Neurocomputing 214 (2016) 445–457

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Optimal local rejection for classifiers

Lydia Fischer [a,b,*], Barbara Hammer [a], Heiko Wersing [b]

[a] Bielefeld University, Universitätsstraße 25, 336615 Bielefeld, Germany
[b] HONDA Research Institute Europe, Carl-Legien-Straße 30, 63065 Offenbach, Germany

ARTICLE INFO

Article history:
Received 14 December 2015
Received in revised form
22 April 2016
Accepted 14 June 2016
Communicated by Zhaohong Deng
Available online 23 June 2016

Keywords:
Classification
Prototype-based
Distance-based
Local rejection
Multiple choice knapsack

ABSTRACT

We analyse optimal rejection strategies for classifiers with input space partitioning, e.g. prototype-based classifiers, support vector machines or decision trees using certainty measures such as the distance to the closest decision border. We provide new theoretical results: we link this problem to the multiple choice knapsack problem and devise an exact polynomial-time dynamic programming (DP) scheme to determine optimal local thresholds on given data. Further, we propose a linear time, memory efficient approximation thereof. We show in experiments that the approximation has a competitive performance compared to the full DP. Further, we evaluate the performance of classification with rejection in various benchmarks: we conclude that local rejection is beneficial in particular for simple classifiers, while the improvement is less pronounced for advanced classifiers. An evaluation on speech prosody and biomedical data highlights the benefit of local thresholds.

1. Introduction

1.1. Motivation

Classification is a standard technique of machine learning: its application ranges from automated digit recognition, computer vision [1] up to fraud detection, and numerous machine learning classifiers are available for this task [2]. Often, besides classification accuracy, the treatment of uncertain classifications is important. Classifiers providing a certainty level together with predicted class labels offer such a treatment, i.e. data with uncertain predictions are rejected. In safety critical areas like driver assistance systems, health care, or biomedical data analysis, the certainty of the prediction is almost as important as the class label itself. Further tests or expert opinions can be consulted for uncertain predictions to avoid critical consequences of a misclassification for instance in health care. For driver assistance systems, a high degree of uncertainty can lead to turning off the system and passing the responsibility back to the human driver.

An early reject option was introduced and analysed by Chow [3]: if the costs for a misclassification versus a rejected data point are known, one can determine an optimal rejection threshold based on the probability of misclassification. This finding allows to extend probabilistic classifiers directly with a reject option based on their internal probabilistic model. There are approaches which equip deterministic classifiers efficiently with certainty values, like the extensions of the support vector machine (SVM) [4–7], extreme leaning machines [8,9], decision trees (DT) [10,11], or prototype-based classifiers [12]. Since these reject options use an estimate of the underlying probabilities only, their validity relies on results of the investigation of so-called plugin rules for the true probability measures [13].

Rejection based on deterministic certainty measures is an alternative. Deterministic reject options directly address extensions of the 0–1-classification loss. Many deterministic certainty measures are based on geometric quantities such as the distance to the decision border, e.g. [14,15]. Experiments showed that distance-based rejection can reach the accuracy of probabilistic counterparts [16,17]. Distance-based reject options can be used in a post-processing step or they can be involved in the training of the classifier itself. One of the first embedded reject options for SVM has been proposed in [18]. Later, alternative formulations which approximate the 0–1-loss by a convex surrogate and which also prove the validity of embedded rejection have been proposed in [19–21].

So far reject options deal with a single threshold and mostly with binary classification only. Extensions to more general settings like multi-label classification [22,23] and multiple classes [24–26] have been considered. Still, these proposals rely on a reject option with one global threshold. It has been shown that reject rules which rely on plugins for the involved

[*] Corresponding author at: Bielefeld University and HONDA Research Institute Europe, Carl-Legien-Straße 30, 63065 Offenbach, Germany.
E-mail address: lfischer@uni-bielefeld.de (L. Fischer).

http://dx.doi.org/10.1016/j.neucom.2016.06.038
0925-2312/© 2016 Elsevier B.V. All rights reserved.

probabilities can benefit from local rejection thresholds such as class-wise thresholds [27]. Up to our knowledge, there do not yet exist strategies how to optimally determine local thresholds in the case of deterministic rejection, and there does not yet exist an extensive comparison how optimal local rejection compares to global rejection for deterministic classifiers.

In this article we tackle these problems related to local reject options: how to efficiently choose optimal local thresholds based on a given partitioning of the input space e.g. according to the predicted output classes. We rely on first promising results as reported in [28] which proposes an efficient greedy algorithm. We extend this work by a general formalisation of the problem to optimally choose local thresholds for any given classifier. We phrase this problem as an optimisation in form of a multiple choice knapsack problem [29] and provide an optimal solution for finding local thresholds in form of a polynomial time dynamic programming (DP) scheme. We compare this scheme to a linear-time greedy alternative from [28], experimentally validating competitive behaviour of the latter. While our optimal threshold selection strategy can be used for any classifier, we focus in the experiments on three popular classifiers: learning vector quantisation (LVQ) and its derivatives [30,12,31,32], SVM, and DTs. We evaluate the rejection strategies using benchmarks, and two real-life data sets. As performance measure, we use the accuracy-reject curve [33,34]. To summarise, the contribution of this article consists in the following (i) theoretical and (ii) experimental results:

(i) An optimal, polynomial time DP scheme which determines local thresholds on given data, by linking it to a specific case of the multiple choice knapsack problem; a memory efficient linear-time greedy algorithm thereof.
(ii) We demonstrate that the results of the greedy approximation are competitive to the full DP scheme. Further, we test the gain of local reject options versus a global choice for several popular classifiers and benchmark data sets.

This article is structured as follows: In Section 1.2 we review existing reject options as proposed in the literature. The general problem setting of global and local rejection can be found in Section 2. Afterwards in Section 3, we develop a polynomial time scheme based on DP allowing an optimal choice of local thresholds, and a time and memory efficient greedy approximation thereof. In the experiments Section 4, we briefly explain used classifiers and their related certainty measure. Thereafter we test the rejection strategies on artificial data, benchmarks, and two real-life data sets. We illustrate the suitability of rejection, and we show a comparison of local versus global rejection on the one hand, and the comparison of an optimal computation of local thresholds by means of DP versus an efficient greedy scheme on the other hand. The article ends with a discussion of the main findings.

1.2. Related work

The following section summarises the state of the art for rejection strategies and related certainty measures for classification. Vailya and Jain [14] highlight two reasons for rejection: ambiguity and outliers. There exist approaches explicitly addressing one of these reasons or a combination of both. Mostly, rejection is based on a measure which provides a certainty value about whether a given data point is correctly classified.

Probabilistic approaches: Common certainty measures are based on probabilities. Chow [3] proposed an optimal reject option, given the true probability density function is known. In this case, global rejection is an adequate strategy and local rejection would not offer any benefit in such a setting. 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. the Bayes classifier, or the estimation is done in addition to a non-probabilistic classifier.

The work of [35] proved that in the limit case, the rejection strategy [3] provides a bound for any other measure in the sense of the error-reject trade-off and they provide illustrative examples. The authors introduce a general scaling approach to compare error-reject curves of several independent experiments even with different classifiers or data sets. The authors of [13] link the work of [3] 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 are particularly important in medical studies where the classification of an ill patient as healthy is worse than vice versa. The approach [27] directly builds on [3] and they state that class-related 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.

Due to this theoretical background, many approaches estimate the data distribution first, e.g. with Gaussian mixture models (GMM) [36,14]. The reliable estimation of GMMs is particularly problematic for high dimensional data. Therefore, [37] proposes 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 either one uses a probabilistic counterpart of the desired algorithm (e.g. the robust soft LVQ [12] instead of distance-based LVQ variants) or one does a probabilistic modelling of the data in a post-processing step. 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: Platt [6] proposed an approach to turn the activity (the distance of a data point to the decision border) of a binary SVM into an approximation of a classification confidence. A transfer of this method for multi-class tasks is provided by [7] and it is implemented in the LIBSVM toolbox [38].

Deterministic approaches: Rejection based on deterministic measures uses the output activation of a given classifier or a geometric alternative, e.g. the distance to the decision border or similar [10,15]. The approach [39] focuses on effective outlier detection, relying on the distances of a new data point from elements of a randomly chosen subset of given data. An outlier score is then given by the smallest distance. The resulting method outperforms state of the art approaches like [40] in efficiency and accuracy. The authors of [41] introduce a reject option identifying ambiguous regions in binary classifications. Their approach is based on a data replication method with the advantage that no threshold has to be set externally, rather the method itself provides a suitable cut-off. The approach [42] addresses 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 [3,13]. Furthermore, different certainty measures based on the activation of the output neurons are investigated.

Conclusion: The aforementioned approaches mostly consider a global or class-related thresholds in binary classification settings.

We consider multi-class settings with local thresholds (their number is at least equal to the number of classes). Further, for the first time, we study how to optimally choose local thresholds for any classifier based on an optimisation of its related classification error.

Subsequently, we consider a deterministic setting and rely on two elements for rejection: (I) a real-valued certainty measure for the rejection of an unreliable classification, and (II) a partitioning of the data space into local patches for local rejection, e.g. a decomposition of the space based on the predicted class label in case of a multi-class classification. We analyse, how to optimally determine local thresholds in the following. In particular, we propose efficient schemes how to optimise local thresholds which are attached to the given partitioning of the input space. We test global and local rejection for three popular classifiers given by prototype-based classifiers, SVM, and DT on several data sets.

2. General setting for global and local rejection

General setting: 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 provides a function $c: \mathbb{R}^M \to \{1, \dots, Z\}$. For classical settings, a classifier aims at minimising the classification error

$$\mathrm{error}(c) := \int \mathrm{loss}(c(\mathbf{x}), y) d_P(\mathbf{x}, y), \quad \text{with}$$
$$\mathrm{loss}(c(\mathbf{x}), y) := \begin{cases} 0 & \text{if } c(\mathbf{x}) = y \\ 1 & \text{otherwise} \end{cases} \tag{1}$$

Since P is unknown, standard classifiers often optimise the empirical error (2) instead, or a related, numerically simpler (e.g. convex) surrogate loss. For popular classifiers, results from computational learning theory guarantee that the empirical error allows us to uniformly bound the true error provided data are i.i.d. and suitable regularisation takes place [43].

$$\widehat{\mathrm{error}}(c, X) := \frac{1}{N} \sum_{i=1}^{N} \mathrm{loss}(c(\mathbf{x}_i), y_i) \tag{2}$$

In this article, we study how to efficiently extend a multi-class classifier by a reject option posterior to training. Hence, we assume that a trained classifier is given. We use deterministic and probabilistic classifiers (Section 4.1). In addition to the predicted class label, many classifiers provide a certainty value of its classification like the class probability or the distance to the decision border. This certainty value is used whenever classification is extended by a reject option and points with a low certainty value are rejected. Formally, a reject option extends the classifier to a mapping (denoted with the same symbol) $c: \mathbb{R}^M \to \{1, \dots, Z, \circledR\}$ where the symbol $\circledR$ denotes the rejection of the classification of input $\mathbf{x}$ which is typically defined by an extended 0–1-loss

$$\mathrm{loss}(c(\mathbf{x}), y) := \begin{cases} 0 & \text{if } c(\mathbf{x}) = y \\ b & \text{if } c(\mathbf{x}) = \circledR \\ 1 & \text{if } c(\mathbf{x}) \neq y, \ c(\mathbf{x}) \neq \circledR \end{cases} \tag{3}$$

where costs $b \in (0, 1)$ are assigned to a reject $\circledR$ [44,20]. 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. Depending on the application the user is willing to accept the rejection of many correct classified points, if they are only focussing on classification of highly reliable data as in [45]. The ratio between the rejection of correctly classified data and errors can be controlled with the cost parameter b. Hence, our proposed method can be applied to any use case. The threshold (or threshold vector) is a crucial parameter in a reject option. The question how to choose thresholds which optimise the modified classification error is the key topic of this article. While threshold optimisation is straightforward in the case of one global threshold, the optimisation of local thresholds is more difficult [27].

In the following, we will investigate two aspects of this setting:

- Optimisation of local thresholds: Assume that a trained classifier is given. How can we efficiently determine optimal thresholds for rejection in particular if local thresholds are considered, such that the extended loss is minimised? Thereby we focus on empirical risk minimisation of the expected error for given training data since the true risk is unavailable, and we test the performance of the determined thresholds in a cross-validation for unseen data and we evaluate its generalisation ability. Further, we do not assume fixed, known costs b, rather we provide a method which finds optimal threshold vectors for every choice of b for a given classifier. We phrase local threshold optimisation as multiple-choice-knapsack problem and propose a polynomial time exact solution as well as an efficient linear time, constant memory greedy approximation.
- Evaluation of the efficiency of local and global reject options for different classifiers: We experimentally analyse the effect of local and global rejection strategies for popular classifiers which display different characteristics. This gives insight on the question which rejection strategy, certainty measure, and partitioning of the input space is suited for which classifier. Note that an optimal reject option can be explicitly computed whenever the Bayesian risk is known [3], and we use this Gold standard in artificial settings with known ground truth. Provided a classifier does not closely resemble the Bayes risk (e.g. due to the characteristics of the training data, unknown priors, or technical constraints posed on the form and complexity of the classifier), alternative reject options can be beneficial, as initially investigated in [27]. We confirm this finding and study the effect of local and global reject options which are optimised based on the empirical error.

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

$$r(\mathbf{x}): \mathbb{R}^M \to \mathbb{R}, \ \mathbf{x} \mapsto r(\mathbf{x}) \tag{4}$$

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

$$r(\mathbf{x}) < \theta. \tag{5}$$

A rejection strategy performs optimal if only labelling errors are rejected. In general this is not the case and a certainty measure causes the rejection of few correctly classified data points together with errors.

Local reject option: Global rejection relies on the assumption of equal scaling of the certainty measure $r(\mathbf{x})$ for all inputs $\mathbf{x}$. We relax this assumption by using local thresholds. A local rejection strategy relies on a partition of the input space $\mathbb{R}^M$ into ζ disjunct, non-empty sets Υ_j such that $\mathbb{R}^M = \cup_{j=1}^{\zeta} \Upsilon_j$. Using a different threshold θ_j in every set Υ_j enables a finer control of rejection [14,28].

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 $\boldsymbol{\theta} = (\theta_1, \dots, \theta_\zeta)$ of the

Table 1
Definitions of quantities.

X	Data set with N elements	$X = L \cup E$
L/E	Correctly/wrongly classified data	
Rejection quantities		
$\mathcal{X}_\theta$	Rejected data	$X = \mathcal{X}_\theta \cup X_\theta$
$\mathcal{X}_{\theta_j}$	Rejected data in Υ_j	
X_θ	Accepted data	$X_{\boldsymbol{\theta}} = \ \cup_j \ X_{\theta_j}$
X_{θ_j}	Accepted data in Υ_j	
L_j	Correctly classified data in Υ_j	$L_j = L \cap \Upsilon_j$
$\mathcal{L}_\theta$	False rejects	$\mathcal{L}_\theta = \mathcal{X}_\theta \cap L$
$\mathcal{L}_{\theta_j}$	False rejects in Υ_j	$\mathcal{L}_{\theta_j} = \Upsilon_j \cap L$
		$\mathcal{L}_{\boldsymbol{\theta}} = \ \cup_j \ \mathcal{L}_{\theta_j}$
E_j	Errors in Υ_j	$E_j = E \cap \Upsilon_j$
$\mathcal{E}_\theta$	True rejects	$\mathcal{L}_\theta = \mathcal{X}_\theta \cap L$
		$\mathcal{E}_{\boldsymbol{\theta}} = \ \cup_j \ \mathcal{E}_{\theta_j}$
$\mathcal{E}_{\theta_j}$	True rejects in Υ_j	$\mathcal{E}_{\theta_j} = \Upsilon_j \cap E$

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

3. Optimal choices of global/local thresholds

Rejection strategies crucially depend on the threshold θ or threshold vector $\boldsymbol{\theta}$. Subsequently, we analyse how to choose those in an optimal way which refers to a multiple objective: 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. First we formalise this setting (definition of quantities see Table 1) and its evaluation.

3.1. Pareto front

Assume that labelled data X is given to determine optimal thresholds thereon. A classifier decomposes X into a set of correctly classified data L and a set of wrongly classified data (errors) E. These sets split with respect to the partition Υ_j into L_j and E_j. An optimal rejection would reject all points in E, while classifying all points in L. This is usually impossible using any reject option. Applying a global threshold θ, the data X decomposes into rejected data $\mathcal{X}_\theta$ and accepted data X_θ. We refer to false rejects as $\mathcal{L}_\theta$ and to true rejects as $\mathcal{E}_\theta$. Note that, when increasing θ, $|\mathcal{X}_\theta|$, $|\mathcal{L}_\theta|$, and $|\mathcal{E}_\theta|$ are 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 as X_{θ_j}. These relate to false rejects $\mathcal{L}_{\theta_j}$ and true rejects $\mathcal{E}_{\theta_j}$ for Υ_j. As before, 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 in Υ_j.

We report the accuracy-reject curve (ARC) [33] as performance measure. A given threshold θ leads to the accuracy of the classified points $t_a(\theta) := (|L \backslash \mathcal{L}_\theta|)/|X_\theta|$ versus the ratio of the classified points $t_c(\theta) := |X_\theta|/|X|$. Both 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 complete data). The same quantities can be defined for a threshold vector $\boldsymbol{\theta}$, we refer to as threshold θ in the following, for simplicity. We optimise θ, such that the value t_a is maximised, and t_c is minimised. Hence, not all possible thresholds and related pairs $(t_a(\theta), t_c(\theta))$ are of interest, only optimal choices corresponding 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 $|\mathcal{L}_{\theta'}| \le |\mathcal{L}_\theta|$ and $|\mathcal{E}_{\theta'}| \ge |\mathcal{E}_\theta|$ and for at least one term, inequality holds. We aim at the *Pareto front*

$$\mathcal{P}_\theta := \{(|\mathcal{L}_\theta|, |\mathcal{E}_\theta|) | \ \theta \text{ is not dominated by any } \theta'\}. \tag{6}$$

Each dominated threshold (threshold vector) corresponds to a sub-optimal choice only: We 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.

To evaluate the efficiency of a threshold strategy, it turns out that a slightly different set is more easily accessible. 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*

$$\widetilde{\mathcal{P}}_\theta := \{(|\mathcal{L}_\theta|, |\mathcal{E}_\theta|) | \ \theta \text{ is not dominated by any } \theta' \\ \text{with respect to the true rejects}\}. \tag{7}$$

Hence, $\mathcal{P}_\theta$ can be computed as the subset of $\widetilde{\mathcal{P}}_\theta$ by taking the minima over the false rejects. $\widetilde{\mathcal{P}}_\theta$ 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 $\widetilde{\mathcal{P}}_\theta$ and the corresponding thresholds, we report the efficiency of a rejection strategy by the corresponding ARC curve, i.e. the pairs $(t_a(\theta), t_c(\theta))$: These pairs correspond to a graph, where we report the ratio of classified points (starting from a ratio 1 down to 0) versus the obtained accuracy for the classified points. For good strategies, this graph should be increasing as fast as possible. In the following, we discuss efficient strategies to compute the extended Pareto front for global and local rejection strategies.

3.2. Optimal thresholds for given rejection costs

In the preceding section, we introduced the Pareto front rather than a single threshold which is optimised according to the extended empirical risk $\widehat{\text{error}}(c, X)$ (2) for given rejection costs b. This has the benefit that, given *any* $b \in (0, 1)$, an optimal threshold can be extracted from the Pareto front due to the following relation: assume that a threshold $\theta \in \mathcal{P}_\theta$ is chosen; using the notation from above, we can rephrase

$$\widehat{\text{error}}(c, X) = \frac{1}{N}\left(|E| - (1 - b) \cdot |\mathcal{E}_\theta| + b \cdot |\mathcal{L}_\theta|\right). \tag{8}$$

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

$$\theta_{\text{opt}}(b) = \arg \max_\theta \left(|\mathcal{E}_\theta| - \frac{b}{1 - b} \cdot |\mathcal{L}_\theta|\right). \tag{9}$$

This can be extracted from the Pareto front.

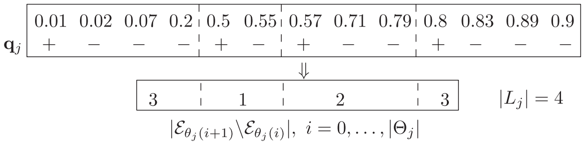


Fig. 1. Local thresholds for a partition with 13 points. The first row shows the sorted certainty values $r(\mathbf{x}_i)$, the second row depicts if a point is correct (+)/wrong (−) classified. Here are 4 thresholds corresponding to the Pareto front, according to the number of signs + (due to the fact that point 13 is in E). The third row shows the gain when increasing the threshold value θ_j.

3.3. Optimal global rejection

Global rejection needs only one threshold θ. We compute thresholds leading to the extended Pareto front and the corresponding pairs $(t_a(\theta), t_c(\theta))$ in time $O(N \log N)$ 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 $r(\mathbf{x}_{i_1}) \leq \dots \leq r(\mathbf{x}_{i_N})$ (Fig. 1). We sort certainty values which are identical such that the points in L come first. The following holds:

- Each pair $(|\mathcal{L}_\theta|, |\mathcal{E}_\theta|) \in \widetilde{\mathcal{P}}_\theta$ 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:

$$\Theta := \{\theta = r(\mathbf{x}_{i_k}) | \mathbf{x}_{i_k} \in L\} \cup \{\infty | \text{if } x_{i_N} \notin L\}. \tag{10}$$

$|\Theta| \in \{|L|, |L| + 1\}$ depending on whether the last point in this list is classified correctly or not. Fig. 1 shows an exemplary setting. We refer to thresholds obtained this way as $\theta(0), \dots, \theta(|\Theta| - 1)$, and we assume ascending sorted values.

3.4. Optimal local rejection

Computing the extended Pareto front for local rejection is harder than global rejection since the number of parameters (thresholds) in the optimisation rises from one to ζ. First, we derive an optimal solution via DP [46,47]. Second, we introduce a faster greedy solution which provides 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 $j \in \{1, \dots, \zeta\}$ 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

$$\Theta_j := \{\theta_j(0), \dots, \theta_j(|\Theta_j| - 1)\} \tag{11}$$

equivalent to (10) with $|\Theta_j| \in \{|L_j|, |L_j| + 1\}$.

We look for 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 Table 2. 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.

Table 2
Example rejects for three partitions and their losses/gains.

Threshold i	$\lvert\mathcal{L}_{\theta_j(i)}\rvert$				$\lvert\mathcal{E}_{\theta_j(i)}\rvert$			
	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

Hence the question arises how to efficiently compute optimal combinations of the single values in Θ_j. There exist at most $|\Theta_1| \cdot \ldots \cdot |\Theta_\zeta| = O(|L|^\zeta)$ different combinations (using the trivial upper bound $O(|L_j|) \leq O(|L|)$ for each $|\Theta_j|$). This number is infeasible for large ζ, i.e. a fine grained decomposition. We propose two methods to compute the Pareto front which are linear with respect to ζ, which depend on its formalisation as multiple choice knapsack problem [29].

3.4.1. Formulation as multiple choice knapsack problem

Assume that the number of false rejects n is fixed. 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) [29] as follows:

$$\begin{aligned} &\max_{a_{ji}} \sum_{j=1}^{\zeta} \sum_{i=0}^{|\Theta_j|-1} |\mathcal{E}_{\theta_j(i)}| \cdot a_{ji} \\ &\quad \text{subject to} \sum_{j=1}^{\zeta} \sum_{i=0}^{|\Theta_j|-1} |\mathcal{L}_{\theta_j(i)}| \cdot a_{ji} = n \\ &\quad \forall\, 1 \leq j \leq \zeta\colon \quad \sum_{i=0}^{|\Theta_j|-1} a_{ji} = 1 \\ &\quad \forall\, 1 \leq j \leq \zeta, \forall\, 0 \leq i \leq |\Theta_j| - 1\colon \quad a_{ji} \in \{0, 1\} \end{aligned} \tag{12}$$

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 (12) are polynomial with respect to the number of data points $|X|$, this leads to a polynomial solution of this problem. As an example, the contributions [29,48,49] investigate efficient exact solutions, mostly based on linear programming relaxations which simplify the original MCKP such that it can be solved optimally by enumeration. In our case, we directly derive an efficient and very intuitive alternative with the same computational complexity relying on the fact that thresholds in every partition Υ_j have a linear ordering according to their gain/costs. This enables us to derive a quadratic time and linear memory algorithm very similar to the DP scheme of the classical (simple) knapsack problem.

3.4.2. Local threshold adaptation by DP

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

$$\begin{aligned} \mathrm{opt}(n, j, i) := \max_{\boldsymbol{\theta}} \{ &|\mathcal{E}_{\boldsymbol{\theta}}| \mid |\mathcal{L}_{\boldsymbol{\theta}}| = n, \theta_k \in \{\theta_j(0), \dots, \theta_j(|\Theta_j| - 1)\} \\ &\forall\, k < j, \theta_j \in \{\theta_j(0), \dots, \theta_j(i)\}, \theta_k = \theta_k(0) \ \ \forall\, k > j\} \end{aligned} \tag{13}$$

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 $\text{opt}(n, \zeta, |\Theta_\zeta| - 1)$ for $n \le |L|$, since these parameters correspond to the optimal number of true rejects provided n false rejects and free choice of the thresholds. Hence an efficient computation scheme for the quantities $\text{opt}(n, j, i)$ allows to efficiently compute the Pareto front.

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

$$\text{opt}(n, j, i) = \begin{cases} \text{if } n = 0: & \sum_{k=1}^{\zeta} |\mathcal{E}_{\theta_k(0)}| \\ \text{if } n > 0, j = 0: & -\infty \\ \text{if } n > 0, j > 0, i = 0: & \text{opt}(n, j-1, |\Theta_{j-1}| - 1) \\ \text{if } 0 < n < i, j > 0: & \text{opt}(n, j, i-1) \\ \text{if } n \ge i > 0, j > 0: & \max\{\text{opt}(n, j, i-1), \text{opt}(n-i, j-1, \\ & \quad |\Theta_{j-1}| - 1) + |\mathcal{E}_{\theta_j(i)} \setminus \mathcal{E}_{\theta_j(0)}|\} \end{cases} \tag{14}$$

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 equal 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 (13). This is exactly the same as the term $\text{opt}(n, j-1, \left|\Theta_{j-1}\right| - 1)$.
- 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 $\text{opt}(n, j, i-1)$ 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 $\text{opt}(n-i, j-1, |\Theta_{j-1}| - 1)$ 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 $\left|\mathcal{E}_{\theta_j(i)} \setminus \mathcal{E}_{\theta_j(0)}\right|$.

This recursive scheme can be computed by DP, since the value i or j is decreased in every recursion, using loops over n, j and i (Algorithm 1); for memory efficiency we reduce the tensor $\text{opt}(n, j, i)$ to a matrix $\text{opt}(n, j)$ denoting the maximal number of true rejects with n false rejects and flexible thresholds in partitions $1, \ldots, j$. Since every evaluation of (14) itself is constant time, the computation scheme has an effort of $O(|L| \cdot \zeta \cdot \max_k |\Theta_k|)$ with memory efficiency $O(|L| \cdot \zeta)$. A standard back-tracing scheme reveals the related optimal threshold vectors.

3.4.3. *Local threshold adaptation by an efficient greedy strategy*

Albeit enabling an optimal choice of the local threshold vectors for given data, DP as proposed above (14) 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 $O(N/\zeta)$. We propose a greedy approximation scheme which yields to an only linear method (besides pre-processing).

The basic idea is to start with the initial setting analogical to $\text{opt}(0, \zeta, |\Theta_\zeta| - 1)$: 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 $|\mathcal{E}_{\theta_j(k+1)} \setminus \mathcal{E}_{\theta_j(k)}|$ 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. Relying on a greedy strategy, the algorithm may yield suboptimal solutions. However, as we see in experiments, results tightly approximate the optimal choices. Unlike the exact algorithm, the greedy strategy only requires $O(|L| \cdot \zeta)$ time and $O(\zeta)$ memory.

4. Experiments

4.1. *Classifiers*

We use the following classifiers with their related certainty measure: prototype-based, DT and SVM classifiers. We ground local rejection on a natural tessellation Υ_j of the input space induced by these classifiers.

4.1.1. *Prototype-based classifiers*

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 a prototype $\mathbf{w}_j$ has a class label c_j. A data point $\mathbf{x}$ is classified according to its closest prototype

$$c(\mathbf{x}) = c_l \quad \text{with } l = \arg \min_{j=1,\ldots,\xi} d(\mathbf{w}_j, \mathbf{x}) \tag{15}$$

where d is a distance measure; e.g. the Euclidean distance. Prototype-based models aim at a spars data representation by a predefined number of prototypes. Such a classifier partitions the data into *Voronoi cells* or *receptive fields*

$$\Upsilon_j = \{\mathbf{x} \mid d(\mathbf{w}_j, \mathbf{x}) \le d(\mathbf{w}_k, \mathbf{x}), \forall\, k \ne j\}, \quad j = 1, \ldots, \xi; \tag{16}$$

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

Prototype locations are usually learned based on a given training data set X with N data points $(\mathbf{x}_i, y_i) \in \mathbb{R}^M \times \{1, \ldots, Z\}$ of Z different classes, aiming at a high classification performance. While classical training schemes are based on heuristics such as the Hebbian learning paradigm [30], more recent training schemes rely on a cost function, including generalised LVQ (GLVQ) [50], its extension to an adaptive matrix: generalised matrix LVQ (GMLVQ) [31], its local version (LGMLVQ) [31] with local adaptive metrics, and statistical counterparts, the robust soft LVQ (RSLVQ) [12]. Lately there are variants using kernels like the SVM as well [51].

GMLVQ: The GMLVQ [31] realises a stochastic gradient decent on the cost function in [50] with a more general metric d_Λ than the standard Euclidean one. This cost function is differentiable and it approximates the 0–1 loss:

$$E_{\text{GMLVQ}} = \sum_{i=1}^{N} \Phi((d_\Lambda^+ - d_\Lambda^-)/(d_\Lambda^+ + d_\Lambda^-)). \tag{17}$$

The metric d_Λ is defined as general quadratic form

$$d_\Lambda(\mathbf{w}, \mathbf{x}) = (\mathbf{x} - \mathbf{w})^T \Lambda (\mathbf{x} - \mathbf{w}) \tag{18}$$

with a semi-positive definite matrix Λ. The value $d_\Lambda^+ = d_\Lambda(\mathbf{w}_j, \mathbf{x}_i)$ is the distance of a data point $\mathbf{x}_i$ to the closest prototype $\mathbf{w}_j$ belonging to the same class and $d_\Lambda^- = d_\Lambda(\mathbf{w}_k, \mathbf{x}_i)$ is the distance of a data point $\mathbf{x}_i$ to the closest prototype $\mathbf{w}_k$ belonging to a different class. Φ is a monotonically increasing function, e.g. the Sigmoid.

Using a local metric $d_{\Lambda_j} = (\mathbf{x} - \mathbf{w}_j)^T \Lambda_j (\mathbf{x} - \mathbf{w}_j)$ for every prototype $\mathbf{w}_j$ instead of (18), one gets the LGMLVQ [31].

Rejection measure: *RelSim*. The relative similarity (RelSim) is a certainty measure which is related to the cost function of GLVQ [50,16]. It relies on the normalised distance d^+ 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^-:

$$r(\mathbf{x}) = \text{RelSim}(\mathbf{x}) := (d^- - d^+)/(d^- + d^+) \tag{19}$$

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

RSLVQ: RSLVQ [12] optimises the data log likelihood of a probabilistic model with a gradient ascent with respect to the prototypes:

$$\log\ L := \sum_{1 \le i \le N} \log(p(\mathbf{x}_i, y_i \mid W)/p(\mathbf{x}_i \mid W)).$$

The quantity $p(\mathbf{x} \mid W) = \sum_{1 \le j \le \xi} P(j) \cdot p(\mathbf{x} \mid j)$ is a mixture of Gaussians with uniform prior probability $P(j)$ and Gaussian probability $p(\mathbf{x}|j)$ centred in $\mathbf{w}_j$ which is isotropic with fixed variance. There exist schemes adapting the bandwidth additionally [52,53]. The probability $p(\mathbf{x}, y \mid W) = \sum_{j: c(\mathbf{w}_j) = y} P(j) \cdot p(\mathbf{x} \mid j)$ restricts to mixture components with correct labelling. Relying on a probability model, RSLVQ provides an explicit confidence value $p(y|\mathbf{x}, W)$ for every pair $\mathbf{x}$ and y, paying the price of a higher computational complexity for learning.

Rejection measure: *empirical estimation of the Bayesian confidence*. Probabilistic models like the RSLVQ provide explicit estimations of the probability $\hat{P}(j \mid \mathbf{x})$ of class j given a data point $\mathbf{x}$ [16] leading to the certainty measure

$$r(\mathbf{x}) = \text{Conf}(\mathbf{x}) := \max_{1 \le j \le Z} \hat{P}(j \mid \mathbf{x}). \tag{20}$$

4.1.2. Basic decision trees for classification

A DT for classification [11] is a rooted tree with one 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 ends defines the class label. We denote the decision border induced by the DT as Γ; a data point $\mathbf{x}$ is mapped to the class label $c(\mathbf{x})$ defined by the DT by a leaf. Note that the split-criterion sets a threshold for a specific dimension of the input space, e.g. on the first one: $\mathbf{x}(1) < \beta$, hence it defines a hyperplane. We consider axis parallel decision borders only. We use the receptive fields of the leaves of a given DT as partition:

$$\Upsilon_j := \{\mathbf{x} \mid \mathbf{x} \text{ falls into leaf } \alpha_j\}, \quad j = 1, \ldots, \Xi; \tag{21}$$

Rejection measure: *distance to decision border for DT* [10]. 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. Then the minimal distance of $\mathbf{x}$ and this set of projections gives $d(\mathbf{x}, \Gamma)$ and the certainty measure

$$r(\mathbf{x}) = \text{Dist}(\mathbf{x}) := d(\mathbf{x}, \Gamma). \tag{22}$$

An extension towards DTs with more general decision borders (such as non-axes parallel cuts, or borders induced by a general quadratic form) is provided in [54].

4.1.3. Support vector machine for classification

The popular SVM classifier uses an implicit nonlinear embedding of the data into a high dimensional kernel space. For a binary setting, it realises the generalised nonlinear classification

$$\mathbf{x} \mapsto H(\mathbf{w}^T \cdot \Phi(\mathbf{x})) \tag{23}$$

with Heaviside function H, linear weighing $\mathbf{w}$, and feature map Φ which is usually implicitly realised efficiently via a suitable kernel mapping. Training is phrased as 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 $(\mathbf{w}_{ij})^T \cdot \Phi(\mathbf{x})$ where $\mathbf{w}_{ij}$ refers to the decision 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, then we define the input space partitioning according to the classes (24). Note that this is a general partitioning of the space which can be applied for any classifier:

$$\Upsilon_j = \{\mathbf{x} \mid c(\mathbf{x}) = j\}, \quad j = 1, \ldots, Z. \tag{24}$$

Rejection measure: *class probability estimates of SVM.* The approach by Platt [6] 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. Wu et al. [7] extend this method for multi-class tasks and it is integrated in the LIBSVM toolbox [38]. The method leads to a certainty measure similar to (20), but the empirical probability estimates are extracted from the SVM.

4.2. Data sets

For evaluation, we consider the following benchmark data sets:

Gaussian clusters: This data set contains two overlapping 2D Gaussian clusters with means $\mu_x = (-4, 4.5)$, $\mu_y = (4, 0.5)$, and standard deviations $\sigma_x = (5.2, 7.1)$ and $\sigma_y = (2.5, 2.1)$. The points are overlaid with uniform noise.

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

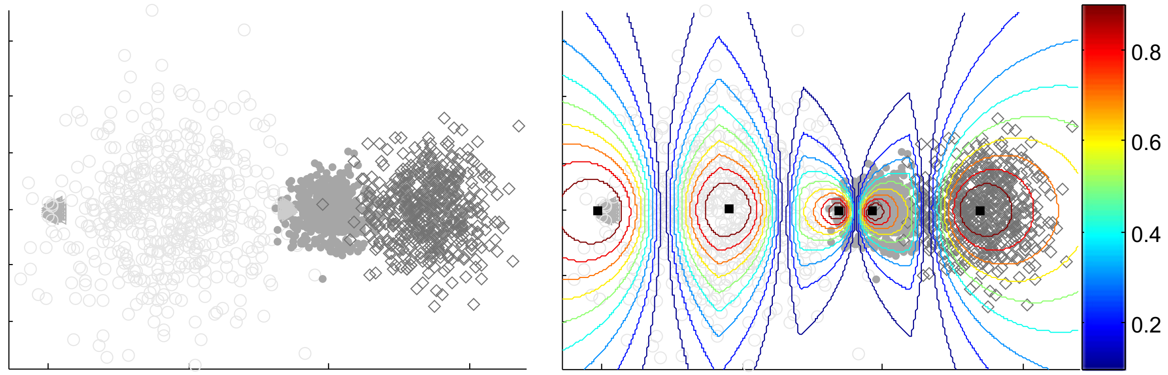


Fig. 2. [28] The Pearl Necklace with prototypes trained by GMLVQ (black squares) without metric adaptation. The coloured curves are the contour lines of RelSim (19). Note that a critical region for a global threshold is between the second and the third cluster from left. The third cluster needs a high threshold because the data points are very compact. Applying the same threshold for the second cluster would reject most data points in this cluster which is not optimal. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

Image segmentation: The image segmentation data consist of 2310 data points which contain 19 real-valued image descriptors. The data represent small patches from outdoor images of 7 different classes, equally distributed such as grass, cement, etc. [55].

Tecator: The Tecator data [56] consist of 215 spectra of meat probes. The 100 spectral bands range from 850 nm to 1050 nm. The task is predicting the fat content (high/low) of the probes. It is a small, balanced classification problem.

Coil: The Columbia Object Image Database Library contains gray scaled images of 20 objects [57]. Each object is rotated in 5° steps, resulting in 72 images per object. It contains 1440 vectors with 16, 384 dimensions that are reduced with PCA [58] to 30.

Since the complete ground truth is available for the first two, artificial data sets, we use the optimal Bayesian rejection [3] as a Gold standard for comparison in these two cases.

4.3. Experimental set-up

We evaluate the exact and approximate algorithms to determine optimal thresholds as proposed above. We use a 10-fold repeated cross-validation with ten repeats. We evaluate the models obtained by RSLVQ, GMLVQ, and LGMLVQ with one prototype per class,[1] DT, and SVM. Since RSLVQ provides probability estimates, we combine it with the certainty measure Conf (20). In turn, GMLVQ and LGMLVQ lend itself to the certainty measure RelSim (19). For the DT we use the default settings of the Statistics Toolbox of Matlab[2] except of the *splitmin*-parameter and use the related certainty measure Dist (22). We use the SVM [38] with a rbf-kernel and choose the best parameters of a cross-validation and a standard certainty measure [6,7].

In Figs. 4–6, 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.

4.4. Comparison of DP versus greedy optimisation

First, we compare the performance of the greedy optimisation and the optimal DP scheme for all benchmark data sets and algorithms. 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 optimised using DP.

We show only the results for RSLVQ and GMLVQ but the results of LGMLVQ, DT, and the SVM look similar. The mean squared error of both curves created by DP and the greedy strategy is below 0.0015 for all experiments, for all but two it is even below $2.1 \cdot 10^{-5}$ (Fig. 3). Hence the greedy optimisation provides near optimal results for realistic settings, while requiring less time and memory complexity. In the following, we use the greedy optimisation for the local reject options.

4.5. 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 first two data sets, we compare local and global reject options with the optimal Bayes rejection (Fig. 4). Thereby, RSLVQ is combined with Conf as certainty measure, while RelSim is used for deterministic LVQ models, relying on the insights as gained in the studies [50,16,17,28], the DT model uses Dist as certainty measure and the SVM uses the estimated class probabilities. For all settings, the performance of the classifier on the test set is shown, after optimising classifier parameters and threshold values on the training set. Results of a repeated cross-validation are shown, as specified before.

Gaussian clusters: For this data, 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, but a computationally more efficient global reject option suffices. Only for DT, the curves are different and local rejection boosts the performance significantly. For SVM, 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 as can be seen in the left part of the ARCs. RSLVQ even enables a close to optimal rejection for the full regime as well as the local rejection for DT (Fig. 4).

Pearl necklace: The pearl necklace data set is designed to show the advantage of local rejection (Fig. 2). 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. 4, neither RSLVQ nor GMLVQ reach the optimal decision quality, but the ARC curves are greatly improved when using a local instead of a global rejection 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

[1] We use the LVQ toolbox at: http://matlabserver.cs.rug.nl/gmlvqweb/web/.

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

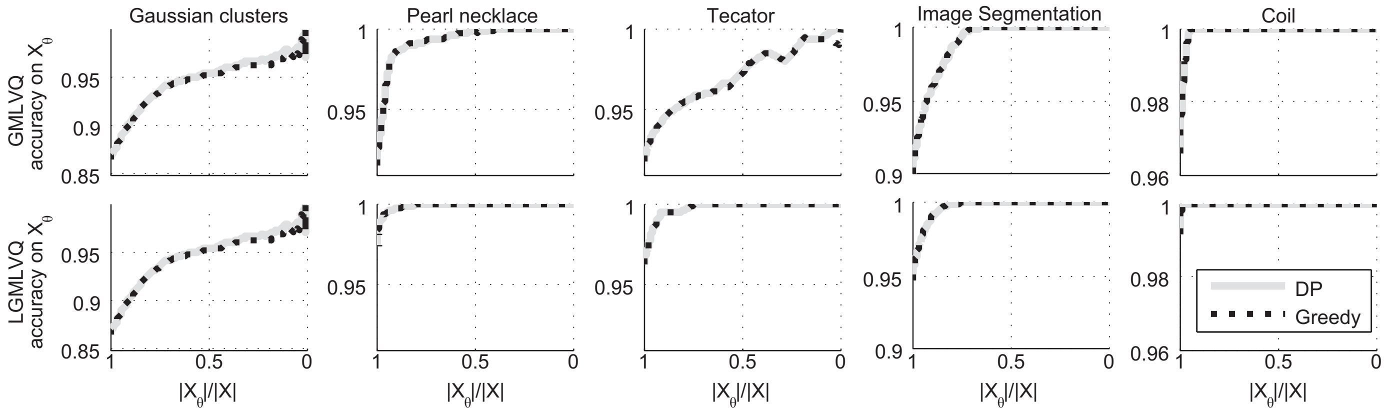


Fig. 3. Averaged accuracy reject curves for dynamic programming (DP) and the greedy optimisation applied on artificial and benchmark data sets for the relative similarity.

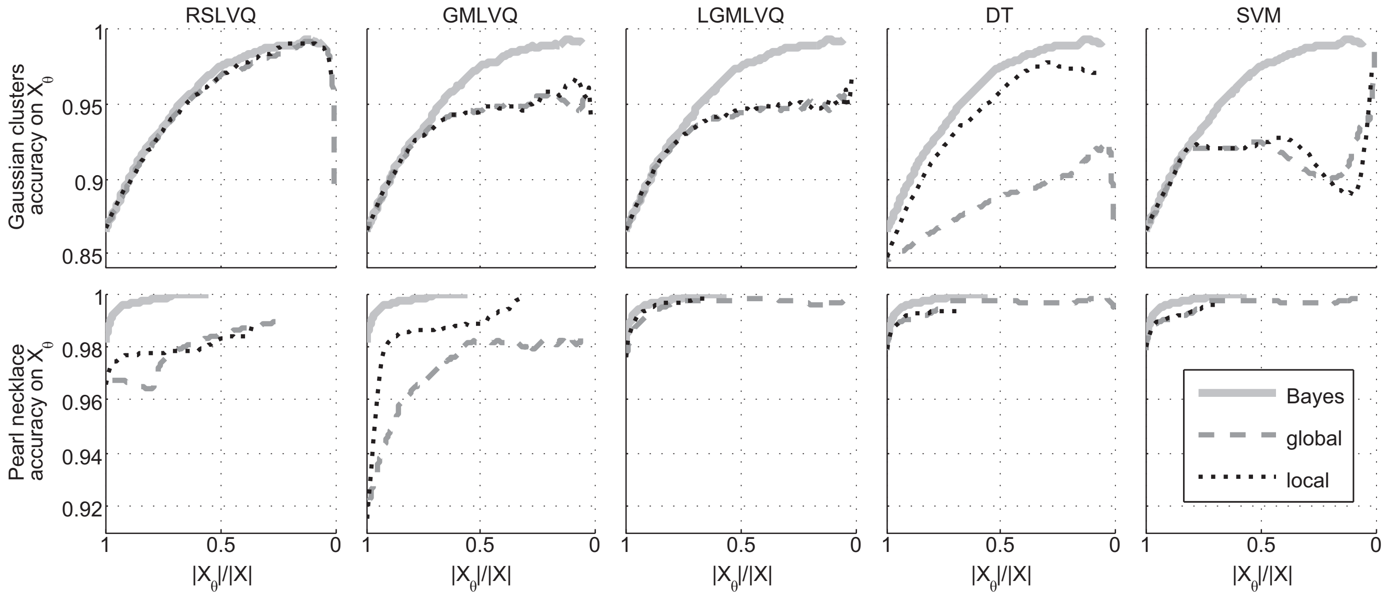


Fig. 4. Averaged ARCs for global and local rejection evaluated on the test sets. For RSLVQ Conf (20) serves as certainty measure and for the other two LVQ models RelSim (19) serves as certainty measure. For DT the certainty measure Dist is applied and the SVM uses the estimated class probabilities. The Bayes rejection with known class probabilities provides a Gold standard for comparison.

data. In comparison, LGMLVQ is capable of reaching the optimal Bayesian reject border 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 in between local and global rejection for well performing classifiers.

4.6. *Experiments on benchmarks*

For the benchmark data sets, the underlying density models are unknown, hence we do not report the result of optimal Bayes rejection. Fig. 5 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 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 is not necessarily given.
- Threshold optimisation by means of a linear time greedy strategy displays the same accuracy as computationally more complex optimal choices.

4.7. *Prosody data*

The prosody data contains 1,866 data points related to prominent

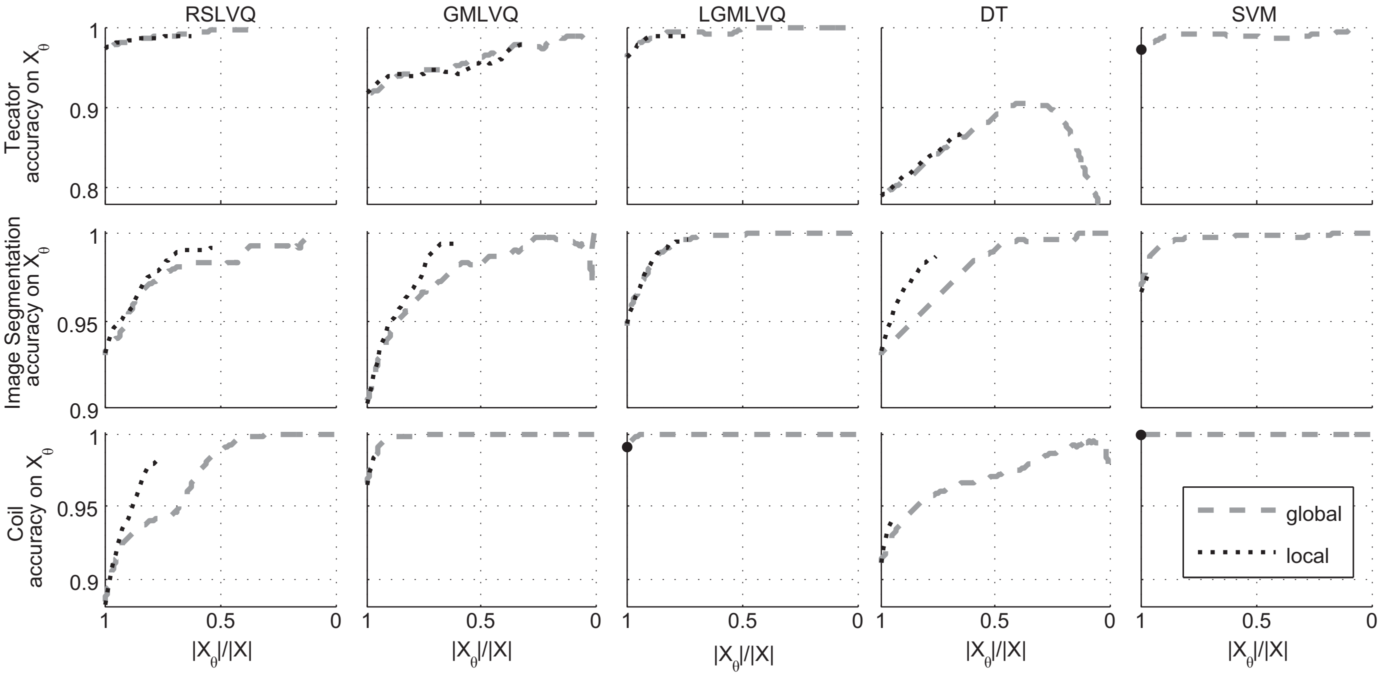


Fig. 5. Averaged ARCs for global and local rejection of the test sets. For RSLVQ Conf (20) serves as certainty measure, for the other two LVQ models the RelSim (19), for the DT the Dist (22) and the SVM uses its estimated class probabilities. The black points show the performance of the related classifier without local rejection, if the obtained local thresholds from the training set do not affect the test set.

words and 12,191 data points of non-prominent words. Hence, it is a very imbalanced data set. A single data point consists of 159 audio features. The data was collected with a Wizard-of-Oz setting in a small cartoon game [59]. The eight (male/female) subjects had to interact with a computer using spoken advises with simple grammar. Some words were misunderstood by the computer and the subjects had to correct the sentence. They repeated the sentence with emphasis on the misunderstood word. The classification task is to distinguish prominent from non-prominent words in order to correct the misunderstood word. Note that, each subject has a highly individual characteristic of emphasising the corrected word. For further informations and recent results we refer to [60].

This data set contains highly imbalanced classes which are also strongly overlapping. Therefore, we trained the LVQ classifiers with 15 prototypes per class. Fig. 6 contains the results of the classifiers on this data set. As one can see, local rejection strongly enhances the performance of the classifiers except in case of the SVM. On the one hand the SVM has the highest accuracy without rejection and on the other hand there are only two local thresholds which can be chosen (class-wise). Two thresholds offer less flexibility as for instance the 15 prototypes per class in the LVQ classifiers. This data set is an example where global rejection can be less effective while local rejection performs much better (GMLVQ). For DT, the RSLVQ and the LGMLVQ, global rejection works but local rejection is clearly a better choice since this strategy performs much better for this data.

4.8. Medical application

We conclude with a recent example from the medical domain. The adrenal tumours data [61] contains 147 data points composed of 32 steroid marker values. Two unbalanced classes are present: patients with benign adrenocortical adenoma (102 points) or malignant carcinoma (45 points). The 32 steroid marker values are measured from urine samples using gas chromatography/mass spectrometry. For further medical details we refer to [61,62].

Our analysis of the data and the pre-processing follow the evaluations in [61,62]: we train a GMLVQ model with one prototype per class. For the evaluation of rejection we split the data into a training set (90%) and a test set (10%). We evaluate the ARC of

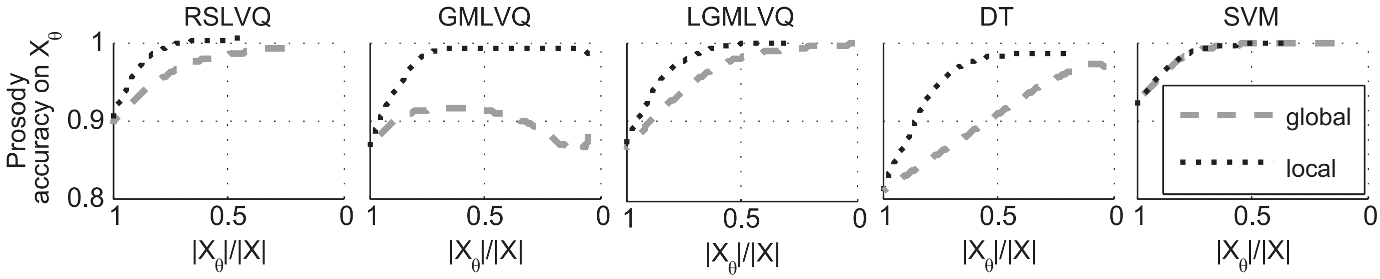


Fig. 6. Averaged ARCs for global and local rejection of the prosody data test sets. For RSLVQ Conf (20) serves as certainty measure, for the other two LVQ models the RelSim (19), for the DT the Dist (22) and the SVM uses its estimated class probabilities.

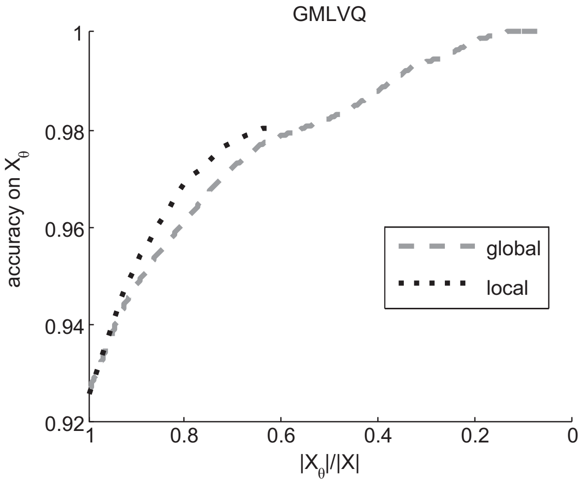


Fig. 7. Averaged ARCs for global and local rejection based on (19) (test set).

1000 random data splits and the corresponding GMLVQ models. Fig. 7 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 provides insight into potentially relevant biomarkers and prototypical representatives of the classes [62]. As a conclusion, the GMLVQ together with the proposed rejection offers a reliable and compact classifier for this medical application. Lately results from the medical domain using learning vector quantisation [63] could probably benefit from a reject option, too, in order to improve the performance.

5. Conclusion

In this article, we derived theoretical and practical results for classifiers with rejection. A rejection strategy can be global (one threshold) or local (several thresholds).

In our theory part, we introduced two algorithms for determining optimal local thresholds for the latter strategy: (i) an optimal technique based on DP and (ii) a fast greedy approximation. While the first is provably optimal, the latter is based on heuristics. The time complexity of the greedy approximation 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. Also we linked the optimisation problem of finding optimal local thresholds with the multiple choice knapsack problem.

In our practical part, we firstly compared the results of DP with the results of the greedy approximation. Our experiments show that both solutions have very similar results such that the fast greedy solution instead of the more complex DP solution seems a reasonable choice. Secondly, we compared global and local rejection strategies on several benchmarks and two real-life data sets for three classifier types: prototype-based, DT and SVM classifiers. The benefit of local strategies becomes apparent especially 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. Nevertheless, as the results of the prosody data showed, local rejection can provide a much better performance than the global counterpart. Interestingly, the proposed rejection strategies in combination with the intuitive deterministic LGMLVQ lead to results which are comparable to SVM and related reject options. Thereby, the LVQ techniques base the rejection on their distance to few prototypes only, hence they open the way towards efficient techniques for online scenarios.

So far, the rejection strategies have been designed and evaluated for offline training scenarios only, disregarding the possibility of trends present in life long learning scenarios, or its coupling to possibly varying costs for rejects versus errors. We will analyse in future work how to extend the proposed methods to online scenarios and life long learning, where accordingly thresholds are picked automatically based on the proposed results in this article.

Acknowledgements

The authors would like to thank Stephan Hasler for the basic idea of the greedy optimisation of local thresholds and helpful discussions thereon. The authors are grateful for the provided medical data and the support in related issues by Wiebke Arlt and Michael Biehl. The authors would like to thank Andrea Schnall for providing the Prosody data set and the support in related questions. This research was partly supported by the Cluster of Excellence Cognitive Interaction Technology 'CITEC' (EXC 277) at Bielefeld University, funded by the German Research Foundation (DFG).

Appendix A

Algorithm 1. DP(X, classifier).

```
$h := \sum_{k=1}^{\zeta} |\mathcal{E}_{\theta_k(0)}|$;
for $k := 0, \ldots, \zeta$
  do opt$(0,k) := h$;
for $n := 1, \ldots, |L|$
  do for $k := 0, \ldots, \zeta$
       do opt$(n,k) := -\infty$;
for $n := 1, \ldots, |L|$                //loop over number of false rejects
  do for $j := 1, \ldots, \zeta$         //loop over partitions
       do opt$(n,j) :=$ opt$(n,j-1)$;
          //loop over thresholds in partition j that agree with false rejects
          for $i := 1, \ldots, \min\{n, |\Theta_j| - 1\}$
            do $n' := n - i$;
               $gain := |\mathcal{E}_{\theta_j(i)} \backslash \mathcal{E}_{\theta_j(0)}|$;
               $h :=$ opt$(n', j-1) + gain$;
               if $h >$ opt$(n,j)$
                 then opt$(n,j) := h$;
// compute threshold vector by back-tracing; init with default value
for $n := 0, \ldots, |L|$
  do for $k := 1 \ldots \zeta$
       do $\boldsymbol{\theta}(n,k) := \theta_k(0)$;
for $n := 1, \ldots, |L|$                //back-tracing in the matrix opt
  do $j := \zeta$; // start in last partition
     $n' := n$;    $i := \min(n', |\Theta_j| - 1)$;
     while $j > 0$
       do if $i = 0$
            then $j := j - 1$;          //take threshold 0
                 $i := \min(n', |\Theta_j| - 1)$;
            else $n'' := n' - i$;
                 $gain := |\mathcal{E}_{\theta_j(i)} \backslash \mathcal{E}_{\theta_j(0)}|$;
                 $h :=$ opt$(n'', j-1) + gain$;
                 if opt$(n', j) = h$
                   then $\boldsymbol{\theta}(n,j) := \theta_j(i)$;     //take threshold  $i$
                        $n' := n''$;    $j := j - 1$;
                        $i := \min(n', |\Theta_j| - 1)$;
                   else $i := i - 1$;      //take smaller threshold
return (matrices opt$(n,k)$ and $\boldsymbol{\theta}(n,k)$)
```

Appendix B

Algorithm 2. GREEDY(X, classifier).

```
for $j := 1, \dots, \zeta$                         //initialisation by first thresholds
  do $I(j) := 0$;
$h := \sum_{k=1}^{\zeta} |\mathscr{E}_{\theta_k(0)}|$
$|\mathscr{E}_{\boldsymbol{\theta}}| := h$;     $n := 0$;     $s := 1$;
$t_c(s) := 1 - |\mathscr{E}_{\boldsymbol{\theta}}|/|X|$;  $t_a(s) := |L|/(|X| - |\mathscr{E}_{\boldsymbol{\theta}}|)$;
while $|\mathscr{E}_{\boldsymbol{\theta}}| \neq |E|$                 //loop while true rejects can be increased
do
    $gain := \max_j \{|\mathscr{E}_{\theta_j(I(j)+1)} \setminus \mathscr{E}_{\theta_j(I(j))}|\}$; //most improvement locally
    $I_{gain} := \arg\max_j \{|\mathscr{E}_{\theta_j(I(j)+1)} \setminus \mathscr{E}_{\theta_j(I(j))}|\}$;
    $GAIN := \max_j \{|\mathscr{E}_{\theta_j(n+1)} \setminus \mathscr{E}_{\theta_j(0)}|\}$;    //most improvement globally
    $I_{GAIN} := \arg\max_j \{|\mathscr{E}_{\theta_j(n+1)} \setminus \mathscr{E}_{\theta_j(0)}|\}$;
    if $GAIN > (gain + |\mathscr{E}_{\boldsymbol{\theta}}| - h)$
      then
        for $j := 1, \dots, \zeta$
          do $I(j) := 0$;
        $I(I_{GAIN}) := n$;
        $|\mathscr{E}_{\boldsymbol{\theta}}| := GAIN + h$;
        $n := n + 1$;
      else
        if $I_{gain}$ is unique
          then
            $I(I_{gain}) := I(I_{gain}) + 1$;
            $|\mathscr{E}_{\boldsymbol{\theta}}| := |\mathscr{E}_{\boldsymbol{\theta}}| + gain$;
            $n := n + 1$;
          else
            // increase false rejects
            $o := 1$;
            repeat
              $o := o + 1$;
              $gain := \max_j \{|\mathscr{E}_{\theta_j(I(j)+o)} \setminus \mathscr{E}_{\theta_j(I(j))}|\}$;
              $I_{gain} := \arg\max_j \{|\mathscr{E}_{\theta_j(I(j)+o)} \setminus \mathscr{E}_{\theta_j(I(j))}|\}$;
            until $I_{gain}$ is unique;
            $n := n + o$;
            $I(I_{gain}) := I(I_{gain}) + o$;
            $|\mathscr{E}_{\boldsymbol{\theta}}| := |\mathscr{E}_{\boldsymbol{\theta}}| + gain$;
    $s := s + 1$;
    $\mathbf{t}_c(s) := 1 - (n + |\mathscr{E}_{\boldsymbol{\theta}}|)/|X|$; $\mathbf{t}_a(s) := (|L| - n)/(|X| - (n + |\mathscr{E}_{\boldsymbol{\theta}}|))$;
return $(\mathbf{t}_c, \mathbf{t}_a)$
```

References

[1] J. Yu, D. Tao, R. Hong, X. Gao, Recent developments on deep big vision, Neurocomputing 187 (2016) 1–3.

[2] C.M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

[3] C.K. Chow, On optimum recognition error and reject tradeoff, IEEE Trans. Inf. Theory 16(1) (1970) 41–46.

[4] B.E. Boser, I. Guyon, V. Vapnik, A training algorithm for optimal margin classifiers, in: Proceedings of the 5th Annual ACM Conference on Computational Learning Theory (COLT), 1992, pp. 144–152.

[5] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (3) (1995) 273–297.

[6] J.C. Platt, Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods, in: Advances in Large Margin Classifiers, MIT Press, 1999, pp. 61–74.

[7] T.-F. Wu, C.-J. Lin, R.C. Weng, Probability estimates for multi-class classification by pairwise coupling, J. Mach. Learn. Res. 5 (2004) 975–1005.

[8] H. Lu, S. Wei, Z. Zhou, Y. Miao, Y. Lu, Regularised extreme learning machine with misclassification cost and rejection cost for gene expression data classification, Int. J. Digit. Multimed. Broadcast. 12 (3) (2015) 294–312.

[9] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (1–3) (2006) 489–501.

[10] I. Alvarez, S. Bernard, G. Deffuant, Keep the decision tree and estimate the class probabilities using its decision boundary, in: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI), 2007, pp. 654–659.

[11] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and Regression Trees, Wadsworth, 1984.

[12] S. Seo, K. Obermayer, Soft learning vector quantization, Neural Comput. 15 (7) (2003) 1589–1604.

[13] R. Herbei, M.H. Wegkamp, Classification with reject option, Can. J. Stat. 34 (4) (2006) 709–721.

[14] A. Vailaya, A.K. Jain, Reject option for VQ-based Bayesian classification, in: International Conference on Pattern Recognition (ICPR), 2000, pp. 2048–2051.

[15] R. Hu, S.J. Delany, B.M. Namee, 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, 2009, pp. 181–192.

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

[17] L. Fischer, D. Nebel, T. Villmann, B. Hammer, H. Wersing, Rejection strategies for learning vector quantization – a comparison of probabilistic and deterministic approaches, in: Advances in Self-Organizing Maps and Learning Vector Quantization, Advances in Intelligent Systems and Computing, vol. 295, 2014, pp. 109–118.

[18] G. Fumera, F. Roli, Support vector machines with embedded reject option, in: Proceedings of the Pattern Recogition with SVM, 2002, pp. 68–82.

[19] Y. Grandvalet, A. Rakotomamonjy, J. Keshet, S. Canu, Support vector machines with a reject option, in: Advances in Neural Information Processing Systems (NIPS), 2008, pp. 537–544.

[20] P.L. Bartlett, M.H. Wegkamp, Classification with a reject option using a hinge loss, J. Mach. Learn. Res. 9 (2008) 1823–1840.

[21] M. Yuan, M. Wegkamp, Classification methods with reject option based on convex risk minimization, J. Mach. Learn. Res. 11 (2010) 111–130.

[22] I. Pillai, G. Fumera, F. Roli, Threshold optimisation for multi-label classifiers, Pattern Recognit. 46 (7) (2013) 2055–2065.

[23] I. Pillai, G. Fumera, F. Roli, Multi-label classification with a reject option, Pattern Recognit. 46 (8) (2013) 2256–2266.

[24] D.M.J. Tax, R.P.W. Duin, Growing a multi-class classifier with a reject option, Pattern Recognit. Lett. 29 (10) (2008) 1565–1570.

[25] H.G. Ramaswamy, A. Tewari, S. Agarwal, Consistent algorithms for multiclass classification with a reject option CoRR, abs/1505.04137.

[26] H.L. Capitaine, A unified view of class-selection with probabilistic classifiers, Pattern Recognit. 47 (2) (2014) 843–853.

[27] G. Fumera, F. Roli, G. Giacinto, Reject option with multiple thresholds, Pattern Recognit. 33 (12) (2000) 2099–2101.

[28] L. Fischer, B. Hammer, H. Wersing, Local rejection strategies for learning vector quantization, in: Proceedings of the International Conference on Artificial Neural Networks (ICANN), 2014, pp. 563–570.

[29] A.K. Chandra, D.S. Hirschberg, C.K. Wong, Approximate algorithms for some generalized knapsack problems, Theor. Comput. Sci. 3 (3) (1976) 293–304.

[30] T. Kohonen, Self-Organization and Associative Memory, Springer Series in Information Sciences, third edition, Springer-Verlag, 1989.

[31] P. Schneider, M. Biehl, B. Hammer, Adaptive relevance matrices in learning vector quantization, Neural Comput. 21 (12) (2009) 3532–3561.

[32] P. Schneider, K. Bunte, H. Stiekema, B. Hammer, T. Villmann, M. Biehl, Regularization in matrix relevance learning, IEEE Trans. Neural Netw. 21 (5) (2010) 831–840.

[33] M.S.A. Nadeem, J.-D. Zucker, B. Hanczar, Accuracy-rejection curves (ARCs) for comparing classification methods with a reject option, in: Workshop on Machine Learning in Systems Biology (MLSB), 2010, pp. 65–81.

[34] T. Landgrebe, D.M.J. Tax, P. Paclík, R.P.W. Duin, The interaction between classification and reject performance for distance-based reject-option classifiers, Pattern Recognit. Lett. 27 (8) (2006) 908–917.

[35] L.K. Hansen, C. Liisberg, P. Salomon, The Error-reject Tradeoff, Technical Report, Electronics Institute, Technical University of Denmark, 1994.

[36] P.R. Devarakota, B. Mirbach, B. Ottersten, Confidence estimation in classification decision: a method for detecting unseen patterns, in: International Conference on Advances in Pattern Recognition (ICAPR), 2007, pp. 136–140.

[37] E. Ishidera, D. Nishiwaki, A. Sato, A confidence value estimation method for handwritten Kanji character recognition and its application to candidate reduction, Int. J. Doc. Anal. Recognit. 6 (4) (2004) 263–270.

[38] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, ACM Trans. Intell. Syst. Technol. 2 (2011) 27:1–27:27.

[39] M. Sugiyama, K.M. Borgwardt, Rapid distance-based outlier detection via sampling, in: Proceedings of the Conference on Neural Information Processing Systems, 2013, pp. 467–475.

[40] S. Ramaswamy, R. Rastogi, K. Shim, Efficient algorithms for mining outliers from large data sets, in: SIGMOD International Conference on Management of Data, 2000, pp. 427–438.

[41] R. Sousa, J.S. Cardoso, The data replication method for the classification with reject option, AI Commun. 26 (3) (2013) 281–302.

[42] C.D. Stefano, C. Sansone, M. Vento, To reject or not to reject: that is the question—an answer in case of neural classifiers, IEEE Trans. Syst. Man Cybern. Part C 30 (1) (2000) 84–94.

[43] A. Tewari, P.L. Bartlett, On the consistency of multiclass classification methods, J. Mach. Learn. Res. 8 (2007) 1007–1025.

[44] R. Herbei, M. Wegkamp, Classification with reject option, Can. J. Stat. 34 (4) (2006) 709–721.

[45] W. Tang, E.S. Sazonov, Highly accurate recognition of human postures and activities through classification with rejection, IEEE J. Biomed. Health Inform. 18 (1) (2014) 309–315.

[46] R. Bellman, Dynamic Programming, Princeton University Press, 1957.

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

[48] D. Pisinger, A minimal algorithm for the multiple-choice knapsack problem, Eur. J. Oper. Res. 83 (2) (1995) 394–410 (EURO Summer Institute Combinatorial Optimization).

[49] K. Dudzinski, S. Walukiewicz, Exact methods for the knapsack problem and its generalizations, Eur. J. Oper. Res. 28 (1) (1987) 3–21.

[50] A. Sato, K. Yamada, Generalized learning vector quantization, in: Advances in Neural Information Processing Systems (NIPS), vol. 7, 1995, pp. 423–429.

[51] T. Villmann, S. Haase, M. Kaden, Kernelized vector quantization in gradient-descent learning, Neurocomputing 147 (2015) 83–95 (Advances in Self-Organizing Maps Subtitle of the special issue: Selected Papers from the Workshop on Self-Organizing Maps 2012 (WSOM 2012)).

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

[53] S. Seo, K. Obermayer, Dynamic hyperparameter scaling method for LVQ algorithms, in: Proceedings of the IJCNN, 2006, pp. 3196–3203.

[54] N. Tóth, B. Pataki, On classification confidence and ranking using decision trees, in: International Conference on Intelligent Engineering Systems, IEEE, 2007, pp. 133–138.

[55] K. Bache, M. Lichman, UCI Machine Learning Repository (2013).

[56] H.H. Thodberg, Tecator data set, contained in StatLib Datasets Archive, 1995.

[57] S.A. Nene, S.K. Nayar, H. Murase, Columbia Object Image Library (COIL-20), Technical Report CUCS-005-96, 1996.

[58] L.J.P. van der Maaten, Matlab Toolbox for Dimensionality Reduction (2013).

[59] M. Heckmann, Audio-visual evaluation and detection of word prominence in a human-machine interaction scenario, in: INTERSPEECH 2012, 13th Annual Conference of the International Speech Communication Association, Portland, Oregon, USA, September 9–13, 2012, ISCA, 2012, pp. 2390–2393.

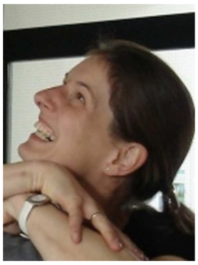
[60] A. Schnall, M. Heckmann, Speaker adaptation for word prominence detection with support vector machines, in: Speech Prosody, accepted, 2016.

[61] M. Biehl, P. Schneider, D. Smith, H. Stiekema, A. Taylor, B. Hughes, C. Shackleton, P. Stewart, W. Arlt, Matrix relevance LVQ in steroid metabolomics based classification of adrenal tumors, in: European Symposium on Artificial Neural Networks (ESANN), 2012, pp. 423–428.

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

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

Lydia Fischer received her M. Sc. in Discrete and Computer-oriented Mathematics at the University of Applied Sciences Mittweida in 2012. Since 2013 she is a PhD student at the CoR-Lab of the University of Bielefeld in cooperation with the HONDA Research Institute Europe.



Barbara Hammer received her Ph.D. in Computer Science in 1995 and her venia legendi in Computer Science in 2003, both from the University of Osnabrück, Germany. From 2000 to 2004, she was leader of the junior research group 'Learning with Neural Methods on Structured Data' at the University of Osnabrück before accepting an offer as professor for Theoretical Computer Science at Clausthal University of Technology, Germany, in 2004. Since 2010, she is holding a professorship for Theoretical Computer Science for Cognitive Systems at the CITEC cluster of excellence at Bielefeld University, Germany. Several research stays have taken her to Italy, UK, India, France, the Netherlands, and the USA. Her areas of expertise include hybrid systems, self-organizing maps, clustering, and recurrent networks as well as applications in bioinformatics, industrial process monitoring, or cognitive science. She is currently chairing the IEEE CIS Technical Committee on Data Mining, and the Fachgruppe Neural Networks of the GI. She has published more than 200 contributions to international conferences/journals, and she is co-author/editor of four books.

Heiko Wersing received the diploma in physics in 1996 from Bielefeld University, Germany. In 2000, he received his Ph.D. in science from the Faculty of Technology, Bielefeld University. In 2000, he became a member of the Future Technology Research Group of Honda R&D Europe, GmbH, Offenbach, Germany. Currently he holds a position as a chief scientist in the Honda Research Institute Europe, at Offenbach. From 2007 to 2013 he was co-speaker of the graduate school of the CoR-Lab Research Institute for Cognition and Robotics, Bielefeld University. His current research interests include recurrent neural networks, models of perceptual grouping and feature binding, principles of sparse coding, biologically motivated object recognition and online learning.