Explaining Reject Options of Learning Vector Quantization Classifiers

André Artelt[1,2][a], Johannes Brinkrolf[1][b], Roel Visser[1] and Barbara Hammer[1][c]
[1]*Faculty of Technology, Bielefeld University, Bielefeld, Germany*
[2]*KIOS – Research and Innovation Center of Excellence, University of Cyprus, Nicosia, Cyprus*

Keywords: XAI, Contrasting Explanations, Learning Vector Quantization, Reject Options.

Abstract: While machine learning models are usually assumed to always output a prediction, there also exist extensions in the form of reject options which allow the model to reject inputs where only a prediction with an unacceptably low certainty would be possible. With the ongoing rise of eXplainable AI, a lot of methods for explaining model predictions have been developed. However, understanding why a given input was rejected, instead of being classified by the model, is also of interest. Surprisingly, explanations of rejects have not been considered so far. We propose to use counterfactual explanations for explaining rejects and investigate how to efficiently compute counterfactual explanations of different reject options for an important class of models, namely prototype-based classifiers such as learning vector quantization models.

1 INTRODUCTION

Nowadays, machine learning (ML) based decision making systems are increasingly used in safety-critical or high-impact applications like autonomous driving (Sallab et al., 2017), credit (risk) assessment (Khandani et al., 2010) and predictive policing (Stalidis et al., 2018). Because of this, there is an increasing demand for transparency which was also recognized by the policy makers and emphasized in legal regulations like the EU's GDPR (European parliament and council, 2016). It is a common approach to realize transparency by explanations – i.e. providing an explanation of why the system behaved in the way it did – which gave rise to the field of explainable artificial intelligence (XAI or eXplainable AI) (Tjoa and Guan, 2019; Samek et al., 2017). Although it is still unclear what exactly makes up a "good" explanation (Doshi-Velez and Kim, 2017; Offert, 2017), a lot of different explanation methods have been developed (Guidotti et al., 2019; Molnar, 2019). Popular explanations methods (Molnar, 2019; Tjoa and Guan, 2019) are feature relevance/importance methods (Fisher et al., 2019) and examples based methods (Aamodt and Plaza., 1994). Instances of example based methods are contrasting explanations like counterfactual explanations (Wachter et al., 2017; Verma et al., 2020) and prototypes & criticisms (Kim et al., 2016) – these methods use a set or a single example for explaining the behavior of the system.

Another important aspect, in particular in safety-critical and high-risk applications, is reliability: if the system is not "absolutely" certain about its decision/prediction it might be better to refuse making a prediction and, for instance, pass the input back to human – if making mistakes is "expensive" or critical, the system should reject inputs where it is not certain enough in its prediction. As a consequence, a mechanism called reject options has been pioneered (Chow, 1970), where optimal reject rates are determined based on costs assigned to misclassifications (e.g. false positives and false negatives) and rejects. Many realizations of reject options are based on probabilities, like class probabilities in classification (Chow, 1970). However, not all models output probabilities along with their predictions or if they do, their computed probabilities might be of "bad quality" (e.g. just a score in $[0,1]$ without any probabilistic/statistical foundation). One possible remedy is to use a general (i.e. model agnostic) post-processing method or wrapper on top of the model for computing reliable certainty scores, as is done in (Herbei and Wegkamp, 2006), or use a method like conformal prediction (Shafer and Vovk, 2008) in which a non-conformity measure together with a hold-out data set is used for computing certain-

[a] https://orcid.org/0000-0002-2426-3126
[b] https://orcid.org/0000-0002-0032-7623
[c] https://orcid.org/0000-0002-0935-5591

Artelt, A., Brinkrolf, J., Visser, R. and Hammer, B.
Explaining Reject Options of Learning Vector Quantization Classifiers.
DOI: 10.5220/0011389600003332
In *Proceedings of the 14th International Joint Conference on Computational Intelligence (IJCCI 2022)*, pages 249-261
ISBN: 978-989-758-611-8; ISSN: 2184-3236

ties and confidences of predictions. Another option is to develop model specific certainty measures and consequently reject options, which for instance was done for prototype-based methods like learning vector quantization (LVQ) models. These are a class of models that maintain simplicity (and thus interpretability), while still being powerful predictive models, and have also excelled in settings like life-long learning and biomedical applications (Nova and Estévez, 2014; Kirstein et al., 2012; Xu et al., 2009; Owomugisha et al., 2020; Brinkrolf and Hammer, 2020a; Fischer et al., 2015b; Fischer et al., 2015a).

We think that while explaining the prediction of the model is important, similarly explaining why a given input was rejected is also important – i.e. explaining why the model "thinks" that it does not know enough for making a proper and reliable prediction. *For instance, consider a biomedical application where a model is supposed to assist in some kind of early cancer detection – such a scenario could be considered a high-risk application where the addition of a reject option to the model would be necessary. In such a scenario, it would be very useful if a reject is accompanied with an explanation of why this sample was rejected, because by this we could learn more about the model and the domain itself – e.g. an explanation could state that some combination of serum values is very unusual compared to what the model has seen before.* Surprisingly, (to the best of our knowledge) explaining rejects have not been considered so far.

Contributions. In this work, we focus on prototype-based models such as learning vector quantization (LVQ) models and propose to explain LVQ reject options by means of counterfactual explanations – i.e. explaining why a particular sample was rejected by the LVQ model. Whereby we consider different reject options separately and study how to efficiently compute counterfactual explanations in each of these cases.

The remainder of this work is structured as follows: after reviewing the foundations of learning vector quantization (Section 2.1), popular reject options for LVQ (Section 2.2), and counterfactual explanations (Section 2.3), we propose a modeling and algorithms for efficiently computing counterfactual explanations of different LVQ reject options (Section 3). In Section 4, we empirically evaluate our proposed modelings and algorithms on several different data sets with respect to different aspects. Finally, this work closes with a summary and conclusion in Section 5. Note that for the purpose of readability, all proofs and derivations are given in Appendix 5.

2 FOUNDATIONS

2.1 Learning Vector Quantization

In this work, we focus on learning vector quantization (LVQ) models. In modern variants, these models constitute state-of-the-art classifiers which can also be used for incremental, online, and federated learning (Gepperth and Hammer, 2016; Losing et al., 2018; Brinkrolf and Hammer, 2021). Modern versions area based on cost functions which have the benefit that they can be easily extended to a more flexible metric learning scheme (Schneider et al., 2009). First, we introduce generalized LVQ (GLVQ) (Sato and Yamada, 1995) and then take a look at extensions for metric learning. In the following, we assume that $\mathcal{X} = \mathbb{R}^d$ and $\{1,\ldots,k\} = \mathcal{Y}$. An LVQ model is characterized by m labeled prototypes $(\vec{\mathrm{p}}_j, c(\vec{\mathrm{p}}_j)) \in \mathbb{R}^d \times \mathcal{Y}$, $j \in \{1,\ldots,m\}$, whereby the labels $c(\vec{\mathrm{p}}_j)$ of the prototypes are fixed. Classification of a sample $\vec{x} \in \mathbb{R}^d$ takes place by a winner takes all scheme: $\vec{x} \mapsto c(\vec{\mathrm{p}}_j)$ with $j = \arg\min_{j\in\{1,\ldots,m\}} \mathrm{d}(\vec{x}, \vec{\mathrm{p}}_j)$ where the squared Euclidean distance is used:

$$\mathrm{d}(\vec{x}, \vec{\mathrm{p}}_j) = (\vec{x} - \vec{\mathrm{p}}_j)^\top (\vec{x} - \vec{\mathrm{p}}_j) \tag{1}$$

Note that the prototypes' positions not only allow an interpretable classification but also act as a representation of the data and its underlying classes. Training of LVQ models is done in a supervised fashion – i.e. based on given labeled samples $(\vec{x}_i, y_i) \in \mathbb{R}^d \times \mathcal{Y}$ with $i \in \{1,\ldots,N\}$. For GLVQ (Sato and Yamada, 1995), the learning rule originates from minimizing the following cost function:

$$E = \sum_{i=1}^{N} \mathrm{sgd}\left(\frac{\mathrm{d}(\vec{x}_i, \vec{\mathrm{p}}_+) - \mathrm{d}(\vec{x}_i, \vec{\mathrm{p}}_-)}{\mathrm{d}(\vec{x}_i, \vec{\mathrm{p}}_+) + \mathrm{d}(\vec{x}_i, \vec{\mathrm{p}}_-)}\right) \tag{2}$$

where $\mathrm{sgd}(\cdot)$ is a monotonously increasing function – e.g. the logistic function or the identity. Furthermore, $\vec{\mathrm{p}}_+$ denotes the closest prototype with the same label – i.e. $c(\vec{x}) = c(\vec{\mathrm{p}}_+)$ – and $\vec{\mathrm{p}}_-$ denotes the closest prototype which belongs to a different class. Note that the models' performance heavily depends on the suitability of the Euclidean metric for the given classification problem – this is often not the case, in particular in case of different feature relevances. Because of this, a powerful metric learning scheme has been proposed in (Schneider et al., 2009) which substitutes the squared Euclidean metric in Eq. (1) by a weighted alternative:

$$\mathrm{d}(\vec{x}, \vec{\mathrm{p}}) = (\vec{x} - \vec{\mathrm{p}})^\top \Omega (\vec{x} - \vec{\mathrm{p}}) \tag{3}$$

where $\Omega \in \mathcal{S}_+^d$ refers to a positive semidefinite matrix encoding the relevance of each feature. This matrix is

treated as an additional parameter which is, together with the prototypes' positions, chosen such that the cost function Eq. (2) is minimized – usually, a gradient based method like LBFGS is used. Due to the parameterization of the metric, this LVQ variant is often named generalized matrix LVQ (GMLVQ). Further details can be found in (Schneider et al., 2009).

2.2 Reject Options

LVQ schemes provide a classification rule which assigns a label to every possible input no matter how reliable & reasonable such classifications might be. Reject options allow the classifier to reject a sample if a certain prediction is not possible – i.e. the sample is too close to the decision boundary, or it is very different from the observed training data and therefore a classification based on the learned prototypes would not be reasonable. In order to realize such a reject option, we extend the set of possible predictions by a new class which represents a reject – i.e. for a classifier $h : \mathcal{X} \to \mathcal{Y}$, we construct $h : \mathcal{X} \to \mathcal{Y} \cup \{R\}$. Several methods for doing so have been proposed and evaluated in (Fischer et al., 2015a). Those methods use a function $r : \mathcal{X} \to \mathbb{R}$ for computing the certainty of a prediction by the classifier (Chow, 1970). If the certainty is below some threshold θ, the sample is rejected, more formally if $r(\vec{x}) < \theta$ then $h(\vec{x}) = R$. As demonstrated in the work (Chow, 1970), this strategy is optimum if the certainty reliably estimates the class posterior probability. In the following, we consider three popular realizations of the certainty function $r(\cdot)$ for LVQ models.

Relative Similarity. In (Fischer et al., 2015a), a very natural realization of $r(\cdot)$ called relative similarity (RelSim) yields excellent results for LVQ models:

$$r_{\text{RelSim}}(\vec{x}) = \frac{\mathrm{d}(\vec{x},\vec{\mathrm{p}}_-) - \mathrm{d}(\vec{x},\vec{\mathrm{p}}_+)}{\mathrm{d}(\vec{x},\vec{\mathrm{p}}_-) + \mathrm{d}(\vec{x},\vec{\mathrm{p}}_+)} \tag{4}$$

where $\vec{\mathrm{p}}_+$ denotes the closest prototype and $\vec{\mathrm{p}}_-$ denotes the closest prototype belonging to a different class than $\vec{\mathrm{p}}_+$. Note that this fraction is always non-negative and smaller than 1. Obviously, it is 0 if the distance between the sample and the closest prototype $\vec{\mathrm{p}}_+$ equals the distance to $\vec{\mathrm{p}}_-$. At the same time, RelSim gets close to 0 if the sample is far away – i.e. both distances are large (Brinkrolf and Hammer, 2018).

Decision Boundary Distance. Another, similar, realization of a certainty measure is the distance to the decision boundary (Dist). In case of LVQ, this can be formalized as follows (Fischer et al., 2015a):

$$r_{\text{Dist}}(\vec{x}) = \frac{|\,\mathrm{d}(\vec{x},\vec{\mathrm{p}}_+) - \mathrm{d}(\vec{x},\vec{\mathrm{p}}_-)|}{2\|\vec{\mathrm{p}}_+ - \vec{\mathrm{p}}_-\|_2^2} \tag{5}$$

where $\vec{\mathrm{p}}_+$ and $\vec{\mathrm{p}}_-$ are defined in the same way as in RelSim Eq. (4). Note that Eq. (5) is not normalized and depends on the prototypes and their distances to the sample $\vec{x}$. It is worthwhile mentioning that Eq. (5) is closely related to the reject option of an SVM (Platt, 1999) – in case of binary classification problem and a single prototype per class in a LVQ model, both models determine a separating hyperplane.

Probabilistic Certainty. The third certainty measure is a probabilistic one. The idea is to obtain proper class probabilities $p(y \mid \vec{x})$ for each class $y \in \mathcal{Y}$ and reject a sample $\vec{x}$ if the probability for the most probable class is lower than a given threshold θ. We denote the probabilistic certainty measure as follows:

$$r_{\text{Proba}}(\vec{x}) = \max_{y \in \mathcal{Y}} p(y \mid \vec{x}). \tag{6}$$

In the following, we use the fastest method from (Brinkrolf and Hammer, 2020b) for computing class probabilities given a trained LVQ models. The method (Price et al., 1994) combines estimates from binary classifiers in order to obtain class probabilities for the final prediction. First, to obtain estimates of class-wise binary classifiers, we train a single LVQ model for each pair of classes using only the samples belonging to those two classes. This yields a set of $|\mathcal{Y}|(|\mathcal{Y}|-1)/2$ binary classifiers. Next, a data-dependent scaling of the predictions follows to yield pairwise probabilities. Here, we use RelSim Eq. (4) and fit a sigmoid function to the real-valued scores. This mimics the approach by Platt (Platt, 1999) which is very popular in the context of SVMs. This post-processing yields estimates $r_{i,j}(\vec{x})$ of pairwise probabilities for every sample $\vec{x}$ and pairs of classes i and j: $r_{i,j} \equiv p(y = i \mid y = i \vee j, \vec{x})$. Given those pairwise probabilities and assuming a symmetric completion of the pairs $r_{i,j} + r_{j,i} = 1$, the posterior probabilities are obtained as follows:

$$p(y = i \mid \vec{x}) = \frac{1}{\sum_{j \neq i} \frac{1}{r_{i,j}} - (|\mathcal{Y}| - 2)} \tag{7}$$

where $i, j \in \mathcal{Y}$ (Price et al., 1994). After computing all probabilities, a normalization step is required such that $\sum_{i=1}^{k} p(y = i \mid \vec{x}) = 1$. We refer to (Brinkrolf and Hammer, 2020b) for further details on all these steps.

2.3 Counterfactual Explanations

Counterfactual explanations (often just called *counterfactuals*) are a prominent instance of contrasting

explanations, which state a change to some features of a given input such that the resulting data point, called the counterfactual, causes a different behavior of the system than the original input does. Thus, one can think of a counterfactual explanation as a suggestion of actions that change the model's behavior/prediction. One reason why counterfactual explanations are so popular is that there exists evidence that explanations used by humans are often contrasting in nature (Byrne, 2019) – i.e. people often ask questions like *"What would have to be different in order to observe a different outcome?"*. For illustrative purposes, consider the example of loan application: *imagine you applied for a credit at a bank, but your application is rejected. Now, you would like to know why and what you could have done to get accepted. A possible counterfactual explanation might be that you would have been accepted if you had earned 500$ more per month and if you had not had a second credit card.* Despite their popularity, the missing uniqueness of counterfactuals could pose a problem: often there exist more than one possible & valid counterfactual – this is called the Rashomon effect (Molnar, 2019) – and in such cases, it is not clear which or how many of them should be presented to the user. One common modeling approach is to enforce uniqueness by a suitable formalization or regularization.

In order to keep the explanation (suggested changes) simple – i.e. easy to understand – an obvious strategy is to look for a small number of changes so that the resulting sample (counterfactual) is similar/close to the original sample, which is aimed to be captured by Definition 1.

Definition 1 ((Closest) Counterfactual Explanation (Wachter et al., 2017))**.** *Assume a prediction function $h : \mathbb{R}^d \to \mathcal{Y}$ is given. Computing a counterfactual $\vec{x}_{cf} \in \mathbb{R}^d$ for a given input $\vec{x}_{orig} \in \mathbb{R}^d$ is phrased as an optimization problem:*

$$\underset{\vec{x}_{cf} \in \mathbb{R}^d}{\arg\min}\ \ell\left(h(\vec{x}_{cf}), y'\right) + C \cdot \phi(\vec{x}_{cf}, \vec{x}_{orig}) \tag{8}$$

where $\ell(\cdot)$ denotes a loss function, y' the target prediction, $\phi(\cdot)$ a penalty for dissimilarity of $\vec{x}_{cf}$ and $\vec{x}_{orig}$, and $C > 0$ denotes the regularization strength.

The counterfactuals from Definition 1 are also called *closest counterfactuals* because the optimization problem Eq. (8) tries to find an explanation $\vec{x}_{\text{cf}}$ that is as close as possible to the original sample $\vec{x}_{\text{orig}}$. However, other aspects like plausibility and actionability are ignored in Definition 1, but are covered in other work (Looveren and Klaise, 2021; Artelt and Hammer, 2020; Artelt and Hammer, 2021).

3 COUNTERFACTUAL EXPLANATIONS OF REJECT

In this section, we elaborate our proposal of using counterfactual explanations for explaining LVQ reject options. First, we introduce the general modeling in Section 3.1, and then study the computational aspects of each reject option in Section 3.2.

3.1 Modeling

Because counterfactual explanations (see Section 2.3) proved to be an effective and useful explanation, we propose to use counterfactual explanations for explaining reject options of LVQ models (see Section 2.1). Therefore, a counterfactual of a reject provides the user with actionable feedback of what to change in order to be able to classify. Furthermore, such an explanation also communicates why the model is too uncertain for making a prediction in this particular case.

Since there exist evidence that people prefer low complexity (i.e. "simple") explanations, we are looking for sparse counterfactuals. Similar to Definition 1, we phrase a counterfactual explanation $\vec{x}_{\text{cf}}$ of a given input $\vec{x}_{\text{orig}}$ as the following optimization problem:

$$\min_{\vec{x}_{\text{cf}} \in \mathbb{R}^d} \|\vec{x}_{\text{orig}} - \vec{x}_{\text{cf}}\|_1 \quad \text{s.t. } r(\vec{x}_{\text{cf}}) \geq \theta \tag{9}$$

where $r(\cdot)$ denotes the specific reject option and the l_1-norm objective is supposed to yield a sparse and hence a "low complexity explanation".

3.2 Computational Aspects of LVQ Reject Options

In the following, we propose modeling and algorithms which phrase problem (9) as convex optimizations problems, for efficiently computing counterfactual explanations of LVQ rejects – whereby we consider each of the three reject options from Section 2.2 separately. For the purpose of readability, we moved all proofs and derivations to Appendix 5.

3.2.1 Relative Similarity

In case of the relative similarity reject option Eq. (4), the optimization problem Eq. (9) can be solved by using a divide & conquer approach, where we need to solve a number of convex quadratic programs of the following form:

$$\begin{aligned} &\min_{\vec{x}_{\text{cf}} \in \mathbb{R}^d} \|\vec{x}_{\text{orig}} - \vec{x}_{\text{cf}}\|_1 \\ &\text{s.t. } \vec{x}_{\text{cf}}^\top \Omega \vec{x}_{\text{cf}} + \vec{q}_j^\top \vec{x}_{\text{cf}} + c_j \leq 0 \quad \forall \vec{p}_j \in \mathcal{P}(o_+) \end{aligned} \tag{10}$$

where $\mathcal{P}(o_+)$ denotes a set of prototypes, $\vec{q}_j$ and c_j are defined in Eq. (20) and Eq. (21) (see Appendix 5 for details). Note that convex quadratic programs can be solved efficiently (Boyd and Vandenberghe, 2014).

The complete algorithm for computing a counterfactual explanation is given in Algorithm 1.

Algorithm 1: Counterfactual under RelSim/DistanceToDecisionBoundary reject option.

Input: Original input $\vec{x}_{\text{orig}}$, reject threshold θ, the LVQ model
Output: Counterfactual $\vec{x}_{\text{cf}}$

1: $\vec{x}_{\text{cf}} = \vec{0}$ ▷ Initialize dummy solution
2: $z = \infty$ ▷ Sparsity of the best solution so far
3: **for** $\vec{p}_+ \in \mathcal{P}$ **do** ▷ Loop over every possible prototype
4: Solving Eq. (10)/Eq. (19) yields a counterfactual $\vec{x}_{\text{cf}*}$
5: **if** $\|\vec{x}_{\text{orig}} - \vec{x}_{\text{cf}*}\|_1 < z$ **then** ▷ Keep this counterfactual if it is sparser than the currently "best" counterfactual
6: $z = \|\vec{x}_{\text{orig}} - \vec{x}_{\text{cf}*}\|_1$
7: $\vec{x}_{\text{cf}} = \vec{x}_{\text{cf}*}$
8: **end if**
9: **end for**

3.2.2 Distance to Decision Boundary

Similar to the relative similarity reject option, we again use a divide & conquer approach for solving Eq. (9). But in contrast to the relative similarity reject option, we have to solve a number of linear programs only, which can be solved even faster than convex quadratic programs (Boyd and Vandenberghe, 2014):

$$\min_{\vec{x}_{\text{cf}} \in \mathbb{R}^d} \|\vec{x}_{\text{orig}} - \vec{x}_{\text{cf}}\|_1 \quad \text{s.t. } \vec{q}_j^\top \vec{x}_{\text{cf}} + c_j \geq 0 \quad \forall \vec{p}_j \in \mathcal{P}(o_+) \tag{11}$$

where $\mathcal{P}(o_+)$ denotes a set of prototypes, $\vec{q}_j$ and c_j are defined in Eq. (27) (see Appendix 5 for details).

The final algorithm for computing a counterfactual explanation is equivalent to Algorithm 1 for the relative similarity reject option, except that instead of solving Eq. (10) in line 4, we have to solve Eq. (19).

3.2.3 Probabilistic Certainty Measure

In case of the probabilistic certainty measure as a reject option Eq. (6), it holds:

$$\begin{aligned} r_{\text{Proba}}(\vec{x}) \geq \theta \quad &\Leftrightarrow \quad \max_{y \in \mathcal{Y}} p(y \mid \vec{x}) \geq \theta \\ &\Leftrightarrow \exists i \in \mathcal{Y} : \; p(y = i \mid \vec{x}) \geq \theta \end{aligned} \tag{12}$$

Applying the divide & conquer paradigm over $i \in \mathcal{Y}$ for solving Eq. (9), yields optimization problems of the following form:

$$\begin{aligned} &\min_{\vec{x}_{\text{cf}} \in \mathbb{R}^d} \|\vec{x}_{\text{orig}} - \vec{x}_{\text{cf}}\|_1 \quad \text{s.t.} \\ &\sum_{j \neq i} \exp\left(\alpha \frac{\mathrm{d}_{i,j}(\vec{x}_{\text{cf}}, \vec{p}_-) - \mathrm{d}_{i,j}(\vec{x}_{\text{cf}}, \vec{p}_+)}{\mathrm{d}_{i,j}(\vec{x}_{\text{cf}}, \vec{p}_-) + \mathrm{d}_{i,j}(\vec{x}_{\text{cf}}, \vec{p}_+)} + \beta\right) + \\ &\quad 1 - \frac{1}{\theta} \leq 0 \end{aligned} \tag{13}$$

While we could solve Eq. (13) directly – yielding Algorithm 2 –, it is a rather difficult optimization problem because of its lack of any structure like convexity – e.g. general (black-box) solvers might be the only applicable choice. We therefore, additionally, propose a convex approximation where we can still guarantee feasibility (i.e. validity of the final counterfactual) at the price of losing closeness – i.e. we might not find the sparsest possible counterfactual, although finding a global optimum of Eq. (13) might be difficult as well. Approximating the constraint in Eq. (13)

Algorithm 2: Counterfactual under the probabilistic certainty reject option.

Input: Original input $\vec{x}_{\text{orig}}$, reject threshold θ, the LVQ model
Output: Counterfactual $\vec{x}_{\text{cf}}$

1: $\vec{x}_{\text{cf}} = \vec{0}$ ▷ Initialize dummy solution
2: $z = \infty$ ▷ Sparsity of the best solution so far
3: **for** $i \in \mathcal{Y}$ **do** ▷ Loop over every possible class
4: Solving Eq. (13) yields a counterfactual $\vec{x}_{\text{cf}*}$
5: **if** $\|\vec{x}_{\text{orig}} - \vec{x}_{\text{cf}*}\|_1 < z$ **then** ▷ Keep this counterfactual if it is sparser than the currently "best" counterfactual
6: $z = \|\vec{x}_{\text{orig}} - \vec{x}_{\text{cf}*}\|_1$
7: $\vec{x}_{\text{cf}} = \vec{x}_{\text{cf}*}$
8: **end if**
9: **end for**

yields a convex quadratic constraint, which then results in a convex quadratic program as a final approximation of Eq. (13) – for details see Appendix 5:

$$\begin{aligned} &\min_{\vec{x}_{\text{cf}} \in \mathbb{R}^d} \|\vec{x}_{\text{orig}} - \vec{x}_{\text{cf}}\|_1 \\ &\text{s.t. } \vec{x}^\top \Omega \vec{x} + \vec{q}_j^\top \vec{x} + c_j \leq 0 \quad \forall \vec{p}_j \in \mathcal{P}(o_i) \end{aligned} \tag{14}$$

Using the approximation Eq. (14) requires us to iterate over every possible prototype, every possible class different from the i-th class, and finally over every possible class – i.e. finally yielding Algorithm 3. Although our proposed approximation Eq. (14) can be computed quite fast (because it is a convex quadratic program), it comes at the price of "drastically" increasing the complexity of the final divide & conquer algorithm. While we have to solve $|\mathcal{Y}|$ optimization problems Eq. (13) in Algorithm 2, we get a quadratic

Algorithm 3: Counterfactual under the probabilistic certainty reject option – Approximation.

Input: Original input $\vec{x}_{\text{orig}}$, reject threshold θ, the LVQ model
Output: Counterfactual $\vec{x}_{\text{cf}}$

$\vec{x}_{\text{cf}} = \vec{0}$ ▷ Initialize dummy solution
$z = \infty$ ▷ Sparsity of the best solution so far
for $i \in \mathcal{Y}$ **do** ▷ Loop over every possible class
  for $j \in \mathcal{Y} \setminus \{i\}$ **do** ▷ Loop over all other classes
    for $\vec{p}_i \in \mathcal{P}$ **do** ▷ Loop over every possible prototype
      Solving Eq. (14) yields a counterfactual $\vec{x}_{\text{cf}*}$
      if $\|\vec{x}_{\text{orig}} - \vec{x}_{\text{cf}*}\|_1 < z$ **then** ▷ Keep this counterfactual if it is sparser than the currently "best" counterfactual
        $z = \|\vec{x}_{\text{orig}} - \vec{x}_{\text{cf}*}\|_1$
        $\vec{x}_{\text{cf}} = \vec{x}_{\text{cf}*}$
      end if
    end for
  end for
end for

complexity (quadratic in the number of classes) for our proposed convex approximation (Algorithm 3):

$$|\mathcal{Y}|^2 \cdot P_N - |\mathcal{Y}| \cdot P_N \tag{15}$$

where P_N denotes the number of prototypes per class used in the pair-wise classifiers. This quadratic complexity could become a problem in case of a large number of classes and a large number of prototypes per class.

To summarize, we propose two divide & conquer algorithms for computing counterfactual explanations of the probabilistic certainty reject option Eq. (6). In Algorithm 2, we have to solve a rather complicated (i.e. unstructured) optimization problem, but we have to do this only a few times, whereas in Algorithm 3 we have to solve many convex quadratic programs. Although we have to solve more optimization problems in Algorithm 3 than in Algorithm 2, solving the optimization problems in Algorithm 3 is much easier and it is also possible to easily extend the optimization problems with additional constraints, such as plausibility constraints (Artelt and Hammer, 2020). Hence both algorithms have their areas of application and it is worthwhile to enable practitioners to choose between them depending on their needs and the specific scenario.

4 EXPERIMENTS

We empirically evaluate all proposed algorithms for computing counterfactual explanations of different reject options (see Sections 2.1, 2.2) for GMLVQ on several different data sets. We evaluate two different properties:

- Algorithmic properties like sparsity, validity (in case of black-box solvers).
- Goodness of the explanations – i.e. whether our proposed counterfactual explanations of reject are able to find and explain known ground truth reasons for rejects.

All experiments are implemented in Python and the source code is available on GitHub[1].

4.1 Data Sets

We consider the following data sets for our empirical evaluation:

4.1.1 Wine

The "Wine data set" (S. Aeberhard and de Vel, 1992) is used for predicting the cultivator of given wine samples based on their chemical properties. The data set contains 178 samples and 13 numerical features such as alcohol, hue and color intensity.

4.1.2 Breast Cancer

The "Breast Cancer Wisconsin (Diagnostic) Data Set" (William H. Wolberg, 1995) is used for classifying breast cancer samples into benign and malignant. The data set contains 569 samples and 30 numerical features such as area and smoothness.

4.1.3 Flip

This data set (Sowa et al., 2013) is used for the prediction of fibrosis. The set consists of samples of 118 patients and 12 numerical features such as blood glucose, BMI and total cholesterol. As the data set contains some rows with missing values, we chose to replace these missing values with the corresponding feature mean.

4.1.4 T21

This data set (Nicolaides et al., 2005) is used for early diagnosis of chromosomal abnormalities, such as trisomy 21, in pregnant women. The data set consists of 18 numerical features such as heart rate and weight,

[1] https://github.com/andreArtelt/explaining_lvq_reject

Table 1: Algorithmic properties – Mean sparsity (incl. variance) of different counterfactuals, smaller values are better.

	DataSet	BbCfFeasibility	BbCf	TrainCf	ClosestCf
Relative Similarity Eq. (4)	Wine	1.0 ± 0.0	11.14 ± 1.41	13.0 ± 0.0	5.68 ± 11.36
	Breast Cancer	0.99 ± 0.0	27.64 ± 4.38	30.0 ± 0.0	12.69 ± 52.31
	t21	0.99 ± 0.0	16.14 ± 1.91	11.0 ± 1.0	4.39 ± 12.79
	Flip	1.0 ± 0.0	10.52 ± 1.38	12.0 ± 0.0	4.07 ± 8.02
Distance DecisionBoundary Eq. (5)	Wine	1.0 ± 0.0	10.98 ± 1.12	13.0 ± 0.0	2.8 ± 9.13
	Breast Cancer	1.0 ± 0.0	27.5 ± 2.78	30.0 ± 0.0	4.16 ± 48.19
	t21	1.0 ± 0.0	16.02 ± 1.43	11.0 ± 1.0	2.12 ± 5.19
	Flip	1.0 ± 0.0	10.31 ± 1.2	12.0 ± 0.0	1.88 ± 3.11
Probabilistic Certainty Eq. (6)	Wine	0.45 ± 0.21	13.0 ± 0.0	13.0 ± 0.0	12.67 ± 0.22
	Breast Cancer	0.94 ± 0.02	30.0 ± 0.0	30.0 ± 0.0	26.04 ± 60.5
	t21	0.4 ± 0.24	17.39 ± 0.64	11.0 ± 0.0	12.55 ± 7.41
	Flip	0.4 ± 0.24	11.75 ± 0.23	12.0 ± 0.0	11.5 ± 1.47

Table 2: Goodness of counterfactual explanations – Mean and variance recall of identified relevant features (larger numbers are better).

	DataSet	BbCfFeasiblility	BbCf	TrainCf	Cf
Relative Similarity Eq. (4)	Wine	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0
	Breast Cancer	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	0.99 ± 0.01
	t21	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	0.98 ± 0.0
	Flip	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	0.96 ± 0.02
Distance DecisionBoundary Eq. (5)	Wine	0.8 ± 0.16	1.0 ± 0.0	1.0 ± 0.0	0.72 ± 0.15
	Breast Cancer	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	0.45 ± 0.18
	t21	0.6 ± 0.24	1.0 ± 0.0	1.0 ± 0.0	0.63 ± 0.15
	Flip	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	0.62 ± 0.22
Probabilistic Certainty Eq. (6)	Wine	0.8 ± 0.16	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0
	Breast Cancer	0.8 ± 0.16	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0
	t21	0.85 ± 0.02	1.0 ± 0.0	1.0 ± 0.0	0.94 ± 0.01
	Flip	0.73 ± 0.06	1.0 ± 0.0	1.0 ± 0.0	0.75 ± 0.19

and contains over 50000 samples but only 0.8 percent abnormal samples (e.g. cases of trisomy 21).

4.2 Setup

For finding the best model parameters, we perform a grid search on the GMLVQ hyperparameters and rejection thresholds. In order to reduce the risk of overfitting, we cross validate the GMLVQ models' accuracy and rejection rates on the different reject thresholds using a 5-fold cross validation.

For each GMLVQ model hyperparameterization, the rejection rates for each of the thresholds is computed as well as the impact of this on the accuracy of the model. Based on these rejection rates and accuracies, the accuracy-rejection curve (ARC) (Nadeem et al., 2010) can be computed. The area under the ARC measure (AU-ARC) gives an indication of how well the reject model performs given the GMLVQ model type and its hyperparameterization. For each combination of data set, GMLVQ model type, and reject option method, we can then determine the best GMLVQ model parameters and rejection threshold. We do this by selecting the GMLVQ hyperparameters with the highest accuracy.

Following the selection of the best GMLVQ hyperparameters, the best rejection threshold is found by determining the "optimal" threshold in the ARC by finding the so-called knee-point using the Kneedle algorithm (Satopaa et al., 2011). By finding the optimal hyperparameters for each model, our evaluation of the different model types will be less dependent on potentially poor hyperparamerization.

We run all experiments (5-fold cross validation) for each data set, each reject option and each method for computing the counterfactuals – i.e black-box solver for solving Eq. (9)[2], our proposed algorithms from Section 3) and the closest sample from the training data set which is not rejected (this "naive" way of computing a counterfactual serves as a baseline).

4.2.1 Algorithmic Properties

We evaluate the sparsity of the counterfactual explanations by using the l_0-norm[3]. Since our methods

[2]We use the penalty method (with equal weighting) to turn Eq. (9) into an unconstrained problem, solve it by using the Nelder-Mead black-box method.

[3]For numerical reasons, we use a threshold at which we consider a floating point number to be equal to zero – see provided source code for details.

and algorithms are guaranteed to output feasible (i.e. valid) counterfactuals, we only evaluate validity (reported as percentage) of the counterfactuals computed by the black-box solver.

4.2.2 Goodness of Counterfactual Explanations

For evaluating the ground truth recovery rate (goodness) of the counterfactuals, we create scenarios with ground truth as follows: for each data set, we select a random subset of features (30%) and perturb these in the test set by adding Gaussian noise – we then check which of these samples are rejected due to the noise (i.e. applying the reject option before and after applying the perturbation), and compute counterfactual explanations of these samples only. We then evaluate for each counterfactual how many of the relevant features (from the known ground truth) are detected and included in the counterfactual explanation.

4.3 Results & Discussion

When reporting the results, we use the following abbreviations: *BbCfFeasibility* – Feasibility of the counterfactuals computed by the black-box solver, in case of the probabilistic reject option Eq. (6), we report the results of using Algorithm 2 (the results for the "true" black-box solver can be found in Appendix 5); *BbCf* – Counterfactuals computed by the black-box solver, *TrainCf* – Counterfactuals by selecting the closest sample from the training set which is not rejected; *ClosestCf* – Counterfactuals computed by our proposed algorithms.

Note that we round all values to two decimal points.

4.3.1 Algorithmic Properties

In Table 1, we report the mean sparsity (along with the variance) of the counterfactuals for different reject options and different data sets. We observe that our proposed methods for computing counterfactual explanation of reject options is consistently able to compute very sparse (i.e. low complexity) counterfactuals – however, the variance is often quite large which suggests that there exist a few outliers in the data set for which it is not possible to compute a sparse counterfactuals. As it is to be expected, we observe the worst performance when choosing a sample from the training set as a counterfactual. The counterfactuals computed by a black-box solver are often a bit better than those from the training set but still far away from the counterfactuals computed by our proposed algorithms. While the black-box solver works quite well in case of the relative similarity and distance to decision boundary reject options, the performance drops significantly (for many but not all data sets) in case of the probabilistic certainty reject option. We think this might be due to the increased complexity of the reject option, compared to the other two reject options which have much simpler mathematical form. For this reject option, our proposed algorithm is still able to consistently yield the sparsest counterfactuals but the difference to other counterfactuals is not that significant like it is the case for the other two reject options.

4.3.2 Goodness of Counterfactual Explanations

The mean recall (along with the variance) of recovered (ground truth) relevant features for different counterfactuals, data sets and reject options, is given in Table 2. We observe that all methods are able to identify the relevant features that caused the reject. There exist few instances where counterfactuals computed by our proposed algorithms miss a few relevant features – this is most likely due to the fact that feature correlations might exist and the objective to find a sparse counterfactual can yield to different resolutions of such redundancies. Here, other regularization terms (such as L2) might yield better stability possibly at the cost of decreased interpretability.

5 SUMMARY & CONCLUSION

In this work we proposed to explain reject options of LVQ models by means of counterfactual explanations. We considered three popular reject options and for each, proposed (extendable) modelings and algorithms for efficiently computing counterfactual explanations under the particular reject option. We empirically evaluated all our proposed methods under different aspects – in particular, we demonstrated that our algorithms delivers sparse (i.e. "low-complexity") explanations, and that counterfactual explanations in general seem to be able to detect and highlight relevant features in scenarios where the ground truth is known.

Although our proposed idea of using counterfactual explanations for explaining rejects is rather general, our proposed methods and algorithms are tailored towards LVQ models and thus not applicable to other ML models. Therefore, it would be of interest to see other, either model specific or even more general (e.g. model agnostic) methods for computing counterfactual explanations of reject under other ML models.

Our evaluation focused on algorithmic properties such as sparsity and feature relevances for assessing the ground truth recovery rate (goodness) of the computed counterfactuals. However, it is still unclear how and if these kinds of explanations of reject are useful and helpful to humans – since it is difficult to implement "human usefulness" as a scoring function, a proper user study for evaluating the usefulness is necessary.

We leave these aspects as future work.

ACKNOWLEDGEMENT

We gratefully acknowledge funding from the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) for grant TRR 318/1 2021 – 438445824, from the BMWi for grant 01MK20007E, and the VW-Foundation for the project *IMPACT* funded in the frame of the funding line *AI and its Implications for Future Society*.

REFERENCES

Aamodt, A. and Plaza., E. (1994). Case-based reasoning: Foundational issues, methodological variations, and systemapproaches. *AI communications*.

Artelt, A. and Hammer, B. (2020). Convex density constraints for computing plausible counterfactual explanations. In Farkas, I., Masulli, P., and Wermter, S., editors, *Artificial Neural Networks and Machine Learning - ICANN 2020 - 29th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 15-18, 2020, Proceedings, Part I*, volume 12396 of *Lecture Notes in Computer Science*, pages 353–365. Springer.

Artelt, A. and Hammer, B. (2021). Convex optimization for actionable \& plausible counterfactual explanations. *CoRR*, abs/2105.07630.

Boyd, S. P. and Vandenberghe, L. (2014). *Convex Optimization*. Cambridge University Press.

Brinkrolf, J. and Hammer, B. (2018). Interpretable machine learning with reject option. *Autom.*, 66(4):283–290.

Brinkrolf, J. and Hammer, B. (2020a). Time integration and reject options for probabilistic output of pairwise LVQ. *Neural Comput. Appl.*, 32(24):18009–18022.

Brinkrolf, J. and Hammer, B. (2020b). Time integration and reject options for probabilistic output of pairwise LVQ. *Neural Comput. Appl.*, 32(24):18009–18022.

Brinkrolf, J. and Hammer, B. (2021). Federated learning vector quantization. In *29th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2021, Online event (Bruges, Belgium), October 6-8, 2021*.

Byrne, R. M. J. (2019). Counterfactuals in explainable artificial intelligence (XAI): evidence from human reasoning. In Kraus, S., editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 6276–6282. ijcai.org.

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

Doshi-Velez, F. and Kim, B. (2017). Towards a rigorous science of interpretable machine learning.

European parliament and council (2016). General data protection regulation: Regulation (eu) 2016/679 of the european parliament.

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

Fischer, L., Hammer, B., and Wersing, H. (2015b). Optimum reject options for prototype-based classification. *CoRR*, abs/1503.06549.

Fisher, A., Rudin, C., and Dominici, F. (2019). All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. *J. Mach. Learn. Res.*, 20:177:1–177:81.

Gepperth, A. and Hammer, B. (2016). Incremental learning algorithms and applications. In *24th European Symposium on Artificial Neural Networks, ESANN 2016, Bruges, Belgium, April 27-29, 2016*.

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., and Pedreschi, D. (2019). A survey of methods for explaining black box models. *ACM Comput. Surv.*, 51(5):93:1–93:42.

Herbei, R. and Wegkamp, M. H. (2006). Classification with reject option. *Canadian Journal of Statistics*, 34(4):709–721.

Khandani, A. E., Kim, A. J., and Lo, A. (2010). Consumer credit-risk models via machine-learning algorithms. *Journal of Banking & Finance*, 34(11).

Kim, B., Koyejo, O., and Khanna, R. (2016). Examples are not enough, learn to criticize! criticism for interpretability. In Lee, D. D., Sugiyama, M., von Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2280–2288.

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 : the official journal of the International Neural Network Society*, 28:90–105.

Looveren, A. V. and Klaise, J. (2021). Interpretable counterfactual explanations guided by prototypes. 12976:650–665.

Losing, V., Hammer, B., and Wersing, H. (2018). Incremental on-line learning: A review and comparison of state of the art algorithms. *Neurocomputing*, 275:1261–1274.

Molnar, C. (2019). *Interpretable Machine Learning*.

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.

Nicolaides, K. H., Spencer, K., Avgidou, K., Faiola, S., and Falcon, O. (2005). Multicenter study of first-trimester screening for trisomy 21 in 75 821 pregnancies: results and estimation of the potential impact of individual risk-orientated two-stage first-trimester screening. *Ultrasound in Obstetrics & Gynecology*, 25(3):221–226.

Nova, D. and Estévez, P. A. (2014). A review of learning vector quantization classifiers. *Neural Comput. Appl.*, 25(3-4):511–524.

Offert, F. (2017). "i know it when i see it". visualization and intuitive interpretability.

Owomugisha, G., Nuwamanya, E., Quinn, J. A., Biehl, M., and Mwebaze, E. (2020). Early detection of plant diseases using spectral data. In Petkov, N., Strisciuglio, N., and Travieso-González, C. M., editors, *APPIS 2020: 3rd International Conference on Applications of Intelligent Systems, APPIS 2020, Las Palmas de Gran Canaria Spain, 7-9 January 2020*. ACM.

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.

Price, D., Knerr, S., Personnaz, L., and Dreyfus, G. (1994). Pairwise neural network classifiers with probabilistic outputs. 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 1109–1116. MIT Press.

S. Aeberhard, D. C. and de Vel, O. (1992). Comparison of classifiers in high dimensional settings. *Tech. Rep. no. 92-02*.

Sallab, A. E., Abdou, M., Perot, E., and Yogamani, S. (2017). Deep reinforcement learning framework for autonomous driving. *Electronic Imaging*, 2017(19):70–76.

Samek, W., Wiegand, T., and Müller, K. (2017). Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *CoRR*, abs/1708.08296.

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, USA, November 27-30, 1995*, pages 423–429. MIT Press.

Satopaa, V., Albrecht, J. R., Irwin, D. E., and Raghavan, B. (2011). Finding a "kneedle" in a haystack: Detecting knee points in system behavior. In *31st IEEE International Conference on Distributed Computing Systems Workshops (ICDCS 2011 Workshops), 20-24 June 2011, Minneapolis, Minnesota, USA*, pages 166–171. IEEE Computer Society.

Schneider, P., Biehl, M., and Hammer, B. (2009). Adaptive Relevance Matrices in Learning Vector Quantization. *Neural Computation*, 21(12):3532–3561.

Shafer, G. and Vovk, V. (2008). A tutorial on conformal prediction. *J. Mach. Learn. Res.*, 9:371–421.

Sowa, J.-P., Heider, D., Bechmann, L. P., Gerken, G., Hoffmann, D., and Canbay, A. (2013). Novel algorithm for non-invasive assessment of fibrosis in nafld. *PLOS ONE*, 8(4):1–6.

Stalidis, P., Semertzidis, T., and Daras, P. (2018). Examining deep learning architectures for crime classification and prediction. abs/1812.00602.

Tjoa, E. and Guan, C. (2019). A survey on explainable artificial intelligence (XAI): towards medical XAI. *CoRR*, abs/1907.07374.

Verma, S., Dickerson, J., and Hines, K. (2020). Counterfactual explanations for machine learning: A review.

Wachter, S., Mittelstadt, B., and Russell, C. (2017). Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31:841.

William H. Wolberg, W. Nick Street, O. L. M. (1995). Breast cancer wisconsin (diagnostic) data set. https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic).

Xu, Y., Furao, S., Hasegawa, O., and Zhao, J. (2009). An online incremental learning vector quantization. In *Advances in Knowledge Discovery and Data Mining, 13th Pacific-Asia Conference, PAKDD 2009, Bangkok, Thailand, April 27-30, 2009, Proceedings*, pages 1046–1053.

APPENDIX

Proofs & Derivations

Counterfactual Explanations

Relative Similarity. In order for a sample $\vec{x} \in \mathbb{R}^d$ to be classified, it must hold that:

$$r_{\text{RelSim}}(\vec{x}) \geq \theta \tag{16}$$

Using the shorter notation $\mathrm{d}^+(\vec{x}) = \mathrm{d}(\vec{x}, \vec{\mathrm{p}}_+)$ and $\mathrm{d}^-(\vec{x}) = \mathrm{d}(\vec{x}, \vec{\mathrm{p}}_-)$, respectively, where $\vec{\mathrm{p}}_+$ refers to the closest prototype and $\vec{\mathrm{p}}_-$ revers to the closest one belongs to a different class, this further translates into:

$$\begin{aligned} & r_{\text{RelSim}}(\vec{x}) \geq \theta \\ \Leftrightarrow\ & \frac{\mathrm{d}^-(\vec{x}) - \mathrm{d}^+(\vec{x})}{\mathrm{d}^-(\vec{x}) + \mathrm{d}^+(\vec{x})} \geq \theta \\ \Leftrightarrow\ & (1-\theta)\mathrm{d}^-(\vec{x}) - (1+\theta)\mathrm{d}^+(\vec{x}) \geq 0 \end{aligned} \tag{17}$$

Assuming that the closest prototype $\vec{\mathrm{p}}_+$ is fixed, we can rewrite Eq. (17) as follows:

$$(1-\theta)\mathrm{d}(\vec{x}, \vec{\mathrm{p}}_j) - (1+\theta)\mathrm{d}(\vec{x}, \vec{\mathrm{p}}_+) \geq 0 \quad \forall \vec{\mathrm{p}}_j \in \mathcal{P}(o_+) \tag{18}$$

where $\mathcal{P}(o_+)$ denotes the set of all prototypes that are not labeled as o_+. The intuition behind the translation

of Eq. (17) into Eq. (18) is that, if Eq. (17) holds for all possible $\vec{p}_-$, then also for the particular choice of $\vec{p}_-$ in Eq. (16).

These constraints can be rewritten as the following convex quadratic constraints – for a given $\vec{p}_j$:

$$\begin{aligned}
&(1-\theta)d(\vec{x},\vec{p}_j) - (1+\theta)d(\vec{x},\vec{p}_+) \geq 0 \\
&\Leftrightarrow d(\vec{x},\vec{p}_j) - d(\vec{x},\vec{p}_+) - \theta\Big(d(\vec{x},\vec{p}_j) + d(\vec{x},\vec{p}_+)\Big) \geq 0 \\
&\Leftrightarrow -2(\vec{p}_+^\top \Omega - \vec{p}_j^\top \Omega)\vec{x} - \vec{p}_j^\top \Omega \vec{p}_j + \vec{p}_+^\top \Omega \vec{p}_+ + \\
&\quad \theta\Big(2\vec{x}^\top \Omega \vec{x} - 2(\vec{p}_j^\top \Omega + \vec{p}_+^\top \Omega)\vec{x} + \vec{p}_j^\top \Omega \vec{p}_j + \vec{p}_+^\top \Omega \vec{p}_+\Big) \leq 0 \\
&\Leftrightarrow \vec{x}^\top \Omega \vec{x} + \vec{q}_j^\top \vec{x} + c_j \leq 0
\end{aligned} \tag{19}$$

where

$$\vec{q}_j^\top = \left(-\frac{1}{\theta} - 1\right)\vec{p}_+^\top \Omega + \left(\frac{1}{\theta} - 1\right)\vec{p}_j^\top \Omega \tag{20}$$

$$c_j = \frac{1}{2}\left(\left(1 - \frac{1}{\theta}\right)\vec{p}_j^\top \Omega \vec{p}_j + \left(1 + \frac{1}{\theta}\right)\vec{p}_+^\top \Omega \vec{p}_+\right) \tag{21}$$

We therefore get the following convex quadratic optimization problem – note that convex quadratic programs can be solved efficiently (Boyd and Vandenberghe, 2014):

$$\begin{aligned}
&\min_{\vec{x}_{\text{cf}} \in \mathbb{R}^d} \|\vec{x}_{\text{orig}} - \vec{x}_{\text{cf}}\|_1 \\
&\text{s.t. } \vec{x}_{\text{cf}}^\top \Omega \vec{x}_{\text{cf}} + \vec{q}_j^\top \vec{x}_{\text{cf}} + c_j \leq 0 \quad \forall \vec{p}_j \in \mathcal{P}(o_+)
\end{aligned} \tag{22}$$

We simply solve this optimization problem Eq. (22) for all possible target prototypes and select the counterfactual which is the closest to the original sample – since every possible prototype could be a potential closest prototype $\vec{p}_+$, we have to solve as many optimization problems as we have prototypes. However, one could introduce some kind of early stopping by adding an additional constraint on the distance of the counterfactual to the original sample – i.e. use the currently best known solution as a upper bound on the objective, which might result in an infeasible program and hence will be aborted quickly. Alternatively, one could solve the different optimization problems in parallel because they are independent.

Distance to Decision Boundary. In order for a sample $\vec{x} \in \mathbb{R}^d$ to be accepted (i.e. not being rejected), it must hold that:

$$r_{\text{Dist}}(\vec{x}) \geq \theta \tag{23}$$

This further translates into:

$$\begin{aligned}
&r_{\text{Dist}}(\vec{x}) \geq \theta \\
&\Leftrightarrow \frac{|d^+(\vec{x}) - d^-(\vec{x})|}{2\|\vec{p}_+ - \vec{p}_-\|_2^2} \geq \theta \\
&\Leftrightarrow d^-(\vec{x}) - d^+(\vec{x}) - 2\theta\|\vec{p}_+ - \vec{p}_-\|_2^2 \geq 0
\end{aligned} \tag{24}$$

Assuming that the closest prototype $\vec{p}_+$ is fixed, we get:

$$d(\vec{x},\vec{p}_j) - d(\vec{x},\vec{p}_+) - 2\theta\|\vec{p}_+ - \vec{p}_j\|_2^2 \geq 0 \quad \forall \vec{p}_j \in \mathcal{P}(o_+) \tag{25}$$

where $\mathcal{P}(o_+)$ denotes the set of prototypes that are not labeled as o_+. Again, the intuition behind this translation is that, if Eq. (24) holds for all possible $\vec{p}_-$, then also for the particular choice of $\vec{p}_-$ in Eq. (23).

These constraints can be rewritten as the following linear constraints – for a given $\vec{p}_j$:

$$\begin{aligned}
&d(\vec{x},\vec{p}_j) - d(\vec{x},\vec{p}_+) - 2\theta\|\vec{p}_+ - \vec{p}_j\|_2^2 \geq 0 \\
&\Leftrightarrow (\vec{x} - \vec{p}_j)^\top \Omega(\vec{x} - \vec{p}_j) - (\vec{x} - \vec{p}_+)^\top \Omega(\vec{x} - \vec{p}_+) - \\
&\quad 2\theta(\vec{p}_+ - \vec{p}_j)^\top \Omega(\vec{p}_+ - \vec{p}_j) \geq 0 \\
&\Leftrightarrow \Big(2\vec{p}_+^\top \Omega - 2\vec{p}_j^\top \Omega\Big)\vec{x} + \vec{p}_j^\top \Omega \vec{p}_j - \vec{p}_+^\top \Omega \vec{p}_+ - \\
&\quad 2\theta(\vec{p}_+ - \vec{p}_j)^\top \Omega(\vec{p}_+ - \vec{p}_j) \geq 0 \\
&\Leftrightarrow \vec{q}_j^\top \vec{x} + c_j \geq 0
\end{aligned} \tag{26}$$

where

$$\begin{aligned}
\vec{q}_j^\top &= 2\vec{p}_+^\top \Omega - 2\vec{p}_j^\top \Omega \\
c_j &= \vec{p}_j^\top \Omega \vec{p}_j - \vec{p}_+^\top \Omega \vec{p}_+ - 2\theta(\vec{p}_+ - \vec{p}_j)^\top \Omega(\vec{p}_+ - \vec{p}_j)
\end{aligned} \tag{27}$$

Finally, we get the following linear optimization problem – note that linear programs can be solved even faster than convex quadratic programs (Boyd and Vandenberghe, 2014):

$$\begin{aligned}
&\min_{\vec{x}_{\text{cf}} \in \mathbb{R}^d} \|\vec{x}_{\text{orig}} - \vec{x}_{\text{cf}}\|_1 \\
&\text{s.t. } \vec{q}_j^\top \vec{x}_{\text{cf}} + c_j \geq 0 \quad \forall \vec{p}_j \in \mathcal{P}(o_+)
\end{aligned} \tag{28}$$

Again, we try all possible target prototypes $\vec{p}_+$ and select the best counterfactual – everything from the relative similarity case applies (see Section 5).

Probabilistic Certainty Measure. In order for a sample $\vec{x} \in \mathbb{R}^d$ to be classified (i.e. not being rejected), it must hold that:

$$r_{\text{Proba}}(\vec{x}) \geq \theta \tag{29}$$

This further translates into:

$$\begin{aligned}
&r_{\text{Proba}}(\vec{x}) \geq \theta \quad \Leftrightarrow \quad \max_{y \in \mathcal{Y}} p(y \mid \vec{x}) \geq \theta \\
&\Leftrightarrow \exists i \in \mathcal{Y} : \; p(y = i \mid \vec{x}) \geq \theta
\end{aligned} \tag{30}$$

Table 3: Algorithmic properties – Mean sparsity (incl. variance) of different counterfactuals, smaller values are better, for feasibility, larger values are better for feasibility.

DataSet	Bb Eq. (9) Feasibility	Algo. 2 Feasibility	BlackBox Eq. (9)	Algo. 2	Algo. 3
Wine	0.45 ± 0.15	0.45 ± 0.21	11.1 ± 2.09	13.0 ± 0.0	12.67 ± 0.22
Breast Cancer	0.69 ± 0.12	0.49 ± 0.02	26.43 ± 3.45	30.0 ± 0.0	26.04 ± 60.5
t21	0.82 ± 0.03	0.4 ± 0.24	15.94 ± 2.34	17.39 ± 0.64	12.55 ± 7.41
Flip	0.5 ± 0.2	0.4 ± 0.24	9.5 ± 2.25	11.75 ± 0.23	11.5 ± 1.47

For the moment, we assume that i is fixed – i.e. using a divide & conquer approach. It follows that:

$$\begin{aligned} & p(y = i \mid \vec{x}) \geq \theta \\ \Leftrightarrow & \frac{1}{\sum_{j \neq i} \frac{1}{p(y=i \mid (i,j), \vec{x})} - (|\mathcal{Y}| - 2)} \geq \theta \\ \Leftrightarrow & 1 \geq \theta \left(\sum_{j \neq i} \frac{1}{p(y = i \mid (i,j), \vec{x})} - (|\mathcal{Y}| - 2) \right) \\ \Leftrightarrow & \sum_{j \neq i} \frac{1}{p(y = i \mid (i,j), \vec{x})} - c \leq 0 \end{aligned} \tag{31}$$

where

$$c = |\mathcal{Y}| - 2 + \frac{1}{\theta} \tag{32}$$

Further simplifications of Eq. (31) yield:

$$\begin{aligned} & \sum_{j \neq i} \frac{1}{p(y = i \mid (i,j), \vec{x})} - c \leq 0 \\ \Leftrightarrow & \sum_{j \neq i} 1 + \exp(\alpha \cdot r_{i,j}(\vec{x}) + \beta) - c \leq 0 \\ \Leftrightarrow & \sum_{j \neq i} \exp\left(\alpha \frac{\mathrm{d}^-_{i,j}(\vec{x}_{\mathrm{cf}}) - \mathrm{d}^+_{i,j}(\vec{x}_{\mathrm{cf}})}{\mathrm{d}^-_{i,j}(\vec{x}_{\mathrm{cf}}) + \mathrm{d}^+_{i,j}(\vec{x}_{\mathrm{cf}})} + \beta \right) + c' \leq 0 \end{aligned} \tag{33}$$

where

$$c' = |\mathcal{Y}| - 1 - c = 1 - \frac{1}{\theta} \tag{34}$$

We therefore get the following optimization problem:

$$\begin{aligned} & \min_{\vec{x}_{\mathrm{cf}} \in \mathbb{R}^d} \|\vec{x}_{\mathrm{orig}} - \vec{x}_{\mathrm{cf}}\|_1 \\ & \text{s.t.} \sum_{j \neq i} \exp\left(\alpha \frac{\mathrm{d}^-_{i,j}(\vec{x}) - \mathrm{d}^+_{i,j}(\vec{x})}{\mathrm{d}^-_{i,j}(\vec{x}) + \mathrm{d}^+_{i,j}(\vec{x})} + \beta \right) + c' \leq 0 \end{aligned} \tag{35}$$

Because of the divide & conquer paradigm, we would have to try all possible target classes $i \in \mathcal{Y}$ and finally select the one yielding the lowest objective – i.e. the closest counterfactual. While one could do this using constraint Eq. (33), the optimization would be rather complicated because the constraint is not convex and rather "ugly" – although it could be tackled by an evolutionary optimization method. However, the number of optimization problems we have to solve is rather small – it is equal to the number of classes.

We therefore, additionally, propose a surrogate constraint which captures the same "meaning/intuition" as Eq. (33) does, but is easier to optimize over – however, note that by using a surrogate instead of the original constraint Eq. (33) we give up closeness which, in our opinion, would be acceptable if the solutions stay somewhat close to each other[4]. We try out and compare both approaches in the experiments (see Section 4).

First, we apply the natural logarithm to Eq. (33) and then bound it by using the maximum:

$$\begin{aligned} & \log\left(\sum_{j \neq i} \exp\left(\alpha \frac{\mathrm{d}^-_{i,j}(\vec{x}) - \mathrm{d}^+_{i,j}(\vec{x})}{\mathrm{d}^-_{i,j}(\vec{x}) + \mathrm{d}^+_{i,j}(\vec{x})} + \beta \right) \right) \\ & \leq \max_{j \neq i} \left(\alpha \frac{\mathrm{d}^-_{i,j}(\vec{x}) - \mathrm{d}^+_{i,j}(\vec{x})}{\mathrm{d}^-_{i,j}(\vec{x}) + \mathrm{d}^+_{i,j}(\vec{x})} + \beta \right) + \log(|\mathcal{Y}| - 1) \end{aligned} \tag{36}$$

We therefore approximate the constraint Eq. (33) by using Eq. (36), which yields the following constraint:

$$\begin{aligned} \max_{j \neq i} \left(\alpha \frac{\mathrm{d}^-_{i,j}(\vec{x}) - \mathrm{d}^+_{i,j}(\vec{x})}{\mathrm{d}^-_{i,j}(\vec{x}) + \mathrm{d}^+_{i,j}(\vec{x})} + \beta \right) \leq & \log(-c') \\ & - \log(|\mathcal{Y}| - 1) \end{aligned} \tag{37}$$

Note that, in theory it could happen that $-c' \leq 0$ – we fix this by simply taking $\max(-c', \varepsilon)$, which results in a feasible solution but the approximation gets a bit worse.

Assuming that the maximum j is fixed, we get the following constraint:

$$\frac{\mathrm{d}^-_{i,j}(\vec{x}) - \mathrm{d}^+_{i,j}(\vec{x})}{\mathrm{d}^-_{i,j}(\vec{x}) + \mathrm{d}^+_{i,j}(\vec{x})} \leq \gamma \tag{38}$$

where

$$\gamma = \frac{\log(-c') - \log(|\mathcal{Y}| - 1) - \beta}{\alpha} \tag{39}$$

Further simplifications reveal that:

$$\begin{aligned} & \frac{\mathrm{d}^-_{i,j}(\vec{x}) - \mathrm{d}^+_{i,j}(\vec{x})}{\mathrm{d}^-_{i,j}(\vec{x}) + \mathrm{d}^+_{i,j}(\vec{x})} \leq \gamma \\ \Leftrightarrow & (1 - \gamma)\mathrm{d}^-_{i,j}(\vec{x}) - (1 + \gamma)\mathrm{d}^+_{i,j}(\vec{x}) \leq 0 \\ \Leftrightarrow & (\gamma - 1)\mathrm{d}^-_{i,j}(\vec{x}) + (1 + \gamma)\mathrm{d}^+_{i,j}(\vec{x}) \geq 0 \end{aligned} \tag{40}$$

[4]Furthermore, in case of additional plausibility & actionability constraints, closeness becomes even less important.

Next, we assume that the closest prototype with the correct label is fixed and denote it by $\vec{p}_i$ – we denote prototypes from the other class as $\vec{p}_j$. In the end, we iterate over all possible closest prototypes $\vec{p}_i$ and select the one that minimizes the objective (i.e. closeness to the original sample) – note that this approximation drastically increases the number of optimization problems that must be solved and thus the overall complexity of the final algorithm. We then can rewrite Eq. (40) as follows – we make sure that Eq. (40) is satisfied for every possible $\vec{p}_-$:

$$\begin{aligned}&(\gamma-1)\mathrm{d}^-_{i,j}(\vec{x})+(1+\gamma)\mathrm{d}^+_{i,j}(\vec{x})\geq 0\\ &\Leftrightarrow(\gamma-1)\mathrm{d}(\vec{x},\vec{p}_j)+(1+\gamma)\mathrm{d}(\vec{x},\vec{p}_i)\geq 0\quad\forall\vec{p}_j\in\mathcal{P}(o_i)\end{aligned}\tag{41}$$

Applying even more simplifications yield:

$$\begin{aligned}&(\gamma-1)\mathrm{d}(\vec{x},\vec{p}_j)+(1+\gamma)\mathrm{d}(\vec{x},\vec{p}_i)\geq 0\\ &\Leftrightarrow(\gamma-1)\left(\vec{x}^\top\Omega\vec{x}-2\vec{p}_j^\top\Omega\vec{x}+\vec{p}_j^\top\Omega\vec{p}_j\right)+\\ &\quad(1+\gamma)\left(\vec{x}^\top\Omega\vec{x}-2\vec{p}_i^\top\Omega\vec{x}+\vec{p}_i^\top\Omega\vec{p}_i\right)\geq 0\\ &\Leftrightarrow 2\gamma\vec{x}^\top\Omega\vec{x}-2(\gamma-1)\vec{p}_j^\top\Omega\vec{x}-2(1+\gamma)\vec{p}_i^\top\Omega\vec{x}+\\ &\quad(\gamma-1)\vec{p}_j^\top\Omega\vec{p}_j+(1+\gamma)\vec{p}_i^\top\Omega\vec{p}_i\geq 0\\ &\Leftrightarrow\vec{x}^\top\Omega\vec{x}+\vec{q}_j^\top\vec{x}+c_j\leq 0\end{aligned}\tag{42}$$

where

$$\begin{aligned}\vec{q}_j^\top&=\frac{1}{-2\gamma}\left(-2(\gamma-1)\vec{p}_j^\top\Omega-2(1+\gamma)\vec{p}_i^\top\Omega\right)\\ c_j&=\frac{1}{-2\gamma}\left((\gamma-1)\vec{p}_j^\top\Omega\vec{p}_j+(1+\gamma)\vec{p}_i^\top\Omega\vec{p}_i\right)\end{aligned}\tag{43}$$

Finally, we get the following convex quadratic optimization program:

$$\begin{aligned}&\min_{\vec{x}_{\mathrm{cf}}\in\mathbb{R}^d}\|\vec{x}_{\mathrm{orig}}-\vec{x}_{\mathrm{cf}}\|_1\\ &\text{s.t. }\vec{x}^\top\Omega\vec{x}+\vec{q}_j^\top\vec{x}+c_j\leq 0\quad\forall\vec{p}_j\in\mathcal{P}(o_i)\end{aligned}\tag{44}$$

Note that we have to solve Eq. (44) for every possible closest prototype, every possible class different from the i-th class and finally for every possible class. Thus, we get the following number of optimization problems (quadratic in the number of classes):

$$|\mathcal{Y}|\cdot(|\mathcal{Y}|-1)\cdot P_N=|\mathcal{Y}|^2\cdot P_N-|\mathcal{Y}|\cdot P_N\tag{45}$$

where P_N denotes the number of prototypes per class used in the pair-wise classifiers.

Note that this number is much larger than $|\mathcal{Y}|$ which we got without introducing any surrogate or approximation. However, in contrast to Eq. (35), the surrogate Eq. (44) is much easier to solve because it is a convex quadratic program which are known to be solved very fast (Boyd and Vandenberghe, 2014).

Additional Empirical Results

Probabilistic Reject Option

The results for the different methods for computing counterfactual explanations of the probabilistic reject option Eq. (6) are given in Table 3.