLVQ}}(X) = \sum_{1 \leq i \leq N} \Phi(\mu(\mathbf{x}_i)) \quad \text{with} \quad \mu(\mathbf{x}_i) = \left(\frac{d^+(\mathbf{x}_i) - d^-(\mathbf{x}_i)}{d^+(\mathbf{x}_i) + d^-(\mathbf{x}_i)} \right) . \tag{3.2}$$

The value $d^+(\mathbf{x}_i) = d(\mathbf{x}_i, \mathbf{w}^+)$ and $d^-(\mathbf{x}_i) = d(\mathbf{x}_i, \mathbf{w}^-)$ denotes the dissimilarity between a data point $\mathbf{x}_i$ and the closest prototype $\mathbf{w}^+$ belonging to the same class and the dissimilarity of the closest prototype $\mathbf{w}^-$ of any different class (see Fig. 3.2). The function $\Phi(\cdot)$ is a monotonic increasing function, e. g., the identity or the sigmoid function. The summands of (3.2) are negative if and only if the classification of the corresponding point is correct, hence the costs correlate to the overall error. In this way it optimises the so-called hypothesis margin of the classifier (Schneider et al., 2009a). Note that the LVQ cost function (3.2) approximates the loss (2.1), since it aims at minimising the empirical classification error as explained. Minimising E_{GLVQ} (3.2) is done by a stochastic gradient descent

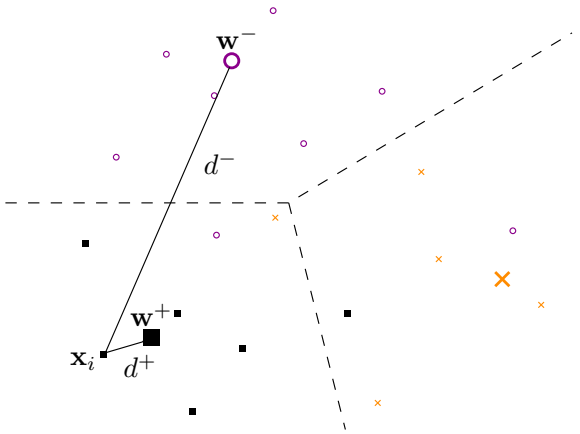


Figure 3.2.: Scheme of a three class setting (different colours, symbols). Bigger symbols are prototypes. For one data point $\mathbf{x}_i$ the quantities $\mathbf{w}^+, \mathbf{w}^-, d^+$ and d^- are displayed.

with respect to the prototypes $\mathbf{w}^+$ and $\mathbf{w}^-$, leading to the update

$$\mathbf{w}^{\pm} := \mathbf{w}^{\pm} - \varepsilon \cdot \frac{\partial E_{\text{GLVQ}}}{\partial \mathbf{w}^{\pm}} \tag{3.3}$$

with a learning rate $\varepsilon > 0$. Using the squared Euclidean distance $d(\mathbf{x}, \mathbf{w}) = \|\mathbf{x} - \mathbf{w}\|^2$, the derivation in the update rule (3.3) becomes

$$\frac{\partial E_{\text{GLVQ}}}{\partial \mathbf{w}^{\pm}} = \frac{\partial \Phi}{\partial \mu} \cdot \frac{\partial \mu}{\partial d^{\pm}} \cdot \frac{\partial d^{\pm}}{\partial \mathbf{w}^{\pm}}$$

$$\frac{\partial \mu}{\partial d^{\pm}} = \frac{\pm 2 d^{\mp}}{(d^+ + d^-)^2} \qquad \frac{\partial d^{\pm}}{\partial \mathbf{w}^{\pm}} = -2(\mathbf{x} - \mathbf{w}^{\pm})$$

Hence the GLVQ's learning rule can be specified as:

$$\mathbf{w}^{\pm} := \mathbf{w}^{\pm} \pm \varepsilon \cdot \frac{\partial \Phi}{\partial \mu} \cdot \frac{4 d^{\mp}}{(d^+ + d^-)^2} \cdot (\mathbf{x} - \mathbf{w}^{\pm}) \; .$$

One limitation of the GLVQ is the chosen dissimilarity measure. Often it is difficult to chose a well suited measure since one does not know which aspects of the data are relevant for classification a priori. More advanced models, e. g., GMLVQ (next section) and LGMLVQ (sec. 3.3), learn the prototypes and the metric parameters. These methods limit the general structure of the used dissimilarity measure but the related parameters are learned in order to improve the overall performance. Hence the learned measure stresses useful aspects of the data for classification.

3.2. Generalised Matrix LVQ – GMLVQ

The GMLVQ (Schneider et al., 2009a) is established on a cost function E_{GMLVQ} which is the same as (3.2) but it replaces the dissimilarity measure with

$$d_{\Lambda}(\mathbf{w}, \mathbf{x}) = (\mathbf{x} - \mathbf{w})^T \Lambda (\mathbf{x} - \mathbf{w}) \; , \tag{3.4}$$

a general quadratic form. The matrix $\Lambda = \Omega^T \Omega$ is positive semi definite and hence $d_{\Lambda}(\mathbf{w}, \mathbf{x}) = [\Omega(\mathbf{x} - \mathbf{w})]^2$. Since every positive symmetric Λ has a symmetric root Ω with $\Lambda = \Omega^2$, we will rely thereon in the following. The GMLVQ performs a stochastic gradient descent on E_{GMLVQ} with respect to the prototypes and with respect to the metric parameters. A similar metric learning concept exists for k-NN and nearest class mean classifiers (Bellet et al., 2013; Mensink et al., 2013) which are related to LVQ approaches.

A substitution of the dissimilarity measure (3.4) in the prototype update (3.3)

leads to a replacement of the used dissimilarity measure and its derivative

$$\frac{\partial d_\Lambda^\pm}{\partial \mathbf{w}^\pm} = -2\Omega\Omega(\mathbf{x} - \mathbf{w}^\pm) = -2\Lambda(\mathbf{x} - \mathbf{w}^\pm)$$

and hence

$$\frac{\partial E_{\text{GMLVQ}}}{\partial \mathbf{w}^\pm} = \frac{\partial \Phi}{\partial \mu} \cdot \frac{\partial \mu}{\partial d_\Lambda^\pm} \cdot \frac{\partial d_\Lambda^\pm}{\partial \mathbf{w}^\pm} = \frac{\partial \Phi}{\partial \mu} \cdot \frac{\pm 2 d^\mp}{(d^+ + d^-)^2}(-2\Lambda(\mathbf{x} - \mathbf{w}^\pm)) \; . \tag{3.5}$$

Therefore the prototype update is

$$\mathbf{w}^\pm := \mathbf{w}^\pm \pm \varepsilon_1 \cdot \frac{\partial \Phi}{\partial \mu} \cdot \frac{4 d_\Lambda^\mp}{(d_\Lambda^+ + d_\Lambda^-)^2} \cdot \Lambda(\mathbf{x} - \mathbf{w}^\pm) \; .$$

Additionally the single elements $\Omega(l,k)$ are learned through a stochastic gradient descent (3.6) on E_{GMLVQ} with respect to these elements.

$$\Omega(l,k) := \Omega(l,k) - \varepsilon_2 \cdot \frac{\partial E_{\text{GMLVQ}}}{\partial \Omega(l,k)} \tag{3.6}$$

To obtain the update rule for the elements of Ω we need the derivation

$$\frac{\partial d_\Lambda}{\partial \Omega(l,k)} = (\mathbf{x}(l) - \mathbf{w}(l)) \cdot [\Omega(\mathbf{x} - \mathbf{w})]_k + (\mathbf{x}(k) - \mathbf{w}(k)) \cdot [\Omega(\mathbf{x} - \mathbf{w})]_l$$

since it is required in the gradient (3.7) of the cost function with respect to $\Omega(l,k)$. The parameter l denotes the l-th component of the vector.

$$\frac{\partial E_{\text{GMLVQ}}}{\partial \Omega(l,k)} = \frac{\partial \Phi}{\partial \mu} \left(\frac{\partial \mu}{\partial d_\Lambda^+} \cdot \frac{\partial d_\Lambda^+}{\partial \Omega(l,k)} + \frac{\partial \mu}{\partial d_\Lambda^-} \cdot \frac{\partial d_\Lambda^-}{\partial \Omega(l,k)} \right) \tag{3.7}$$

Putting all parts of (3.6) together, leads to

$$\begin{aligned} \Omega(l,k) :=& \Omega(l,k) - \varepsilon_2 \cdot \frac{\partial \Phi}{\partial \mu} \cdot \frac{2}{(d_\Lambda^+ + d_\Lambda^-)^2} \cdot \\ & \Big(d_\Lambda^- \Big(\left[\Omega(\mathbf{x} - \mathbf{w}^+)\right]_k \left(\mathbf{x}(l) - \mathbf{w}^+(l)\right) + \left[\Omega(\mathbf{x} - \mathbf{w}^+)\right]_l \left(\mathbf{x}(k) - \mathbf{w}^+(k)\right) \Big) \\ & - d_\Lambda^+ \Big(\left[\Omega(\mathbf{x} - \mathbf{w}^-)\right]_k \left(\mathbf{x}(l) - \mathbf{w}^-(l)\right) + \left[\Omega(\mathbf{x} - \mathbf{w}^-)\right]_l \left(\mathbf{x}(k) - \mathbf{w}^-(k)\right) \Big) \Big) \; . \end{aligned}$$

The parameter ε_1 and ε_2 are the learning rates of the updates. They can be chosen independently but usually $\varepsilon_1 > \varepsilon_2$ holds. In order to prevent the algorithm from degeneration, the matrix Λ should be normalised (Schneider et al., 2009a). It is possible to use any optimisation solver instead of a stochastic gradient descent

(de Vries, 2013, 2014) but then it is no longer applicable in online settings.

The next section describes a straightforward extension of the GMLVQ towards local dissimilarity measures.

3.3. Localised Generalised Matrix LVQ – LGMLVQ

Attaching a local dissimilarity measure d_{Λ_j} (3.8) to every prototype $\mathbf{w}_j$ instead of using a global measure (3.4), leads to the LGMLVQ (Schneider et al., 2009a) which can stress local characteristics of the data that are beneficial for classification.

$$d_{\Lambda_j}(\mathbf{w}_j, \mathbf{x}) = (\mathbf{x} - \mathbf{w}_j)^T \Lambda_j (\mathbf{x} - \mathbf{w}_j) \tag{3.8}$$

Note that the Voronoi cells become more complex due to the usage of local metrics. The respective derivative of the dissimilarity measure (3.8) is

$$\frac{\partial d^{\pm}}{\partial \mathbf{w}^{\pm}} = -2\Omega_{\pm}\Omega_{\pm}(\mathbf{x} - \mathbf{w}^{\pm}) = -2\Lambda_{\pm}(\mathbf{x} - \mathbf{w}^{\pm}) \tag{3.9}$$

and the index $\pm$ refers to the local matrices belonging to $\mathbf{w}^{\pm}$. Using (3.9) in (3.5) leads to the prototype update of the LGMLVQ. The related matrices $\Lambda_{\pm}$ respectively $\Omega_{\pm}$ are again trained with a stochastic gradient descend, using

$$\frac{\partial E_{\text{LGMLVQ}}}{\partial \Omega_{\pm}(l,k)} = \frac{\partial \Phi}{\partial \mu} \cdot \frac{\pm 2 d^{\mp}}{(d^{+} + d^{-})^2} \cdot \Big(\left[\Omega_{\pm}(\mathbf{x} - \mathbf{w}^{\pm})\right]_k \left(\mathbf{x}\,(l) - \mathbf{w}^{\pm}\,(l)\right) + \left[\Omega_{\pm}(\mathbf{x} - \mathbf{w}^{\pm})\right]_l \left(\mathbf{x}\,(k) - \mathbf{w}^{\pm}\,(k)\right) \Big) .$$

Further information to (L)GMLVQ can be found in (Schneider et al., 2009a).

So far the LVQ approaches are based on deterministic non-generative cost functions. A probabilistic member of the LVQ family is described in the next section.

3.4. Robust Soft LVQ – RSLVQ

The objective function of RSLVQ (Seo and Obermayer, 2003) is a statistical generative modelling of the setting with the benefit of offering class probabilities for given data. Assume a Gaussian mixture model (GMM) creates the data. The probability of mixture component j generating a point $\mathbf{x}$ is

$$p(\mathbf{x} \,|\, j) = K(j) \cdot \exp(f(\mathbf{x}, \mathbf{w}_j)) = \frac{1}{(2\pi\sigma_j^2)^{\frac{M}{2}}} \cdot \exp\left(-\frac{d(\mathbf{w}_j, \mathbf{x})}{2\sigma_j^2}\right) \tag{3.10}$$

where $d(\cdot)$ denotes the squared Euclidean distance. This induces a mixture model

$$p(\mathbf{x} \mid W) = \sum_{1 \leq j \leq \xi} P(j) \cdot p(\mathbf{x} \mid j)$$

describing the probability of having observed the (unlabelled) data. The priors sum to one $\sum_j P(j) = 1$. Label information is integrated into the model by assigning every mixture component (i. e., every prototype) with a class label. Then the probability of having observed the labelled data is given by

$$p(\mathbf{x}, y \mid W) = \sum_{\substack{1 \leq j \leq \xi \\ c_j = y}} P(j) \cdot p(\mathbf{x} \mid j) \ .$$

The objective function of RSLVQ describes the log likelihood ratio of the observed data

$$\log L := \sum_{1 \leq i \leq N} \log \left(\frac{p(\mathbf{x}_i, y_i \mid W)}{p(\mathbf{x}_i \mid W)} \right)$$

which corresponds to the optimisation of the likelihood of the observed class labels assuming an underlying mixture model and independence of the data. Training optimises the log likelihood by means of a gradient ascend with respect to the prototypes

$$\mathbf{w}_l := \mathbf{w}_l + \varepsilon \cdot \frac{\partial (\log L)}{\partial \mathbf{w}_l} \tag{3.11}$$

with a learning rate $\varepsilon > 0$. The bandwidth σ_j is usually set identically for all mixture components, and it is treated as a meta-parameter. There exist schemes adapting the bandwidth additionally (Schneider et al., 2010a; Seo and Obermayer, 2006). Executing the derivation of (3.11) leads to

$$\begin{aligned} \frac{\partial}{\partial \mathbf{w}_l} \left[\log \frac{p(\mathbf{x}, y|W)}{p(\mathbf{x}|W)} \right] =& \delta(c_l = y) \cdot (P_y(l|\mathbf{x}) - P(l|\mathbf{x})) \cdot \frac{\partial f(\mathbf{x}, \mathbf{w}_l)}{\partial \mathbf{w}_l} \\ & - \delta(c_l \neq y) \cdot P(l|\mathbf{x}) \cdot \frac{\partial f(\mathbf{x}, \mathbf{w}_l)}{\partial \mathbf{w}_l} \end{aligned} \tag{3.12}$$

where $P_y(l|\mathbf{x})$ and $P(l|\mathbf{x})$ are assignment probabilities

$$P_y(l|\mathbf{x}) = \frac{p(l) \cdot \exp(f(\mathbf{x}, \mathbf{w}_l))}{\sum\limits_{\substack{1 \leq j \leq \xi \\ c_j = y}} p(j) \exp(f(\mathbf{x}, \mathbf{w}_j))}, \quad P(l|\mathbf{x}) = \frac{p(l) \cdot \exp(f(\mathbf{x}, \mathbf{w}_l))}{\sum\limits_{1 \leq j \leq \xi} p(j) \exp(f(\mathbf{x}, \mathbf{w}_j))} \tag{3.13}$$

and with respect to (3.10)

$$\frac{\partial f(\mathbf{x}, \mathbf{w}_l)}{\partial \mathbf{w}_l} = \frac{1}{\sigma^2} (\mathbf{x} - \mathbf{w}_l) \ . \tag{3.14}$$

Assuming $p(j) = 1/\xi,\ \forall j$ and using equations (3.10) to (3.14) for the stochastic gradient ascent (3.11) leads to the update rule for the prototypes

$$\mathbf{w}_l =:= \mathbf{w}_l + \varepsilon \cdot \begin{cases} (P_y(l|\mathbf{x}) - P(l|\mathbf{x})) \cdot (\mathbf{x} - \mathbf{w}_l), & c_l = y \\ -P(l|\mathbf{x}) \cdot (\mathbf{x} - \mathbf{w}_l), & c_l \neq y \end{cases}$$

with

$$P_y(l|\mathbf{x}) = \frac{\exp\left(-\frac{(\mathbf{x}-\mathbf{w}_l)^2}{2\sigma^2}\right)}{\sum\limits_{\substack{1\leq j\leq \xi \\ c_j = y}} \exp\left(-\frac{(\mathbf{x}-\mathbf{w}_j)^2}{2\sigma^2}\right)}, \qquad P(l|\mathbf{x}) = \frac{\exp\left(-\frac{(\mathbf{x}-\mathbf{w}_l)^2}{2\sigma^2}\right)}{\sum\limits_{1\leq j\leq \xi} \exp\left(-\frac{(\mathbf{x}-\mathbf{w}_j)^2}{2\sigma^2}\right)} \ .$$

This closes the chapter about the classifiers used later on. The next chapter deals with global rejection in the context of prototype-based classifiers which we introduced in this chapter.

4. Global Reject Option

Chapter overview *In this chapter, we discuss global rejection strategies which enable classifiers to reject data in regions where only an uncertain decision could be made. We analyse several dissimilarity-based certainty measures, e. g., a cost function based measure, the distance to the closest prototype, and the distance to the closest decision border, as counterparts to probabilistic ones, suited for prototype-based classifiers like LVQ. We study properties and results of these measures on various data sets. From the executed experiments, we draw the conclusion, that the measures perform well for the used prototype-based models except one measure. The performance of the measures keeps up with the performance of the optimal rejection in some settings and to a state of the art certainty measure. In addition, the analysed deterministic measures can perform equally good compared to integrated certainty measures of probabilistic classifiers independently of their type (generative, discriminative).*

Parts of this chapter are based on:

[J15] L. Fischer, B. Hammer, and H. Wersing. Efficient Rejection Strategies for Prototype-based Classification. *Neurocomputing*, 169 (2015) 334–342.

[C14a] L. Fischer, B. Hammer, and H. Wersing. Rejection Strategies for Learning Vector Quantization. In *ESANN*, pages 41–46, 2014.

[C14b] L. Fischer, D. Nebel, T. Villmann, B. Hammer, and H. Wersing. Rejection Strategies for Learning Vector Quantization – A Comparison of Probabilistic and Deterministic Approaches. In *WSOM*, pages 109–118, 2014.

4.1. Motivation

Improved sensor technology and the increasing availability of high quality digital information provide new possibilities for machine learning technology in high impact domains like personalised medicine (Weiss et al., 2012). In biomedical applications or safety-related domains, a wrong classification can severely affect the applicability of a classifier. The reliability of a classification constitutes a critical property of any method used in such domains (Rudin and Wagstaff, 2014; Vellido et al., 2012). In these areas, the reliability of classification results is as important as the accuracy of a classifier. It is often better to refuse the classification of a given data point rather than to predict a class with uncertain assignment (Hanczar and Dougherty, 2008). In case of doubt, data can then be analysed by a human expert or it can be marked for further tests instead of a direct, uncertain classification.

In the following we discuss several simple, efficient prototype-based reject options: We consider rejection based on the distance of the point to the classi-

fication border, the dissimilarity to the closest prototype d^+ as indication of the point being an outlier, a combination of both, and a simple direct measurement inspired by the GLVQ cost function, which we dub *relative similarity* (section 4.3). These certainty measures take ambiguous and outlier rejection into account to varying degrees. Further, the measures differ according to their scaling, allowing an uniform threshold θ iff $r(\mathbf{x})$ is normalised, and they differ according to their computational complexity and online computability, i. e., efficiency. Afterwards the sections introduce the used measures, discuss their properties, and results on several data sets in order to answer the research questions of section 4.2.

4.2. Research Questions

A common choice for a certainty measure is the estimated probability of a data point belonging to the assigned class of a classifier. For non-probabilistic classifiers, e. g., LVQ, it is often costly to get such an estimation. The next sections tackle the following questions related to the proposed certainty measures for prototype-based classifiers, focussing on LVQ approaches.

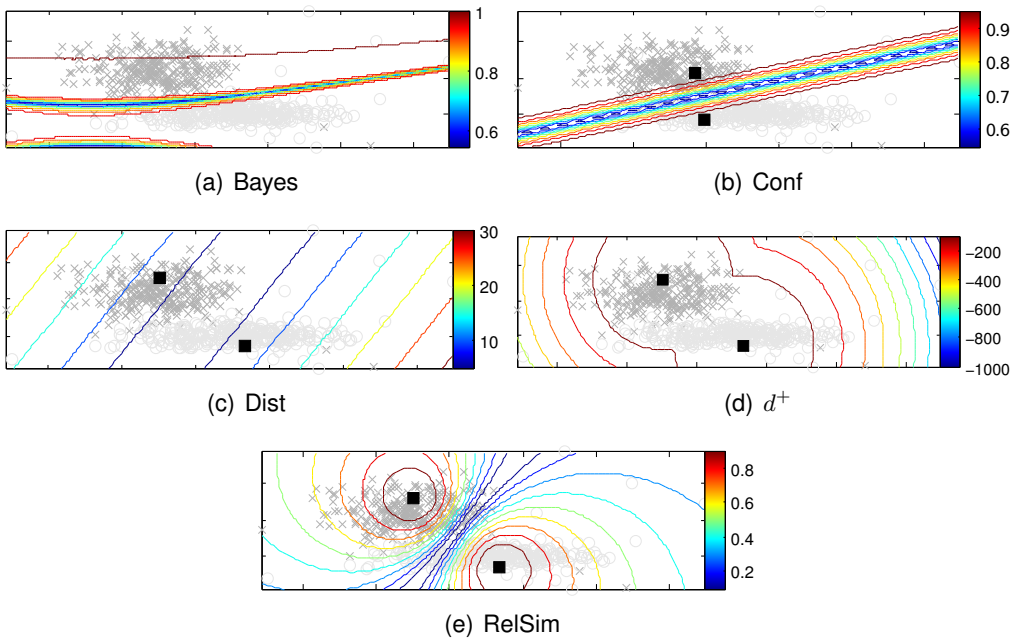
1. Are dissimilarity-based certainty measures which are easy to compute, suited for efficient rejection?
2. How do these measures perform in comparison to probabilistic counterparts?
3. Which general properties do they have?

4.3. Certainty Measures

Common choices for a certainty measure r are based on estimated probabilities or on heuristics. Probabilistic measures often either require a probabilistic classifier (Chow, 1970) or a probabilistic model on top of the trained classifier estimating the probabilities (Vailaya and Jain, 2000; Ishidera et al., 2004). Both approaches are computationally expensive. Heuristic measures can be based on distances (Suutala et al., 2004; Platt, 1999; Wu et al., 2004) or on the neighbouring class labels (Hu et al., 2009). In the following we use two probabilistic measures based on a probabilistic classifier and several heuristic measures based on dissimilarities.

Note that we use d as symbol for all dissimilarity measures. The definition of d for the different used algorithms, is displayed in the list below:

- GLVQ: $d(\mathbf{w}_j, \mathbf{x}) = (\mathbf{x} - \mathbf{w}_j)^T(\mathbf{x} - \mathbf{w}_j)$
- GMLVQ: $d(\mathbf{w}_j, \mathbf{x}) = (\mathbf{x} - \mathbf{w}_j)^T\Lambda(\mathbf{x} - \mathbf{w}_j)$
- LGMLVQ: $d_j(\mathbf{w}_j, \mathbf{x}) = (\mathbf{x} - \mathbf{w}_j)^T\Lambda_j(\mathbf{x} - \mathbf{w}_j)$ for each prototype $\mathbf{w}_j$



(a) Bayes (b) Conf

(c) Dist (d) d^+

(e) RelSim

Figure 4.1.: [C14a] Contour lines of the measures for artificial 2D data. This binary problem consists of data of two Gaussians (symbols: $\times, \circ$). Black squares are GLVQ/RSLVQ prototypes.

4.3.1. Bayes

The Bayes classifier provides class probabilities for each class provided that the data distribution is known. The reject option related to the certainty measure

$$r_{\text{Bayes}}(\mathbf{x}) = \max_{1 \leq z \leq Z} p(z|\mathbf{x}) \tag{4.1}$$

is optimal in the sense of an error-reject trade-off (Chow, 1970). In general, the class probabilities are unknown. Hence this optimal Bayes rejection can serve as Gold standard for artificially designed settings with ground truth only as in Fig. 4.1(a). It shows the contour lines of Bayes for an artificial two class problem with known class densities. The Bayes measure indicates ambiguous data.

4.3.2. Conf

Classifiers based on probabilistic models such as RSLVQ provide a confidence value of the classification:

$$r_{\text{Conf}}(\mathbf{x}) = \max_{1 \leq z \leq Z} \hat{p}(z|\mathbf{x}) \tag{4.2}$$

with the estimated probability $\hat{p}(\cdot)$ as obtained during training. We use Conf in settings with available probabilities during training. Hence these settings fall under the framework of plugin-rules as studied in (Herbei and Wegkamp, 2006). One problem is that it is often unclear how good the empirical estimation $\hat{p}(\cdot)$ resembles the underlying probability. This is especially problematic in supervised settings where the aim is often a good classification accuracy of the model rather than an exact estimation of the probabilities. Figure 4.1(b) shows the contour lines of Conf for an RSLVQ model trained on an artificial two class setting. Conf has low values near the class border and it realises a rejection due to ambiguity.

4.3.3. RelSim – The Relative Similarity

The RelSim is a GLVQ cost function (3.2) related measure. It takes the dissimilarity of the closest prototype $d^+(\mathbf{x})$ and the dissimilarity of the closest prototype of a different class $d^-(\mathbf{x})$ for a new unlabelled data point into account. The measure calculates values according to:

$$r_{\mathrm{RelSim}}(\mathbf{x}) = \frac{d^-(\mathbf{x}) - d^+(\mathbf{x})}{d^-(\mathbf{x}) + d^+(\mathbf{x})} \,. \tag{4.3}$$

Hence the closest prototype belonging to $d^+(\mathbf{x})$ defines the class label of this unlabelled data point if it is accepted. The relation $r_{\mathrm{RelSim}}(\mathbf{x}) = -\mu(\mathbf{x})$ holds for the function $\mu(\mathbf{x})$ of Sato and Yamada (1995) in the case of a GLVQ classifier. The measure ranges in the interval $(0, 1)$ where values near 1 indicate a certain classification and values near 0 are an indicator for uncertain class labels.

The values of $d^+(\mathbf{x})$ and $d^-(\mathbf{x})$ are already calculated by the used LVQ algorithms and therefore no additional computational costs are caused. Further, RelSim depends only on the stored prototypes W and the new unlabelled data point $\mathbf{x}$. The measure is well suited for online computation because it needs no additional storage. Figure 4.1(e) shows the contour lines of RelSim for an artificial two class problem with prototypes trained by GLVQ. The values near the class border are low. This means the measure correctly performs rejection due to ambiguity. In addition, as can be seen from the circular contour lines (squared Euclidean metric), a rejection of outliers is included. Therefore, RelSim seems a good compromise between an efficient measure and a richness of its usability.

4.3.4. Dist

The Dist considers the distance of a point to the closest decision border of the classifier as certainty measure. The distance of a point $\mathbf{x}$ to the hyperplane

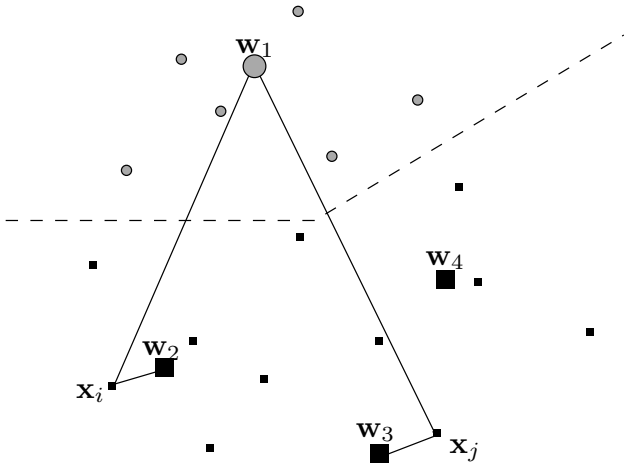


Figure 4.2.: Sketch of an artificial binary setting. The two decision borders are defined by the prototype pairs $(\mathbf{w}_1, \mathbf{w}_2)$ and $(\mathbf{w}_1, \mathbf{w}_4)$. Hence $r_{\mathrm{Dist}}(\mathbf{x}_i)$ states the exact distance value to the decision border while $r_{\mathrm{Dist}}(\mathbf{x}_j)$ is just an approximation due to the fact that the closest prototype $\mathbf{w}_3$ defines no decision border.

separating the receptive fields of $\mathbf{w}^+$ and $\mathbf{w}^-$ is given by

$$r_{\mathrm{Dist}}(\mathbf{x}) = \frac{|d^+(\mathbf{x}) - d^-(\mathbf{x})|}{2\|\mathbf{w}^+ - \mathbf{w}^-\|^2} \ . \tag{4.4}$$

Note the distance between the prototypes $\|\mathbf{w}^+ - \mathbf{w}^-\|^2$ has to be calculated as the distances $d^{\pm}(\mathbf{x})$, i. e., as $d(\mathbf{w}^+, \mathbf{w}^-)$ and d is the chosen measure in the used algorithm. Figure 4.1(c) shows the contour lines of Dist for an artificial two class problem with prototypes trained by the GLVQ. For settings with one prototype per class Dist calculates the exact distance (Fig. 4.2). In cases with more than one prototype per class, the underlying topology has to be estimated using e. g., Hebbian learning (Martinetz and Schulten, 1994). Then, (4.4) can be used for the pairs of prototypes that define the corresponding class border. An experimental evaluation has shown that the approximation of Dist based on $d^+(\mathbf{x}), d^-(\mathbf{x}), \mathbf{w}^+, \mathbf{w}^-$ (even if $\mathbf{w}^+$ and $\mathbf{w}^-$ do not define a class border) provides good results with less effort compared to the correct calculation. Hence, we always use this approximation, avoiding additional computational burden. Dist can be computed efficiently. Unfortunately its range is $(0, \infty)$ which makes it more difficult to choose a good threshold for rejection. Dist depends on the stored prototypes W, the distance calculation, i. e., $d^+(\mathbf{x}), d^-(\mathbf{x})$ and the new data point $\mathbf{x}$. This means no additional storage is needed and the needed values for the Dist calculation can be used directly without much additional computational effort.

In a binary setting with one prototype per class, the classifier defines one separating hyperplane. Dist uses the *pure* distance of a data point to the hyperplane

for rejection. This scheme is related to SVM rejection. A binary SVM defines a separating hyperplane too and its rejection is based on distances to its hyperplane. The SVM rejection (Platt, 1999) scales these distances with an adapted Sigmoid function which can cause a shifting of the decision border especially for unbalanced classes.

4.3.5. d^+

Outliers are indicated by their dissimilarity to the closest prototype $d^+(\mathbf{x})$ (Fig. 4.1(d)). Using this information for an outlier-based certainty measure leads to:

$$r_{d^+}(\mathbf{x}) = -d^+(\mathbf{x}) \ . \tag{4.5}$$

The measure $d^+(\mathbf{x})$ uses the stored prototypes W and the dissimilarity calculation. Hence this measure is efficient. Note, that the measure $d^+(\mathbf{x})$ is not normalised.

4.3.6. Comb

This measure combines the previous two reject options

$$r_{\mathrm{Comb}}(\mathbf{x}) = (r_{\mathrm{Dist}}(\mathbf{x}), r_{d^+}(\mathbf{x}))$$

leading to a reject option based on a threshold vector $\theta = (\theta_1, \theta_2)$: $\mathbf{x}$ is rejected iff

$$r_{\mathrm{Dist}}(\mathbf{x}) < \theta_1 \text{ or } r_{d^+}(\mathbf{x}) < \theta_2 \ .$$

This option takes ambiguity and outliers into account, but it requires two thresholds. For evaluation, we refer to the best combination of both thresholds determined via exhaustive search. This combination is inefficient since it requires a loop over the regime of threshold vectors, but it excellently serves as a baseline for comparison.

4.3.7. Characteristics of the Certainty Measures

In the following we compare some general properties of the measures which are listed in Tab. 4.1. SVM refers to the SVM rejection (Platt, 1999; Wu et al., 2004) implemented in the LIBSVM (Chang and Lin, 2011). The first row specifies requirements for online training of the rejection strategies based on these measures. The measures RelSim, d^+, Dist and Comb do only rely on a set of prototypes W, whereas the measures Conf and Bayes need class probabilities or its estimations. For the reject option as provided by an SVM, the training set needs to be stored since the parameters of the Sigmoid rely on the distances of training data to the separating hyperplane. Updating the SVM rejection because of new training data

requires a retraining of the parameters of the Sigmoid. In case of offline training and testing only, the SVM rejection requires only the trained parameters of the Sigmoid and for the other measures it stays the same.

The next property specifies the co-domains of the measures. This is particularly interesting since it indicates whether a natural predefined choice of the threshold θ is possible (for a normalised co-domain) or not (unlimited co-domain). Still, even for the same co-domains such as for RelSim and Conf, it is unclear whether their interpretation coincides and hence if similar thresholds have the same meaning.

The next property, comparable scaling, makes this more precise. It refers to the question whether the provided value displays the same range independent of the location of data points in the data space, or whether the scaling can severely change with different locations in the data space and e. g., in different Voronoi cells. In the latter case, it is likely that global threshold strategies do not provide satisfactory results, and local thresholds have to be used. Provided measurements refer to probabilities such as for Conf and Bayes, a uniform scaling is present. For all other measures (RelSim, d^+, Dist, Comb) a uniform scaling is not guaranteed since relative dissimilarities can vary severely across the input space.

The next two lines refer to the type of rejection offered by the measures: Do they detect outliers and/or ambiguous regions, respectively?

The final row contains an interesting property: Can the measures be used in online settings, i. e., is it computable based on a small number of parameters of the classifier? The proposed measures (RelSim, d^+, Dist, Comb, Conf) are suited for online settings because they only depend on the prototypes W. This means if the classifier is adapted online, the certainty measures take these changes immediately into account because they are based on dissimilarities from data to prototypes only. For Bayes and SVM rejection this is not possible in an online way although there exist online training schemes of these algorithms: The rejection strategies require the whole training data for its computation, hence an update of the certainty measure cannot be done in online settings with a limited amount of memory. Therefore a previously calculated certainty measure no longer fits to the permanently updated model of a Bayes or SVM classifier trained in an online way.

Property	RelSim	d^+	Dist	Comb	Conf	Bayes	SVM
Requirements	W	W	W	W	$\hat{p}(z\|\mathbf{x})$	$p(z\|\mathbf{x})$	data, model
Co-domain	$(0,1)$	$(-\infty,0)$	$(0,\infty)$	$(-\infty,\infty)$	$(0,1)$	$(0,1)$	$(0,1)$
Equal scaling	-	-	-	-	$\surd$	$\surd$	-
Outliers	$\surd$	$\surd$	-	$\surd$	-	-	-
Ambiguity	$\surd$	-	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
Online	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	-	-

Table 4.1.: Properties of the studied measures. The symbols indicate yes ($\surd$) and no (-).

4.4. Experiments for Global Rejection

After these theoretical inspections, we evaluate the results of the rejection strategies for different data sets. In all cases, we use a 10-fold cross-validation with ten repeats for RSLVQ, GLVQ, and (L)GMLVQ with one prototype per class[1]. Hence we average 100 runs for each experiment. We compare our results with the results of a standard certainty measure of SVM (Platt, 1999; Wu et al., 2004) implemented in the LIBSVM toolbox (Chang and Lin, 2011).

We use ARC as evaluation measure for our experiments, as explained in chapter 2. In Fig. 4.3 to 4.5, the ARC are averaged over 100 runs per data set and certainty measure. The data set X for evaluation is the test fold of the related cross validation. For numerical reasons, we omit the point for $|X_\theta| = 0$ where no point is classified. Since single curves can end in different points with maximal threshold value, we only report those points where at least 80 % of the repetitions deliver a value to ensure a reliable display.

4.4.1. Artificial and Benchmark Data

We report experiments on one artificial data set with known ground truth for the Bayes optimal rejection and four benchmarks. (In section A.2 the properties of all data sets of this thesis are listed in a table.)

Gaussian clusters: This data set contains two artificially generated overlapping 2D Gaussian clusters which are overlaid with uniform noise. (parameters: means at the x-axis $\mu_x = (-4, 4.5)$, means at the y-axis $\mu_y = (4, 0.5)$, standard deviations at the x-axis $\sigma_x = (5.2, 7.1)$, and standard deviations at the y-axis $\sigma_y = (2.5, 2.1)$)

Tecator data: The task is to predict the fat content of meat probes, which is turned into a binary classification problem to predict a high/low fat content by binning into two classes of equal size. The data set consists of 215 spectra with 100 spectral bands ranging from 850 nm to 1,050 nm (Thodberg, 1995).

Image Segmentation: The image segmentation data set consists of 2,310 data points representing small patches from outdoor images with 7 different classes with equal distribution such as brick-face, sky, ... (Bache and Lichman, 2013). Each data point consists of 19 real-valued image descriptors.

Haberman: The Haberman survival data set contains 306 instances from two classes indicating the survival for more than 5 years after breast cancer surgery

[1] We use the LVQ toolbox at: `http://matlabserver.cs.rug.nl/gmlvqweb/web/` and the RSLVQ code of Sambu Seo (Seo and Obermayer, 2003)

(Bache and Lichman, 2013). Data are represented by three attributes related to the age, the year, and the number of positive axillary nodes detected.

Coil: The Columbia Object Image Database Library (Coil) consists of grey scaled images (128x128 pixels) of twenty objects (Nene et al., 1996). The task is to classify each object. They are rotated in 5° steps, so that there are 72 images per object. The data set contains 1,440 data points which are 16,384 dimensional. We use the principal component analysis implementation of van der Maaten (2013) to reduce the dimensionality to 30.

4.4.2. Results

We report the effect of the different rejection strategies for the different classifiers: RSLVQ, GLVQ, and (L)GMLVQ. We combine RSLVQ with Conf since the former provides explicit probabilities. All classifiers can be combined with d^+, Dist, and Comb, since these measures depend on the provided prototypes only. For GLVQ and GMLVQ, the measure RelSim is already computed during the classification. For the artificial data set, the ground truth of the data distribution is available. Hence we can compare against the optimal Bayes decision.

Experiments on Artificial Data

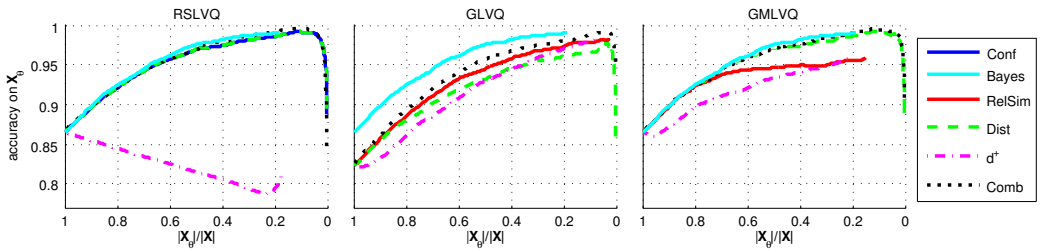


Figure 4.3.: [C14a] Accuracy-reject-curves for different reject options when applying RSLVQ, GLVQ and GMLVQ models trained on Gaussian clusters.

Figure 4.3 shows the results for the Gaussian clusters data set. Note that errors mostly stem from ambiguity in the overlapping region, such that a reject option due to outliers is less efficient for this setting. We observe that the probabilistic model RSLVQ together with its certainty measure Conf well resembles the optimal rejection strategy of a Bayes classifier. GLVQ does not reach the performance of the Bayes classifier because it relies on the squared Euclidean distance. Hence it cannot account for the different standard variations in the two axes of the two Gaussians. Matrix adaptation improves this behaviour, and RelSim as well as Dist and Comb reach the performance of the optimal Bayes rejection in the (important)

regime of up to 25 % rejected data points. For more rejected data, the accuracy of RelSim drops since the chance is higher to reject correctly classified data that are confound with outliers. Assume the underlying prototype-based classifier is sufficiently flexible to capture the nature of the data. Then we can conclude for this setting that the used reject options based on dissimilarities to the decision border or the certainty value are well suited for a close to optimal rejection in the relevant regime.

Experiments on Benchmark Data

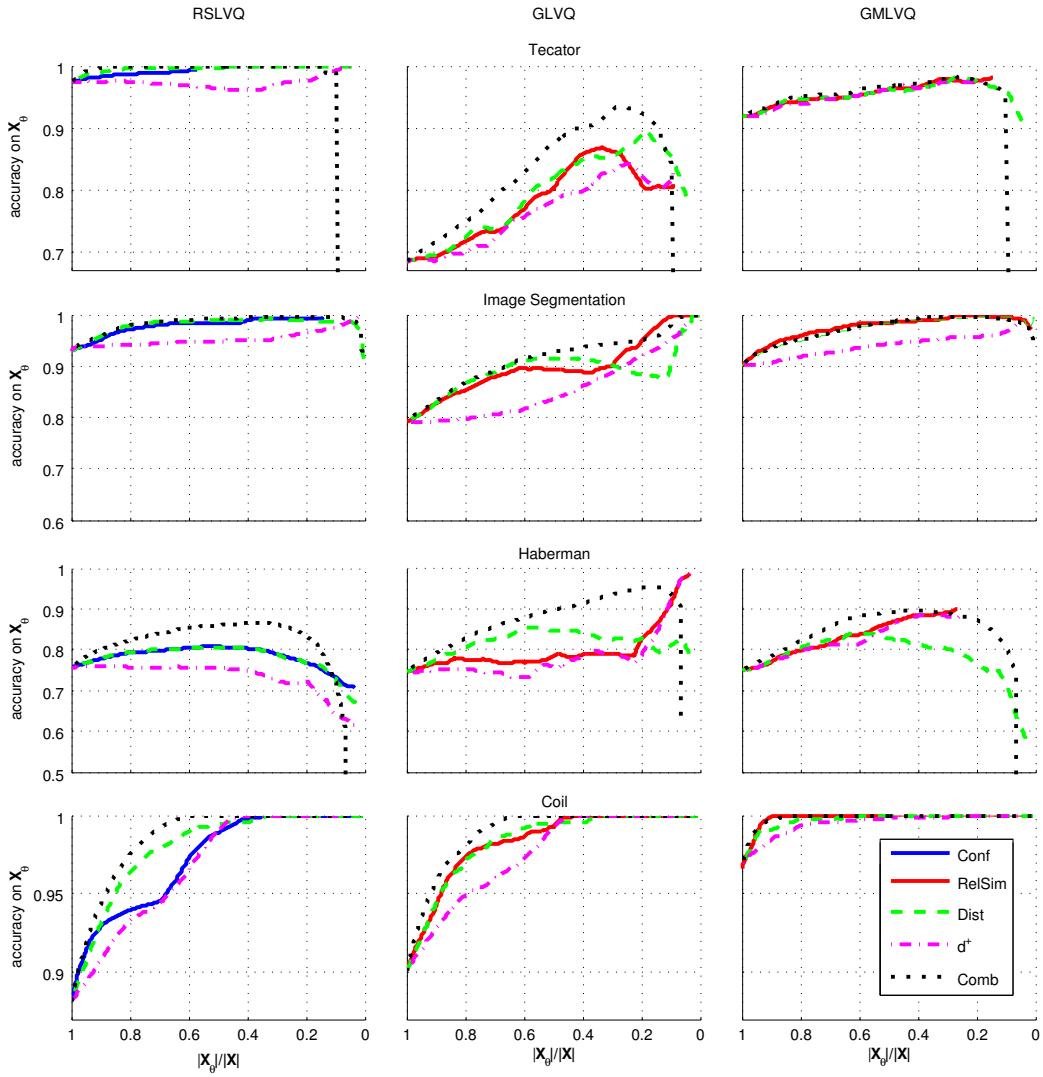


Figure 4.4.: [C14a] Accuracy-reject-curves for several prototype-based classifiers trained on benchmark data sets.

Figure 4.4 displays the average ARCs of the rejection strategies for the benchmark data sets. Mere outlier detection d^+ does not work well on average, which can be attributed to the fact that most errors can be accounted for ambiguities rather than outliers. This principle might become more important in online scenarios where the underlying distribution is subject to trend.

Dist and Conf show similar results for the RSLVQ models. This finding indicates that the more efficient measure Dist can be sufficient for a reliable reject option, making the (more complex) estimation of probability values superfluous. For the Coil data, Dist is even superior, which indicates that the internal estimation of the probability by the RSLVQ is not a good estimator for this data set.

The results of GLVQ are mostly inferior to RSLVQ, while GMLVQ can reach the same or even better accuracy. GLVQ uses a static dissimilarity measure for the whole input space and for every prototype. The RSLVQ offers more flexibility since it can change the bandwidth as a meta parameter and the GMLVQ adapts the global metric according to the data. The better accuracies of RSLVQ and GMLVQ are due to this flexibility. For GLVQ, Dist and RelSim mostly provide comparable results in the relevant regime of up to 25 % rejected data points. Interestingly, taking into account an optimal combination with outlier detection can improve this performance, albeit outlier detection alone (d^+) is not very well performing. This combination, however, does not offer an efficient strategy since it requires an additional loop over possible threshold vectors.

For GMLVQ, RelSim and Dist both provide excellent results similar or even better than a fully probabilistic modelling as offered by RSLVQ and Conf. Hence they offer a good compromise between classifier accuracy and efficiency of the reject option. In addition, they have the benefit that an adaptation to online scenarios is easily possible since the measures depend on the position of the prototypes only.

Comparison to SVM Rejection

We compare the results of the certainty measures Conf, Comb, Dist, and RelSim with a state of the art reject option on top of an SVM. This enables us to compare the efficiency of the used reject options to alternative strategies which are not based on prototypes. The SVM reject option for binary classes uses a strategy which rescales the distance to the border (Platt, 1999): A sigmoid function is fitted against the binned distances of the training data points to the separating hyperplane such that probabilities which are estimated from the data are matched as closely as possible. This approach can be extended to multi-class settings by means of a pairwise coupling (Wu et al., 2004).

Figure 4.5 shows the results of RSLVQ and GMLVQ as in Fig. 4.4 and results for the LGMLVQ in comparison to SVM for the relevant reject options and the data sets: Tecator, Image Segmentation, Habermann, and Coil. For GMLVQ and

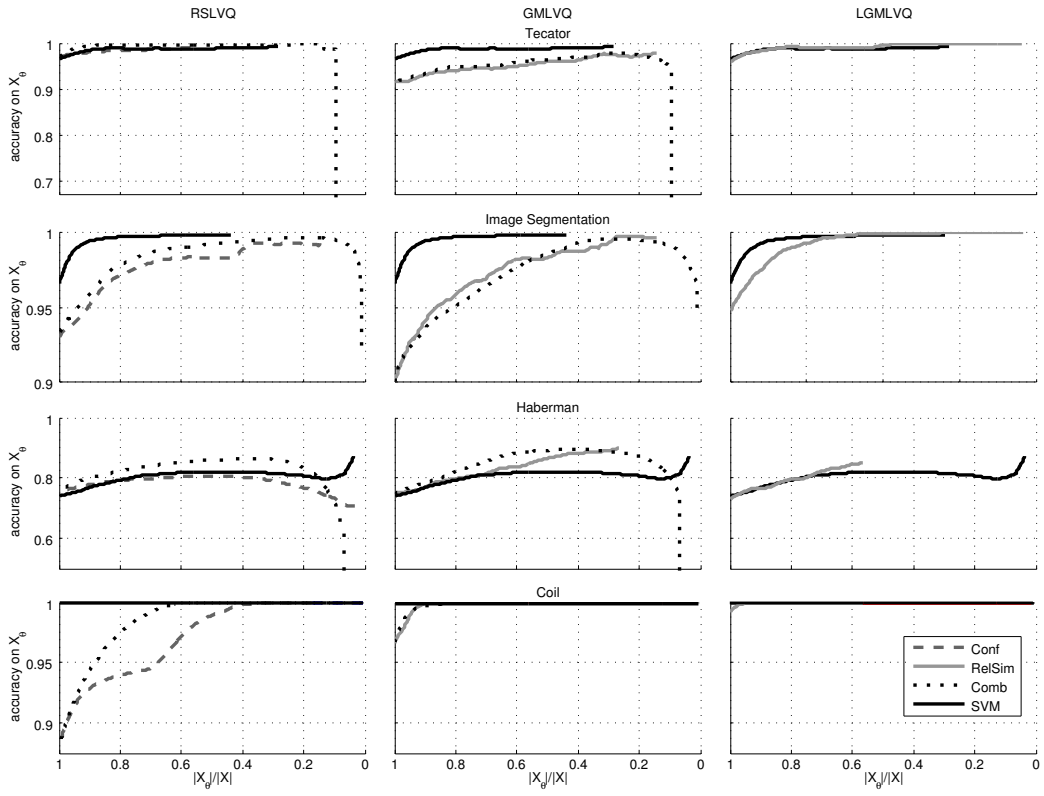


Figure 4.5.: Accuracy-reject-curves of the proposed measures in comparison to them of the support vector machine rejection on benchmark data sets.

RSLVQ all data but Habermann, SVM obtains a better accuracy at the price of a more complex model: The average number of support vectors per model is 14.96 for Tecator, 265.81 for Image Segmentation, and 145.51 for Haberman. For the sake of completeness, we show the results for Coil, although the SVM reaches an accuracy close to 100 %, hence rejection cannot be evaluated in a meaningful way. The difference between the accuracy of the SVM and the LGMLVQ is very small because the latter is powerful due to its trained local metrics.

Interestingly, the reject options decrease the difference of the accuracy provided by SVM and the accuracy of RSLVQ or GMLVQ. Hence the used reject options seem to be suited as concerns the acquired performance. For the Haberman data set, the results are even superior as compared to SVM. Hence prototype-based classifiers such as (L)GMLVQ or RSLVQ together with efficient reject options such as Conf or RelSim offer a good compromise of a sparse classification model enhanced with the possibility of rejection, and a good classification accuracy.

4.4.3. Summary of the Main Findings

We have proposed and systematically compared several reject options for prototype-based classifiers using the example of learning vector quantisation. In particular, we have proposed efficient geometric certainty measures for prototype-based classifiers which have the potential of direct online applicability. We have compared these direct measures with statistical rejection strategies which are based on a full (more demanding) probabilistic modelling, and with state of the art rejection for SVM on benchmark data sets. Interestingly these settings constitute typical representatives of popular classification paradigms: (L)GMLVQ as a popular LVQ scheme based on a cost function and motivated by large margin optimisation, incorporating the powerful framework of metric learning in its model; RSLVQ as statistically motivated discriminative model; and, in comparison, SVM as discriminative large margin model which, unlike sparse prototype-based representations, relies on a representation of class borders in terms of support vectors.

We have demonstrated that efficient geometrically motivated measures (Dist, RelSim) can be used as efficient reject options, providing results which are comparable to optimal Bayes rejection strategies where available, but releasing the burden of explicit statistical modelling. Interestingly, geometric measures reach the accuracy of fully probabilistic models used as plugin-rules. However, SVM usually displays a better overall accuracy for the full model due to its ability to use a flexible description of the class borders in terms of support vectors rather than a sparse prototype-based representation only. We would like to stress the fact that the proposed certainty measures are not restricted to LVQ classifiers but they have a broader scope: On the one hand, the training technique is not relevant for the scenario, rather any prototype-based classifier can be enhanced accordingly, such as unsupervised techniques equipped with posterior labels. On the other hand, some of the concepts transfer to alternative classifiers such as the distance to the class border. E. g. any classifier where one can define the closest distance to the class border of a data point one can apply a reject option based on this measure. There exist approaches for example for decision trees (Alvarez et al., 2007).

These findings open the way towards the design of efficient lifelong model adaptation for popular prototype-based classifiers such as (L)GMLVQ: The model complexity can easily be tailored online towards regions with a low certainty of the classification, e. g., introducing novel prototypes which are capable of representing novel aspects of the data (further details in chapter 6).

The next section contains a deeper analysis of the performance of the measures in comparison to probabilistic approaches since up to now only a comparison to RSLVQ model with Conf is done.

4.5. Comparison with Probabilistic Approaches

Since we know that the used certainty measures work properly for LVQ methods, it is interesting how this compares to probabilistic approaches. We consider the key question: Are these geometric approaches comparable to rejection strategies based on certainty values of probabilistic models which can be optimal as shown in Chow (1970), and if so under which conditions? Hence, we systematically compare the behaviour of the measures to rejection strategies for probabilistic classifiers. We vary

- the rejection strategy, ranging from deterministic, geometric measures to reject options based on confidence values,
- the data set, ranging from artificial data to typical benchmarks, and
- the nature of the prototype-based classifier for which the reject option is taken, considering purely discriminative models (RSLVQ) in comparison to generative ones (GMM).

Albeit both classifiers are derived as explicit probabilistic models, they have a different design. Purely discriminative ones are tailored to the classification task rather than the data, such that it is unclear whether rejection strategies can be based on their certainty values. Similarly, it is unclear whether efficient deterministic strategies based on simple geometric quantities reach the performance of rejection strategies on certainty values, the latter is supposed to require valid probabilistic models of the data. We show that this is indeed the case for real-life settings: heuristic rejection strategies based on geometric quantities offer an alternative to measures based on a probabilistic value. The description of the RSLVQ can be found in section 3.4 while the Gaussian mixture model is explained afterwards.

4.5.1. Gaussian Mixture Model and its Certainty Measure

Assume as before a data set X with elements $\mathbf{x} \in \mathbb{R}^M$. As already stated earlier, a prototype-based classifier is characterised by a set of prototypes $W = \{\mathbf{w}_i \in \mathbb{R}^M\}_{i=1}^{\xi}$, which are equipped with labels $c_j \in \{1, \ldots, Z\}$ considering a classification into Z classes.

In practice, generative models are often trained in an unsupervised way, directly aiming at a representation of the data distribution $p(\mathbf{x})$, popular examples being GMM for density estimation. Here we consider a class-wise GMM which aims at a

representation of every class by optimising the following data log-likelihood

$$E_{\text{GMM}}(X) = \sum_{1 \leq i \leq N} \log \left(\sum_{1 \leq j \leq \xi} \delta^{c_j}_{c(\mathbf{x}_i)} \cdot p(\mathbf{w}_j) \cdot p(\mathbf{x}_i | \mathbf{w}_j) \right) \tag{4.6}$$

where $p(\mathbf{x}_i|\mathbf{w}_j)$ is a Gaussian distribution centred on $\mathbf{w}_j$, and $p(\mathbf{w}_j)$ is the class-wise prior of the prototype with

$$\sum_{1 \leq j \leq \xi} \delta^{c_j}_{c(\mathbf{x}_i)} \cdot p(\mathbf{w}_j) = 1 \ .$$

The model parameters can be optimised by means of a gradient technique or, alternatively, a classical expectation maximisation scheme (Dempster et al., 1977) for every class, since the objective decomposes according to the class labels (Bishop, 2006). A GMM provides for each class y an explicit confidence measure

$$p(y|\mathbf{x}, W) = \frac{p(y) \cdot p(\mathbf{x}, y|W)}{\sum_{z \in \{1,\ldots,Z\}} p(z) \cdot p(\mathbf{x}, z|W)} \ . \tag{4.7}$$

A generative model, the GMM, represents the distribution on $\mathbf{x}$ with respect to the training scheme and $p(y)$ is the prior of the class with $\sum_{y \in \{1,\ldots,Z\}} p(y) = 1$.

Certainty Measure: Since GMM is a probabilistic model like RSLVQ, the classification of a data point $\mathbf{x}$ can be based on the most likely class

$$c(\mathbf{x}) = \arg \max_{1 \leq z \leq Z} p(z|\mathbf{x}, W) \ .$$

In practice, the resulting maximum z often corresponds to the class of the closest prototype such that a close resemblance to a classical winner takes all scheme (3.1) is obtained. As certainty measure one can immediately use the estimated class probabilities like for RSLVQ. Hence the structure is the same as r_{Conf} (4.2) but a GMM model provides the probabilities.

4.5.2. Experiments

We analyse the behaviour of the different certainty measures, focusing on the following questions: What is the behaviour of the measures regarding different characteristics of the classifier model ranging from a discriminative to a generative one? What is the behaviour of simple deterministic heuristics in comparison to rejection strategies based on confidence measures and do the latter require valid probabilistic models? Since probabilistic models are needed for an evaluation of Conf, we use the two probabilistic models RSLVQ and GMM. For all settings,

RSLVQ and GMM are trained using one prototype per class. For RSLVQ, a global parameter σ^2 is optimised via cross-validation. For GMM, correlations are set to zero and local scalings of the dimensions are adapted by means of diagonal matrices attached to the prototypes which are optimised in an expectation maximisation scheme. Training takes place until convergence using random initialisation. Convergence is assumed if the training error changes less than 10^{-5} during two sequenced training steps. We use the following data sets: Gaussian Clusters, Image Segmentation, Tecator, and Haberman (section 4.4.1).

For all data sets, two models are trained: a probabilistic generative model by means of class-wise GMM, and a probabilistic discriminative model by means of RSLVQ. For the resulting models, the effect of a reject option is compared for different possible strategies as introduced above. As in the previous experimental section, we vary the rejection threshold θ in small steps from no rejection (which corresponds to the original model) to full rejection i. e., no data point is classified). For Comb, a threshold vector is varied accordingly, and we report the result of the respective best combination. The results are depicted as ARCs.

Figure 4.6 shows the results obtained for the different rejection strategies and data sets. The resulting ARCs display a smooth transition from the accuracy of the model without reject options to the limit value 1 (in the case of Gaussian clusters it goes to 0) which results if $|X_\theta|$ approaches 0 (we leave out the value for the empty set at $|X_\theta| = 0$). The classification accuracy on X_θ does not change with θ if the classification accuracy is already 100 % (as is the case for the Tecator data set for RSLVQ), or if the errors are uniformly distributed over the range of the certainty measure r which is the case for the Haberman data set, for example. In the latter case, classes are imbalanced with the second class accounting for roughly one third of the data only, and LVQ models tend to represent only class one properly, such that class two accounts for errors equally distributed according to r. Note that the graphs are subject of noise if the size $|X_\theta|$ approaches 0 which can be attributed to the small sample size X_θ. Accordingly, the graphs are not reliable for $|X_\theta|/|X| < 0.1$, and the corresponding parts of the graphs should be seen as an indicator only. We choose the values of θ equidistant between the extremal values of each single measure.

Interestingly, the control of the number of points which are not rejected, $|X_\theta|$, depending on the threshold θ partially has gaps, as indicated in Fig. 4.6 by the straight parts of the curves and the ending of the curves at some size of $|X_\theta| \gg 0$. Such gaps can occur provided the size of X_θ changes abruptly with the threshold, which seems to be the case in some settings where a further increase of the thresholds leads to a rejection of all remaining data points. This is the fact for Conf for Gaussian clusters, Image Segmentation and Tecator for the GMM model, indicating that no point with confidence larger than a maximum threshold value

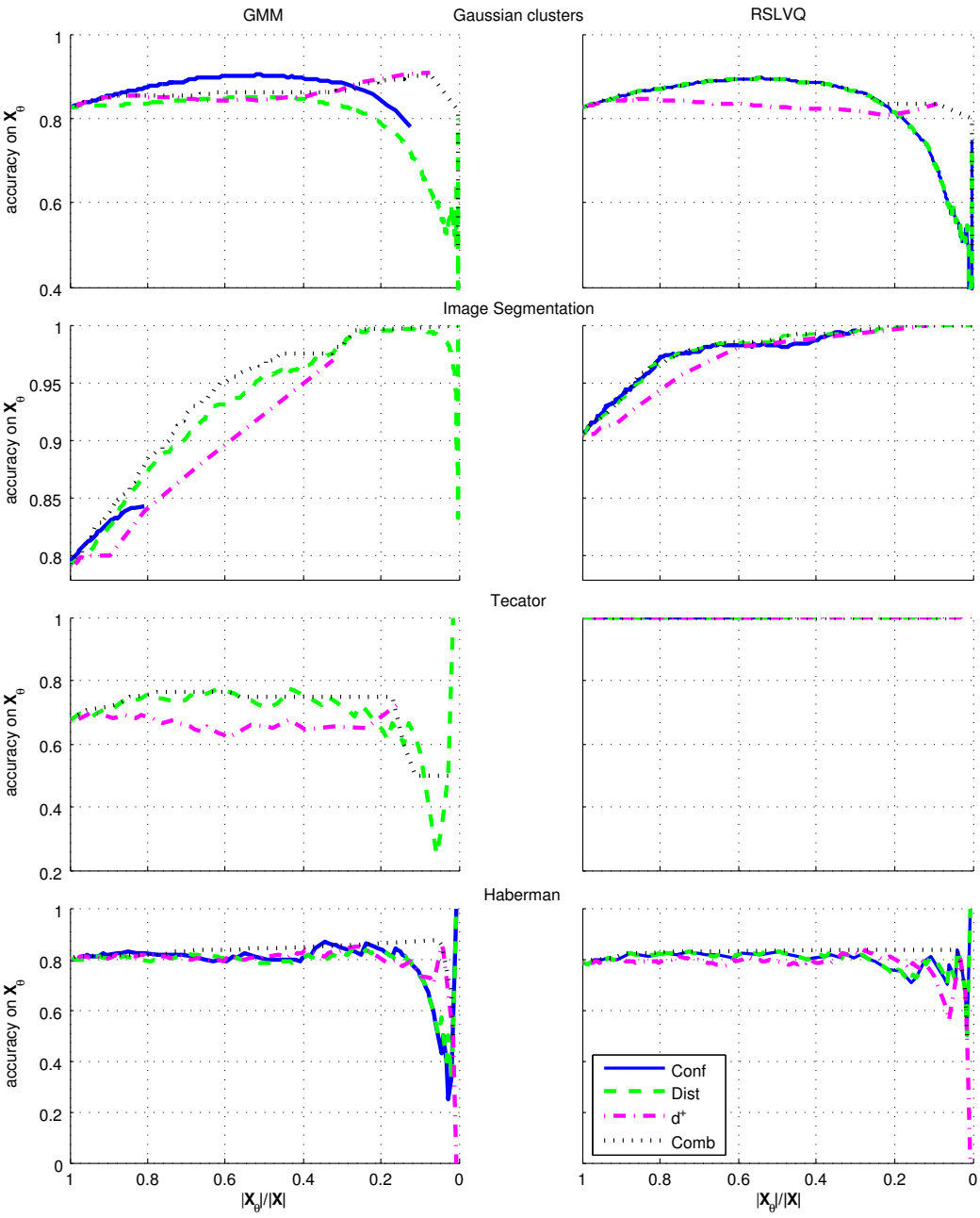


Figure 4.6.: Accuracy-reject-curves of different reject options when applied to generative or discriminative models, trained on several data sets. The Conf ARC for the GMM model of Tecator is rarely visible, since the probabilities of the data are all near one. For RSLVQ no errors are in the test set and hence all points of the different ARCs have a accuracy of one.

θ exist. Interestingly these gaps can be observed for Conf for the generative models only. Further, this behaviour is observed for d^+ for the data sets Gaussian clusters and Image Segmentation (both models) and Tecator (generative model). In contrast, the graphs of Dist and Comb do not have large gaps.

We can draw a few general conclusions from the graphs displayed in Fig. 4.6: In all cases, the discriminative model RSLVQ yields the same or better results as compared to generative GMM models, albeit the latter have a higher degree of freedom because of an adaptive diagonal matrix per prototype unlike RSLVQ, which relies on a global bandwidth only. This also holds for the full range of certainty values taken for the rejection strategies, regardless of whether deterministic of probabilistic certainty measures are used. Thus, it seems advisable to focus on the discriminative task, where confidence-based or deterministic measures can be used. As expected, rejection strategies based on confidence yield the best behaviour in most cases, but it does not allow a smooth variation of the size of X_θ for a large range in two of the settings. As mentioned, Conf cannot exclude outliers. This is apparently not a problem for the used data sets, highlighting the applicability of the optimality criterion of Chow (1970). Dist seems to offer a reasonable strategy in all other settings, whereby the behaviour is universally good for generative as well as discriminative classifiers, and it relaxes the burden of computing an explicit confidence value. d^+ gives better results than Dist in only one case (Gaussian clusters, GMM), and worse results than Dist in three cases (Gaussian cluster, RSLVQ; Image segmentation, both models; Tecator, GMM). Thus, in general, focusing on the discriminative nature seems advisable also as concerns the rejection strategy. As expected, Comb shows results comparable to the best of the two geometric reject options Dist and d^+, but also requiring a more complex rejection strategy by the combination of both values.

4.5.3. Conclusion with Respect to Probabilistic Approaches

We have compared direct geometric reject options and their combination with Bayesian motivated reject options in a couple of benchmarks using classifiers with different characteristics. The resulting observations are that geometric measures such as Dist can behave equally good as probabilistic measures, while often allowing a more fine-grained control of the size of the rejected data. In addition, they do not require explicit probabilistic models.

While allowing for simple measures which are applicable for a wider range of classifiers, the scaling of appropriate thresholds is unclear a priori and it depends on the data set at hand. In the literature, a few proposals how to automatically determine data-adapted values have been proposed (Vailaya and Jain, 2000), which can be transferred to our setting.

4.6. Conclusion: Answering the Research Questions

From the summary of global rejection in section 4.4.3 and from the summary of the comparison of probabilistic approaches in section 4.5.3, we can extract the following answers to the stated research questions at the beginning of this chapter:

1. Are the easily available dissimilarity-based certainty measures suited?

   From the executed experiments, we draw the conclusion, that the measures perform well for the used prototype-based LVQ models except the outlier detection d^+ but the used data sets seem to have less outliers than ambiguous data points. The performance of the measures keeps up with the performance of the optimal Bayes rejection in some settings and the rejection of the SVM.

2. How do these measures perform in comparison to probabilistic counterparts?

   The analysed deterministic measures can perform equally good compared to the natural certainty measure of probabilistic classifiers independently of their type (generative, discriminative).

3. Which properties do they have?

   Section 4.3.7 discusses the properties of the measures. Just pointing to a main finding: There is no guarantee that a given value $r(\mathbf{x})$ has the same semantic meaning at every position $\mathbf{x}$ of the space $\mathbb{R}^M$.

The main finding regarding the third question indicates that one global threshold for rejection may not suffice in cases where the semantic meaning of a given value $r(\mathbf{x})$ varies a lot for different positions $\mathbf{x}$. Extending the concept of a global threshold towards several thresholds valid in different partitions of the feature space leads to a local rejection strategy which can balance the different semantic meanings (local characteristics) if they are occurring. The next chapter 5 focusses on local rejection strategies.

5. Local Reject Option

Chapter overview *In this chapter, we discuss local rejection strategies for classifiers providing a partitioning of the feature space, e. g., learning vector quantisation, decision trees (DT) and SVM. We propose an efficient greedy algorithm and an optimal dynamic programming (DP) solution for defining the local thresholds. We analyse properties and performance of both solutions on various data sets and we compare them to global rejection. Our experiments show that the results of both solutions are very similar such that the fast greedy solution instead of the more complex DP solution seems a reasonable choice. When investigating rejection for benchmarks, the benefit of local strategies gets apparent in particular for simple prototype-based classifiers and DT. The effect is less distinct for more complex classifiers such as the SVM or classifiers that involve local metric learning like LGMLVQ.*

Parts of this chapter are based on:

[J16] L. Fischer, B. Hammer, and H. Wersing. Optimal Local Rejection for Classifiers. *Neurocomputing*, submitted

[TR15] L. Fischer, B. Hammer, and H. Wersing. Optimum Reject Options for Prototype-based Classification. In *CoRR*, abs/1503.06549, 2015.

[C14c] L. Fischer, B. Hammer, and H. Wersing. Local Rejection Strategies for Learning Vector Quantization. In *ICANN*, pages 563–570, 2014.

5.1. Motivation

Global rejection is restricted through its single threshold. More flexibility is available for local rejection strategies since every threshold is only valid in a defined partition of the input space as already mentioned in chapter 2. Local thresholds allow to consider local characteristics of the data within the reject option and they give the user better control over the complete rejection process. Using multiple thresholds leads to the problem of finding suitable ones. So far we did not find any method in the literature which solves this. Therefore, we analyse how to efficiently choose local optimal thresholds based on a given partition of the input space e. g., according to the predicted output classes. We rely on first promising results of an efficient greedy strategy. We extend this work by a general formalisation of the problem to optimally choose local thresholds for any given classifier. This problem can be treated as an optimisation problem in the form of a multiple choice knapsack problem (Chandra et al., 1976), and we derive a polynomial time dynamic programming (DP) scheme optimally solving the problem of local

threshold selection. We compare both solutions experimentally. While our optimal threshold selection strategy can be used for any classifier, we are focusing in the experiments on three popular classifiers: learning vector quantisation and its derivatives (Kohonen, 1989; Seo and Obermayer, 2003; Schneider et al., 2009a, 2010b), SVM, and DTs. We evaluate the rejection strategies extensively using different benchmark data and one real-life example from the medical domain whereby the evaluation of all experiments relies on the ARC (Nadeem et al., 2010; Landgrebe et al., 2006).

In the next section we state the research questions which we tackle thereafter in this chapter.

5.2. Research Questions

Global rejection implicitly assumes a suitable global scaling of the underlying certainty measure. Usually this is not met in a given setting. Using local thresholds instead releases the burden of a globally appropriate scaling of the underlying certainty measure. In this chapter we tackle the following questions:

1. How to determine efficiently local thresholds?
2. How good is the generalisation of the optimal thresholds on new data sets?
3. Does a local rejection outperform its global counterpart?
4. Is the usage of local rejection always beneficial and when is it better to stick with a global rejection?

Before answering these questions, the next section describes the used classifier.

5.3. Classifiers

In the following we introduce the classifiers used later on: prototype-based, DT and SVM classifiers. We explain how these classifiers induce a natural partition of the input space, on which to ground local rejection. We denote these sets defined by the respective partition as Υ_j.

5.3.1. Prototype-based Classifiers

As introduced in chapter 3, a prototype-based classifier consists of a set W of ξ prototypes $(\mathbf{w}_j, c_j) \in \mathbb{R}^M \times \{1, \ldots, Z\}$ and each prototype $\mathbf{w}_j$ has a class label c_j.

Classification is done by a winner takes all rule: A data point $\mathbf{x}$ gets the label c_l as the closest prototype $\mathbf{w}_l$ with

$$l = \arg \min_{j=1,\ldots,\xi} d(\mathbf{w}_j, \mathbf{x})$$

where $d(\cdot)$ is a dissimilarity measure; e. g., the Euclidean distance. By means of rule (3.1), a prototype-based classifier partitions the data into *Voronoi cells* or *receptive fields*

$$\Upsilon_j = \{\mathbf{x} \,|\, d(\mathbf{w}_j, \mathbf{x}) \leq d(\mathbf{w}_k, \mathbf{x}), \forall\, k \neq j\},\ j = 1, \ldots, \xi; \tag{5.1}$$

and it defines a constant classification on any Voronoi cell given by the label of its representative prototype.

5.3.2. Basic Decision Trees for Classification

A DT for classification (Breiman et al., 1984) is a rooted tree with a single root-node and interior-nodes which are equipped with a splitting criterion given by a dimension and a threshold, and Ξ leaves α_j which are equipped with a class label. A data point $\mathbf{x}$ is passed through the DT with respect to the split-criteria at each internal node. The leaf-node in which the data point $\mathbf{x}$ ends, defines the class label $c(\mathbf{x})$. We denote the decision border induced by the DT as Γ. A split-criterion sets a threshold for a specific dimension of the input space, e. g., on the first dimension: $\mathbf{x}(1) < \beta$, hence it defines a hyperplane. We consider axis parallel decision borders in this case. We use the receptive fields of the leaves of a given DT as partition (Fig. 5.1):

$$\Upsilon_j := \{\mathbf{x} \,|\, \mathbf{x} \text{ falls into leaf } \alpha_j\},\ j = 1, \ldots, \Xi; \tag{5.2}$$

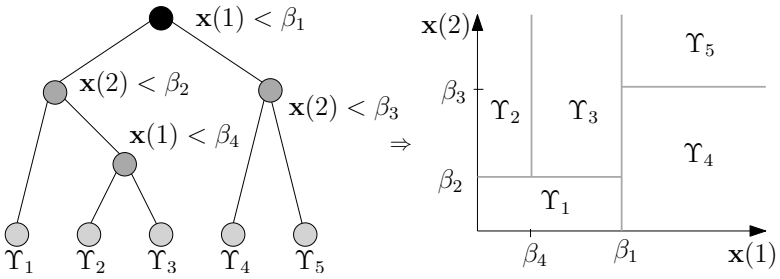


Figure 5.1.: Left: An example decision tree. The root-node in black, the terminal-nodes in grey with their split-criteria, the leave-nodes in light-grey. Right: The partitions of the space induced by the decision tree.

5.3.3. Support Vector Machine for Classification

The SVM is a classifier which uses an implicit non-linear embedding of the data into a high dimensional kernel space. For a binary setting, it realises the generalised non-linear classification

$$\mathbf{x} \mapsto H(\mathbf{w}^T \cdot \Psi(\mathbf{x}))$$

with Heaviside function $H(\cdot)$, linear weighing $\mathbf{w}$, and feature map $\Psi(\cdot)$ which is usually implicitly executed efficiently via a suitable kernel mapping. Training is stated as a constrained optimisation problem, which can be solved efficiently based on quadratic programming. For multiple classes, there exist different encoding schemes which transfer the problem into several binary classification problems. One popular approach is the one-versus-one scheme, which separates all pairs of classes by a binary SVM. Coupling can be done by means of the output activation $(\mathbf{w}_{ij})^T \cdot \Psi(\mathbf{x})$ where $\mathbf{w}_{ij}$ refers to the separation border of classes i and j.

Let $c(\mathbf{x})$ be the class label of a new data point $\mathbf{x}$ with respect to the SVM model, then we define the partition of the input space according to the classes:

$$\Upsilon_j = \{\mathbf{x} \,|\, c(\mathbf{x}) = j\},\ j = 1, \ldots, Z\ . \tag{5.3}$$

Note that this is a general partitioning of the space usable for any classifier.

5.4. Local Rejection

We focus on rejection strategies for classifiers which partition the input space and we rely on the following main parts:

- A certainty measure assigns a certainty value $r(\mathbf{x})$ to a data point $\mathbf{x}$ indicating the certainty of its classification,
- and a strategy how to reject a classification based on the certainty value; suitable rejection has to consider that often $r(\mathbf{x})$ is not scaled in an easily interpretable or uniform way. Hence, the value $r(\mathbf{x})$ does not necessarily coincide with the statistical confidence (which would be uniformly scaled in $[0, 1]$), and the scaling of the value $r(\mathbf{x})$ might even change depending on the location of the data point $\mathbf{x}$.

Later, we compare two strategies for rejection: a global strategy with one global threshold, and a local strategy based on the partitioning Υ_j of the input space, and an optimised vector of thresholds. First, we briefly review suitable certainty measures $r(\mathbf{x})$ before discussing optimal rejection strategies based thereon. We use some of the explained certainty measures for prototype-based classifiers (section 4.3). Further, we use a popular certainty measure designed for DTs

(Alvarez et al., 2007) and a particularly powerful certainty measure for SVM which already strives at an approximation of the underlying probabilities (Platt, 1999; Wu et al., 2004).

5.4.1. Certainty Measures

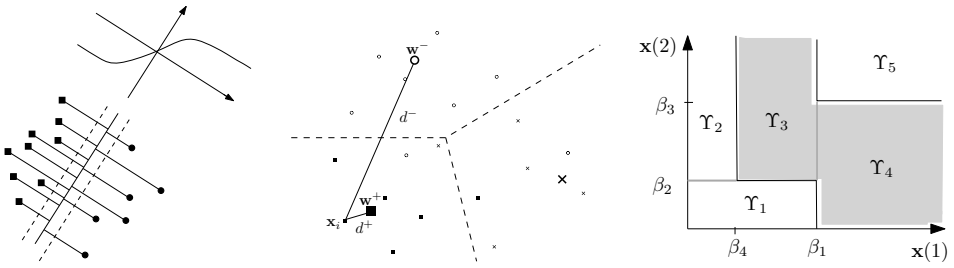
Reminder: Bayesian Confidence Chow (1970) analysed the error-reject trade-off of Bayes classifiers. He proposed an optimal certainty measure in this sense. The certainty value of a data point $\mathbf{x}$ in case of a Bayes classifier is defined as:

$$r_{\text{Bayes}}(\mathbf{x}) = \max_{1 \le z \le Z} p(z \mid \mathbf{x})$$

where $p(z \mid \mathbf{x})$ is the known probability of class z for a given data point $\mathbf{x}$.

Reminder: Empirical Estimation of the Bayesian Confidence Probabilistic classifiers like the RSLVQ provide explicit estimations of the probability $\hat{p}(z \mid \mathbf{x})$ of class z given a data point $\mathbf{x}$ leading to the certainty measure

$$r_{\text{Conf}}(\mathbf{x}) = \max_{1 \le z \le Z} \hat{p}(z \mid \mathbf{x}) \ .$$



(a) SVM rejection (b) Prototype-based rejection (c) DT rejection

Figure 5.2.: Certainty measures: (a) A binary classification setting with SVM. A Sigmoid is fitted against the values of the bins of the distances from the data points to the separating hyperplane. (b) A prototype-based classifier for an artificial three-class setting (different symbols, bigger ones are prototypes). For a single data point d^+, d^- are shown. (c) A partition of the input space of a DT. The leave-color (white/grey) encodes the class. The grey pieces of the border are no real borders since they divide areas of the same class while the black borders represent real borders dividing areas of different classes.

Reminder: Class Probability Estimates of SVM A popular approach by Platt (1999) turns the activity of a binary SVM into an approximation of a classification confidence. This activity (the distance of a data point to the decision border) is

transformed into a certainty value by a Sigmoid. The parameters of the Sigmoid are fitted on the given training data (Fig. 5.2(a)). Wu et al. (2004) provide an extension of this method to multi-class tasks, which is integrated in the LIBSVM toolbox (Chang and Lin, 2011). The method leads to a certainty measure like (4.2), but the empirical probability estimates are extracted from the SVM.

Reminder: RelSim for Prototype-based Classifiers The RelSim is a cost function based certainty measure (sec. 4.3). It used the normalised distance $d^+(\mathbf{x})$ of a data point $\mathbf{x}$ to the closest prototype and the distance of $\mathbf{x}$ to a closest prototype of a different class $d^-(\mathbf{x})$ (Fig. 5.2(b)):

$$r_{\text{RelSim}}(\mathbf{x}) = \frac{d^-(\mathbf{x}) - d^+(\mathbf{x})}{d^-(\mathbf{x}) + d^+(\mathbf{x})}$$

whereby d is either d_Λ (3.4) or d_{Λ_j} (3.8). Note that the prototype which belongs to d^+ also defines the class label of $\mathbf{x}$.

Distance to Decision Border for DT (Alvarez et al., 2007) The distance to the closest decision border (Dist) $d(\mathbf{x}, \Gamma)$ denotes the distance of a data point $\mathbf{x}$ to the closest decision border as defined by Γ; this border is formed by the hyperplanes defined by the split-criteria of the internal-nodes. Since the partition Υ_j of leaf α_j of a DT is bounded by axes-parallel hyperplanes, it is easy to compute the distance $d(\mathbf{x}, \Gamma)$ for a given point $\mathbf{x} \in \Upsilon_j$: $\mathbf{x}$ is projected orthogonally onto all hyperplanes which bound the leaf α_j, whereby we have to make sure to restrict to points which are on the decision border only (Fig. 5.2(c)). Then the minimal distance of $\mathbf{x}$ and this set of projections gives $d(\mathbf{x}, \Gamma)$ and the certainty measure

$$r_{\text{Dist}}(\mathbf{x}) = d(\mathbf{x}, \Gamma) \; . \tag{5.4}$$

Tóth and Pataki (2007) offer an extension for DTs with more general decision borders (e. g., non-axes parallel cuts, borders induced by a general quadratic form).

5.4.2. Local Reject Option

Reminder: Global Rejection A global reject option extends a certainty measure by a global threshold for the whole input space. Assume that

$$r(\mathbf{x}) : \mathbb{R}^M \to \mathbb{R}, \; \mathbf{x} \mapsto r(\mathbf{x})$$

refers to a certainty measure where a higher value indicates higher certainty. Given a real-valued threshold $\theta \in \mathbb{R}$, a data point $\mathbf{x}$ is rejected if and only if

$$r(\mathbf{x}) < \theta \ .$$

Figure 5.3 shows an artificial five class data set consisting of five Gaussians with very different variances where global rejection would fail for the considered certainty measure (RelSim). The reason is that the measure neglects the local characteristics (different variances of the Gaussians) of the data. This *Pearl Necklace* data set serves as example where local thresholds should perform much better than a global threshold for rejection.

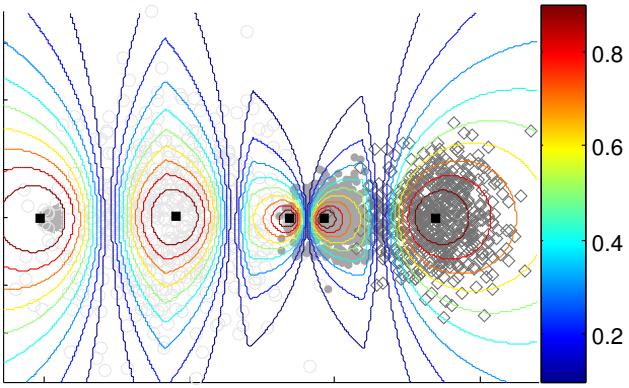


Figure 5.3.: Pearl Necklace data set: An example where a global reject option fails – Sketch of an artificial five-class setting (different symbols). The black squares are the prototypes of a classifier. Each of the five classes consists of one Gaussian. The variances of the Gaussians are very different which causes the problem for global rejection. Assume the RelSim as certainty measure (its contour lines are displayed) and a global threshold $\theta = 0.8$: While applying this threshold on the data of the middle Gaussian would lead to a reasonable reject option, the same threshold applied on data from the second Gaussian from left would lead to a rejection of lots of correctly classified data. In case of a single threshold per prototype this could be avoided by choosing different values for the thresholds, hence using a local rejection strategy.

Reminder: Local Reject Option Global rejection relies on the assumption of an equal scaling of the certainty measure $r(\mathbf{x})$ for all inputs $\mathbf{x}$. We relax this assumption by using local thresholds. A local threshold strategy relies on a partition of the input space $\mathbb{R}^M$ into ζ disjunct, non-empty sets Υ_j such that

$$\mathbb{R}^M = \bigcup_{1 \leq j \leq \zeta} \Upsilon_j \ .$$

For prototype-based classifiers we use the natural partition of the input space: the Voronoi-cells (5.1) as proposed in Vailaya and Jain (2000). The leaf areas (5.2)

are a proper partition for DT and for SVM we use a class-wise partitioning (5.3).

A separate threshold $\theta_j \in \mathbb{R}$ is chosen for every set Υ_j, and the reject option is given by a threshold vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_\zeta)$ of the dimension ζ equal to the number of sets in the partition. A data point $\mathbf{x}$ is rejected iff

$$r(\mathbf{x}) < \theta_j \quad \text{where } \mathbf{x} \in \Upsilon_j \ .$$

Rejection strategies crucially depend on the choice of the threshold θ or threshold vector $\boldsymbol{\theta}$. Afterwards, we analyse how to choose those in an optimal way.

5.5. Optimal Choices of Rejection Thresholds

As mentioned in chapter 2, finding an optimal threshold (vector) for rejection refers to multiple objectives: A threshold θ or threshold vector $\boldsymbol{\theta}$, should be chosen such that the rejection of errors (*true rejects*) is maximised, while the rejection of correctly classified points (*false rejects*) is minimised. Remember the following symbols, introduced in chapter 2:

$L, (L_j)$**:** correctly classified data points (in partition Υ_j)

$E, (E_j)$**:** wrongly classified data points (in partition Υ_j)

$\mathcal{L}_\theta, (\mathcal{L}_{\theta_j})$**:** false rejects (related to partition Υ_j)

$\mathcal{E}_\theta, (\mathcal{E}_{\theta_j})$**:** true rejects (related to partition Υ_j)

5.5.1. Extended Pareto Front

Analysing the efficiency of a threshold strategy, it turns out that a slightly different set is more easily accessible than the Pareto front $\mathcal{P}_\theta$ (2.2). Without loss of generality, we say that θ' *dominates* θ *with respect to the true rejects* if $|\mathcal{L}_{\theta'}| = |\mathcal{L}_\theta|$ and $|\mathcal{E}_{\theta'}| > |\mathcal{E}_\theta|$. This induces the *extended Pareto front*

$$\hat{\mathcal{P}}_\theta := \{(|\mathcal{L}_\theta|, |\mathcal{E}_\theta|) |\ \theta \text{ is not dominated by any } \theta' \text{ with respect to the true rejects}\} \ . \tag{5.5}$$

Hence, $\mathcal{P}_\theta$ can be computed as the subset of $\hat{\mathcal{P}}_\theta$ by taking the minima over the false rejects. $\hat{\mathcal{P}}_\theta$ has the benefit that it can be understood as a graph where $|\mathcal{L}_\theta|$ varies in between 0 and $|L|$ and $|\mathcal{E}_\theta|$ serves as function value. Having computed $\hat{\mathcal{P}}_\theta$ and the corresponding thresholds, we report the efficiency of a rejection strategy by the related ARC. In the following, we discuss efficient strategies to compute the extended Pareto front for global and local rejection.

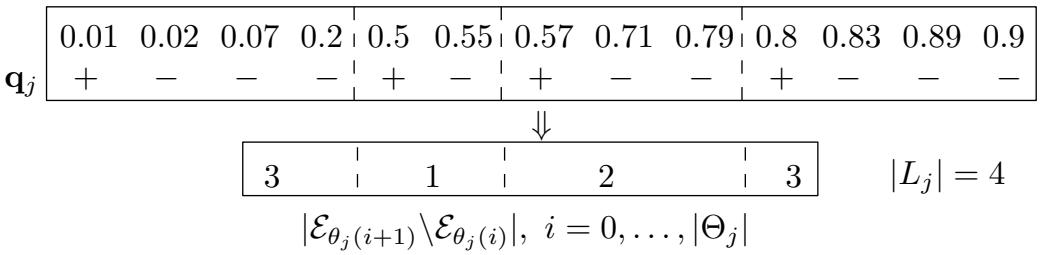


Figure 5.4.: Rejection thresholds for a partition with 13 points. The first row shows the sorted certainty values $r(\mathbf{x}_i)$, the second row depicts if a point is correctly $(+)$/wrongly $(-)$ classified. Here are 4 thresholds corresponding to the Pareto front, according to the number of signs $+$ (since point 13 belongs to E). The third row shows the gain when increasing the threshold value θ_j.

5.5.2. Optimal Global Rejection

Global rejection needs only one threshold θ. We compute thresholds leading to the extended Pareto front and the related pairs $(t_a(\theta), t_c(\theta))$ in time $\mathcal{O}(N \log N)$ due to the following observation: Consider a certainty measure $r(\mathbf{x}_i)$ for all points $\mathbf{x}_i \in X$ and sort the values $r(\mathbf{x}_{i_1}) \leq \ldots \leq r(\mathbf{x}_{i_N})$ (Fig. 5.4). We sort certainty values which are identical such that the points in L come first. The following holds:

- Each pair $(|\mathcal{L}_\theta|, |\mathcal{E}_\theta|) \in \hat{\mathcal{P}}_\theta$ is generated by some $\theta = r(\mathbf{x}_{i_j})$ related to a certainty value in this list or related to ∞ (i. e. rejecting all points), since values in between do not alter the number of rejected points on X.

- Values $r(\mathbf{x}_{i_k})$ with $\mathbf{x}_{i_k} \in E$ are dominated by $r(\mathbf{x}_{i_{k+1}})$ (or ∞ for the largest value) with respect to true rejects since the latter threshold accounts for the same number of false rejects, adding one true reject $\mathbf{x}_{i_k}$.

- Contrary, values $r(\mathbf{x}_{i_k})$ with $\mathbf{x}_{i_k} \in L$ are not dominated with respect to the number of true rejects. Increasing this threshold always increases the number of false rejects by adding $\mathbf{x}_{i_k}$ to the rejected points.

Therefore, the extended Pareto front is induced by the set of thresholds Θ corresponding to correctly classified points:

$$\Theta := \{\theta = r(\mathbf{x}_{i_k}) \mid \mathbf{x}_{i_k} \in L\} \cup \{\infty \mid \text{if } x_{i_N} \notin L\} \,. \tag{5.6}$$

$|\Theta| \in \{|L|, |L|+1\}$ depending on whether the last point in this list is classified correctly or not. An exemplary setting is depicted in Fig. 5.4. We refer to thresholds obtained this way as $\theta(0), \ldots, \theta(|\Theta| - 1)$ assuming ascending sorted values.

5.5.3. Optimal Local Rejection

Computing $\hat{\mathcal{P}}_\theta$ (5.5) for local rejection is harder than for a global one since the number of parameters (thresholds) in the optimisation rises from one to ζ. First, we derive an optimal solution via DP (Bellman, 1957; Cormen et al., 2001). Secondly, we introduce a faster greedy solution providing a good approximation of DP.

For every single partition Υ_j, the optimal choice of a threshold and its corresponding extended Pareto front is given in exactly the same way as for global rejection: We use the same notation as for global rejection, but indicate via an additional index $j \in \{1, \ldots, \zeta\}$ that these values refer to partition Υ_j. For any Υ_j, optimal thresholds as concerns the number of true rejects are induced by the certainty values of correctly classified points in this partition, possibly adding ∞. These thresholds are referred to as

$$\Theta_j := \{\theta_j(0), \ldots, \theta_j(|\Theta_j| - 1)\} \tag{5.7}$$

equivalent to (5.6) with $|\Theta_j| \in \{|L_j|, |L_j| + 1\}$.

We are interested in threshold vectors describing the extended Pareto front of the overall strategy, i. e., parameters $\boldsymbol{\theta}$ such that no $\boldsymbol{\theta}' \neq \boldsymbol{\theta}$ exists which dominates $\boldsymbol{\theta}$ with respect to the true rejects. The following relation holds: $\boldsymbol{\theta}$ is optimal $\Rightarrow$ every θ_j is optimal in Υ_j. Otherwise, we could easily improve $\boldsymbol{\theta}$ by improving its suboptimal component. The converse is false: E. g., assume a partition and thresholds as shown in Tab. 5.1. Here, we compare the threshold vectors (1,1,1) and (0,0,3). While both choices lead to 3 false rejects, the first one causes 9 true rejects and the second one leads to 25 true rejects. Hence the second vector dominates the first one, albeit thresholds are optimal within each Υ_j.

	$\lvert\mathcal{L}_{\theta_j(i)}\rvert$				$\lvert\mathcal{E}_{\theta_j(i)}\rvert$			
threshold i	0	1	2	3	0	1	2	3
Υ_1	0	1	2	3	3	4	6	9
Υ_2	0	1	2	-	2	3	6	-
Υ_3	0	1	2	3	1	2	10	20

Table 5.1.: Sample rejects for three partitions Υ_j and their losses $|\mathcal{L}_{\theta_j(i)}|$ and gains $|\mathcal{E}_{\theta_j(i)}|$.

It arises the question how to efficiently compute optimal combinations of the single values in Θ_j. There exist at most $|\Theta_1| \cdot \ldots \cdot |\Theta_\zeta| = \mathcal{O}(|L|^\zeta)$ different combinations (using the trivial upper bound $\mathcal{O}(|L_j|) \leq \mathcal{O}(|L|)$ for each $|\Theta_j|$). This number is infeasible for large ζ, i. e., a fine grained decomposition. We describe two methods to compute the Pareto front which are linear with respect to ζ, which depend on its formalisation as multiple choice knapsack problem (Chandra et al., 1976).

5.5.4. Formulation as Multiple Choice Knapsack Problem

Assume a fixed number n of false rejects. Then the problem of finding a threshold vector $\boldsymbol{\theta}$ which leads to a maximal number of true rejects can be formulated as multiple choice knapsack problem (MCKP, Chandra et al., 1976) as follows:

$$\begin{aligned} \max_{a_{ji}} \quad & \sum_{j=1}^{\zeta} \sum_{i=0}^{|\Theta_j|-1} |\mathcal{E}_{\theta_j(i)}| \cdot a_{ji} \\ \text{subject to} \quad & \sum_{j=1}^{\zeta} \sum_{i=0}^{|\Theta_j|-1} |\mathcal{L}_{\theta_j(i)}| \cdot a_{ji} = n \\ & \forall\, 1 \leq j \leq \zeta : \quad \sum_{i=0}^{|\Theta_j|-1} a_{ji} = 1 \\ & \forall\, 1 \leq j \leq \zeta, \forall\, 0 \leq i \leq |\Theta_j| - 1 : \quad a_{ji} \in \{0,1\} \end{aligned} \tag{5.8}$$

where the variable $a_{ji} \in \{0,1\}$ denotes whether the local threshold $\theta_j(i)$ is chosen for rejection in the partition Υ_j. The constraints guarantee that exactly one threshold is chosen in each Υ_j, and that the sum of false rejects equals n. The objective maximises the obtained number of true rejects. $|\mathcal{E}_{\theta_j(i)}|$ is the gain obtained in partition Υ_j and $|\mathcal{L}_{\theta_j(i)}|$ are the costs which are paid for this choice.

In general, the MCKP allows a pseudo-polynomial algorithm. Since the involved costs and gains in the formulation (5.8) are polynomial with respect to the number of data points $|X|$, this allows a polynomial solution of this problem. For instance, Chandra et al. (1976); Dudzinski and Walukiewicz (1987); Pisinger (1995) analyse efficient exact solutions, mostly based on linear programming relaxations which simplify the original MCKP such that it can be solved optimally by enumeration. We derive an efficient and intuitive alternative with the same computational complexity relying on the fact that thresholds in every partition Υ_j have an ordering according to their gain/costs in our case. This enables us to derive a quadratic time and linear memory algorithm similar to the DP scheme of the classical knapsack problem.

5.5.5. Local Threshold Adaptation by Dynamic Programming

For any number $0 \leq n \leq |L|$, $1 \leq j \leq \zeta$, $0 \leq i \leq |\Theta_j| - 1$ we define:

$$\begin{aligned} \mathrm{opt}(n,j,i) := \max_{\boldsymbol{\theta}} \{ |\mathcal{E}_{\boldsymbol{\theta}}| \mid & |\mathcal{L}_{\boldsymbol{\theta}}| = n, \theta_k \in \{\theta_j(0), \ldots, \theta_j(|\Theta_j|-1)\} \\ & \forall k < j, \theta_j \in \{\theta_j(0), \ldots, \theta_j(i)\}, \theta_k = \theta_k(0)\, \forall k > j \} \end{aligned} \tag{5.9}$$

The term $\mathrm{opt}(n,j,i)$ measures the maximum number of true rejects that we can obtain with n false rejects, and a threshold vector that is restricted as follows: the threshold in partition j is one of the first i thresholds, it is any threshold value for partition $k < j$, and the threshold for any partition $k > j$ is fixed to the first

threshold value. For technical reasons, it is useful to extend the index range of the partitions with 0 that refers to the initial case that all thresholds are set to 0 which serves as an easy initialisation. Since there are no thresholds to pick in partition Υ_0, we define $|\Theta_0| = 1$, i. e., the index i is the constant 0 in this virtual partition Υ_0.

The extended Pareto front can be recovered from the values $\mathrm{opt}(n, \zeta, |\Theta_\zeta|-1)$ for $n \leq |L|$, since these parameters correspond to the optimal number of true rejects provided n false rejects and free choice of the thresholds. An efficient computation scheme for the quantities $\mathrm{opt}(n, j, i)$ allows to efficiently compute the Pareto front.

For the values $\mathrm{opt}(n, j, i)$, the following Bellmann equality holds:

$$\mathrm{opt}(n,j,i) = \begin{cases} \text{if } n = 0: & \sum_{k=1}^{\zeta} |\mathcal{E}_{\theta_k(0)}| \\ \text{if } n > 0, j = 0: & -\infty \\ \text{if } n > 0, j > 0, i = 0: & \mathrm{opt}(n, j-1, |\Theta_{j-1}| - 1) \\ \text{if } 0 < n < i, j > 0: & \mathrm{opt}(n, j, i-1) \\ \text{if } n \geq i > 0, j > 0: & \max\{\mathrm{opt}(n, j, i-1), \\ & \quad \mathrm{opt}(n-i, j-1, |\Theta_{j-1}| - 1) + |\mathcal{E}_{\theta_j(i)} \backslash \mathcal{E}_{\theta_j(0)}|\} \end{cases} \tag{5.10}$$

This recursion captures the decomposition of the problem along the partitions:

- In the first case, no false rejects are allowed and the gain equals the sum of the gains $|\mathcal{E}_{\theta_j(0)}|$ obtained by the smallest thresholds in the partitions.
- In the second case, the number of false rejects has to be n, and only a trivial threshold with no rejects is allowed which is impossible (reflected with $-\infty$).
- In the third case, the threshold of partition j and all partitions with index larger than j are fixed to the first one by definition of opt (5.9). This is exactly the same as the term $\mathrm{opt}(n, j-1, |\Theta_{j-1}| - 1)$.
- In the fourth case, the i-th threshold is allowed, but it would account for i false rejects in partition j with only $n < i$ allowed false rejects. Hence we cannot pick number i but a smaller one only.
- In the fifth case there are two options, and the better of these two yields the result: Either a threshold with index smaller than i in partition j is chosen, or the threshold i in partition j is chosen. The first option leads to $\mathrm{opt}(n, j, i-1)$ true rejects. The second option causes i false rejects in partition j, hence at most $n-i$ further false rejects are allowed in partitions 1 to $j-1$, leading to a number of $\mathrm{opt}(n-i, j-1, |\Theta_{j-1}|-1)$ true rejects caused by thresholds in partition 1 to $j-1$. In addition, by picking threshold i in partition j, we gain $|\mathcal{E}_{\theta_j(i)}|$ true rejects as compared to only $|\mathcal{E}_{\theta_j(0)}|$ for the default 0. This is mirrored by the term $|\mathcal{E}_{\theta_j(i)} \backslash \mathcal{E}_{\theta_j(0)}|$.

DP can solve this recursive scheme, since the value i or j is decreased in every recursion, using loops over n, j and i (Algorithm 1, page 124); for memory efficiency we reduce the tensor $\mathrm{opt}(n, j, i)$ to a matrix $\mathrm{opt}(n, j)$ denoting the maximal number of true rejects with n false rejects and flexible thresholds in partitions $1, \ldots, j$. Since every evaluation of (5.10) itself is constant time, the computation scheme has an effort of $\mathcal{O}(|L| \cdot \zeta \cdot \max_k |\Theta_k|)$ with memory efficiency $\mathcal{O}(|L| \cdot \zeta)$. A standard back-tracing scheme reveals the related optimal threshold vectors.

5.5.6. Local Threshold Adaptation by an Efficient Greedy Strategy

Albeit enabling an optimal choice of the local threshold for given data, DP as described above (5.10) is infeasible for large training sets since its time complexity scales quadratically with the number of data: The number of possible thresholds $\max_j |\Theta_j|$ scales with N, we can expect it is of order $\mathcal{O}(N/\zeta)$. We describe a greedy approximation scheme[1] yielding to a linear method (besides pre-processing).

The basic idea is to start with the initial setting in analogy to $\mathrm{opt}(0, \zeta, |\Theta_\zeta| - 1)$: All thresholds are set to the default choice $\theta_j(0)$, hence no false rejects are present. Then, thresholds are increased greedily until the number of true rejects corresponds to the maximal possible number $|E|$. While increasing their values, the respective optima are stored and the values of the ARC are computed.

The greedy step proceeds as follows: In each round, the number of false rejects n increases by at least one to yield the optimal achievable gain, as follows:

- We consider local gains $|\mathcal{E}_{\theta_j(k+1)} \backslash \mathcal{E}_{\theta_j(k)}|$ for each partition Υ_j gained by rising the threshold index k by one. In addition, we evaluate global gains, which are obtained when assigning all false rejects to one partition only.
- If a global gain surpasses the local gains, this setting is taken and greedy optimisation continues.
- If a local gain surpasses the global gain, it is checked whether this choice is unique. If so, the greedy step continues.
- Otherwise, a tie occurs; this is in particular the case when a threshold increase does not increase the number of true rejects e. g., due to clusters of correctly labelled points. In this case, we allow to increase the number of false rejects until the tie is broken.

This procedure is described in detail in Algorithm 2 (page 126). Relying on a greedy strategy, the algorithm can yield suboptimal solutions. As we see in

[1] We would like to thank Stephan Hasler for the basic idea of the greedy optimisation of local thresholds and helpful discussions thereon.

experiments, results tightly approximate the optimal choices. Unlike the exact algorithm, the greed strategy only requires $\mathcal{O}(|L| \cdot \zeta)$ time and $\mathcal{O}(\zeta)$ memory.

5.6. Experiments for Local Rejection

We evaluate both algorithms (greedy and DP) which determine optimal thresholds as stated above for several data sets and classifiers. We use a 10-fold repeated cross-validation with ten repeats. The optimal thresholds are obtained from the training set and then used on the related test set of the cross validation. We use RSLVQ, GMLVQ, and LGMLVQ with one prototype per class. Since RSLVQ provides probability estimates, we combine it with the certainty measure Conf (4.2). The GMLVQ and LGMLVQ lend itself to the RelSim (4.3) measure. For the DT we use the default settings of the Matlab Statistics Toolbox[2] except of the *splitmin*-parameter and the related certainty measure Dist (5.4). We use the SVM (Chang and Lin, 2011) with a RBF-kernel and choose the best parameters of a cross-validation and a standard certainty measure (Platt, 1999; Wu et al., 2004).

In Fig. 5.6 and Fig. 5.7, we display the ARC averaged over 100 runs per data set and classifier. Note that the single curves have different ranges for $|X_\theta|/|X|$ corresponding to different thresholds. To ensure a reliable display, we only report those points $|X_\theta|/|X|$ for which at least 80 runs deliver a value.

5.6.1. Data Sets

For evaluation, we consider the following benchmark data sets: Image Segmentation, Tecator, Haberman, and Coil (section 4.4.1), an the artificial data set Gaussian Clusters (section 4.4.1), and the

Pearl Necklace This data set consists of five 2D Gaussian clusters with overlap (Fig. 5.3). Mean values are given by $\mu_{y_i} = 3\ \forall i,\ \mu_x = (2, 44, 85, 100, 136)$, the standard deviation per dimension is $\sigma_x = (1, 20, 0.5, 7, 11)$, $\sigma_x = \sigma_y$.

Since the complete ground truth is available for the artificial data sets, we use the optimal Bayesian rejection as a Gold standard for comparison.

5.6.2. Dynamic Programming versus Greedy Optimisation

First, we evaluate the performance of a greedy optimisation for the computation of local rejection thresholds versus an optimal DP scheme for all data sets. Since we are interested in the ability of the heuristics to approximate optimal thresholds, ARCs are computed on the training set for which the threshold values are exactly

[2]MATLAB and Statistics Toolbox Release 2008b, The MathWorks, Inc.

optimised using DP. The mean squared error of the two curves created by DP and the greedy solution is below 0.0015 for all experiments, for all but two it is even below $2.1 \cdot 10^{-5}$, see Fig. 5.5. Hence the greedy optimisation provides near optimal results for realistic settings, while requiring less time and memory consumption. In the following, we use the greedy optimisation for the local reject options.

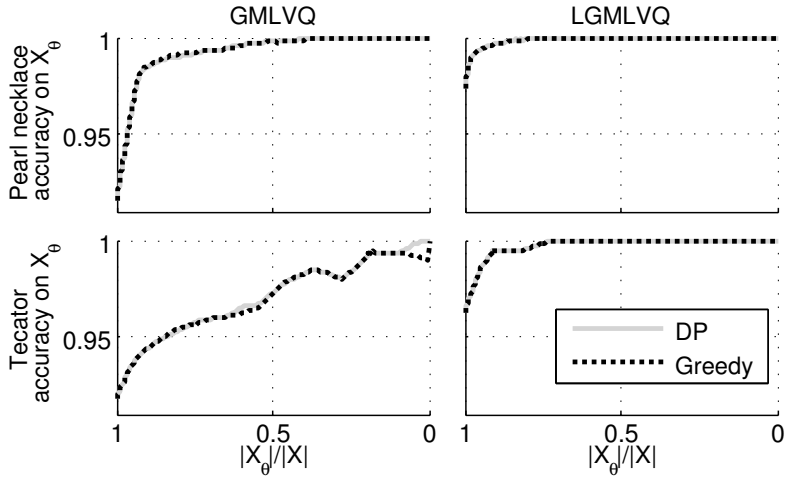


Figure 5.5.: Averaged accuracy-reject-curves for dynamic programming (DP) and the greedy optimisation applied on artificial and benchmark data sets for the relative similarity. The other settings are not shown since they look similar.

5.6.3. Experiments on Artificial Data

We report the ARC obtained on a hold out test set not used for training or threshold selection in order to judge the generalisation error of the classifiers with rejection. For the artificial data sets, we compare local and global reject options with the optimal Bayes rejection (Fig. 5.6). Thereby, RSLVQ is combined with Conf as certainty measure, while RelSim is used for deterministic LVQ classifiers, relying on the insights as gained in the studies (Sato and Yamada, 1995), and the findings of chapter 4, the DT uses Dist (5.4) as certainty measure and the SVM uses its estimated class probabilities. For all settings, the performance of the classifier on the test set is depicted, after optimising model parameters and threshold values on the training set. Results of a repeated cross-validation are shown, as specified.

Gaussian Clusters: For Gaussian clusters, global and local rejection ARCs are almost identical for all three LVQ classifiers. In this setting, it is not necessary to carry out a local strategy, rather a computationally more efficient global reject option suffices. Only for the DT the curves are different and the local rejection boosts the performance significantly. For SVM the local rejection does not improve

the performance over the global one. Interestingly, rejection strategies reach the quality of optimal Bayesian rejection in the relevant regime of up to 25 % rejected data points as can be seen in the left part of the ARCs. RSLVQ, due to its foundation on a probabilistic model, even enables a close to optimal rejection for the full regime as well as the local rejection for DT (Fig. 5.6).

Pearl Necklace: The pearl necklace data set is designed to show the advantage of local rejection. Here, local rejection performs better than global rejection for RSLVQ and GMLVQ and slightly better for LGMLVQ and DT. Global and local rejection show the same performance for SVM. As can be seen from Fig. 5.6, neither RSLVQ nor GMLVQ reach the optimal decision quality, but the ARC curves are greatly improved when using a local instead of a global threshold strategy. This observation is attributed to the fact that the scaling behaviour of the certainty measure is not the same for the full data space in these settings: RSLVQ is restricted to one global bandwidth, similarly, GMLVQ is restricted to one global quadratic form. This enforces a scaling of the certainty measure which does not scale uniformly with the (varying) certainty as present in the data. In comparison, LGMLVQ is capable of reaching the optimal Bayesian bound for both, local and global rejection strategies, caused by the local scaling of the quadratic form in the classifier. The analysis on these artificial data sets is a first indicator showing that local reject options can be superior to global ones in particular for simple classifiers. On the other side, there might be a small difference only between local and global rejection for well performing classifiers.

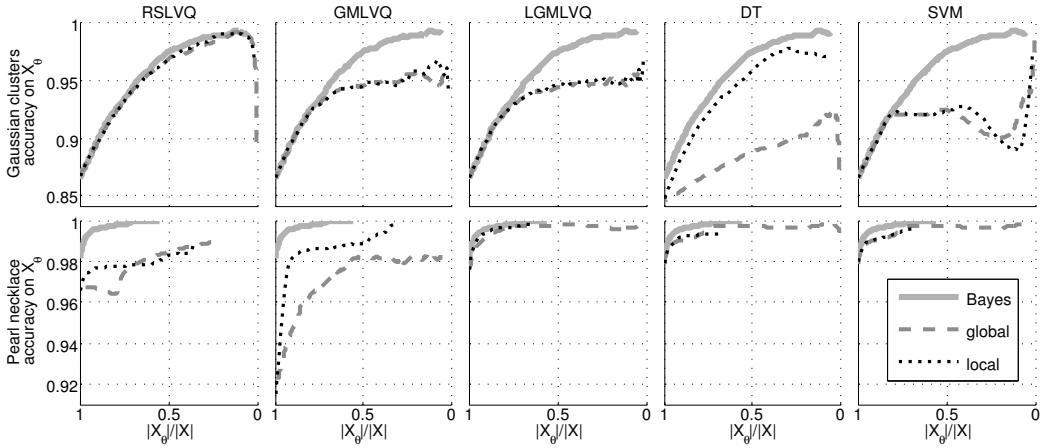


Figure 5.6.: Averaged accuracy-reject-curves for global and local rejection evaluated on test data. For RSLVQ Conf (4.2) serves as certainty measure and for the other LVQ classifiers, RelSim (4.3) serves as certainty measure. The decision tree (DT) uses the certainty measure Dist (5.4) and the support vector machine (SVM) uses the estimated class probabilities. The Bayes rejection with known class probabilities provides a Gold standard for comparison.

5.6.4. Experiments on Benchmarks

For the benchmark data sets, the underlying densities are unknown, hence we omit the result of optimal Bayes rejection. Figure 5.7 displays all results.

The experiments show that local rejection performs at least as good as global rejection for the important range from 0 % to 25 % rejection rate of the data. If the used classifier is already performing well there are less (e. g., LGMLVQ: Coil) or no (e. g., SVM: Coil) errors in the training data leading to bad or no local thresholds which can be applied on the test data. For simpler classifiers such as GMLVQ, RSLVQ, and DT, local thresholds improve the performance for several data sets. Therefore local thresholds seem beneficial in particular for simple classifiers where they can balance the local characteristics of the data neglected by the classifiers.

Based on these experiments, we conclude the following:

- Rejection can greatly enhance the classification performance, provided the classification accuracy is not yet optimal.
- Local rejection yields better results than global ones, whereby this effect is stronger for simple classifiers for which the classification accuracy on the full data set is not yet optimal. For more flexible classifiers with excellent classification accuracy for the full data set, this effect vanishes.
- Threshold optimisation by means of a linear time greedy strategy displays the same accuracy as computationally more complex optimal choices.
- If the distribution of the training and the test data is too different, the optimised thresholds on the training set will not likely perform well on the test set.

We would like to comment on the generalisation ability from the training to the test set. We observe desired behaviour of the local rejection in the experiments. But there are effects of overfitting, e. g., for Haberman, fortunately in regions which are less important (rejection rates $> 80\,\%$). It can also happen that the local thresholds do not affect the test data. This effect seems to occur only when the accuracy of the classifier is already high and hence there are only less data points for determining local thresholds, e. g., Tecator: SVM or Coil: LGMLVQ.

5.6.5. Medical Application – The Adrenal Tumours Data

We conclude with a recent instance from the medical area. The adrenal tumours data[3] (Biehl et al., 2012) contains 147 data points composed of 32 steroid marker values. These values are measured from urine samples using gas chromatography/mass spectrometry. We refer to (Biehl et al., 2012; Arlt et al., 2011) for

[3] We would like to thank Wiebke Arlt and Michael Biehl for providing the medical data and the support in related issues.

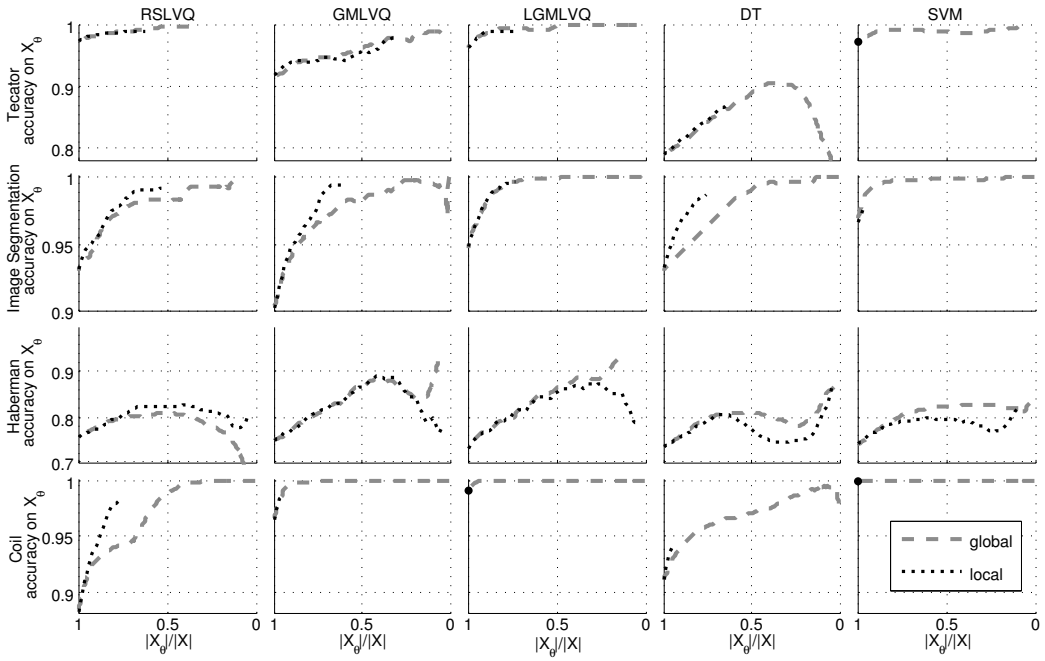


Figure 5.7.: Averaged accuracy-reject-curves for global and local rejection of test data. For RSLVQ Conf (4.2) serves as certainty measure, for the other LVQ classifiers the RelSim (4.3), for the decision tree (DT) the Dist (5.4) and the SVM uses its estimated class probabilities. The black points show the performance of the related classifier without local rejection, if the obtained local thresholds from the training set do not affect the test data.

further medical details. Two unbalanced classes are present: Patients with benign adrenocortical adenoma (102 points) or malignant carcinoma (45 points).

Our analysis of the data and the pre-processing follow the evaluation in (Biehl et al., 2012; Arlt et al., 2011): We train a GMLVQ model with one prototype per class. For the analysis of rejection we split the data into a training (90 %) and a test set (10 %). We evaluate the ARC of 1000 random splits of the data and the related GMLVQ models. Figure 5.8 shows the averaged ARCs of the tested rejections.

There is nearly no difference between the curves of the global and the local rejection for small rejection rates (up to 10 %). For more than 10 % rejection, the local rejection strategy improves the accuracy as compared to the global one. Further, the GMLVQ model provides insight into potentially relevant biomarkers and prototypical representatives of the classes (Arlt et al., 2011). As a conclusion, the GMLVQ model together with the proposed rejection offers a reliable and compact classifier for this medical application.

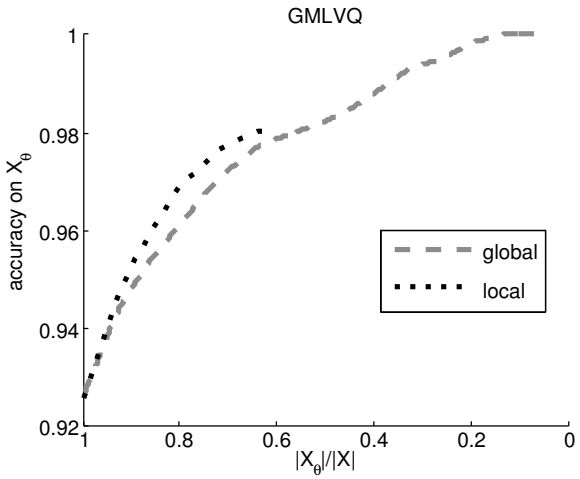


Figure 5.8.: Averaged accuracy-reject-curves for global and local rejection based on the RelSim (4.3) (test data).

5.7. Conclusion: Answering the Research Questions

In this chapter, we introduced rejection strategies for prototype-based, DT and SVM classifiers and extensively evaluated the proposed methods for diverse data sets. In particular, we explained global and local rejection and addressed the problem of their efficient computation. We explained two algorithms to derive optimal local thresholds: (i) An optimal solution based on DP and (ii) a fast greedy approximation. While the first is provably optimal, the latter is based on heuristics. Our experiments show that the results of both solutions are very similar such that the fast greedy solution instead of the more complex DP solution seems a reasonable choice. Its time complexity is only linear with respect to the number of data, while DP requires quadratic time, and its memory complexity is constant as concerns the number of data, while DPs memory size depends linearly on the number of data points.

When investigating these approaches for benchmarks, the benefit of local strategies becomes apparent in particular for simple prototype-based classifiers and DT. The effect is less pronounced for more complex classifiers that involve local metric learning like LGMLVQ or the SVM. Interestingly, the proposed rejection strategies in combination with the intuitive deterministic method LGMLVQ lead to results which are comparable to SVM and corresponding reject options. Thereby, the LVQ classifiers base the rejection on their dissimilarity to few prototypes only, hence they open the way towards efficient strategies for online scenarios.

Regarding the stated research questions (section 5.2), we provide the following answers:

1. How to determine efficiently local thresholds?

   The experiments have shown that the greedy algorithm 2 provides efficiently good approximations of optimal local threshold vectors.

2. How good is the generalisation of the optimal thresholds on new data sets?

   The local thresholds are optimised on the training set. We observed effects of overfitting but not in relevant regions. This is subject to future research.

3./4. Does a local rejection outperform its global counterpart? Is the usage of local rejection always beneficial and when it is better to stick with a global rejection?

   A clear yes or no can not be given as the answer to the fourth question since it strongly interacts with the third one. Due to the experiments we can say that simple classifiers like the RSLVQ, GMLVQ, and DT strongly benefit from local rejection for some data. For more advanced classifiers such as the LGMLVQ and the SVM only less benefit can be gained since these classifiers are able to deal with local characteristics of the data internally. Hence they are less dependent on local thresholds and often a global threshold suffices. The last two questions can only be adequately answered for a specific setting containing technical and user requirements.

So far we studied certainty measures and their properties for global rejection as well as for local rejection. Further we derived two algorithms for determining local thresholds for various classifiers with different certainty measures which closes the first part of the thesis. In the second part of the thesis we study the use of certainty measures in the context of lifelong learning architectures. The next chapter deals with an LVQ approach using a certainty measure in this context.

6. Incremental Online LVQ

Chapter overview *In this chapter, we investigate incremental online learning for LVQ, integrating metric learning as well as a certainty measure. Since this measure indicates where classification is uncertain with the current classifier, it is suited to point to areas where a higher classifier complexity could be needed. We analyse for artificial and benchmark data if integrating metric learning for incremental settings works and if using a certainty measure gives a benefit. The results of our experiments have shown that integrating metric learning performs better than the approaches without. Further the obtained results are comparable or even better than the results obtained by batch learning and an incremental SVM version.*

Parts of this chapter are based on:

[C15a**]** L. Fischer, B. Hammer, and H. Wersing. Certainty-based prototype insertion/deletion for classification with metric adaptation. In *ESANN*, pages 7–12, 2015.

6.1. Motivation

The previous chapters dealt with certainty measures used for rejection. Another field of application is lifelong learning which is in the focus of this chapter. Machine learning methods like LVQ or SVM provide state of the art classification schemes for automated data analysis (Schneider et al., 2009a; Seo and Obermayer, 2003; Tsang et al., 2005). In most settings, classifiers are used in offline scenarios like in the former chapters where a suitable classifier complexity can be adjusted based on cross-validation. This procedure gets prohibitive for big or streaming data and lifelong learning, where data cannot be inspected at once, and a proper classifier complexity cannot be identified prior to training. In this setting, online, incremental, or streaming algorithms are interesting, which are capable of adapting their model complexity while training based on the observed data (Read et al., 2012).

For SVM, incremental variants have been proposed which inspect only one data point at a time, but require extensive storage space due to a growing number of support vectors[1] (Cauwenberghs and Poggio, 2000; Diehl and Cauwenberghs, 2003; Laskov et al., 2006). For LVQ, a few online heuristics have been proposed: some only incorporate new training data (Bharitkar and Filev, 2001), while others adjust the number of prototypes either by error-based insertion only (Kirstein et al., 2005, 2008; Kietzmann et al., 2008), or dynamic prototype deletion and insertion

[1] We will use the online SVM code at: `http://www.isn.ucsd.edu/svm/incremental/`

(Grbovic and Vucetic, 2009; Xu et al., 2012). Unsupervised counterparts are for instance the growing neural gas (Fritzke, 1994) and extensions thereof. These methods are based on heuristics rather than a cost function. Further, no classifier dynamically adjusts the model complexity and an underlying general quadratic form.

In this chapter we analyse an incremental LVQ method which inserts and deletes prototypes based on a certainty measure strongly related to the GLVQ costs and we study its interaction with metric adaptation. This method allows a fast adaptation to new training data by prototype insertion, respecting noise or overlaps by prototype deletion and it reaches results comparable to offline variants or the incremental SVM (Cauwenberghs and Poggio, 2000), using less memory.

6.2. Research Questions

We want to study a cost function based LVQ approach which combines increasing and decreasing of the model complexity and metric learning. To our knowledge there are LVQ approaches combining some of those aspects but not all (Tab. 6.1).

property	cost-function	prototype insertion	prototype deletion	metric learning
Kirstein et al. (2005)	$\surd$	$\surd$	-	-
Kietzmann et al. (2008)	$\surd$	$\surd$	-	$\surd$
Grbovic and Vucetic (2009)	-	$\surd$	$\surd$	-
Xu et al. (2012)	$\surd$	$\surd$	$\surd$	-
new	$\surd$	$\surd$	$\surd$	$\surd$

Table 6.1.: Comparison of several online, incremental LVQ approaches. The symbol $\surd$ denotes that the classifier has the property while the symbol - denotes that the property is unavailable.

For this new approach we want to analyse the following questions:

1. Which effects do the model parameters have?
2. Does incremental online learning work when metric learning is integrated?

But first we present related work in the next section and we define the keywords: offline/batch learning, online learning, and incremental learning.

6.3. Related Work

The following section summarises the state of the art in online learning respectively in incremental learning. Since these terms are used differently, we define their meaning for this thesis in the beginning (Fig. 6.1):

Online learning: The assumption of online learning is that training data becomes available in sequences or as single points. The classifier integrates the approaching data in its training. Hence the classifier is changing when new training data arrives but its model complexity stays constant, e. g., the number of prototypes does not change in LVQ approaches like (Bharitkar and Filev, 2001).

Offline/batch learning: A offline/batch learning scenario assumes a complete training data set to train the classifier. The trained classifier stays fixed after training, e. g., SVM (Chang and Lin, 2011) and the integration of new training data is not necessary.

Incremental learning: The assumptions made for online learning are also valid for incremental learning with the extension, that the complexity of the classifier is allowed to change. This means the integration of new classes is possible as well as the deletion of disappeared classes, e. g., adding/removing prototypes in LVQ (Grbovic and Vucetic, 2009). For lifelong learning approaches this type of learning is expected since there are mechanisms for learning new concepts and forgetting obsolete ones.

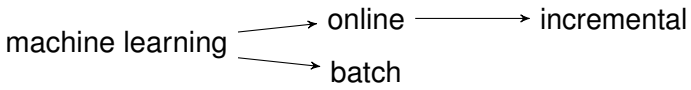


Figure 6.1.: Scheme of machine learning scenarios

Since the main focus of this thesis lies on prototype-based approaches, we provide a paragraph with related literature focusing on such classifiers and a paragraph dealing with other classifiers usable in online and/or incremental settings.

Prototype-based classifiers: Some classifiers only incorporate new training data (Bharitkar and Filev, 2001), while others adjust the number of prototypes either by error-based insertion only (Kirstein et al., 2005, 2008; Kietzmann et al., 2008), or dynamic prototype deletion and insertion (Grbovic and Vucetic, 2009; Xu et al., 2012). A generalisation of Kirstein et al. (2008) towards an classifier usable for category learning in a lifelong learning scenario was proposed by Kirstein et al. (2012). Different insertion strategies for prototypes of an online, incremental LVQ were analysed in Losing et al. (2015) for an application on a mobile robot. Xu et al. (2009) deals with incremental, online learning and represents prototypes as subspaces rather than vectors. This classifier is able to increase and to decrease the number of prototypes. Unsupervised counterparts are for example the growing neural gas (Fritzke, 1994) and extensions thereof.

Non-prototype-based classifiers: A popular classifier is the SVM and there are online versions of it. An early one is the approach of Cauwenberghs and Poggio (2000) usable for binary problems only and implying to learn the whole classifier again. To overcome the implicit retraining, Ralaivola and d'Alché-Buc (2001) consider only status changes (data point being a support vector or not) of data in the local neighbourhood of a new data point. A generalisation of Cauwenberghs and Poggio (2000) towards unlearning and learning of single or multiple data points as well as optimising the regularisation and kernel parameters was proposed in Diehl and Cauwenberghs (2003). Zheng et al. (2013) propose an idea how to deal with large-scale data and multi-class settings. The authors combine ideas from prototype-based classification with SVM. Their approach consists of two parts: learning prototypes and learning support vectors. Diethe and Girolami (2013) give a wide review on online learning with kernel methods, including theoretical comparisons of the methods and experiments on benchmark data sets. Besides kernel methods there are e. g., random forests and online versions thereof (Saffari et al., 2009; Abdulsalam et al., 2011; Schulter et al., 2011; Lakshminarayanan et al., 2014). Random forests offer a high computational efficiency during training and testing. Feed forward networks are also widely used for classification. Liang et al. (2006) present an approach for such networks trainable on chunks of data or even on single data points without retraining the whole network. Another method to fit neural networks in the scenario of incremental online learning is based on ensembles: Learn++ (Polikar et al., 2001) and extensions thereof (e. g., Ditzler et al., 2010). The idea is to combine several so-called weak learners via a weighted majority voting to create a dynamic and powerful classifier. The authors provide a theoretical upper bound on the error of the Learn++ classifier (Polikar et al., 2001).

Summing up, there are a lot of different incremental classifiers. Regarding the prototype-based ones there is no classifier which combines all advantages (Tab. 6.1) of (Kirstein et al., 2005; Kietzmann et al., 2008; Grbovic and Vucetic, 2009; Xu et al., 2012). Subsequently we close this gap.

6.4. Incremental Online LVQ

We introduce a general incremental online learning framework for LVQ (ioLVQ) usable in combination with any of GLVQ, GMLVQ, or LGMLVQ. The classifier gets single training data points $(\mathbf{x}, y) \in \mathbb{R}^M \times \{1, \ldots, Z\}$ one after another aiming at a good representation of the observed data in terms of a set of prototypes W and an adequate trained metric depending on the chosen LVQ approach. Starting with an empty set of prototypes, new prototypes are inserted or deleted on demand. Existing ones are adapted according to the standard GLVQ scheme, possibly including matrix learning. Hence the number of prototypes varies automatically to

mirror the complexity of the observed training data. Remember that the

$$\text{RelSim}(\mathbf{x}) = \frac{d^-(\mathbf{x}) - d^+(\mathbf{x})}{d^-(\mathbf{x}) + d^+(\mathbf{x})}$$

relates to the summands of the GLVQ costs and can be interpreted as a certainty of the classification: It provides values $\text{RelSim}(\mathbf{x}) \in (-1, 1)$, where values near 0 indicate high uncertainty, high values near 1 indicate a high certainty, and values below 0 indicate a wrong classification since $d^+ > d^-$. As discussed in chapter 4 this value can serve as an efficient approximation of a confidence for classification with rejection. In the following, this measure serves as signal which changes of prototypes likely decrease the GLVQ costs. There are three mechanisms to self-adjust the model complexity based on the observed data:

New classes insertion strategy taken from Xu et al. (2012): Each training point $(\mathbf{x}, y)$ with $y \neq c_j \,, \forall\, j$, i. e., a new class, is directly used as new prototype (6.1) with label y reducing the costs for class y.

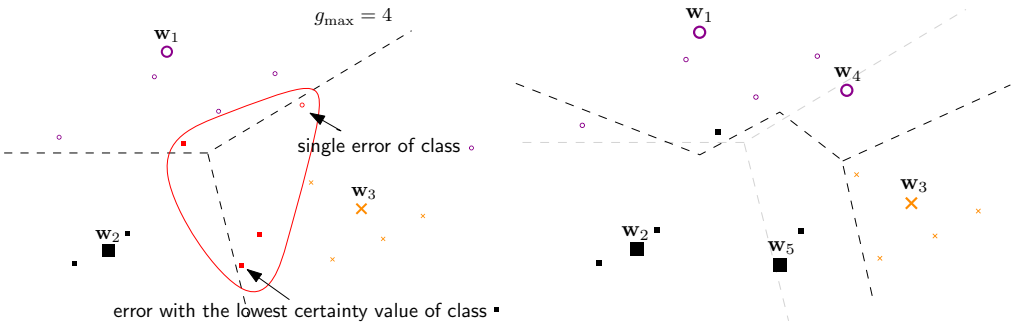
$$W := W \cup \{(\mathbf{x}, y)\} \tag{6.1}$$

Prototype insertion: Wrong classifications increase the costs (3.2), hence lowering their number decrease the costs, provided the remainder is not greatly affected. Based on an idea from (Kirstein et al., 2005; Kietzmann et al., 2008), wrongly classified training data are stored in a set S with maximum storage capacity g_{max}. Once $|S| = g_{\text{max}}$, for each class z for which errors are stored (i. e., $\exists\, i : z = y_i \wedge \mathbf{x}_i \in S$) a prototype with label z is introduced (6.2). We determine its position to minimise costs, i. e., the point $(\mathbf{x}_i, z)$ in the set with lowest $\text{RelSim}(\mathbf{x}_i)$ is chosen as prototype position (Fig. 6.2).

$$W := W \cup (\mathbf{x}_i, z) \quad \text{with} \quad (\mathbf{x}_i, z) = \arg \min_{\substack{(\mathbf{x}_k, y_k) \in S \\ y_k = z}} \text{RelSim}(\mathbf{x}_k) \tag{6.2}$$

Note that this also corresponds to a high degree of uncertainty, that means a potentially useful placement. After this insertion, the set S is cleared, i. e., $S := \emptyset$.

Prototype deletion: Unbounded prototype insertion can generate prototypes with noisy Voronoi cells, hence they harm more than they use. We remove prototypes based on their contribution to the costs (3.2), mimicking the idea underlying Grbovic and Vucetic (2009) that counts correct versus wrong classifications as score (Fig. 6.3). Here, every prototype $\mathbf{w}$ is accompanied by a parameter $\eta(\mathbf{w})$, initialised with zero, that sums up the certainty of the classifications of points in its



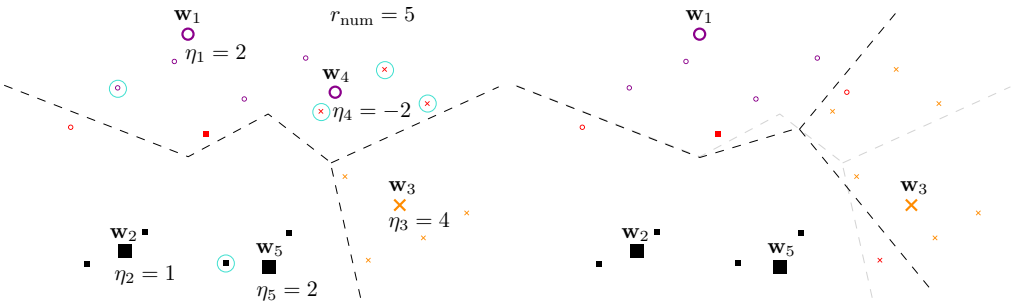
(a) before insertion (b) after insertion

Figure 6.2.: Scheme of error-based prototype insertion – The red symbols are the wrongly classified data points stored in S. The capacity of S gives $g_{\text{max}} = 4$ and the dashed lines are the class borders. The two highlighted errors in (a) got new prototypes $\mathbf{w}_4$, $\mathbf{w}_5$ in (b).

Voronoi cell. Hence a presentation of a data point $(\mathbf{x}, y)$ with its closest prototype $\mathbf{w}_l$ leads to the update:

$$\eta(\mathbf{w}_l) := \eta(\mathbf{w}_l) + \text{RelSim}(\mathbf{x}) \ .$$

The value $\eta(\mathbf{w}_I)$ is negative if and only if, on average, the prototype accounts for more wrongly classified points than correct ones, weighted by their certainty. In periods of r_{num} seen training data points, prototypes $\mathbf{w}$ with $\eta(\mathbf{w}) < 0$ are removed since their deletion directly improves the GLVQ costs.



(a) before deletion (b) after deletion

Figure 6.3.: Scheme of prototype deletion – (a) shows the last training instances encircled (cyan). After r_{num} training data points each parameter $\eta_j := \eta(\mathbf{w}_j)$ is checked. If $\eta_j < 0$ the related prototype is removed. This is the case for $\mathbf{w}_4$, hence it is removed, see (b).

6.5. Experiments

Unless stated otherwise, we consider randomly ordered data presented in a single pass, like in a streaming setting, using constant learning rates. Evaluation is based on ten repeats of a 10-fold cross validation. Considered data sets are: Image Segmentation, Coil (see section 4.4.1, page 36), and

Outdoor[2]: The Outdoor obstacle data set (Losing et al., 2015) contains 4000 elements consisting of 21 values of a 6 bin rg chromaticity diagram (Jain and Li, 2005) and they belong to 40 objects like dog, red ball, book, etc. lying in grass (Fig. 6.4). There are images with different perspectives (rotations, different sizes) of the objects (see Fig. 6.5).

Figure 6.4.: The 40 objects of the Outdoor data set.

Figure 6.5.: Different perspectives of an object of the Outdoor data set

In first experiments, we analyse the impact of the parameters which control insertion and deletion of prototypes. Then we move in the direction of integrating metric learning. Therefore we study the impact of several initialisation schemes for

[2]Thanks to Viktor Losing for providing this real-life data set and the related images 6.4 and 6.5.

the adaptive metric. Experiments with several classifiers for three different settings close the experiments section.

6.5.1. Influence of Parameters for Incremental Learning

We first analyse the effect of the parameters g_{max} and r_{num} of ioGLVQ without metric learning (Fig. 6.6). Because both parameters have a crucial influence on how the classifier deals with the stability plasticity dilemma, like the parameters of an unsupervised counterpart as discussed in Hamker (2001). The results with square symbols show results varying g_{max} only without prototype deletion ($r_{\text{num}} = \infty$). The parameter g_{max} controls the speed of prototype insertion, with small values accounting for a fast reacting but possibly noise-fragile system. A prototype removal strategy makes the system more robust with respect to noise.

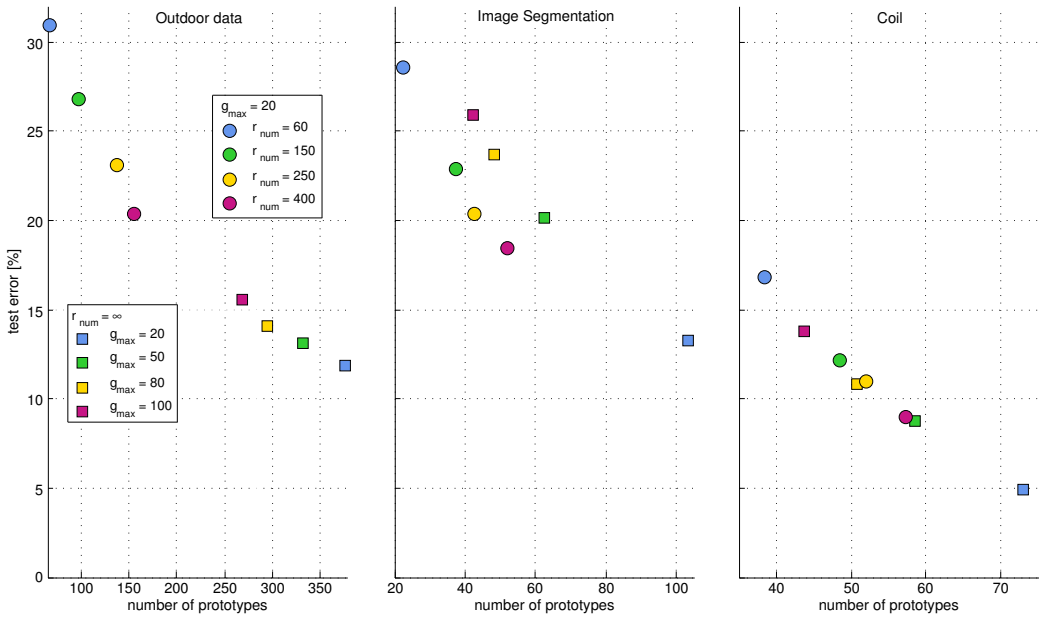


Figure 6.6.: Effect of the parameters g_{max} and r_{num} – We report the average test errors and the average prototype number. The image shows the effect of the parameter g_{max} without removing and of the parameter r_{num} with a fixed $g_{\text{max}} = 20$ (ioGLVQ). The precise values can be found in [C15a].

The results with circle symbols show the effect of r_{num} for a fixed g_{max}. Changing r_{num} to high values increases the number of prototypes. Too small values of r_{num} prohibit a sufficient adaptation of prototype positions, and prototypes are merely replaced. The parameters g_{max} and r_{num} control the trade-off between the error rate and the number of prototypes. In the following, we arbitrarily choose $g_{\text{max}} \approx 40$ and $r_{\text{num}} \approx 0.1 \cdot$ data set size based on these findings.

6.5.2. Compatibility with Metric Learning

We analyse the effect of metric learning for ioLVQ using several initialisation methods for the metric:

a) random values in (-1,1)

b) the unit matrix

c) a unit diagonal and random elements otherwise

d) initialisation with pre-trained global matrix obtained from a previous trained model

e) initialisation with local matrix of the next prototype from the same class

Figure 6.7 shows the results. Clearly, differences are only minor. Only between global and local metrics there are larger differences. Due to the results, we use initialisations c) respectively e) as robust ones in the following.

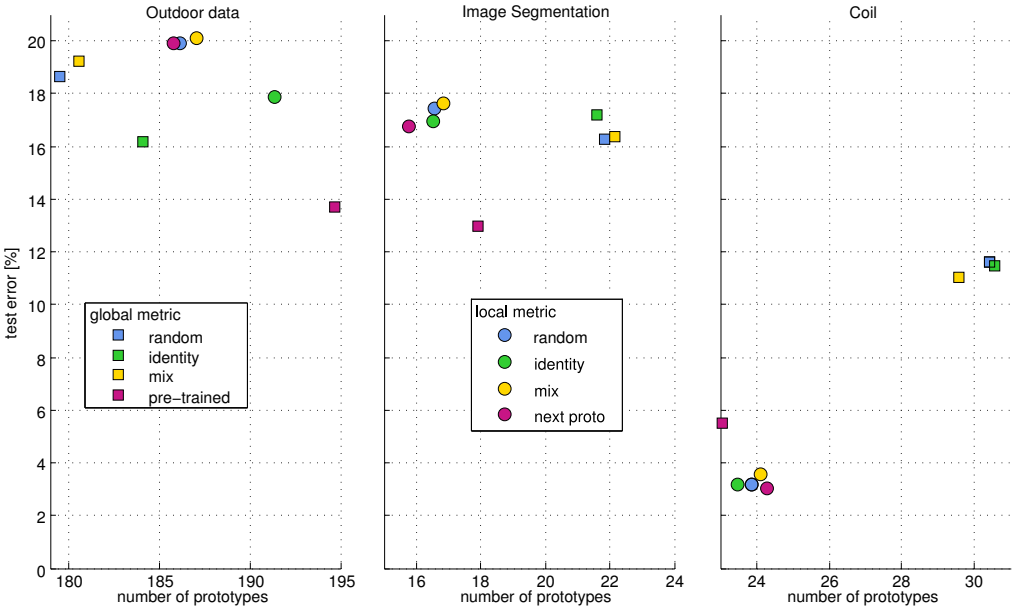


Figure 6.7.: The effect of the initialisation of the global/local metric. We report the average test errors and the average prototype number.The precise values can be found in [C15a].

6.5.3. Comparative Evaluation

We show three experimental settings:

1) a single streaming pass through the data,

Data	Err	$\|W\|$	Err	$\|W\|$	Err	$\|W\|$	Err	$\|SV\|$
1)	GLVQ		GMLVQ		LGMLVQ			
Image	20.4	42.8	**16.3**	**21.9**	**15.9**	**23.5**		
Coil	9.0	57.3	11.1	29.6	**3.0**	**24.3**		
Outdoor	**20.3**	**155.9**	**16.2**	**184.1**	17.9	191.3		
2)							increm. SVM	
Image	16.6	70.1	4.8	58.3	**3.4**	**55.0**	4.2	611.4
Coil[3]	3.0	83.6	0.7	67.1	**0.0**	**36.1**	0.7	1546.3
Outdoor	**16.6**	**241.9**	**14.8**	**255.8**	16.7	245.2	16.5	446.3
3)							batch SVM	
Image	20.9	7	9.9	7	*5.2*	7	2.8	265.8
Coil	9.9	20	3.8	20	*0.8*	20	0.0	773.8
Outdoor	17.7	160	*12.6*	160	18.5	160	4.7	1537.4

Table 6.2.: Comparison of the online and batch version of LVQ, batch SVM (Chang and Lin, 2011) and incremental SVM (Cauwenberghs and Poggio, 2000). For SVM, multiple classes are addressed by one vs. rest encoding. We display the average test error Err and the average number of prototypes $|W|$, support vectors $|SV|$. For settings 1) and 2) Pareto optima are set boldface. In setting 3) the best error rate (omitting batch SVM) is set italic.

2) multiple streaming passes (Image: 70, Coil: 120, Outdoor: 150; the number of passes is adjusted according to the convergence speed of batch LVQ; such that both methods have similar computational effort) these results are compared with an incremental SVM (Cauwenberghs and Poggio, 2000), and

3) batch versions of the LVQ approaches, initialised with one prototype per class (Image, Coil) and four prototypes per class (larger values hardly influence the classification accuracy), this is compared with a batch SVM (Chang and Lin, 2011)

In 1) and 2) learning rates are constant while in 3) the learning rates are annealed. The learning rates (prototypes, metric) of ioLVQ are roughly one magnitude smaller than in the batch version. Figure 6.8 and 6.9 show the results given in Tab. 6.2.

Learning a metric improves the performance and lowers the demand of prototypes, except for the Outdoor data. The reached performances are better than the performances of the batch counterparts for LVQ (except Outdoor: GMLVQ) but the ioLVQ classifiers use more prototypes than the batch classifier, in the worst case (Image: GLVQ) ten times more. In general, this higher number of prototypes is acceptable since the number is still mostly in the same order of magnitude. The results of ioGMLVQ are comparable to the results of an incremental SVM, while ioLGMLVQ is even better for Image and Coil.

[3] Features are scaled to [0,1] for both SVMs.

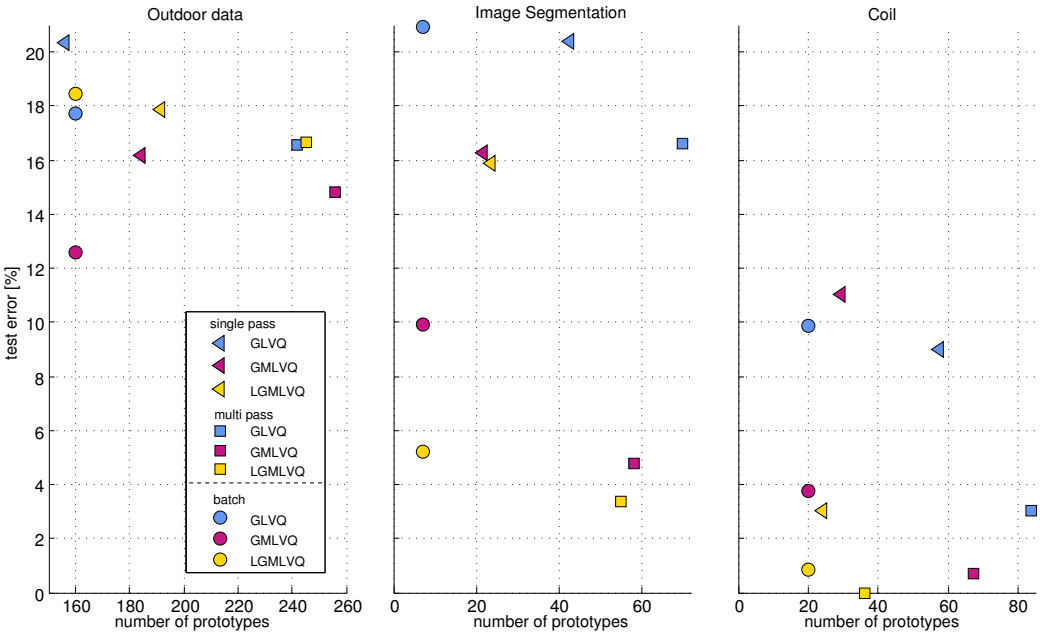


Figure 6.8.: The image shows the results of ioLVQ in three settings contained in Tab. 6.2.

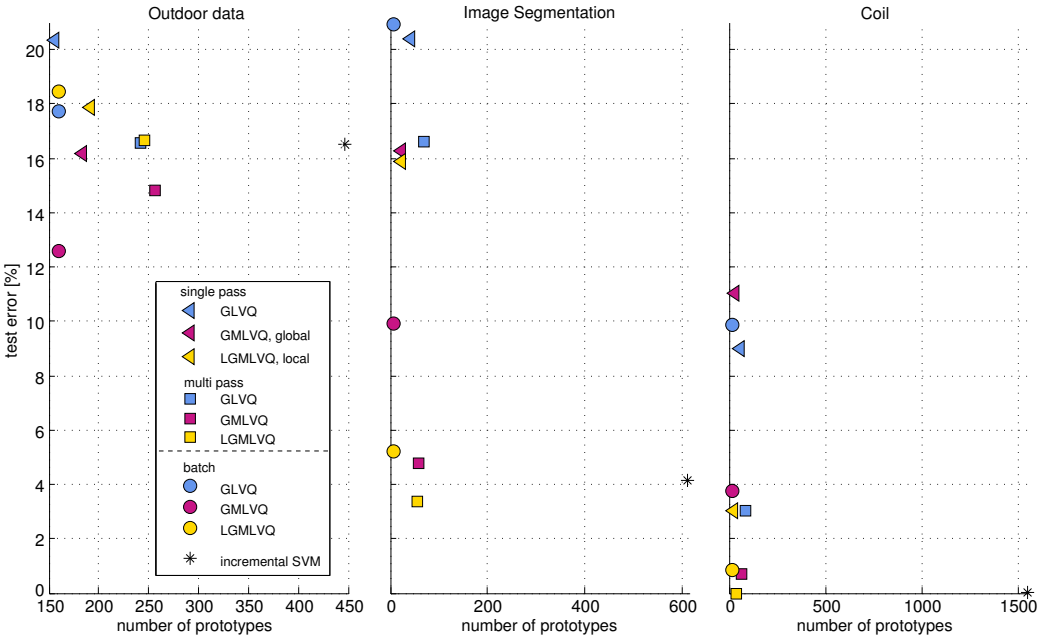


Figure 6.9.: Comparison of ioLVQ in several settings with an incremental SVM – This figure is the same as Fig. 6.8. Only the results of the incremental SVM ($*$) are added.

6.6. Conclusion: Answering the Research Questions

We proposed an efficient strategy for inserting/deleting prototypes based on a certainty measure for online incremental prototype-based classification, in particular with metric adaptation, and we demonstrated its performance using GLVQ, GMLVQ, LGMLVQ in benchmarks.

1. Which effects do the model parameters have?

   It turned out that the metric parameter initialisation only mildly effects the performance, while proper (i. e., small) learning rates are often crucial. Further, the models are very robust to the choice of the initialisation schemes and parameters. As expected, the parameters g_{max} and r_{num} control complexity and accuracy of the resulting models. Interestingly, most of the obtained values are Pareto optimal when varying these parameters.

2. Does incremental online learning work when metric learning is integrated?

   Yes the results have shown that using metric learning performs better than the plain ioGLVQ. Further the obtained results are comparable or even better than the results obtained by batch learning and an incremental SVM version, displaying the great promise of the proposed classifier. One striking property of the results which we obtained in real life benchmarks is the comparably small complexity with a number of prototypes which is larger than for batch variants but considerably smaller than the respective number of support vectors in SVM training. This observation identifies ioLVQ as a classifier with high promises for efficient online learning in the context of big data.

So far we assumed i. i. d. data and did not analyse the so-called *catastrophic forgetting effect* which can occur especially in the setting of non i. i. d. data including the issue of the so-called *concept drift*. In the next chapter several lifelong learning architectures are discussed for such a scenario. One architecture is directly designed to avoid this effect by combining offline and online learning.

7. Combined Offline and Online Learning

Chapter overview *In this chapter, we investigate a classifier architecture which combines offline and online learning via a dynamic classifier selection. Further we compare this architecture with other lifelong learning approaches on several data sets and we discuss their properties. It turned out that none of the analysed architectures surpasses all the others, such that a decision which one to use depends strongly on the desired application scenario. Furthermore, it came up an effect regarding the architecture which we call confidence drift and which has a strong impact on the dynamic classifier selection. We analysed this effect and we proposed a method in order to avoid negative impact on the performance of the architecture regarding this effect.*

Parts of this chapter are based on:

[C16] L. Fischer, B. Hammer, and H. Wersing. Online Metric Learning for an Adaptation to Confidence Drift. In *IJCNN*, accepted, 2016.

[C15b] L. Fischer, B. Hammer, and H. Wersing. Combining Offline and Online Classifiers for Life-long Learning. In *IJCNN*, pages 2808–2815, 2015.

7.1. Motivation

Several trends have caused a rapidly increasing interest in lifelong learning technology (Silver et al., 2013). Electronic systems get smarter and there is a trend for their personalisation such that the capability of systems to learn during their lifetime becomes essential. Application scenarios which require effort concerning lifelong learning, are for instance autonomous robotics/driving or assistive systems (Thrun and Mitchell, 1995; Sykes et al., 2013; Stavens et al., 2007).

Online learning crucially depends on the capability of a system for learning new concepts while preserving already known information over its lifetime (Silver et al., 2013). This setting violates usual assumptions of classical learning scenarios as formalised (see e. g., Vapnik, 1999): here often i. i. d. data are assumed. Furthermore many optimisation schemes depend on a formalisation of the problem using all training data. In contrast, humans always learn in an online setting, while interacting with their environment. Humans keep relevant knowledge from the past and use it to support learning of new content. It is notable that humans employ powerful strategies for lifelong learning which ensure that relevant basic concepts are stable while important changes can rapidly be integrated into their knowledge.

These two goals, encountering stability for ground concepts and at the same time, flexibility to deal with a changing environment or new concepts, constitute conflicting requirements which are difficult to realise in technical systems. Often the so-called *catastrophic forgetting effect* (CFE) can be noticed, i. e., there is no guarantee that ground knowledge (e. g., safety of a system) is respected unless it is explicitly hard-coded in the decisions of the system (Jin and Sendhoff, 2006; Goodfellow et al., 2013).

Within machine learning, the family of incremental learning approaches recently caused a lot of attention (see e. g., Polikar and Alippi, 2014). A typical problem in this context is *concept drift* which occurs in dynamically changing environments (Gama et al., 2014). Here we relate concept drift to the occurrence of new classes and to changing data distributions. We do not assume data points that change their labels. There exist methods which directly try to inspect when concept drift occurs (Minku and Yao, 2012), while we do this implicitly. Ditzler et al. (2015) give an overview on active and passive approaches dealing with concept drift. There are at least two ways how to deal with concept drift: one can rely on a flexible classifier changing with its environment, facing the risk of forgetting known concepts (He et al., 2011; Cauwenberghs and Poggio, 2000; Crammer et al., 2013) or one can rely on a combination of different classifiers, enabling a more flexible control about which information to keep and which one to adapt (Polikar et al., 2001; Hosseini et al., 2013). There exist several approaches using more than two classifiers, ensembles, to gather the diversity of the seen data in diverse models enabling good generalisation performance and dealing with different types of concept drift (Minku and Yao, 2012; Hosseini et al., 2013; Kolter and Maloof, 2007). We address the question of how to efficiently realise a combination of different architectures, and how their performance scales with respect to their ability to reliably deal with both basic known concepts and newly faced events in comparison to alternative model designs.

A lately proposed hybrid architecture (Wersing and Queißer, 2014; Queißer, 2012) combines two complementary classifiers via a dynamic classifier selection (Dasarathy and Sheela, 1979; Sabourin et al., 1993; Woods et al., 1997; Britto Jr. et al., 2014). One classifier is completely static during the application time after it is pre-trained with offline available data (offline classifier). The second classifier starts from scratch and it learns incrementally during its whole lifetime. The online classifier of the architecture allows to deal with concept drift. If the known data distribution changes or new classes occur, the architecture follows this change using the online classifier, while the static classifier preserves the already learned knowledge. This scheme is rather simple, but it uses a heuristic for classifier combination based on error counting.

This chapter describes an extension of this hybrid architecture, a detailed analy-

sis, and comparisons to alternative designs. While keeping the basic structure, we substitute the single parts of the architecture by more fundamental choices which rely on the notion of the certainty of the two parts for efficient classifier selection. Combining a static and a highly flexible incremental classifier in such a way seems promising with respect to the system's stability and plasticity. On an abstract level one can interpret the static classifier as kind of a long-term memory and the online learning part serves as a kind of short-term memory. The advantage of this architecture is the higher robustness against noise and CFE than pure incremental approaches without a backup model, as we will show in several benchmarks.

We demonstrate the capability of this architecture for lifelong learning tasks based on LVQ classifiers (see chapter 3) and the ioLVQ as explained in the last chapter 6. For a comparison we refer to state of the art incremental learners, such as incremental support vector machines (iSVM[1], Cauwenberghs and Poggio, 2000; Diehl and Cauwenberghs, 2003). Note that the latter, unlike LVQ classifiers, often require extensive storage space due to a growing number of support vectors. LVQ, depending on a prototypical data representation rather than a representation of class borders, usually relies on much smaller resources as we demonstrate in a number of experiments.

At first we describe the assumed scenario, followed by a related work section. Then we present the extended architecture combining an online and offline classifier. Later on we explain its parts, especially specifying how decisions on how to combine and how to train can be based on certainty values. Finally, we compare the architecture to purely incremental LVQ schemes as well as iSVM as regards their accuracy and model complexity, using several benchmarks.

7.2. Description of the Scenario

We assume a scenario with two training phases. The first phase is offline assuming complete available training data and it enables multiple passes through the data. A second phase assumes online training, i. e., training data points are available one after another and they can be used for training only once. We assume training data with original class labels for both phases. The original class labels of the test data are used to evaluate the performance of the classifiers explained later on only.

Such a scenario is interesting for systems delivered with a pre-trained classification model which can be personalised/adapted for the users needs, e. g., via direct user interaction (Fig. 7.1). A user can specify new labelled training data, e. g., instances of already known classes or completely new ones.

[1] iSVM code: `https://github.com/diehl/Incremental-SVM-Learning-in-MATLAB`

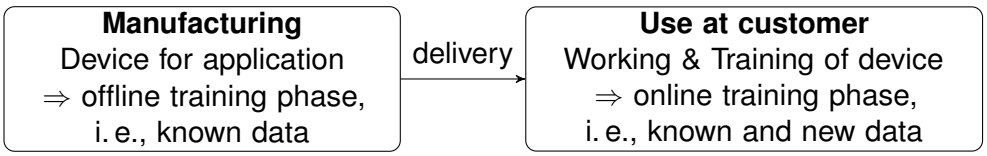
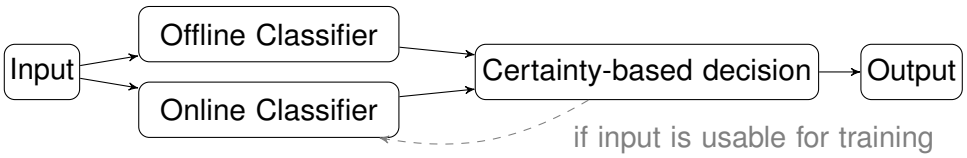


Figure 7.1.: A possible application scenario

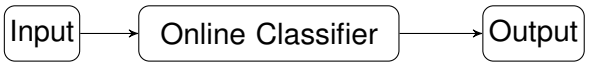
7.3. Research Questions

Regarding the described scenario we are analysing several architectures, e. g., like those shown in Fig. 7.2 with respect to their behaviour. We tackle the questions:

1. What are the strengths and the drawbacks of the analysed architectures?
2. Can they deal with the catastrophic forgetting effect?
3. What is the role of metric learning in this setting, and are specific adjustments required?



(a) A classifier architecture combining offline and online learning



(b) A pure incremental online learning architecture

Figure 7.2.: Two different types of lifelong learning architectures. (a) is a combination of offline and online learning and it will be introduced in this chapter. The second architecture (b) is purely online learning. An example thereof was discussed in the previous chapter.

7.4. Related Work

As mentioned in section 7.2, we want to deal with lifelong learning in a changing environment which is the topic of recent surveys (Gama et al., 2014; Ditzler et al., 2015). One group of approaches dealing with changing environments use *ensembles* of classifiers, e. g., Polikar et al. (2001); Ditzler et al. (2010). Either the output of the classifiers is combined or a single classifier of the ensemble is selected in order to provide a class label for a given data point. An early approach with classifier selection was proposed by Dasarathy and Sheela (1979). They

combine a linear classifier with a k-NN for a binary task. The latter classifier is only selected near the class border where the linear classifier is unreliable. A counterpart to this static selection is the dynamic one proposed by Woods et al. (1997). They compare so-called local accuracies of the classifiers and choose the most reliable one. This selection is dynamic since the accuracies are updated with every new training sample.

We study an architecture with two classifiers and a dynamic classifier selection like Woods et al. (1997), but we use certainty values of the classifiers instead of local accuracies.

7.5. Combining Offline and Online Learning (OOL)

The main idea of the OOL architecture (Fig. 7.3) proposed in Wersing and Queißer (2014); Queißer (2012) combines a pre-trained offline classifier that provides known knowledge of the desired task with an incremental online classifier that learns special characteristics or new classes during the lifelong learning application in its (dynamic) environment. The classifier that is more reliable in its classification with respect to the certainty value for a given data point will define the class label. A schematic system architecture is displayed in Fig. 7.3. In the following, we elaborate the basic components. A crucial point is how to define and train an online classifier, and how to combine both classifiers, using the new RelSim certainty measure.

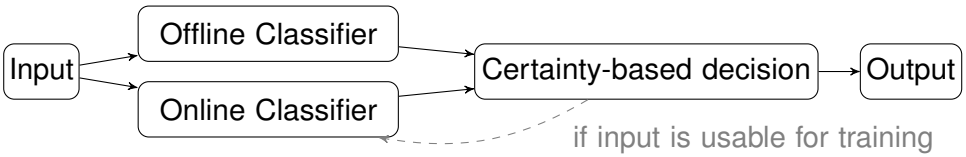


Figure 7.3.: OOL architecture combines an online and offline classifier (Queißer, 2012).

Input A data point $\mathbf{x} \in \mathbb{R}^M$ is the input of the system. It can contain a label y denoting a new or one of the Z known classes. Both classifiers receive the input.

Offline Classifier This container refers to any offline classifier that provides a certainty value in addition to its predicted class label. We choose a pre-trained LGMLVQ model (instead of GMLVQ (Queißer, 2012)), i. e., a set of trained prototypes $\mathbf{w}_j$ for the Z known classes with their local metrics d_j. The offline classifier is static, i. e., it preserves the gained knowledge during the whole application. The output of the offline classifier is a class label y^{off} and the related certainty value ($\text{RelSim}^{\text{off}}(\mathbf{x})$) for an input data point $\mathbf{x}$ that is passed to the classifier selection.

Online Classifier This container refers to any online incremental classifier that provides a certainty value in addition to its predicted class label. It will be the focus of this chapter to analyse how to efficiently set up and integrate ioLVQ to obtain a balance of stability and flexibility of the overall system. Here we choose the ioLVQ based on an LGMLVQ model instead of an incremental GMLVQ which only allows prototype insertion (Queißer, 2012). Update rules of the online classifier are performed under some conditions (see classifier selection) only. During training, it is vital that suitable training data are available for the online classifier in order to gain specific knowledge that is unknown in the offline classifier. The output of the online classifier for an input data point $\mathbf{x}$ is a class label y^{on} and the related certainty value ($\text{RelSim}^{\text{on}}(\mathbf{x})$) which is passed to the classifier selection.

Classifier Selection The classifier selection of the system has two functions: 1) mediating between the online and offline classifier based on their certainty values and 2) deciding if the input is usable as training instance for the online classifier. These two functions work as follows: 1) we mimic the idea of Woods et al. (1997) with a better suited certainty measure instead of a heuristic based on error counting (Queißer, 2012). We choose the more reliable classifier which decides the class label of the system

$$y^{\text{sys}} := \begin{cases} y^{\text{off}}, \text{if } \text{RelSim}^{\text{off}}(\mathbf{x}) \geq \text{RelSim}^{\text{on}}(\mathbf{x}) \\ y^{\text{on}}, \text{else} \end{cases} .$$

2) training of the online classifier is controlled as follows: Firstly the input of the system necessarily needs a class label, i. e., $(\mathbf{x}, y)$. Mainly the system does not use the input data point for training if the offline classifier is reliable and provides the correct class label. If at least one of the three following conditions is valid, the online classifier uses the input for training.

1. The online classifier is more reliable than the offline one

$$\text{RelSim}^{\text{on}}(\mathbf{x}) > \text{RelSim}^{\text{off}}(\mathbf{x}) \tag{7.1}$$

2. The offline classifier is more reliable but it provides a wrong class label.

$$\text{RelSim}^{\text{on}}(\mathbf{x}) < \text{RelSim}^{\text{off}}(\mathbf{x}) \wedge y^{\text{off}} \neq y \tag{7.2}$$

3. Both models are unreliable (here: $\Gamma = 0.6$).

$$\max\{\text{RelSim}^{\text{on}}(\mathbf{x}), \text{RelSim}^{\text{off}}(\mathbf{x})\} < \Gamma \tag{7.3}$$

Output The output of the system, for a given input, is the selected class label y^{sys} with its certainty value.

The extended OOL architecture provides a particularly simple scheme promising an easy control of the model adaptation. It is also designed to perform better than pure incremental approaches with respect to the CFE because of its static offline classifier. We show that the extended OOL architecture enables an effective learning in dynamic settings where the data distribution used for learning is changing, i. e., the crucial assumption of training data being i. i. d is violated.

7.6. Experiments on Artificial and Benchmark Data

We consider experiments on three data sets: a 2D data set (Blossom) for visualisation and analysing the different architectures, the U.S. Postal Service (USPS[2]) Handwritten Digits data as a benchmark data set, and a real-life Outdoor data set (Outdoor) obtained from a robot during its application. Each data set is divided into two partitions. One partition of the data simulates a data distribution which is only accessible in the second training phase of the system while the other partition/distribution provides training instances for both phases.

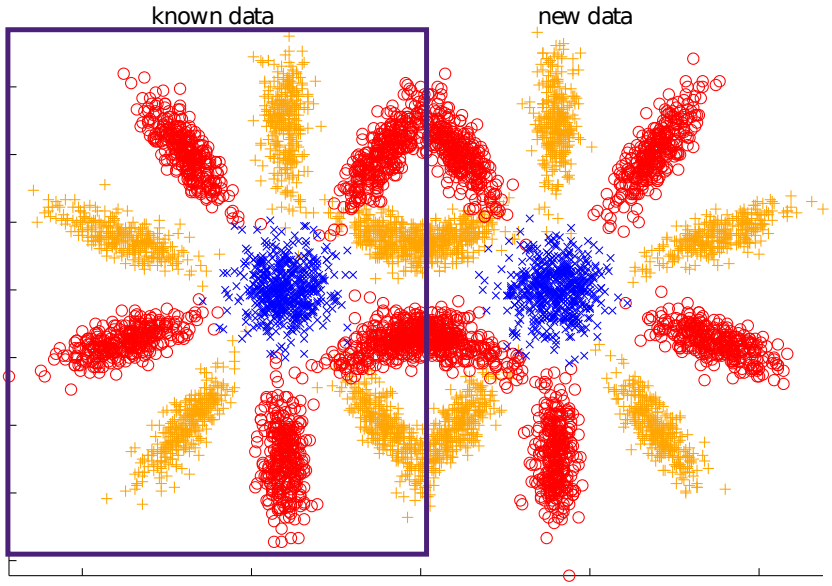


Figure 7.4.: The Blossom data set. The colour encodes the class of the data points. The distribution of the left blossom provides samples for the offline and the online training phase whereas the distribution of the right blossom is only available during the online training phase. This data simulates a changing distribution of known classes.

[2]Data is obtained from: `http://www.cs.nyu.edu/~roweis/data.html`

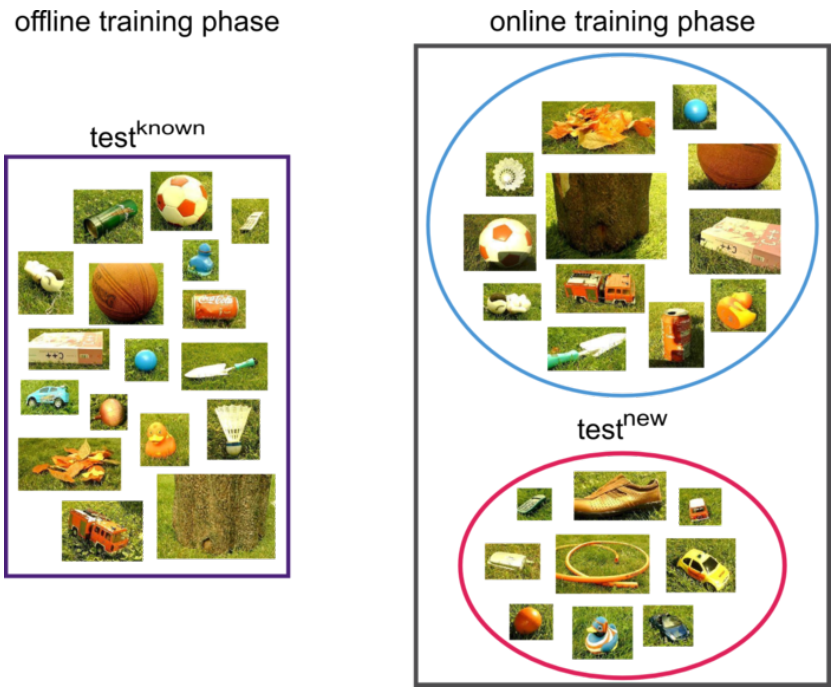


Figure 7.5.: A possible split in offline and online data of the Outdoor data

Blossom: Artificially created three class data set (Fig. 7.4). Each blossom forms one partition of the data.

USPS: This data set contains 8-bit greyscale images of ”0” to ”9”; 1100 samples per class. The dimension of the data is reduced to 30 with principal component analysis (van der Maaten, 2013). Each half of the classes forms a data partition.

Outdoor: Already introduced in section 6.5. Each half of the classes forms a data partition (Fig. 7.5). Figure 7.6 shows a 2D visualisation of the data with the Fisher t-SNE[3] (Gisbrecht et al., 2015) showing highly overlapping classes.

Unless stated otherwise, we consider randomly ordered data presented in a single pass, like in a streaming setting, using constant learning rates (except for the batch LGMLVQ). The evaluation for the Blossom and the USPS data set is based on a 10-fold cross validation with ten repeats as follows: Each data set is decomposed in two balanced partitions as stated before. The first partition symbolises data which is available for offline training and data which can be encountered during use. The second partition symbolises data which can be only encountered during use and it contains new classes or drifted data of known classes (new data distribution, Fig. 7.5). For a 10-fold cross validation, each

[3]Thanks to Alexander Schulz for providing the code.

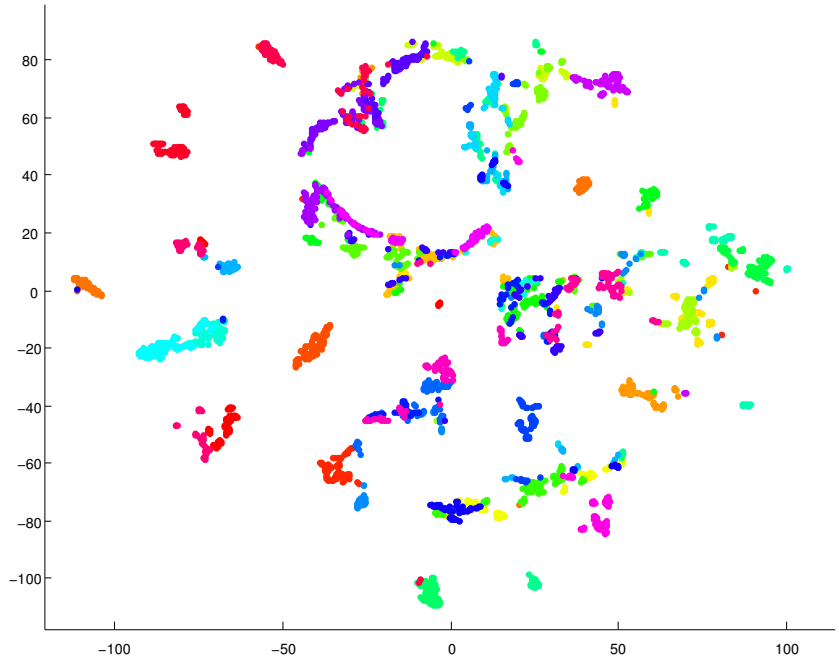


Figure 7.6.: A 2D visualisation of the Outdoor data set with the Fisher t-SNE (Gisbrecht et al., 2015). The different colors denote the different classes.

partition is divided into 10 folds (see Fig. 7.7). The offline training set $\text{train}^{\text{off}}$ consists of four folds of the first half of the data and the online training data train^{on} (encountered during use) consists of four folds of the first half and of eight folds of the second half of the data. To evaluate the different approaches in a proper way we define three different test sets: $\text{test}^{\text{known}}$, test^{new} and test^{all}. The test set $\text{test}^{\text{known}}$ consists of two folds of the first half of the data and its error shows the performance on these data which samples the static ground knowledge (i. e., the performance tests the stability of a learning method). The test set test^{new} contains two folds of the second half of the data and it shows the accuracy of newly gained knowledge in the online part of the system, i. e., it tests the flexibility of a method. The overall performance is checked with the last test set $\text{test}^{\text{all}} = \text{test}^{\text{known}} \cup \text{test}^{\text{new}}$, i. e., it tests the capability to deal with both challenges, stability and plasticity.

In the following experiments we compare the primary architecture by Queißer (2012) and the OOL architecture on an artificial and on the USPS data set. Table 7.1 contains the results and it can be seen that the OOL performs better than the other one. Queißer's architecture needs more prototypes since there is no prototype deletion mechanism. Hence the number of prototypes is monotone increasing over time and presented training data. Furthermore the accuracies are lower than those of the OOL on all three test sets. The reasons therefore are that the OOL has an improved strategy for placing new prototypes and a deletion

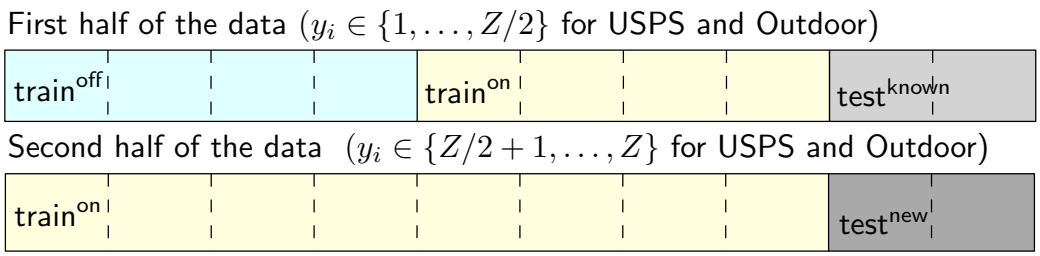


Figure 7.7.: Each data set is partitioned in two balanced sets: the first and the second half of the data. Two folds of each set form a test set: test^{known} and test^{new} respectively. For the Blossom data set holds that the left blossom forms the first half of the data and the right blossom forms the second half (Fig.7.4).

of needless ones as well as an enhanced classifier selection based on certainty values rather than based on error counting. Therefore we will not report further results for Queißer's architecture in the following.

Data	Quantities	OOL	Queißer (2012)
Blossom	$\|W\|$	27.0	40.5
	test^{known}	95.0	94.2
	test^{new}	89.0	84.1
	test^{all}	92.0	89.1
USPS	$\|W\|$	25.6	284.6
	test^{known}	92.6	80.4
	test^{new}	91.7	91.2
	test^{all}	92.1	85.8

Table 7.1.: Comparison of the architecture by Queißer and the proposed architecture. We report the average accuracy on different test sets and the related average prototype number $|W|$ of the architectures. The value $|W|$ denotes the average prototype number of the architecture which is the sum of the prototypes of the online and offline classifier.

The Outdoor data set has the special characteristic that its data points represent sequences of objects which we will use in further experiments. One sequence consists of serial frames of an object approached by a mobile robot. For each object there exist ten different sequences, each with ten data points which are treated as a unit during the experiments (except in batch LGMLVQ). We use the same training and testing scheme as aforementioned with the difference that only 5-partitions are generated (since there are too few sequences for 10-folds) and that we choose randomly the sequences for the different train/test sets. The training data are presented in a single pass with random sequence order. This experiment is performed ten times.

We compare the OOL architecture with four alternatives: ioLVQ, Combi, Batch (Tab. 7.2) and iSVM defined as follows:

	OOL	ioLVQ	Combi
$\text{model}^{\text{off}}$	LGMLVQ	ioLGMLVQ	LGMLVQ
init with $\text{model}^{\text{off}}$	no	yes	yes
model^{on}	ioLGMLVQ	ioLGMLVQ	ioLGMLVQ

Table 7.2.: Explanation of the incremental LVQ approaches used in the experiments.

OOL This is the proposed extension of the architecture (Wersing and Queißer, 2014; Queißer, 2012) described in section 7.5. The static offline classifier (batch LGMLVQ) is trained offline with the training set $\text{train}^{\text{off}}$ and the training set train^{on} is applied to the architecture and data points fulfilling one of the three conditions (7.1), (7.2), (7.3) are used to train the online classifier (ioLVQ). The combination of an offline trained classifier with an online incremental classifier is suggested to avoid the CFE, especially to move in the direction of guaranteeing certain performances on defined data, e. g., for safety relevant applications. The static offline classifier is fine tuned with respect to the offline available data and preserves this knowledge during the whole application. This offline classifier is combined with a flexible online classifier which is capable of adapting to new data and classes.

ioLVQ The ioLVQ is a purely incremental classifier without a guaranteed static offline part as provided by OOL. The resulting classifier uses only incremental learning and serves as comparison. We expect that it more easily suffers from the CFE while potentially being more accurate for the online data. Here, we first train the incremental online LGMLVQ (ioLGMLVQ) using $\text{train}^{\text{off}}$, before using train^{on}.

Combi This classifier is similar to the previous one, but uses batch LGMLVQ (Schneider et al., 2009a) instead of ioLGMLVQ for the first training phase with data $\text{train}^{\text{off}}$. In this setting, we can expect to get the best prototypes with respect to the available offline data. The obtained classifier provides an initialisation for the training of the ioLGMLVQ with train^{on}. Combining these classifiers has the advantage of getting the best prototypes with respect to the offline available data and using this knowledge as reasonable initialisation for the online incremental learning part. This boosts the performance of the online classifier since it does not start from scratch. Because there is no static knowledge of the offline trained data, the CFE can have a bad impact on the performance of this classifier like in ioLVQ.

Batch The batch LGMLVQ (Schneider et al., 2009a) trains on the union of both training sets and it is a powerful classifier if the training data is accessible at once. It is inept for tasks where training data is observed during the application especially

if there is a concept drift or new classes, since the number of classes and the number of prototypes per class have to be set as parameters before training. It serves as comparison to the architecture, showing a baseline performance.

iSVM (Diehl and Cauwenberghs, 2003) The used iSVM code is adapted to bound parts of the memory that we can set the number of the so-called reserve vectors of the realisation[1] to one corresponding to the full online case we consider here. The iSVM trains with the data from train^{off} followed by the train^{on} data. Required parameters are the soft-margin regularisation constant C and the Gaussian kernel parameter γ. Note, that we use the binary classification realisation in a one vs. all mode enabling multi-class classification. The iSVM serves as a comparison to a state of the art approach focusing on class borders rather than typical representatives like LVQ.

	LGMLVQ				ioLVQ				iSVM	
	epochs	$\mathbf{w}$/class	ε_w	ε_m	$g_{\max}$	r_{num}	ε_w	ε_m	C	γ
Blossom	100	5 or 1	0.3	0.07	20	320	0.1	0.01	2^5	$2^{3.5}$
USPS	100	1	0.4	0.01	20	400	0.4	0.01	2^5	$2^{3.5}$
Outdoor	250	5 or 6	$4{\cdot}10^{-5}$	10^{-5}	5	900	$4{\cdot}10^{-5}$	10^{-5}	2^5	0.02

Table 7.3.: Parameters of the offline and the online classifier of the different approaches

Table 7.3 shows the parameters of the experiments and Tab. 7.4 shows the results. It contains the accuracies of the three test sets and the number of prototypes, respectively the number of support vectors. We mark in boldface the results laying on the Pareto-front with respect to the two objectives: accuracy on test^{all} and the number of prototypes. A result is Pareto optimal if there is no other result which is better in both objectives. First we discuss the results of the single approaches and then we draw conclusions from their comparison to each other.

Evaluation of the different approaches

OOL It reaches the highest accuracies on the known data for all three data sets (for the used LVQ classifiers). The good performance on test^{known} highlights the robustness of the architecture only for LVQ classifiers against the CFE. It performs well on the online data, even if the accuracies obtained on test^{new} are slightly lower than for the known data. In particular the accuracy on test^{new} of the Outdoor data set performs worse compared to the other approaches. The analysis of this effect indicates that the classifier selections needs to be improved. The different scaling of the certainty measure in the offline and online classifier seems to cause this effect. Further analysis of this issue are addressed in section 7.8. The overall accuracy of the OOL is mainly the average of the test accuracies on test^{known}

Data	Quantities	OOL	ioLVQ	Combi	iSVM	batch LGMLVQ
Blossom	$\|W\|$	27.0	25.4	**25.2**	**76.8**	22
	test^{known}	95.0	90.7	91.7	97.5*	98.2
	test^{new}	89.0	92.3	93.1	98.6	98.2
	test^{all}	92.0	91.6	**92.4**	**98.1**	98.2
USPS[4]	$\|W\|$	25.6	17.5	**14.8**	**304.3**	11
	test^{known}	92.6*	86.7	91.9	91.1	95.8
	test^{new}	91.7	94.3	94.0	95.6	95.4
	test^{all}	92.1	90.5	**93.1**	**93.3**	95.6
Outdoor	$\|W\|$	**127.9**	128.3	**120.5**	**1035.3**	200
	test^{known}	65.1*	44.8	42.8	52.8	63.1
	test^{new}	38.9	58.2	60.9	65.2	58.0
	test^{all}	**53.2**	52.1	**51.9**	**59.2**	61.3

Table 7.4.: We report the average accuracy on different test sets and the corresponding average prototype number $|W|$ of the approaches. The value $|W|$ denotes the average prototype number of the approaches or the support vectors. For the OOL architecture it is the sum of the prototypes of the online and the offline classifier. Results that are in bold lay on the Pareto front with respect to overall accuracy on the test set test^{all} and the model complexity $|W|$. The results marked with * are the best ones compared to all other approaches on test^{known} disregarding batch LGMLVQ. The t-test for the result of OOL of Blossom is significantly different ($p < 10^{-3}$) to iSVM and batch LGMLVQ. For the USPS data set, every alternative to OOL provides significantly different ($p < 10^{-3}$) results. Since there are only 10 results averaged for Outdoor, no t-test is done.

and test^{new} and competes with the other approaches. Hence there exists a lower bound of the performance (performance of model^{off}) of the architecture, even if the online classifier has little knowledge or if it is disturbed, e. g., by noise.

ioLVQ The ioLVQ reaches the highest accuracies on the test set test^{new} for all three data sets. This shows that the classifier learns new data fast and that it provides high performances especially on data that correspond to the last seen training data. Table 7.4 shows clearly that the accuracies of test^{known} are below test^{new} meaning the classifier suffers from a moderate forgetting effect. The difference is significant for the Outdoor and the USPS data set. The overall accuracy of the test^{all} is comparable to the other approaches.

Combi The Combi classifier gets the highest accuracies on test^{new} but it suffers from a forgetting effect comparing the accuracies of test^{known} and test^{new}, too. This behaviour equals those of the ioLVQ. The offline trained prototypes (batch) used as initialisation for the ioLVQ in the online training affect the needed number

[4]Scaled to [0,1] for the iSVM.

of prototypes (Combi needs less prototypes than ioLVQ) and the accuracies of the classifier in general. Combi has higher accuracies on the Blossom and the USPS data set than the ioLVQ. While the differences among the accuracies of Combi and ioLVQ are small (except $\text{test}^{\text{known}}$ of USPS), they exist. Hence it seems suitable to start training in batch mode for partly available data.

Batch This classifier provides the baseline accuracies, i. e., reachable accuracies if the whole data is available at once with a defined number of prototypes.

iSVM The iSVM reaches its highest accuracy on test^{new}. It exhibits the same effect as Combi and ioLVQ that $\text{test}^{\text{known}}$ has a lower accuracy than test^{new} which means it suffers from a forgetting effect, too. Comparing the results to the other approaches one has to say, that the accuracy is marginally higher except for $\text{test}^{\text{known}}$. Note that the iSVM uses more resources than the other approaches.

Comparison The results in Tab. 7.4 indicate that there is no approach that surpasses all the other approaches in terms of performance and model complexity. It turns out that the approaches have different properties and suit different needs which we discuss subsequently. In general iSVM can often provide a slightly better accuracy at the costs of a greatly enhanced model complexity. This effect can be expected due to the representation of a model in terms of its class borders for iSVM in contrast to its class representatives for LVQ. However, if model complexity is not an issue, a similar hybrid system could easily be built based on SVM as content of the two containers for an online/offline model. Note, that such a system requires an online suited certainty measure usable with an iSVM classifier. When comparing incremental versions which are based on prototypes, the following conclusions can be drawn: The OOL surpasses all the other approaches on the set $\text{test}^{\text{known}}$ which means it preserves offline gained knowledge during the whole application. The accuracy on the test set test^{new} is comparable to the other approaches but slightly lower except for the Outdoor data set. In general the ioLVQ and Combi classifiers are good alternatives if focusing on a high accuracy on previously seen data, especially if no offline data are available. If one is more focused on preserving knowledge and gathering new information during the application, then the OOL is a better choice. In particular for Outdoor which is a real-life data set obtained from a robotic scenario, the OOL performs well since the overall performance on the test set surpasses ioLVQ and Combi and is only slightly lower than the performance of the iSVM while using a fraction of resources. This makes OOL more attractive for devices with limited memory. Additionally the performance of the OOL on $\text{test}^{\text{known}}$ is higher than the performance of the competitors on the Outdoor data which highlights the robustness on known data/classes. The

baseline performance provided by the batch LGMLVQ is nearly reached by the incremental approaches. The number of the prototypes for the OOL, ioLVQ and Combi is still in the same range as the number of prototypes of the LGMLVQ while the number of the support vectors of the iSVM is significantly higher. This shows the effectiveness of the incremental LVQ approaches, i. e., they provide a high performance with a low number of prototypes.

Discussion of the properties of the different approaches

The analysed approaches have different properties and hence are suited for different lifelong learning tasks. In the following we discuss the availability of offline data, the model size, the overall performance of the system and the performance on known data/classes. This might help to choose the best approach for the given circumstances of the desired task.

Offline data If data is offline available two schemes are possible: i) using them as training set for the offline classifier of the OOL or of Combi or ii) simply present them in an online fashion to the other approaches (ioLVQ, iSVM) to train the related classifier. The results in Tab. 7.4 indicate that it is beneficial to choose scheme i). If the offline data is used in the OOL, one gains a high and robust performance on the offline data with respect to the known data/classes during the ongoing application. Using Combi instead provides a highly flexible classifier that is less robust against the CFE but needs less prototypes compared to the ioLVQ or the number of support vectors of the iSVM. If there is no offline data available one can only choose between the ioLVQ and the iSVM.

Model size In the proposed experiments, the approaches OOL, ioLVQ and Combi use less prototypes compared to the batch LGMLVQ while the iSVM needs significantly more support vectors. With the parameters g_{max} and r_{num} one can regulate indirectly the tendency of the complexity of a model (see section 6.5). Basically both parameters influence directly the insertion and the deletion strategy of the algorithm. A hard coded upper bound of prototypes is possible (Grbovic and Vucetic, 2009), e. g., for applications with limited resources.

Overall performance If one is mainly interested in the performance of the system, the iSVM is a proper choice paying the price of a highly complex model, i. e., it needs a high number of support vectors. Also the parameters C and γ have to be defined. They do not directly correspond to an intuitive interpretation which enables the user to set these values by hand. Their choice is critical because they highly influence the performance of the iSVM. The other approaches

reach performances comparable to iSVM using less resources and the required parameters g_{max} and r_{num} are intuitively understandable and easier to choose.

Performance on known data/classes In safety relevant applications it is often very important to guarantee performances on defined data during the whole application time. With this view, the experiments have shown that the OOL provides good performances on test$^{\text{known}}$ highlighting the robustness against the CFE while especially ioLVQ suffers from this effect on the evaluated data sets. Combi and iSVM tend to provide worse results on known data compared to the performance on new data. Hence, the best choice would be the OOL.

7.7. Summary of the Main Findings

The proposed OOL architecture is well suited for lifelong learning tasks or streaming data. The general architecture consists of three parts: a static, pre-trained offline classifier, a flexible online classifier and a dynamic classifier selection part. Combining an offline and an online classifer has the advantage of learning new knowledge with the online classifier while preserving the ground (offline) information with the offline classifier. We have investigated the performance of the OOL using the example of learning vector quantisation: LGMLVQ in batch mode serves as the offline classifier of the architecture, the LGMLVQ in incremental mode offers a flexible online classifier of the architecture and a dynamic classifier selection can be based on a certainty measure that is well suited for LVQ approaches. We analysed the properties and performances of the OOL in comparison to several other incremental approaches (ioLVQ, Combi, iSVM). The experiments on artificial and real-life data have shown that the OOL is more robust against the CFE compared to other state of the art approaches and that it is able to gather new knowledge during application like other incremental learning approaches. Additionally, it turned out that none of the analysed approaches surpasses the others with respect to performance and model complexity for all test sets and criteria. We discussed the properties of the approaches in order to give hints which one to choose for a desired task. The analyses give an overview of some existing lifelong learning approaches highlighting the robustness of the OOL against the CFE.

So far we studied scenarios where the confidence estimation of both classifiers is reasonable. Due to the integrated metric learning of the classifiers, it can happen, that the confidence estimation can become invalid in particular for the offline classifier. When this happens, we will call it confidence drift (explained in the next section). Such a confidence drift can badly influence the OOL performance. Within the next section we explain this drift and we give a solution for the problem, enabling the OOL to deal with confidence drift.

7.8. Online Metric Learning for an Adaptation to Confidence Drift

So far we investigated the performance of the OOL architecture in lifelong learning scenarios. In this section the main focus lies on the confidence estimation of the classifiers and the related classifier selection mechanism. Note, that this general architecture, though conceptually simple and efficient, bears a severe risk: while the offline classifier remains valid for regions of the data space which are covered by the offline training set, virtual concept drift, i. e., a change of the probability $p(\mathbf{x})$, can cause a wrong confidence estimation of the offline classifier for regions which become relevant during online learning. Provided training and test data were known in advance, this setting could be accounted for suitable data by reweighing schemes which is one of the most promising techniques to deal with covariate shift (Sugiyama and Kawanabe, 2012). In our setting, this information is not available a-priori, and there occurs the need to adapt the classifier confidence estimation according to the observed drift. We refer to this problem as *confidence drift*: while training, the estimation of the confidence of the offline or online classifier becomes invalid, such that their combination leads to a false result, albeit the single classifiers are still valid in their respective regions.

Metric learning autonomously adapts the metric parameters of the classifiers, i. e., the internal representation of the given data according to their relevance for the classification task. While the latter aspect allows such classifiers to efficiently deal with high dimensional or heterogeneous data sets where the standard Euclidean metric would be inept, it adapts the data representation according to the given setting. Hence the OOL architecture fails whenever concept drift requires a different data representation, and it leads to a wrong confidence estimation in such cases. In this section, we propose a simple yet efficient approach dealing with such confidence drift: we suggest an online adaptation of the internal data representation by means of online metric learning for both, offline and online classifier, and a self-adjusted weighting scheme adjusting the relevance of the respective data representation for confidence estimation.

An Example of Confidence Drift

Assume a binary 2D 4x4 chequerboard data set (Fig. 7.8). We consider three data splits for the first and second training phase of the architecture (Fig. 7.9). The offline classifier is trained on the known data only (first training phase) while the architecture during use encounters known and new

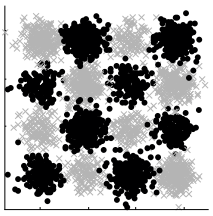
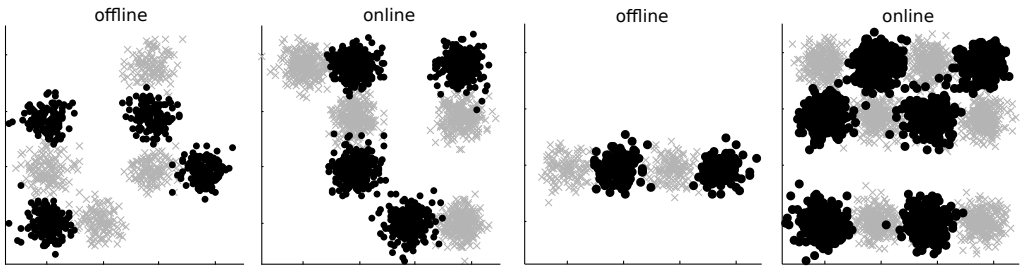


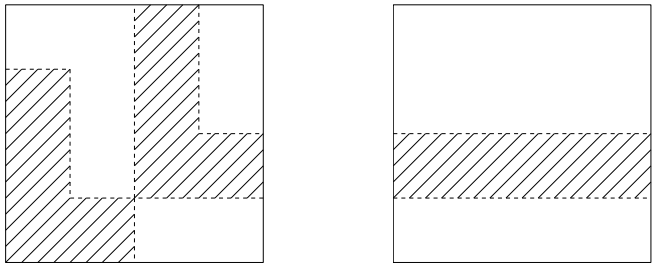
Figure 7.8.: Chequerboard data (Colour encodes class)

data (second training phase). This leads to a desired data space partitioning with respect to the classifiers, as shown in Fig. 7.10.



(a) Setting A (b) Setting B

Figure 7.9.: The chequerboard data is used in three different settings: In setting A both dimensions of the data are needed to classify the data in the known as well as in the new part of the data. In the known data of setting B, one dimension is enough for classification but in the new data both dimension are again needed. Setting C is setting B inverted.

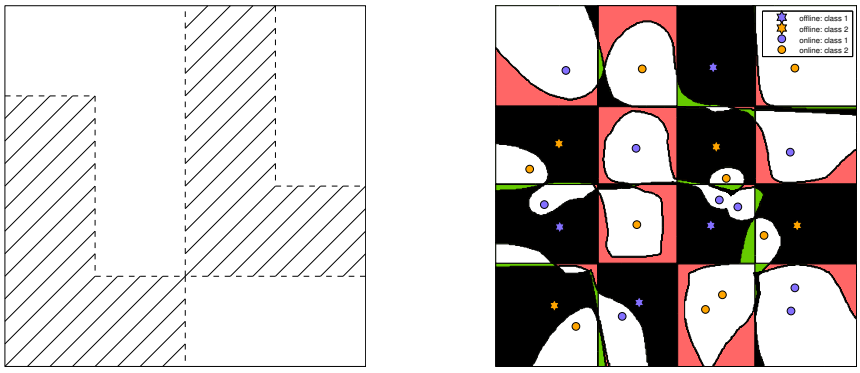


(a) Setting A (b) Setting B

Figure 7.10.: The desired partitioning of setting A and B. The hatched areas should be classified by the offline classifier while the white areas are related to the online classifier. The desired partitioning of setting C is (b), inverted.

In the following, we visualise results of the architecture as shown in Fig. 7.11. A desired result should have a low error rate which means in the visualisation should be as few red and green areas as possible. Furthermore the offline classifier should be responsible for the known data and the online classifier should be responsible for the new data. This can be seen if the desired data space partitioning for a defined setting match the corresponding visualisation of the result (hatched/white areas should match the black/white areas in the visualisation).

Analysing setting A, one recognises that all dimensions are necessary in order to classify the data and this holds for the known and the new data. Contrary in setting B the known data can be classified using only one dimension of the data while classification of the new data still has to use both dimensions. This difference matters for metric learning, in particular for the metric of the offline classifier in

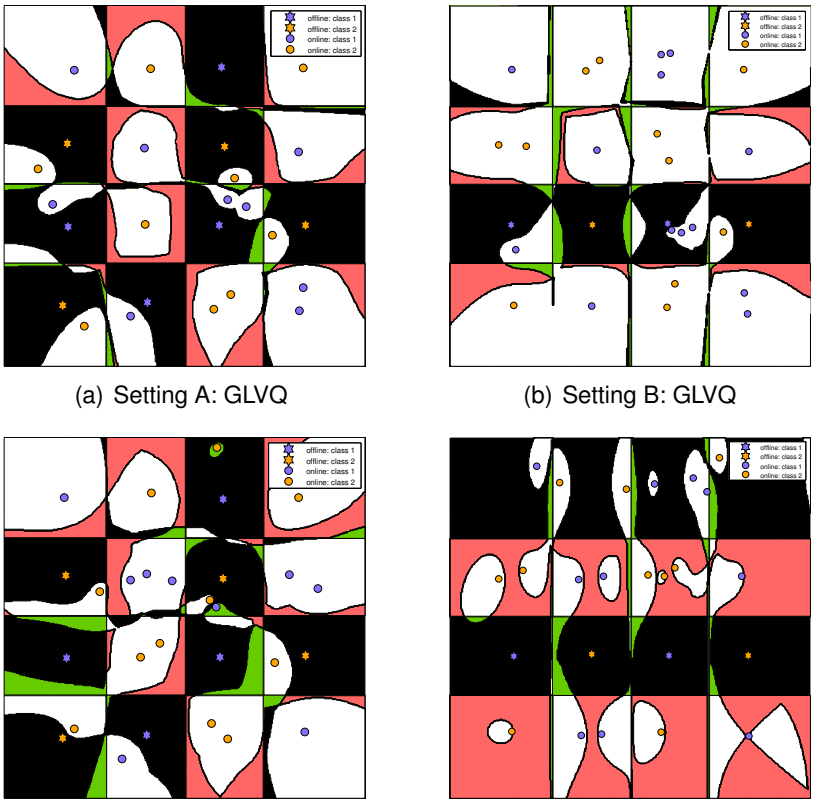


(a) Setting A: desired partitioning (b) Setting A: a GLVQ result

Figure 7.11.: Explanation of the result figures. (a) shows the desired partitioning of setting A and (b) shows a result of the architecture with GLVQ models. The black areas are classified by the offline classifier while the online classifier is responsible for the white areas (correctly classified). Red areas are misclassified by the offline classifier and the green areas are wrong classified by the online classifier. The prototypes of the classifiers are shown as stars and circles. The colour encodes their class. A desired result should have a low error rate which means there should be as few red and green areas as possible. Furthermore the offline classifier should be responsible for the known data and the online classifier should be responsible for the new data. This can be seen if the desired data space partitioning for a defined setting match the corresponding visualisation of the result (hatched/white areas in (a) should match the black/white areas in (b)).

setting B. In this case the matrix Λ will only focus on one dimension $\Lambda_{11} \approx 1$ while all other elements are approximately zero. Figure 7.12 contains exemplary results of setting A and B for the described architecture with metric learning (GMLVQ) and a similar architecture without metric learning, simply using the Euclidean distance (GLVQ). The GLVQ architecture shows a similar data partitioning like the desired ones in Fig. 7.10. Hence for this architecture there is no problem with the classifier selection part. Analysing the GMLVQ architectures, one can see that setting A works fine too but setting B does not due to its trained offline metric. For setting B a confidence drift occurs for the GMLVQ architecture which cannot be followed by the static internal metric since is suited for the known data but which is improper for the new data. The problem occurs due to the fact that the offline classifier deals with a too simplistic data representation: it disregards data dimensions which are irrelevant for the offline training data, but which become relevant for future data points; in consequence, confidence estimation as measured in form of the certainty of the classification which is correlated to the classifier confidence is wrong in the online scenario. Note that an abstraction from irrelevant regions is crucial for LVQ classifiers to provide good generalisations for high dimensional data sets. At the same time, it is a-priori unclear which abstractions are too rigid for future data, hence online adaptation of this confidence measure is necessary.

In the next paragraph, we develop a strategy which allows the offline classifier to adapt its certainty estimation in the case of confidence drift.



(a) Setting A: GLVQ (b) Setting B: GLVQ

(c) Setting A: GMLVQ (d) Setting B: GMLVQ

Figure 7.12.: The images show the results for the specific setting and a specific architecture. The black areas are classified by the offline classifier while the online classifier is responsible for the white areas (correctly classified). Red areas are misclassified by the offline classifier; green areas are wrong classified by the online classifier. The prototypes of the classifiers are shown as stars and circles. Their colour encodes class. GLVQ refers to the architecture using the standard Euclidean distance instead of metric learning.

Online Metric Learning for an Adaptation to Confidence Drift

The offline classifier is static and it consists of a set of prototypes W^{off} and a trained metric using Λ^{off}. Since we want to keep the gained knowledge of the offline classifier, we do not change the classifier itself. Instead, we introduce a metric Λ^{corr} which can be used to correct the wrong data representation for its confidence estimation, taking into account concept drift on the online training set.

The RelSim is computed by using the combination

$$\Lambda^{\text{conf}} = \alpha \cdot \Lambda^{\text{corr}} + (1 - \alpha) \cdot \Lambda^{\text{off}}, \quad \alpha \in [0.1, 0.9] \tag{7.4}$$

where the scaling parameter α is initialised with 0.1 and the matrix Λ^{corr} is initialised with Λ^{off}, and both are adapted in online training, whenever the offline classifier is erroneous, i. e., (7.2) holds. Figure 7.13 shows a schema thereof where Λ^{corr} is adapted using the GMLVQ update rule considering W^{off} together with Λ^{corr} as GMLVQ model (but no prototype update is done). In case of a point with new class label (unknown to the offline GMLVQ), $\mathbf{w}^+$ and d^+ are taken from the online classifier to update Λ^{corr}.

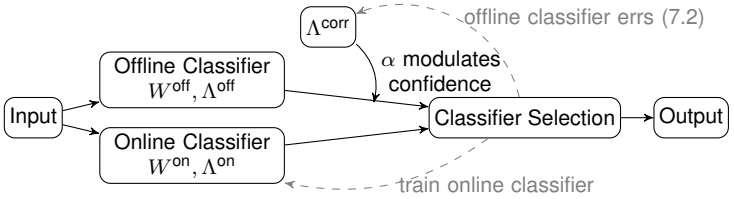


Figure 7.13.: Scheme of OOL extension for confidence drift adaptation. The offline classifier ($W,^{\text{off}}\,\Lambda^{\text{off}}$) still predicts the label y^{off} but in order to calculate RelSim$(\mathbf{x})$ the metric Λ^{conf} (7.4) is used which takes into account the quantities $\Lambda,^{\text{off}}\,\Lambda^{\text{corr}}$ and α.

The choice of α is crucial, and a static prior choice is usually suboptimal. Therefore we use a simple Hebbian adaptation strategy for $\alpha \in [0.1, 0.9]$ as follows: We introduce a counterΓ (initialised with zero) which counts the relevance of Λ^{corr} versus Λ^{off} on the training data. It is decreased by one when ever

$$\text{RelSim}^{\text{on}}(\mathbf{x}) < \text{RelSim}^{\text{off}}(\mathbf{x}) \wedge y^{\text{off}} = y \tag{7.5}$$

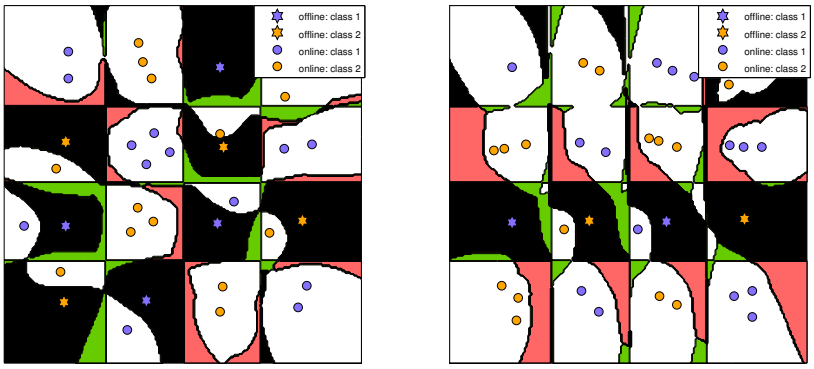
and it is increased by one whenever (7.2) holds. We enforce an upper and lower bound $a_1 \leq \Gamma \leq b_2$ on the counter to prevent plateaus for the optimisation, and we disregard counters approximately equal to zero $a_2 \leq \Gamma \leq b_1$ for an adaptation of α where $a_1{<}a_2{<}0{<}b_1{<}b_2$. Assume a small step size $\varepsilon_\alpha{>}0$, the update rule of the scaling parameter α is

$$\alpha = \alpha \begin{cases} -\varepsilon_\alpha, & a_1 < \Gamma < a_2 \wedge \text{(7.2) holds} \\ +\varepsilon_\alpha, & b_1 < \Gamma < b_2 \wedge \text{(7.2) holds} \end{cases}.$$

This corresponds to a Hebbian scheme since the relevance of Λ^{corr} is increased/ decreased depending on its contribution to a correct classification.

Exemplary results for setting A and B of the architecture with confidence adaptation are visualised in Fig. 7.14. It can be seen that this change enables the hybrid

architecture to learn correctly for both settings. In the next section we evaluate the proposed architecture on several data sets.



(a) Setting A (b) Setting B

Figure 7.14.: The images show the results of OOL with confidence drift adaptation for setting A and B . The black areas are classified by the offline classifier while the online classifier is valid in white areas (correctly classified). Red areas are misclassified by the offline classifier; green areas are wrong classified by the online classifier. The prototypes of the classifiers are shown as stars and circles. Their colour indicates the class.

Experiments

We consider experiments on four data sets: two 2D data sets (Blossom, Chequerboard) mainly for visualisation, the USPS, the Letter (Bache and Lichman, 2013), and the Outdoor data set as benchmarks. Each data set is divided into two partitions: known and new data. Known data are available during both training phases while instances of new data are only accessible in the second phase.

Chequerboard: Artificially created two class data set (Fig. 7.8) with 8000 points per class. Each chequerboard *field* contains 1000 data points.

Chequerboard noise: The Chequerboard data are enhanced by a noisy third dimension which contains values uniformly distributed in (0,1).

Letter: The Letter data (Bache and Lichman, 2013) contain 20,000 instances with 16 dimensions, related to 26 classes (capitals A . . . Z). Fourteen letters form the first partition of the data and the other letters belong to the second partition.

Blossom, USPS, Outdoor[5]: Used as in the previous section 7.6.

We use the same experimental setting as in the previous section 7.6. Reminder: We assume randomly ordered data points which are used only once, like in a

[5]For the current experiments, we use the data randomly ordered, neglecting the internal sequences.

streaming setting, using constant learning rates (except for the batch GMLVQ). The evaluation of the data sets is based on a 10-fold cross validation with ten repeats as follows: Each data set is divided into two parts: known (offline training: $\text{train}^{\text{off}}$) and new data (online training: train^{on}) as stated before. For a 10-fold cross validation, each partition is divided into 10 folds (see Fig. 7.7). For a proper evaluation of the architecture, we consider three different test sets: $\text{test}^{\text{known}}$, test^{new} and test^{all} as before.

		GMLVQ				ioLVQ		
	training epochs	$\mathbf{w}$/class	ε_w	ε_m	g_{max}	r_{num}	ε_w	ε_m
Chequerboard	100	4/2/6 (A/B/C)	0.3	0.07	20	320	0.1	0.01
Letter	300	1	0.01	0.001	20	400	0.01	0.001

Table 7.5.: Parameters of the offline and the online classifiers

Table 7.3 and Tab. 7.5 show the parameters of the experiments and Tab. 7.6 and Tab. 7.7 show the results. They contain the accuracies of the three test sets and the number of prototypes, and (if available) the trained α value.

Data	Setting	α	$\lvert W\rvert$	$\text{test}^{\text{known}}$	test^{new}	test^{all}	
Chequerboard	A	0	28.14	96.45	90.64	93.60	plain OOL
	B	0	23.54	94.94	49.35	60.65	
	C	0	27.42	95.58	43.71	56.81	
	A	0.1127	36.58	93.91	93.78	93.79	with Λ^{corr}
	B	0.5217	36.84	92.06	90.28	90.91	
	C	0.6336	45.90	92.39	93.50	93.12	
Chequerboard noise	A	0	27.20	95.91	90.05	92.93	plain OOL
	B	0	22.25	94.82	49.08	60.54	
	C	0	25.36	95.12	43.51	56.37	
	A	0.1094	35.83	93.49	93.55	93.52	with Λ^{corr}
	B	0.5387	36.81	92.80	90.59	91.14	
	C	0.6348	45.50	93.31	93.56	93.57	

Table 7.6.: The results of the Chequerboard data. We report the average accuracy on the test sets, the related average prototype number $|W|$ of OOL, and the α values. Results with $\alpha = 0$ belong to the plain architecture without confidence drift adaptation.

Evaluation of the Chequerboard Data (Tab. 7.6) Architecture without concept drift adaptation: Firstly, the results of Chequerboard and Chequerboard noise are similar which means that the integrated metric learning detects and neglects the noisy dimension. This indicates the merit of metric learning. Secondly, the number $|W|$ of the prototypes is in the same range for all three settings indicating that the number of prototypes is kind of invariant to the used setting. Thirdly, the accuracy

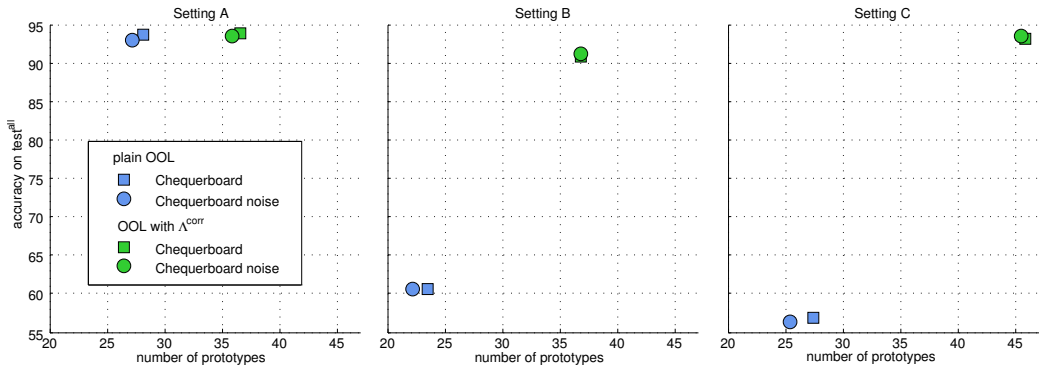


Figure 7.15.: The results of test$^{\text{all}}$ on the Chequerboard data for three settings

values of test$^{\text{known}}$ show a desired performance for all settings. The same holds for test$^{\text{new}}$ but for setting A only. In case of setting B or C the accuracy is low which means that a confidence drift happened and the plain architecture is unable to deal with it. This causes also low test$^{\text{all}}$ accuracies (setting B/C, Fig. 7.15).

Architecture with concept drift adaptation: This extended architecture provides similar results for both Chequerboard data sets. Hence noisy dimensions without information are neglected, too. In comparison to the plain architecture, the number of used prototypes increases slightly. In setting A there is no confidence drift and the results reflect this, since the value of α in both cases is near the lowest possible value. Hence the architecture highlights that there is no need for a confidence adaptation. The other two settings need a confidence adaptation which is nicely reflected by the α values which improve together with the trained Λ^{corr} the performance especially on test$^{\text{new}}$ and test$^{\text{all}}$.

Data	α	$\lvert W\rvert$	test$^{\text{known}}$	test$^{\text{new}}$	test$^{\text{all}}$
Blossom	0	27.33	96.49	97.13	96.80
	0.1	30.72	95.59	97.55	96.61
USPS	0	64.19	91.29	59.54	75.26
	0.6986	77.25	89.13	84.70	86.98
Letter	0	212.88	80.62	70.09	75.70
	0.7797	234.01	80.13	77.91	79.14
Outdoor	0	286.63	89.89	66.65	78.18
	0.4135	286.64	90.03	66.74	78.05

Table 7.7.: The results of the Blossom and the benchmark data. We report the average accuracy on the test sets, the related average prototype number $|W|$, and the α values of OOL. Results with α=0 belong to the plain architecture without confidence drift adaptation.

Evaluation of the Blossom and the Benchmarks (Tab. 7.7) Since the known and the new data of the Blossom data set are similar except an offset in one coordinate, the trained metrics of the online and the offline classifier should be comparable and hence there is no confidence drift. This can also be seen on the already high accuracy values of the plain architecture. We expect that the extended architecture reflects this with a low α value and similar performances than the plain architecture. As can be seen in Tab. 7.7 this expectation is met.

For the USPS, one can see that there might be a confidence drift since the performance on test^{new} and test^{all} of the plain architecture is bad. Comparing those results with those of the extended architecture, it turns out that the adaptation to confidence drift is used and that it helps to improve the performance on these two test sets. The same facts can be seen for the Letter data which emphasises the usefulness of confidence drift adaptation.

The Outdoor data are a difficult task as mentioned previously. The OOL architecture, however, manages to retain known knowledge ($\text{test}^{\text{known}}$) and to gain new information. Comparing the plain architecture with the extended one, it turns out that the results are similar. The reason therefore could be, that there is no confidence drift.

Summing up the Results

The OOL architecture constitutes an approach suitable for lifelong learning scenarios and for streaming data. As pointed out in this section, the classifier selection strategy can cause problems in the case of confidence drift. This pitfall occurs in the context of metric learning whenever the internal data representation of the online and offline classifier mismatch. We have proposed an efficient extension to this architecture which allows an adaptation to confidence drift adaptation based on an online metric learning and weighting scheme. We analysed the OOL architecture based on popular GMLVQ schemes. It turned out that the proposed modification is effective for a number of artificial and benchmark data: it enables an efficient scheme to avoid confidence drift for metric-based classifiers wherever it would be present with the original OOL scheme.

7.9. Conclusion: Answering the Research Questions

In this chapter we analysed several architectures which are suited for lifelong learning. We evaluated them on several data sets and we discovered a problematic effect: confidence drift which can happen in the OOL architecture. In the following, the main findings of this chapter are summarised regarding the research questions.

1. What are the strengths and the drawbacks of the analysed architectures?

A detailed discussion on this topic can be found on page 95. Summing up leads to: The OOL has its strength in combining a static classifier trained on known data with a flexible classifier which learns new information from the online training. It is also a simple scheme usable for various classifier types, e. g., LVQ, SVM, decision trees. The strength of the ioLVQ is its flexibility which enables a fast adaptation to new data. For this approach it would be beneficial to have a mechanism avoiding oscillation prototype insertion and deletion in areas with noise or highly overlapping classes. So far such a mechanism is not included. Combi has its strength in extensively using the available data in order to gain as much information as possible as initialisation for the ioLVQ used in the online training phase of the system. Hence it has the same drawbacks as the ioLVQ and that it assumes access to training data for the first training phase. The mentioned approaches have a moderate model complexity which the iSVM does not. The iSVM has a high model complexity and the choice of needed model parameter is difficult. With suitable parameter settings, the iSVM provides a model with high performance.

2. Can they deal with the catastrophic forgetting effect (CFE)?

   As the experiments in section 7.6 have shown, none of the approaches suffers from a CFE but one can see that the performance on $\text{test}^{\text{known}}$ is often worse than the performance on test^{new}. Hence we have a forgetting effect of different strength except for the OOL. This approach seem not to suffer from a forgetting effect due to its static offline trained classifier.

3. What is the role of metric learning in this setting, and are specific adjustments required?

   Metric learning can be integrated into ioLVQ, Combi and the OOL, and similar to the batch scenario, it plays a crucial role to achieve a good classification accuracy. Including metric learning in the classifiers of the OOL architecture, changes their internal data representation which causes what we refer to as confidence drift. We proposed an elegant solution for this challenge by incorporating an intuitive confidence adaptation scheme.

In the next chapter, we analyse their performance on a real world scenario from the advanced driver assistance systems domain due to their promising results.

8. Application on Road Terrain Detection

Chapter overview *In this chapter we apply the basic ideas of the LVQ approaches for online learning of the preceding chapter for a real world application, i. e., to the detection of road terrain for driver assistant systems. We compare our online trained LVQ approach to a reference system proposed by Fritsch et al. (2014). Therefore, we use a possible online learning strategy which we experimentally test on Kitti[1] data. According to our data analysis, we found out that we can achieve the same performance with our approach using only six instead of eighty-two features as used by the reference system. We show that our online trained LVQ approach is able to outperform the reference system in some cases.*

8.1. Motivation

There are quite a number of machine learning systems which tackle problems in advanced driver assistance. For instance there exist adaptive intersection assistants, lane change assistance, lane departure warning systems, and cruise control. In this context, machine learning techniques are used to detect objects (Ren et al., 2015; Struwe et al., 2013), e. g., cars or pedestrians, or to do behaviour prediction (Platho and Eggert, 2012; Bonnin et al., 2014) for example. Some assistants require a detection of the road terrain in front of the ego car. A representative of such a system is the road terrain detection system (RTDS) proposed by Fritsch et al. (2014).

In related practical applications, often the road detection classifiers are trained offline on a huge data base. Nevertheless such a training data base does not contain data of all possible driving situations. This is also true for the RTDS which detects drivable road on a given image. Although it is performing well on general scenes like e. g., highway, country road and inner city scenes which are well presented in an offline training data base, there are scenes with difficult conditions as e. g., strong shadows, overexposure or a non-standard road appearance. Assuming that such scenes are special, they are a minority compared to frequently occurring scenes. Online learning appears to be a good way to learn the characteristics of this minority if those scenes happen. In the context of online

[1] The data is obtained from: `http://www.cvlibs.net/datasets/kitti/index.php`

learning one major challenge is to get ground truth training data during usage. We want to cope with this challenge and we investigate if online trained classifiers can support the RTDS in such scenes following the basic scheme of the architecture detailed in the former chapter. Since the LVQ schemes used in online learning scenarios performed well (chapter 6 and 7), we study their suitability for the current purpose[2]. Since the RTDS is a very advanced system compared to the online trained LVQ classifiers, we frame-wise compare our approach with a subsystem of the RTDS only. Firstly this is an adequate comparison as we explain latter on and secondly if the online LVQ classifiers perform better than the subsystem of the RTDS, it can be used to enhance the overall RTDS system.

In this chapter we want to tackle the related research questions, as stated in the next section.

8.2. Research Questions

We consider online learning for road terrain detection using LVQ schemes. In this context, the following questions are studied:

1. How do we get ground truth training data for online learning?

2. Are the LVQ approaches suited for road terrain detection in the assumed context?

We tackle these questions after outlining related work and introducing the RTDS.

8.3. Road Terrain Detection – Related Work

In the following we want to point to related approaches which have mechanisms to gather training data without manual labelling. Such a mechanism is important for online learning because in related scenarios a manual labelling is infeasible.

Road terrain detection is a sub domain of terrain assessment. The aim of the later topic is to distinguish traversable and non-traversable areas in the environment while road terrain detection aims at distinguishing road and non-road. The three subsequent approaches are related to both topics and they provide mechanisms for gathering training data for online learning.

[2]Thanks to Stephan Hasler and Thomas Weisswange for fruitful discussions on the topic and the data interface as well as to Jannik Fritsch for the initial idea.

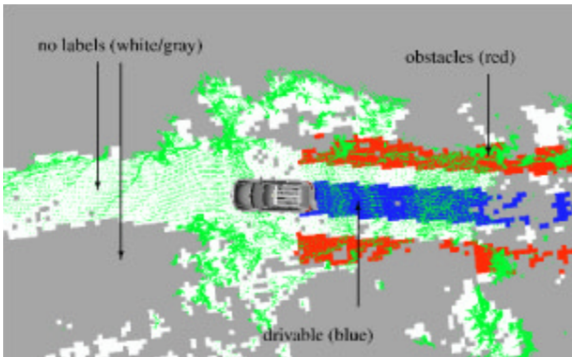


Figure 8.1.: Thrun et al. (2006): Data labelling through human driving. Traversed areas (blue) are assumed as *drivable* and two stripes with a defined distance to the left and to the right are assumed as *non-drivable* (red).

Thrun et al. (2006): A milestone in this domain was Stanley, an autonomous driving car which won the Darpa Grand Challenge 2006. Although the performance was impressive, the authors state that Stanley is unable to navigate in traffic which would be a core capability of an autonomous driving car. A final step in their approach focusses on parameter tuning for the trained Gaussians which are used to model the class probabilities. For this purpose a human drives the car exclusively on drivable terrain such that labelled training data is collected as shown in Fig. 8.1. Hence, the area which is traversed by the car is labelled as drivable and two stripes with a defined distance to the left and right are assumed as non-drivable. This labelled data is used to tune the parameters. The authors claim that even such approximate labelling suffices to improve their terrain detection performance. For our purpose, the idea to collect traversable samples from traversed area is suitable but the collection of non-traversable samples seems too simplified for real traffic scenes, e. g., intersections or roads with multiple lanes.

Álvarez et al. (2013): The authors propose an approach for road detection which is based on an appearance-based model. The benefit of their approach is that a predefined area of a training image is assumed to be road and hence they omit a time consuming hand-labelling of training data. Pixels related to this area are represented by a linear combination of colour planes whose weights yield minimal variance in the road area. Classification is done by comparing input image pixels against the learnt model. One disadvantage is that there is no guarantee that the predefined area really contains road.

Berczi et al. (2015): In general training data is created as follows. In a first step images of road scenes are collected and in a second step the road area is annotated by a human. This process is time consuming and not applicable for

online learning. In the robotics domain, Berczi et al. propose a training scheme with Gaussian processes which automatically assigns labels to data seen during application. They present a new data representation and their approach learns directly from human demonstration. Hence, a controlled robot drives through a region and the traversed areas by the robot in the related images are assumed as traversable. Examples of non-traversable areas are collected when the bumper of the robot is triggered or if the human controller pushes a button indicating the same. The basic idea of collecting samples of traversable areas is similar to Thrun et al. (2006). In order to collect samples for non-drivable area it is impossible to use a bumper or a similar mechanisms in the car domain.

Conclusion: In our scenario, we deal with real-world traffic scenes/data without restrictions. In Álvarez et al., positive samples are taken from a predefined area of the taken traffic scene images which has no guarantee that this area really contains road. The same holds for the collection of negative samples of the approach by Thrun et al., because they also predefine areas where they expect non-traversable terrain. Their approach is inept in real-world traffic scenes, because of e. g., intersections or roads with multiple lanes. In our scenario, we collect training data of road samples with a mechanism mimicking the intuitive approach of Thrun et al. (2006); Berczi et al. (2015). In this way, the area traversed by the car is assumed to be road. Afterwards we briefly introduce the reference system, the RTDS.

8.4. The Road Terrain Detection System

The RTDS (Fritsch et al., 2014) is a vision-based system recognising road. To be more specific, it aims at detecting semantic road which means the area where a car is allowed to drive (e. g., detecting lanes of the road, exit lanes, acceleration and deceleration lanes). This excludes for instance parking lots and breakdown lanes. A simpler task is to detect road-like area which refers to the area of the scene which has the same appearance as the current road. There the classification of parking lots and breakdown lanes as road-like area is correct (Fig. 8.2).

(a) Road-like area (marked in yellow) (b) Semantic road area (marked in green)

Figure 8.2.: There are two road types: road-like area and semantic road. Note it is harder to classify semantic road than road-like area versus non-road. (Image from Kitti data.)

The RTDS (Fig. 8.3) consists of two stages related to the two road types (road-like area, semantic road). The first stage of the RTDS, the local visual appearance stage, detects the road-like area of an image based on colour and texture features (overall 82 features) and it generates a confidence map for the given image where high values indicate road. The second stage uses this confidence map to compute spatial features thereon. These features integrate the confidences of possible road area on a large region and hence integrate geometric knowledge of the scene. The aim of the second spatial stage is to detect semantic road which is the output of the RTDS. Both stages are trained offline on a large data base. For further details we refer to the thesis of T. Kühnl (2013). For our scenario, the RTDS is trained on the Kitti road benchmark (Fritsch et al., 2013) and it serves as reference system for the experiments.

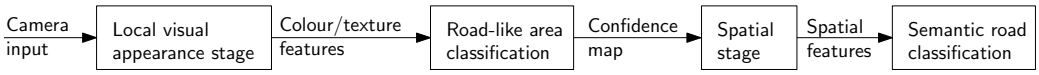


Figure 8.3.: The algorithmic steps of the road terrain detection system

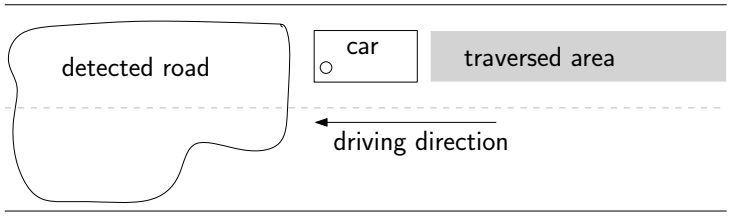


Figure 8.4.: How the system collects ground truth data

8.5. The Scenario

As aforementioned, we assume a good road detection of the RTDS in general traffic scenes and a possibly bad detection in scenes with difficult conditions like strong shadows or overexposure. For such difficult scenes, we assume online available training data which are collected in the following way. We assume a mono camera in front of the car which does the images recordings. These images are stored in a database and when the car traverses areas of the road, the related areas in the images are labelled as road subsequently according to Pomerleau (1993) and Thrun et al. (2006). Hence the ground truth data contains only samples of road (see Fig. 8.4). Samples of non-road cannot be easily collected during the application. If depth information is available, it might be possible to gather instances of non-road. But in our case we have RGB images only. Hence we use a predefined subset of offline available non-road training data for online learning.

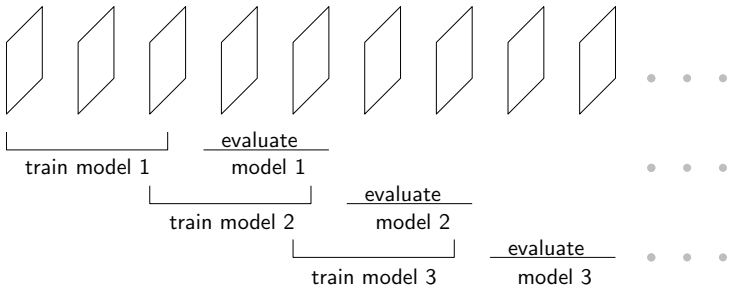


Figure 8.5.: Scheme of the used online training scenario – We assume a stream of frames. On a sliding window of frames with a fixed number (here three) a single model is trained and evaluated on the next frames (here two) till the next model is ready.

We rely on the Kitti road benchmark (Fritsch et al., 2013) for our experiments because it is popular in this area and it contains a huge variation of recordings. The data set contains images with a resolution of 1242 x 375. In this context, we assume the following realisation of the scenario: A GMLVQ model is trained on a sequence of frames and it is used on the next frames for evaluation until the next GMLVQ model is ready (trained on the next sequence of frames, see Fig. 8.5). In the context of this chapter, this learning strategy is referred to as online. We use a stream (0086) with overexposure from the Kitti benchmark because such a stream is challenging for the RTDS. This stream contains a labelling of semantic road and non-road. For our purpose we annotated the ego lane of the car in addition. The Kitti stream 0086 contains highly overexposed frames and it contains strong light changes since sunny regions are followed by strong shadows and vice versa.

The next section consists of a data analysis together with results for the described scenario.

8.6. Experimental Studies

We want to gain insight how good online trained GMLVQ models perform in our scenario, especially in scenes with difficult conditions like strong shadows or overexposure. Since those conditions change the appearance of the scene, we use appearance-based features (colour/texture) to train the GMLVQ models. We rely on the same feature set which is used to train the first stage of the RTDS. We assume that the appearance-based features suit the desired road detection task.

In a first step we do a feasibility study with the GMLVQ on available data (appearance-based features) in order to gain information about performance and the data itself. Therefore we use offline training. The goal of this analysis is to judge whether the model suits the data and to determine a baseline performance that can be achieved when using a GMLVQ model for road detection. After the

feasibility study, we frame-wise compare the performance of the first RTDS stage with the online trained GMLVQ models on a specific stream.

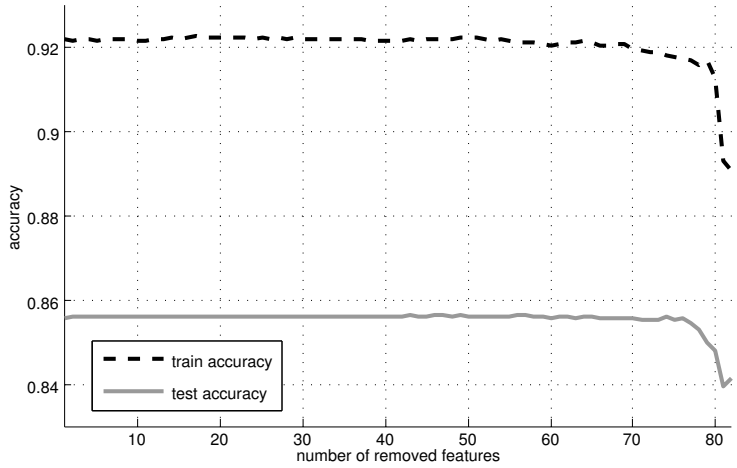


Figure 8.6.: Offline GMLVQ performance when features are removed

Offline Data Analysis

Considering an LVQ model as online classifier, in a feasibility study we check the baseline performance of an offline trained GMLVQ classifier with one prototype per class on Kitti data. From initial experiments we figured out that a higher number of prototypes per class results only in a nominal performance improvement. In this experiment we gain insight into how the appearance-based features contribute to the road-like classification performance. According to a offline trained GMLVQ model the features are ranked with respect to their relevance in the related Λ matrix (3.4) of the model. Subsequently, the feature with the lowest rank (contribution) is removed till only one feature remains. Figure 8.6 shows the accuracy of the GMLVQ models offline trained for each feature set. From this visualisation we draw the conclusion that relying on only 6 out of 82 features is a good choice since the classification performance stays nearly the same. The six most important features consist of five colour features and one basic texture feature. Additionally the computational effort of the metric adaptation of the GMLVQ can be minimised since training of a 82x82 Λ matrix can be replaced by training of a 6x6 Λ matrix. The baseline performance of the offline GMLVQ is acceptable, since the reference system achieves about 85 % test accuracy. Hence we expect accuracies in the same range within the following experiments with online training.

Online Learning from Image Stream Data

We trained and evaluated GMLVQ models as explained in the preceding section (Fig. 8.5). Figure 8.7 shows exemplary results. One can see that the current GMLVQ model of the scene is able to detect the road-like area quite well (Fig. 8.7), doing only few errors at the borders. Hence, the ego lane of the car is safely detected. These promising results are striking since the GMLVQ models are only trained on some frames and only use six features (colour/texture).

(a) (b)

Figure 8.7.: GMLVQ results. Green: correctly classified road-like area; blue: correctly classified non-road; red: misclassified road-like area; yellow: misclassified non-road

In average, the performance of the online trained GMLVQ model is comparable with the RTDS. Therefore we do a frame-wise evaluation in order to see in which situations the online models perform better than the first stage of the RTDS regarding recognising road-like area. Figure 8.8 shows the frame-wise evaluation of the first stage of the RTDS and the GMLVQ models. As indicated through light yellow and light grey, there are sunny and shady images in the stream 0086. The GMLVQ models perform better than the first RTDS stage on sunny frames. In case of shady images, the first RTDS stage mostly performs better than GMLVQ with few exceptions. The classification on such images is more challenging since there can be a high variation in the intensity of the shadows which can badly influence an online trained model if this intensity changes fast. One can also see that on transitions between sunny and shady images the performance of the GMLVQ model drops before it is increasing again. This *delay* is due to the number of training frames. The higher this number is the longer this delay occurs.

Nevertheless the GMLVQ models perform better on some images of the stream. Assuming a mechanism which switches between the GMLVQ and the RTDS such that the classifier with the best performance is chosen, the overall performance can be improved as well. An architecture consisting of two classifiers connected via a classifier selection was discussed in the previous chapter (OOL architecture). There, a classifier is selected for the classification of a single data sample (e. g., single pixel of frame). In the approach of this chapter a classifier would be

selected for a complete frame and hence for several data samples (many pixels). A major challenge in this context is a certainty measure which is suited to judge the reliability of the different classifier types on the given input frame. A reason therefore is that it seems not sufficient to use the entire confidence estimation of the classifiers because they are completely different in their type and scaling for instance. The mechanism, described in section 7.8, tackles this issue but for classifiers of the same type such that it is unclear if the used mechanism can be directly transferred to the issue at hand. This analysis is subject of future work.

8.7. Conclusion: Answering the Research Questions

The main findings of this chapter are summarised in the following in order to answer the research questions of section 8.2.

1. How do we get ground truth training data for online learning?

   Analysing the available online training data leads to the following conclusions: Instances for road-like area can be collected from the traversed road area but the more interesting instances from the borders of the road are difficult to retrieve. Furthermore data from non-road cannot be gathered during the application and has to be stored beforehand. This has the huge drawback that instances of non-road could be of the same appearance as the current road scene which would confuse the classifier during training.

2. Are the LVQ approaches suited for road terrain detection?

   First experiments with the offline trained GMLVQ on the Kitti benchmark give insight into the data. It turns out that the same classification performance can be reached with only 6 out of regularly 82 features. Since the number of features defines the size of the quadratic Λ matrix in the GMLVQ model, the computational effort is reduced. This insight initiated an analysis [3] if the number of features can also be lowered in RTDS without performance loss. Furthermore the current online model uses the appearance-based features of the data only, leading to a model which recognises road-like area rather than semantic road. Hence, a reasonable comparison can only be done between the first RTDS stage and the online trained GMLVQ models. In our experiment on a challenging Kitti stream, we could show that online trained GMLVQ models are suited because they can perform better than the first RTDS stage on particular image scenarios. Especially on sunny images this seems to be the case whereas on shady ones the classification task is more difficult. The reason therefore is the variability of shadows. They can be

[3] At Honda Research Institute Europe

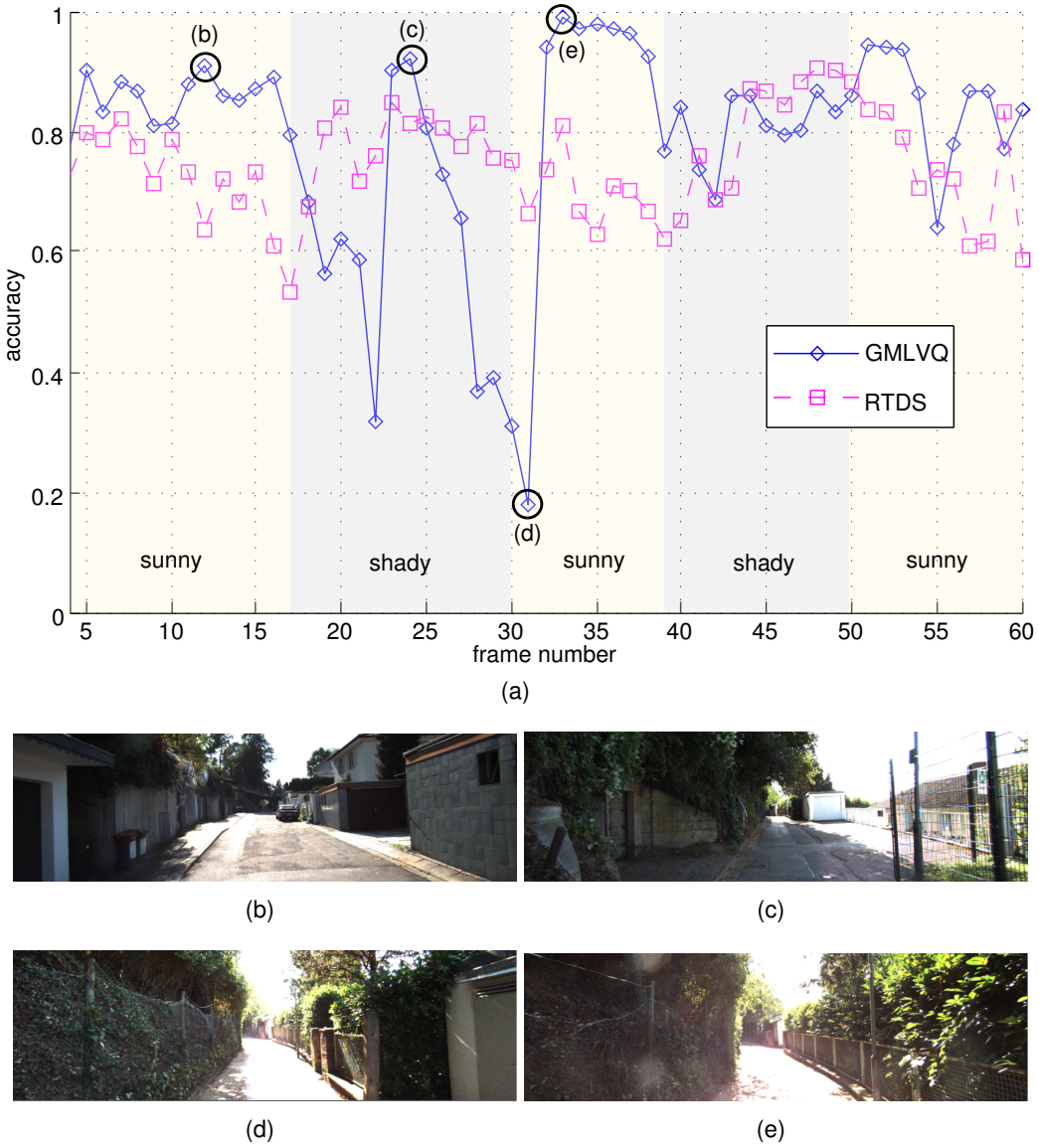


Figure 8.8.: (a) Frame-wise comparison of the GMLVQ and the RTDS (first stage) performance on the Kitti stream 0086 . We omit the first frames since they are used to train the first GMLVQ model and hence no reasonable comparison can be done. (b)-(e) sample images of the stream.

really dark, e. g., from a house or less dark if it is a shadow from a treetop for example. This different shadows cause different colour changes which have an impact on the classification performance especially if they change frequently. Nevertheless we could show that online trained models improve the performance on particular images.

9. Conclusion

Automated data analysis is an important tool to discover complex internal structures of data and it is necessary in settings with so-called big data where a manual processing of data is impossible. One particularly relevant area of automated data analysis, which has been central in our research, is classification which can be equipped with a reject option. In general, a classifier provides a class label for given data without any information if the assigned class labels are reliable. Examples of such classifiers are described in chapter 3.

Within this thesis we focused on prototype-based classifiers in particular LVQ schemes. There are several reasons for this choice. LVQ schemes have a simple classification scheme, they can integrate the powerful concept of metric learning, and they are especially suited for online learning scenarios which was demonstrated in previous work, e. g., in Kirstein et al. (2005, 2008, 2009, 2012); Kietzmann et al. (2008); Queißer (2012). In the context of lifelong learning, certainty measures are particular important. This is why we studied certainty measures suited for prototype-based classifier, for rejection (chapter 4 and 5), lifelong learning (chapter 6) as well as classifier selection (chapter 7).

In chapter 2 we provided the fundamentals of certainty measures as well as for global and local rejection. Also we gave a literature overview and we showed our taxonomy of existing approaches. We found out that there are well studied rejection strategies e. g., for SVM and k-NN, but only few for LVQ approaches. Furthermore, global rejection got lots of attention while local rejection seemed not to be a popular research target.

In chapter 4 we proposed deterministic certainty measures suited for LVQ classifiers which worked well in our experiments. Also we showed that our certainty measures can compete with probabilistic counterparts. Such that a deterministic certainty measure together with an LVQ classifier is a reasonable choice.

In chapter 5 we studied local rejection strategies which rely on a partitioning of the input space, as e. g., the Voronoi cells of LVQ classifiers. We compared global and local rejection strategies for several classifier types (LVQ, SVM, DT) and we gave suggestions when to use which strategy. One main challenge for local rejection is the selection of appropriate local thresholds which gets more complicated with an increasing number of thresholds. A simple straight forward solution is to choose them based on an empirical evaluation which can be time consuming and suboptimal. To overcome these drawbacks, we proposed two algo-

rithms for finding appropriate local thresholds: (i) a dynamic programming scheme which determines the optimal local thresholds for given data with respect to any classifier with a certainty measure, and (ii) a fast greed approximation thereof. Both algorithms provide the full ARC together with optimised local thresholds for any defined costs for rejects and wrong classifications. Hence, if the setting or the costs change, local thresholds can be looked up without recalculation. From a theoretical point of view, we link the problem of finding optimal local thresholds for a given number of false rejects with the multiple choice knapsack problem.

In chapter 6 we directly applied one of our certainty measure in the context of lifelong learning. Classifiers used in lifelong learning have to deal with the so-called stability plasticity dilemma which means they have to be flexible enough to integrate new content without forgetting already acquired knowledge. In lifelong learning, we used certainty information to indicate areas in the data space which are lacking an adequate model representation (uncertain classification) or areas which are noisy (e. g., due to over-fitting). We analysed a new LVQ variant (ioLVQ) for this domain which combines the following benefits of earlier approaches: it is cost function based, it adds and deletes prototypes on demand based on certainty information, and it integrates metric learning. Besides the aforementioned advantages of an LVQ classifier, it offers intuitively understandable parameters which control the classifier complexity and it is easily possible to set an upper bound on the number of prototypes. This possibility is especially important for applications which have limited memory or computational resources, e. g., on autonomous robots. Often such applications have to deal with dynamically changing environments, i. e., changing data distributions which lead to the so-called concept drift. There are mechanism detecting when concept drift happens. They can either be active or passive. Since the ioLVQ adds and removes prototypes with respect to the encountered data, it follows a concept drift passively.

In chapter 7 we studied an alternative to ioLVQ, the OOL architecture which retains defined knowledge over the whole application time due to a static offline classifier which is combined with a flexible online classifier. With a dynamic classifier selection based on certainty measures, the most reliable classifier is chosen for a given input. We demonstrated that the OOL architecture suffers less than other lifelong learning approaches (e. g., ioLVQ) with respect to a forgetting effect on early seen data. Since the offline and the online classifier can integrate metric learning, their entire data space representation can vary a lot. This difference can be caused by a virtual concept drift which means that the underlying data distribution does not change but the current data does not well represent this distribution. In this case the internal data representation of both classifiers varies which results in different certainty measures which are no longer comparable. We introduce this effect as confidence drift and we proposed a solution for this

problem. Our solution uses an additional metric with the aim of correcting the certainty calculation of the offline classifier. This metric is also trained during application time. By design, the OOL architecture can cope with concept and confidence drift during its application time.

In chapter 8 we analyse if the combination of an offline trained road terrain classifier (RTDS) and an online trained one, is beneficial because of the promising results in our experiments and the advantages of the OOL architecture studied in the previous chapter. Hence, we adopted our concepts for road terrain classification tasks. Since both classifiers are from a different classifier type, it is more challenging to find a good measure for dynamic classifier selection. Therefore, we did a frame-wise performance comparison of both classifiers on an image stream. We demonstrated that the online classifier outperforms the offline classifier in some frames although it is challenging to gather ground truth training data for the online classifier during the application. Hence with a suited classifier selection, the combination of an online and offline road detection classifier can be beneficial.

Outlook

The findings as developed in this thesis gave rise to interesting new research areas as follows:

Generalisation guaranty: Theoretical guarantees on the generalisation ability of global and local thresholds extracted from training data towards test data would stress the usefulness and the suitability of the proposed strategies as already demonstrated in this thesis.

General rejection costs: In this thesis, we demonstrated how rejection works with constant rejection and misclassification costs independently of the data. A straight forward extension is to consider more general costs such as class-related ones for instance. This is especially relevant in applications where misclassifications of different classes have different consequences. In the domain of driver assistance systems for instance, two types of misclassifications can happen. The system wrongly initiates an action of the car which confuses the driver or even more severe it badly interacts with other road users. Such an error is more costly than the error, when the system misses a dangerous situation and does nothing, because the human driver is probably aware of the situation and acts appropriately. Another domain where different costs arise is the medical domain. There it is also more 'costly' to classify an ill person as healthy than vice versa.

Partitioning of the input space: We experimented with different classifiers and their natural induced data space partitioning (e. g., Voronoi cells, leaf areas of a decision tree). Maybe a different kind of data space partitioning and hence local thresholds, e. g., class border related thresholds, can even improve upon the results as obtained in this thesis. This could be relevant for applications where

misclassifications between different classes have different costs and therefore they need different treatment/thresholds.

Rejection for online learning: In this thesis we studied rejection applied on offline trained classifiers with successful results. For future research it would be important to study rejection applied on classifiers used in online learning scenarios. We proposed certainty measures which are suited for online LVQ schemes. Online learning gets more and more important because many applications deal with streaming data or the personalisation of systems for example. A main challenge related to online learning is concept drift as aforementioned. In this context, it would be relevant to know if the global/local thresholds for rejection need to be adapted according to observed training data or if for instance the costs for wrong classifications/rejects change over time. Based on this thesis, the upcoming topic of rejection in online learning opens the way to interesting research questions.

Data Fusion[1]: The main aim in this area is to fuse information from several sensors to achieve an improved performance of a system, e. g., classification accuracy. However, in real-world applications an improved classification is useless until one has a robust mechanism indicating whether one can rely on the classification. If the sensors work correct, the data they collect/transmit are reliable and useful for the execution of the system's task. But data of sensors can be noisy or simply wrong due to a temporary functional disorder, a complete breakdown of sensors or due to other environmental conditions. A system should notice if the gathered data on which to ground its behaviour is reliable or not. In case of distorted or wrong data severe consequences can happen. Hence, it is beneficial to neglect such data only using reliable data. A reject option with an appropriate certainty measure for each sensor can be used to indicate when sensor data is unreliable. Since there is a large variety in sensors, data fusion is spread across many domains, e. g., health care, smart homes, and autonomous driving.

[1] Thanks to Harvey Mitchell for pointing to this topic.

A. Appendix

A.1. Publications in the Context of this Thesis

Journal articles

[J16] Lydia Fischer, Barbara Hammer, and Heiko Wersing. Optimal Local Rejection for Classifiers. *Neurocomputing*, http://dx.doi.org/10.1016/j.neucom.2016.06.038, 2016.

[J15] Lydia Fischer, Barbara Hammer, and Heiko Wersing. Efficient Rejection Strategies for Prototype-based Classification. *Neurocomputing*, 169 (2015) 334 – 342.

Conference articles

[C16] Lydia Fischer, Barbara Hammer, and Heiko Wersing. Online Learning for an Adaptation to Confidence Drift. In *Proceedings of International Joint Conference on Neural Networks - IJCNN International Joint Conference on Neural Networks, IJCNN 2016, Vancouver, Canada, July 24-29* , IEEE, pages 748 – 755, 2016.

[C15b] Lydia Fischer, Barbara Hammer, and Heiko Wersing. Combining Offline and Online Classifiers for Life-long Learning. In *Proceedings of International Joint Conference on Neural Networks - IJCNN International Joint Conference on Neural Networks, IJCNN 2015, Killarney, Ireland, July 12-17*, IEEE, pages 2808 – 2815, 2015.

[C15a] Lydia Fischer, Barbara Hammer, and Heiko Wersing. Rejection Strategies for Learning Vector Quantization. In M. Verleysen, editor, *23th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2015, Bruges, Belgium, April 22-24, 2015*, i6doc.com, pages 7 – 12, 2015.

[C14c] Lydia Fischer, Barbara Hammer, and Heiko Wersing. Local Rejection Strategies for Learning Vector Quantization. In *Artificial Neural Networks and Machine Learning - ICANN 2014 - 24th International Conference on Artificial Neural Networks, Hamburg, Germany, September 15-19, 2014. Proceedings*, ser. Lecture Notes in Computer Science, S. Wermter, C. Weber, W. Duch, T. Honkela, P. D. Koprinkova-Hristova, S. Magg, G. Palm, and A. E. P. Villa, editors, Springer, vol. 8681, pages 563 – 570, 2014.

[C14b] Lydia Fischer, David Nebel, Thomas Villmann, Barbara Hammer, and Heiko Wersing. Rejection Strategies for Learning Vector Quantization – A Comparison of

Probabilistic and Deterministic Approaches.[1] In *Advances in Self-Organizing Maps and Learning Vector Quantization, Proceedings of the 10th International Workshop on Self-organizing Maps WSOM 2014, Mittweida, Germany, July 2-4, 2014*, ser. Advances in Intelligent Systems and Computing, T. Villmann, F.-M. Schleif, M. Kaden, and M. Lange, editors, Springer International Publishing, vol. 295, pages 109 – 118, 2014.

[C14a] Lydia Fischer, Barbara Hammer, and Heiko Wersing. Rejection Strategies for Learning Vector Quantization. In M. Verleysen, editor, *22th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2014, Bruges, Belgium, April 23-25, 2014*, i6doc.com, pages 41 – 46, 2014.

Non-refereed publications

[TR16] Lydia Fischer, and Thomas Villmann. A Probabilistic Model with Adaptive Rejection. *Machine Learning Reports*, MLR-01-2016:1–19, 2016. `http://www.techfak.uni-bielefeld.de/~fschleif/mlr/mlr_01_2016.pdf`

[TR15] Lydia Fischer, Barbara Hammer, and Heiko Wersing. Optimum Reject Options for Prototype-based Classification. *CoRR*, abs/1503.06549, 2015.

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

A.2. Data Properties

property	# data points	# dimensions	# classes
Artificial data:			
Gaussian cluster	8030	2	2
Pearl necklace	5000	2	2
Blossom	7000	2	3
Checkerboard	16000	2	2
Checkerboard noise	16000	3	2
Benchmark data:			
Letter	20000	16	26
USPS	11000	30	10
Outdoor	4000	21	40
Image segmentation	2310	19	7
Coil	1440	30	20
Habermann	306	3	2
Tecator	215	100	2
Adrenal	147	32	2

Table A.1.: Data properties

A.3. Algorithms

Algorithm 1: Dynamic Programming

```
/* compute optimal number of true rejects by DP                    */
input  : X, classifier
output : matrices opt(n, k) and θ(n, k)
/* initialisation                                                  */
h := Σ_{k=1}^{ζ} |E_{θ_k(0)}|;
foreach k ∈ {0, ..., ζ} do
    opt(0, k) := h;
end
foreach n ∈ {1, ..., |L|} do
    foreach k ∈ {0, ..., ζ} do
        opt(n, k) := −∞;
    end
end
/* loop over number of false rejects                               */
foreach n ∈ {1, ..., |L|} do
    /* loop over partitions                                        */
    foreach j ∈ {1, ..., ζ} do
        opt(n, j) := opt(n, j − 1);
        /* loop over thresholds in partition j that agree with
           false rejects                                           */
        foreach i ∈ {1, ..., min{n, |Θ_j| − 1}} do
            n′ := n − i;
            gain := |E_{θ_j(i)} \ E_{θ_j(0)}|;
            h := opt(n′, j − 1) + gain;
            if h > opt(n, j) then
                opt(n, j) := h;
            end
        end
    end
end
/* algorithm is continuing on the next page                        */
```

```
/* compute threshold vector by back-tracing; initialisation
   with default value:  first thresholds                          */
foreach n ∈ {0,...,|L|} do
    foreach k ∈ {1,...,ζ} do
        θ(n,k) := θ_k(0);
    end
end
/* back-tracing in the matrix opt                                 */
foreach n ∈ {1,...,|L|} do
    /* start in last partition                                    */
    j := ζ;
    n' := n;
    i := min(n', |Θ_j| − 1);
    while j > 0 do
        if i = 0 then
            /* threshold 0                                        */
            j := j − 1;
            i := min(n', |Θ_j| − 1);
        else
            n'' := n' − i;
            gain := |E_{θ_j(i)} \ E_{θ_j(0)}|;
            h := opt(n'', j − 1) + gain;
            if opt(n', j) = h then
                /* take threshold i                               */
                θ(n,j) := θ_j(i);
                n' := n'';
                j := j − 1;
                i := min(n', |Θ_j| − 1);
            else
                /* take a smaller threshold                       */
                i := i − 1;
            end
        end
    end
end
/* return optimal true reject numbers and corresponding
   threshold vectors                                              */
```

Algorithm 2: Greedy

`/* compute optimal number of true rejects by DP */`
input : X, classifier
output : accuracy reject curve $\mathbf{t}_c, \mathbf{t}_a$
`/* initialisation by first thresholds */`
foreach $j \in \{1, \ldots, \zeta\}$ **do**
  $\boldsymbol{I}(j) := 0$;
end
$h := \sum_{k=1}^{\zeta} |\mathcal{E}_{\theta_k(0)}|$;
$|\mathcal{E}_{\boldsymbol{\theta}}| := h$;
$n := 0$;
$s := 1$;
$t_c(s) := 1 - |\mathcal{E}_{\boldsymbol{\theta}}|/|X|$;
$t_a(s) := |L|/(|X| - |\mathcal{E}_{\boldsymbol{\theta}}|)$;
`/* algorithm is continuing on the next page */`

```
/* loop while true rejects can be increased                    */
while |E_θ| ≠ |E| do
    /* most improvement locally                                */
    gain := max_j{|E_{θ_j(I(j)+1)} \ E_{θ_j(I(j))}|};
    I_gain := arg max_j{|E_{θ_j(I(j)+1)} \ E_{θ_j(I(j))}|};
    /* most improvement globally                               */
    GAIN := max_j{|E_{θ_j(n+1)} \ E_{θ_j(0)}|};
    I_GAIN := arg max_j{|E_{θ_j(n+1)} \ E_{θ_j(0)}|};
    if GAIN > (gain + |E_θ| − h) then
        foreach j ∈ {1, ..., ζ} do
            I(j) := 0;
        end
        I(I_GAIN) := n;
        |E_θ| := GAIN + h;
        n := n + 1;
    else
        if I_gain is unique then
            I(I_gain) := I(I_gain) + 1;
            |E_θ| := |E_θ| + gain;
            n := n + 1;
        else
            /* increase the number of false rejects            */
            o := 1;
            repeat
                o := o + 1;
                gain := max_j{|E_{θ_j(I(j)+o)} \ E_{θ_j(I(j))}|};
                I_gain := arg max_j{|E_{θ_j(I(j)+o)} \ E_{θ_j(I(j))}|};
            until I_gain is unique;
            n := n + o;
            I(I_gain) := I(I_gain) + o;
            |E_θ| := |E_θ| + gain;
        end
    end
    s := s + 1;
    t_c(s) := 1 − (n + |E_θ|)/|X|;
    t_a(s) := (|L| − n)/(|X| − (n + |E_θ|));
end
```

Bibliography

Abdulsalam, H., Skillicorn, D. B., and Martin, P. (2011). Classification using streaming random forests. *IEEE Transactions on Knowledge and Data Engineering*, 23(1):22–36.

Alvarez, I., Bernard, S., and Deffuant, G. (2007). Keep the decision tree and estimate the class probabilities using its decision boundary. In Veloso (2007), pages 654–659.

Álvarez, J. M., Salzmann, M., and Barnes, N. (2013). Learning appearance models for road detection. In *2013 IEEE Intelligent Vehicles Symposium (IV), Gold Coast City, Australia, June 23-26, 2013*, pages 423–429. IEEE.

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

Bache, K. and Lichman, M. (2013). UCI machine learning repository.

Bartlett, P. L. and Wegkamp, M. H. (2008). Classification with a reject option using a hinge loss. *Journal of Machine Learning Research*, 9:1823–1840.

Bellet, A. and Habrard, A. (2015). Robustness and generalization for metric learning. *Neurocomputing*, 151:259–267.

Bellet, A., Habrard, A., and Sebban, M. (2013). A survey on metric learning for feature vectors and structured data. *CoRR*, abs/1306.6709.

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press.

Berczi, L., Posner, I., and Barfoot, T. D. (2015). Learning to assess terrain from human demonstration using an introspective gaussian-process classifier. In *IEEE International Conference on Robotics and Automation, ICRA 2015, Seattle, WA, USA, 26-30 May, 2015*, pages 3178–3185. IEEE.

Bharitkar, S. and Filev, D. (2001). An online learning vector quantization algorithm. In *Proceedings of the Sixth International Symposium on Signal Processing and its Applications, ISSPA 2001, August 13-16 2001, Shmgri-La Hotel, Kuala Lumpur, Malaysia*, pages 394–397. IEEE.

Biehl, M., Bunte, K., and Schneider, P. (2013a). Analysis of flow cytometry data by matrix relevance learning vector quantization. *PLoS ONE*, 8(3):e59401.

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

Biehl, M., Hammer, B., Schneider, P., and Villmann, T. (2009). Metric learning for prototype-based classification. In Bianchini, M., Maggini, M., Scarselli, F., and Jain, L. C., editors, *Innovations in Neural Information Paradigms and Applications*, volume 247 of *Studies in Computational Intelligence*, pages 183–199. Springer.

Biehl, M., Hammer, B., and Villmann, T. (2013b). Distance measures for prototype based classification. In Grandinetti, L., Lippert, T., and Petkov, N., editors, *Brain-Inspired Computing - International Workshop, BrainComp 2013, Cetraro, Italy, July 8-11, 2013, Revised Selected Papers*, volume 8603 of *Lecture Notes in Computer Science*, pages 100–116. Springer.

Biehl, M., Sadowski, P., Bhanot, G., Bilal, E., Dayarian, A., Meyer, P., Norel, R., Rhrissorrakrai, K., Zeller, M. D., and Hormoz, S. (2015). Inter-species prediction of protein phosphorylation in the sbv IMPROVER species translation challenge. *Bioinformatics*, 31(4):453–461.

Biehl, M., Schneider, P., Smith, D., Stiekema, H., Taylor, A., Hughes, B., Shackleton, C., Stewart, P., and Arlt, W. (2012). Matrix relevance LVQ in steroid metabolomics based classification of adrenal tumors. In Verleysen (2012), pages 423–428.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc.

Bonnin, S., Weisswange, T. H., Kummert, F., and Schmüdderich, J. (2014). General behavior prediction by a combination of scenario-specific models. *IEEE Transactions on Intelligent Transportation Systems*, 15(4):1478–1488.

Boser, B. E., Guyon, I., and Vapnik, V. (1992). A training algorithm for optimal margin classifiers. In Haussler, D., editor, *Proceedings of the Fifth Annual ACM Conference on Computational Learning Theory, COLT 1992, Pittsburgh, PA, USA, July 27-29, 1992.*, pages 144–152. ACM.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth.

Britto Jr., A. S., Sabourin, R., and de Oliveira, L. E. S. (2014). Dynamic selection of classifiers - A comprehensive review. *Pattern Recognition*, 47(11):3665–3680.

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

Capitaine, H. L. (2014). A unified view of class-selection with probabilistic classifiers. *Pattern Recognition*, 47(2):843–853.

Cauwenberghs, G. and Poggio, T. (2000). Incremental and decremental support vector machine learning. In Leen, T. K., Dietterich, T. G., and Tresp, V., editors, *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA*, pages 409–415. MIT Press.

Chandra, A. K., Hirschberg, D. S., and Wong, C. K. (1976). Approximate algorithms for some generalized knapsack problems. *Theoretical Computer Science*, 3(3):293–304.

Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Chow, C. K. (1970). On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, 16(1):41–46.

Cordella, L. P., Stefano, C. D., Sansone, C., and Vento, M. (1995). An adaptive reject option for LVQ classifiers. In Braccini, C., Floriani, L. D., and Vernazza, G., editors, *Image Analysis and Processing, 8th International Conference, ICIAP '95, San Remo, Italy, September 13-15, 1995, Proceedings*, volume 974 of *Lecture Notes in Computer Science*, pages 68–73. Springer.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to Algorithms, second edition*. The MIT Press and McGraw-Hill Book Company.

Cover, T. M. and Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27.

Crammer, K., Kulesza, A., and Dredze, M. (2013). Adaptive regularization of weight vectors. *Machine Learning*, 91(2):155–187.

Dasarathy, B. V. and Sheela, B. V. (1979). A composite classifier system design: Concepts and methodology. *Proceedings of IEEE*, 67(5):708–713.

de Vries, H. (2013). Stationarity and uniqueness of generalized matrix learning vector quantization. In *Machine Learning Reports 04/2013*, pages 16–21. `http://www.techfak.uni-bielefeld.de/~fschleif/mlr/mlr_04_2013.pdf`.

de Vries, H. (2014). On the optimization of generalized matrix learning vector quantization. Master thesis, Johann Bernoulli Institute for Mathematics and Computer Science Rijksuniversiteit Groningen. `http://scripties.fwn.eldoc.ub.rug.nl/scripties/Informatica/Master/2014/Vries.H.de./`.

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

Delany, S. J., Cunningham, P., Doyle, D., and Zamolotskikh, A. (2005). Generating estimates of classification confidence for a case-based spam filter. In Muñoz-Avila, H. and Ricci, F., editors, *Case-Based Reasoning, Research and Development, 6th International Conference, on Case-Based Reasoning, ICCBR 2005, Chicago, IL, USA, August 23-26, 2005, Proceedings*, volume 3620 of *Lecture Notes in Computer Science*, pages 177–190, Berlin, Heidelberg. Springer.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38.

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

Devarakota, P. R., Mirbach, B., and Ottersten, B. (2006). Confidence estimation in classification decision: A method for detecting unseen patterns. In *International Conference on Advances in Pattern Recognition (ICAPR 2007)*, pages 290–294.

Diehl, C. P. and Cauwenberghs, G. (2003). SVM incremental learning, adaptation and optimization. In *Proceedings of the 2003 International Joint Conference on Neural Networks*, volume 4, pages 2685–2690.

Diethe, T. and Girolami, M. (2013). Online learning with (multiple) kernels: A review. *Neural Computation*, 25(3):567–625.

Ditzler, G., Muhlbaier, M. D., and Polikar, R. (2010). Incremental learning of new classes in unbalanced datasets: Learn $^{+\,+}$.UDNC. In Gayar, N. E., Kittler, J., and Roli, F., editors, *Multiple Classifier Systems, 9th International Workshop, MCS 2010, Cairo, Egypt, April 7-9, 2010. Proceedings*, volume 5997 of *Lecture Notes in Computer Science*, pages 33–42. Springer.

Ditzler, G., Roveri, M., Alippi, C., and Polikar, R. (2015). Learning in nonstationary environments: A survey. *Computational Intelligence Magazine, IEEE*, 10(4):12–25.

Dudzinski, K. and Walukiewicz, S. (1987). Exact methods for the knapsack problem and its generalizations. *European Journal of Operational Research*, 28(1):3–21.

El-Yaniv, R. and Wiener, Y. (2010). On the foundations of noise-free selective classification. *Journal of Machine Learning Research*, 11:1605–1641.

Fritsch, J., Kühnl, T., and Geiger, A. (2013). A new performance measure and evaluation benchmark for road detection algorithms. In *16th International IEEE Conference on Intelligent Transportation Systems - (ITSC)*, pages 1693–1700.

Fritsch, J., Kühnl, T., and Kummert, F. (2014). Monocular road terrain detection by combining visual and spatial information. *IEEE Transactions on Intelligent Transportation Systems*, 15(4):1586–1596.

Fritzke, B. (1994). A growing neural gas network learns topologies. In Tesauro, G., Touretzky, D. S., and Leen, T. K., editors, *Advances in Neural Information Processing Systems 7, NIPS Conference, Denver, Colorado, USA, 1994*, pages 625–632. MIT Press.

Fumera, G. and Roli, F. (2002). Support vector machines with embedded reject option. In Lee, S. and Verri, A., editors, *Pattern Recognition with Support Vector Machines, First International Workshop, SVM 2002, Niagara Falls, Canada, August 10, 2002, Proceedings*, volume 2388 of *Lecture Notes in Computer Science*, pages 68–82. Springer.

Fumera, G., Roli, F., and Giacinto, G. (2000). Reject option with multiple thresholds. *Pattern Recognition*, 33(12):2099–2101.

Gama, J., Zliobaite, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4):44:1–44:37.

Gao, J. and Tan, P. (2006). Converting output scores from outlier detection algorithms into probability estimates. In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006), 18-22 December 2006, Hong Kong, China*, pages 212–221. IEEE Computer Society.

Giotis, I., Bunte, K., Petkov, N., and Biehl, M. (2013). Adaptive matrices and filters for color texture classification. *Journal of Mathematical Imaging and Vision*, 47(1-2):79–92.

Gisbrecht, A., Schulz, A., and Hammer, B. (2015). Parametric nonlinear dimensionality reduction using kernel t-SNE. *Neurocomputing*, 147:71–82.

Goodfellow, I. J., Mirza, M., Da, X., Courville, A. C., and Bengio, Y. (2013). An empirical investigation of catastrophic forgetting in gradient-based neural networks. *CoRR*, abs/1312.6211.

Grandvalet, Y., Rakotomamonjy, A., Keshet, J., and Canu, S. (2008). Support vector machines with a reject option. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pages 537–544. Curran Associates, Inc.

Grbovic, M. and Vucetic, S. (2009). Learning vector quantization with adaptive prototype addition and removal. In *International Joint Conference on Neural Networks, IJCNN 2009, Atlanta, Georgia, USA, 14-19 June 2009*, pages 994–1001. IEEE Computer Society.

Grossberg, S. (1980). Biological competition: Decision rules, pattern formation, and oscillations. In *Proceedings of the National Academy of Sciences*, volume 77, pages 2338–2342.

Hamker, F. H. (2001). Life-long learning cell structures–continuously learning without catastrophic interference. *Neural Networks*, 14(4-5):551–573.

Hammer, B., Nebel, D., Riedel, M., and Villmann, T. (2014). Generative versus discriminative prototype based classification. In Villmann, T., Schleif, F., Kaden, M., and Lange, M., editors, *Advances in Self-Organizing Maps and Learning Vector Quantization - Proceedings of the 10th International Workshop, WSOM 2014, Mittweida, Germany, July, 2-4, 2014*, volume 295 of *Advances in Intelligent Systems and Computing*, pages 123–132. Springer.

Hanczar, B. and Dougherty, E. R. (2008). Classification with reject option in gene expression data. *Bioinformatics*, 24(17):1889–1895.

Hansen, L. K., Liisberg, C., and Salomon, P. (1994). The error-reject tradeoff. Technical report, Electronics Institute, Technical University of Denmark, Lyngby, Denmark.

He, H., Chen, S., Li, K., and Xu, X. (2011). Incremental learning from stream data. *IEEE Transactions on Neural Networks*, 22(12):1901–1914.

Herbei, R. and Wegkamp, M. H. (2006). Classification with reject option. *Canadian Journal of Statistics*, 34(4):709–721.

Herbrich, R., Graepel, T., and Campbell, C. (2001). Bayes point machines. *Journal of Machine Learning Research*, 1:245–279.

Hosseini, M. J., Ahmadi, Z., and Beigy, H. (2013). Using a classifier pool in accuracy based tracking of recurring concepts in data stream classification. *Evolving Systems*, 4(1):43–60.

Hu, R., Delany, S. J., and Namee, B. M. (2009). Sampling with confidence: Using k-NN confidence measures in active learning. In *Proceedings of the UKDS Workshop at 8th International Conference on Case-based Reasoning*, ICCBR'09, pages 181–192.

Ishidera, E., Nishiwaki, D., and Sato, A. (2004). A confidence value estimation method for handwritten Kanji character recognition and its application to candidate reduction. *International Journal on Document Analysis and Recognition*, 6(4):263–270.

Jain, A. K. and Li, S. Z. (2005). *Handbook of Face Recognition*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Jin, Y. and Sendhoff, B. (2006). Alleviating catastrophic forgetting via multi-objective learning. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN 2006, part of the IEEE World Congress on Computational Intelligence, WCCI 2006, Vancouver, BC, Canada, 16-21 July 2006*, pages 3335–3342. IEEE.

Kietzmann, T. C., Lange, S., and Riedmiller, M. A. (2008). Incremental GRLVQ: learning relevant features for 3D object recognition. *Neurocomputing*, 71(13-15):2868–2879.

Kirstein, S., Denecke, A., Hasler, S., Wersing, H., Gross, H., and Körner, E. (2009). A vision architecture for unconstrained and incremental learning of multiple categories. *Memetic Computing*, 1(4):291–304.

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

Kirstein, S., Wersing, H., and Körner, E. (2005). Rapid online learning of objects in a biologically motivated recognition architecture. In Kropatsch, W. G., Sablatnig, R., and Hanbury, A., editors, *Pattern Recognition, 27th DAGM Symposium, Vienna, Austria, August 31 - September 2, 2005, Proceedings*, volume 3663 of *Lecture Notes in Computer Science*, pages 301–308. Springer.

Kirstein, S., Wersing, H., and Körner, E. (2008). A biologically motivated visual memory architecture for online learning of objects. *Neural Networks*, 21(1):65–77.

Kohonen, T. (1989). *Self-Organization and Associative Memory*. Springer Series in Information Sciences, Springer-Verlag, third edition.

Kolter, J. Z. and Maloof, M. A. (2007). Dynamic weighted majority: An ensemble method for drifting concepts. *Journal of Machine Learning Research*, 8:2755–2790.

Kühnl, T. (2013). *Road Terrain Detection for Advanced Driver Assistance Systems*. Phd thesis, Bielefeld University. `http://pub.uni-bielefeld.de/publication/2633277`.

Lakshminarayanan, B., Roy, D. M., and Teh, Y. W. (2014). Mondrian forests: Efficient online random forests. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3140–3148.

Landgrebe, T., Tax, D. M. J., Paclík, P., and Duin, R. P. W. (2006). The interaction between classification and reject performance for distance-based reject-option classifiers. *Pattern Recognition Letters*, 27(8):908–917.

Langford, J. and Zadrozny, B. (2005). Estimating class membership probabilities using classifier learners. In Cowell, R. G. and Ghahramani, Z., editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, AISTATS 2005, Bridgetown, Barbados, January 6-8, 2005*, pages 198–205. Society for Artificial Intelligence and Statistics.

Laskov, P., Gehl, C., Krüger, S., and Müller, K. (2006). Incremental support vector learning: Analysis, implementation and applications. *Journal of Machine Learning Research*, 7:1909–1936.

Liang, N., Huang, G., Saratchandran, P., and Sundararajan, N. (2006). A fast and accurate online sequential learning algorithm for feedforward networks. *IEEE Transactions on Neural Networks*, 17(6):1411–1423.

Losing, V., Hammer, B., and Wersing, H. (2015). Interactive online learning for obstacle classification on a mobile robot. In *2015 International Joint Conference on Neural Networks, IJCNN 2015, Killarney, Ireland, July 12-17, 2015*, pages 2808–2815. IEEE.

Martinetz, T. and Schulten, K. (1994). Topology representing networks. *Neural Networks*, 7(3):507–522.

Mensink, T., Verbeek, J. J., Perronnin, F., and Csurka, G. (2013). Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2624–2637.

Minku, L. L. and Yao, X. (2012). DDD: A new ensemble approach for dealing with concept drift. *IEEE Transactions on Knowledge and Data Engineering*, 24(4):619–633.

Nadeem, M. S. A., Zucker, J., and Hanczar, B. (2010). Accuracy-rejection curves (ARCs) for comparing classification methods with a reject option. In Dzeroski, S., Geurts, P., and Rousu, J., editors, *Proceedings of the third International Workshop on Machine Learning in Systems Biology, MLSB 2009, Ljubljana, Slovenia, September 5-6, 2009*, volume 8 of *JMLR Proceedings*, pages 65–81. JMLR.org.

Nene, S. A., Nayar, S. K., and Murase, H. (February 1996). Columbia Object Image Library (COIL-20). *Technical Report CUCS-005-96*.

Nova, D. and Estévez, P. A. (2014). A review of learning vector quantization classifiers. *Neural Computing and Applications*, 25(3-4):511–524.

Pillai, I., Fumera, G., and Roli, F. (2013a). Multi-label classification with a reject option. *Pattern Recognition*, 46(8):2256–2266.

Pillai, I., Fumera, G., and Roli, F. (2013b). Threshold optimisation for multi-label classifiers. *Pattern Recognition*, 46(7):2055–2065.

Pisinger, D. (1995). A minimal algorithm for the multiple-choice knapsack problem. *European Journal of Operational Research*, 83(2):394 – 410. EURO Summer Institute Combinatorial Optimization.

Platho, M. and Eggert, J. (2012). Deciding what to inspect first: Incremental situation assessment based on information gain. In *Proceedings 15th International IEEE Conference on Intelligent Transportation Systems*, pages 888–893.

Platt, J. C. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press.

Polikar, R. and Alippi, C. (2014). Guest editorial learning in nonstationary and evolving environments. *IEEE Transactions on Neural Networks and Learning Systems*, 25(1):9–11.

Polikar, R., Upda, L., Upda, S. S., and Honavar, V. (2001). Learn++: An incremental learning algorithm for supervised neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 31(4):497–508.

Pomerleau, D. (1993). Knowledge-based training of artificial neural networks for autonomous robot driving. In Connell, J. and Mahadevan, S., editors, *Robot Learning*, pages 19–43. New York: Kluwer Academic.

Queißer, J. F. (2012). Using context for the combination of offline and online learning. Master thesis, Bielefeld University. `http://www.cor-lab.de/system/files/jqueisse_master_pub.pdf`.

Ralaivola, L. and d'Alché-Buc, F. (2001). Incremental support vector machine learning: A local approach. In Dorffner, G., Bischof, H., and Hornik, K., editors, *Artificial Neural Networks - ICANN 2001, International Conference Vienna, Austria, August 21-25, 2001 Proceedings*, volume 2130 of *Lecture Notes in Computer Science*, pages 322–330. Springer.

Ramaswamy, H. G., Tewari, A., and Agarwal, S. (2015). Consistent algorithms for multiclass classification with a reject option. *CoRR*, abs/1505.04137.

Ramaswamy, S., Rastogi, R., and Shim, K. (2000). Efficient algorithms for mining outliers from large data sets. In Chen, W., Naughton, J. F., and Bernstein, P. A., editors, *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*, pages 427–438. ACM.

Read, J., Bifet, A., Pfahringer, B., and Holmes, G. (2012). Batch-incremental versus instance-incremental learning in dynamic and evolving data. In Hollmén, J., Klawonn, F., and Tucker, A., editors, *Advances in Intelligent Data Analysis XI - 11th International Symposium, IDA 2012, Helsinki, Finland, October 25-27, 2012. Proceedings*, volume 7619 of *Lecture Notes in Computer Science*, pages 313–323. Springer.

Ren, S., He, K., Girshick, R. B., and Sun, J. (2015). Faster R-CNN: towards real-time object detection with region proposal networks. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 91–99.

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

Sabourin, M., Mitiche, A., Thomas, D. S., and Nagy, G. (1993). Classifier combination for hand-printed digit recognition. In *2nd International Conference on Document Analysis and Recognition, ICDAR*, pages 163–166. IEEE.

Saffari, A., Leistner, C., Santner, J., Godec, M., and Bischof, H. (2009). On-line random forests. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 1393–1400. IEEE.

Santos-Pereira, C. M. and Pires, A. M. (2005). On optimal reject rules and ROC curves. *Pattern Recognition Letters*, 26(7):943–952.

Sato, A. and Yamada, K. (1995). Generalized learning vector quantization. In Touretzky, D. S., Mozer, M., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems 8, NIPS, Denver, CO, November 27-30, 1995*, pages 423–429. MIT Press.

Schneider, P., Biehl, M., and Hammer, B. (2009a). Adaptive Relevance Matrices in Learning Vector Quantization. *Neural Computation*, 21(12):3532–3561.

Schneider, P., Biehl, M., and Hammer, B. (2009b). Distance learning in discriminative vector quantization. *Neural Computation*, 21(10):2942–2969.

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

Schneider, P., Bunte, K., Stiekema, H., Hammer, B., Villmann, T., and Biehl, M. (2010b). Regularization in matrix relevance learning. *IEEE Transactions on Neural Networks*, 21(5):831–840.

Schulter, S., Leistner, C., Roth, P. M., Bischof, H., and Gool, L. J. V. (2011). On-line hough forests. In Hoey, J., McKenna, S. J., and Trucco, E., editors, *British Machine Vision Conference, BMVC 2011, Dundee, UK, August 29 - September 2, 2011. Proceedings*, pages 1–11. BMVA Press.

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

Seo, S. and Obermayer, K. (2006). Dynamic hyperparameter scaling method for LVQ algorithms. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN 2006, part of the IEEE World Congress on Computational Intelligence, WCCI 2006, Vancouver, BC, Canada, 16-21 July 2006*, pages 3196–3203. IEEE.

Silver, D. L., Yang, Q., and Li, L. (2013). Lifelong machine learning systems: Beyond learning algorithms. In *Lifelong Machine Learning, Papers from the 2013 AAAI Spring Symposium, Palo Alto, California, USA, March 25-27, 2013*, volume SS-13-05 of *AAAI Technical Report*, pages 49–55. AAAI.

Sousa, R. and Cardoso, J. S. (2013). The data replication method for the classification with reject option. *AI Communications*, 26(3):281–302.

Stavens, D., Hoffmann, G., and Thrun, S. (2007). Online speed adaptation using supervised learning for high-speed, off-road autonomous driving. In Veloso (2007), pages 2218–2224.

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

Struwe, M., Hasler, S., and Bauer-Wersing, U. (2013). Using the analytic feature framework for the detection of occluded objects. In *Proceedings of the 23rd International Conference on Artificial Neural Networks and Machine Learning ICANN 2013*, pages 603–610, New York, NY, USA. Springer-Verlag New York, Inc.

Sugiyama, M. and Borgwardt, K. M. (2013). Rapid distance-based outlier detection via sampling. In Burges, C. J. C., Bottou, L., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 467–475.

Sugiyama, M. and Kawanabe, M. (2012). *Machine Learning in Non-Stationary Environments - Introduction to Covariate Shift Adaptation*. Adaptive computation and machine learning. MIT Press.

Suutala, J., Pirttikangas, S., Riekki, J., and Röning, J. (2004). Reject-optional LVQ-based two-level classifier to improve reliability in footstep identification. In Ferscha, A. and Mattern, F., editors, *Pervasive Computing, Second International Conference, PERVASIVE 2004, Vienna, Austria, April 21-23, 2004, Proceedings*, volume 3001 of *Lecture Notes in Computer Science*, pages 182–187. Springer.

Sykes, D., Corapi, D., Magee, J., Kramer, J., Russo, A., and Inoue, K. (2013). Learning revised models for planning in adaptive systems. In Notkin, D., Cheng, B. H. C., and Pohl, K., editors, *35th International Conference on Software Engineering, ICSE '13, San Francisco, CA, USA, May 18-26, 2013*, pages 63–71. IEEE / ACM.

Tax, D. M. J. and Duin, R. P. W. (2008). Growing a multi-class classifier with a reject option. *Pattern Recognition Letters*, 29(10):1565–1570.

Tewari, A. and Bartlett, P. L. (2007). On the consistency of multiclass classification methods. *Journal of Machine Learning Research*, 8:1007–1025.

Thodberg, H. H. (1995). Tecator data set, contained in StatLib Datasets Archive.

Thrun, S. and Mitchell, T. M. (1995). Lifelong robot learning. *Robotics and Autonomous Systems*, 15(1-2):25–46.

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C. M., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L., Koelen, C., Markey, C., Rummel, C., van Niekerk, J., Jensen, E., Alessandrini, P., Bradski, G. R., Davies, B., Ettinger, S., Kaehler, A., Nefian, A. V., and Mahoney, P. (2006). Stanley: The robot that won the DARPA grand challenge. *Journal of Field Robotics*, 23(9):661–692.

Tóth, N. and Pataki, B. (2007). On classification confidence and ranking using decision trees. In *International Conference on Intelligent Engineering Systems (INES)*, pages 133–138. IEEE.

Tsang, I. W., Kwok, J. T., and Cheung, P. (2005). Core vector machines: Fast SVM training on very large data sets. *Journal of Machine Learning Research*, 6:363–392.

Vailaya, A. and Jain, A. K. (2000). Reject Option for VQ-Based Bayesian Classification. In *15th International Conference on Pattern Recognition, ICPR'00, Barcelona, Spain, September 3-8, 2000*, pages 2048–2051. IEEE Computer Society.

van der Maaten, L. (March 2013). Matlab Toolbox for Dimensionality Reduction.

Vapnik, V. (1999). An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999.

Vellido, A., Martín-Guerrero, J. D., and Lisboa, P. J. G. (2012). Making machine learning models interpretable. In Verleysen (2012), pages 163–172.

Veloso, M. M., editor (2007). *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*.

Verleysen, M., editor (2012). *20th European Symposium on Artificial Neural Networks, ESANN 2012, Bruges, Belgium, April 25-27, 2012*. i6doc.com.

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

Villmann, T., Kaden, M., Bohnsack, A., Saralajew, S., and Hammer, B. (in press 2016). Self-adjusting reject options in prototype based classification. In *11th Workshop on Self-Organizing Maps and Learning Vector Quantization*.

Villmann, T., Kaden, M., Nebel, D., and Biehl, M. (2015). Learning vector quantization with adaptive cost-based outlier-rejection. In Azzopardi, G. and Petkov, N., editors, *Computer Analysis of Images and Patterns - 16th International Conference, CAIP 2015, Valletta, Malta, September 2-4, 2015, Proceedings, Part II*, volume 9257 of *Lecture Notes in Computer Science*, pages 772–782. Springer.

Weiss, J. C., Natarajan, S., Peissig, P. L., McCarty, C. A., and Page, D. (2012). Machine learning for personalized medicine: Predicting primary myocardial infarction from electronic health records. *AI Magazine*, 33(4):33–45.

Wersing, H. and Queißer, J. F. (2014). System for controlling an automated device. EP 2690582 A1, Honda Research Institute Europe GmbH, Offenbach, Germany.

Woods, K. S., Kegelmeyer, W. P., and Bowyer, K. W. (1997). Combination of multiple classifiers using local accuracy estimates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):405–410.

Wu, T., Lin, C., and Weng, R. C. (2004). Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5:975–1005.

Xu, Y., Shen, F., and Zhao, J. (2012). An incremental learning vector quantization algorithm for pattern classification. *Neural Computing and Applications*, 21(6):1205–1215.

Xu, Y., Shen, F., Zhao, J., and Hasegawa, O. (2009). An enhanced incremental prototype classifier using subspace representation scheme. In *The 2nd Workshop of: From Local Patterns to Global Models (In conjunction with ECML 2009). Bled, Slovenia*, pages 139–156.

Yuan, M. and Wegkamp, M. H. (2010). Classification methods with reject option based on convex risk minimization. *Journal of Machine Learning Research*, 11:111–130.

Zhang, J. (2013). Advancements of outlier detection: A survey. *EAI Endorsed Transactions on Scalable Information Systems*, 1(1):e2.

Zheng, J., Shen, F., Fan, H., and Zhao, J. (2013). An online incremental learning support vector machine for large-scale data. *Neural Computing and Applications*, 22(5):1023–1035.

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