Efficient computation of contrastive explanations

1st André Artelt
CITEC - Cognitive Interaction Technology
Bielefeld University
Bielefeld, Germany
aartelt@techfak.uni-bielefeld.de

2nd Barbara Hammer
CITEC - Cognitive Interaction Technology
Bielefeld University
Bielefeld, Germany
bhammer@techfak.uni-bielefeld.de

***Abstract*—With the increasing deployment of machine learning systems in practice, transparency and explainability have become serious issues. Contrastive explanations are considered to be useful and intuitive, in particular when it comes to explaining decisions to lay people, since they mimic the way in which humans explain. Yet, so far, comparably little research has addressed computationally feasible technologies, which allow guarantees on uniqueness and optimality of the explanation and which enable an easy incorporation of additional constraints. Here, we will focus on specific types of models rather than black-box technologies. We study the relation of contrastive and counterfactual explanations and propose mathematical formalizations as well as a 2-phase algorithm for efficiently computing (plausible) pertinent positives of many standard machine learning models.**

***Index Terms*—XAI, Contrastive Explanations, Pertinent Positives**

I. Introduction

The increasing deployment of machine learning (ML) systems in practice led to an increased interest in explainability and transparency. In particular, "prominent failures" of ML systems like predictive policing [1], loan approval [2] and face recognition [3], highlighted the importance of transparency and explainability of ML systems. In addition, the need for explainability was also recognized by policy makers which resulted in a "right to an explanation" in the EUs "General Data Protection Right" (GDPR) [4]. The crucial problem with regard to these demands is the definition and the type of explanations - there exist many different kinds of explanations [5]–[9] but it is still not clear how to properly formalize an explanation [5], [10].

One family of explanations are example-based explanations [11] which are considered to be particularly well suited for lay people, since they allow the inspection of explanations by looking at example data, including the possibility of domain-specific representations of data [5]. Counterfactual explanations [12] and contrastive explanations constitute instantiations of example-based explanations [5], [13], [14]; these will be the focus in this work.

Following the common definition/intuition of a contrastive explanation [5], [14] (in the context of [13]), a contrastive explanation consists of two parts:

- A *pertinent positive* specifies a minimal and interpretable set of features that must be present for obtaining the same prediction as the complete sample does. Meaning that we are looking for a subset of features such that the resulting sample has the same prediction as the original sample.
- A *pertinent negative* specifies a set of features, which *must not* be present to provide the prediction, i.e. it is contrastive, since it relates to elements representative of a different class which are absent; expressed in different words, it refers to a small and interpretable perturbation of the original sample that would lead to a different prediction than the original sample.

Together, a pertinent negative and pertinent positive form a contrastive explanation.

For an example, consider the application of a loan approval system. Imagine that the system rejects a loan application and we now have to explain its decision. A possible contrastive explanation (consisting of a pertinent negative and a pertinent positive) might be: *The loan application was rejected because the pay back of the last loan was delayed, the applicant has a second credit card and because the monthly income is not above a minimum specific threshold, required for acceptance of the loan.* The first two arguments/reasons can be considered as a pertinent positive and the last reason as a pertinent negative. Note that, if more than two classes are present, pertinent negatives always contrast the present class to one specified alternative class.

Related work: There does exist extensive work and experimental evidence, which highlights that explanations provided by people are often contrastive in nature [15]: rather than explaining reasons for an observed event p, people often focus on reasons for observing p rather than another specific event q. The question of how to compute contrastive explanations for technical systems, constitutes an issue, though. In causal models, contrastive arguments of factors, which explain an appearance of p rather than q, can be based on according triangulations within the logical relations [16]. For black box models including deep networks, there exists some work how to compute contrastive explanations in practice [13], [17], [18]. More specifically, the authors of [13] propose an algorithm called "contrastive explanation method (CEM)" that computes a contrastive explanation of a differentiable model such as a Deep Neural Network. The method computes a pertinent positive and a pertinent negative by solving strongly regularized

We gratefully acknowledge funding from the VW-Foundation for the project *IMPACT* funded in the frame of the funding line *AI and its Implications for Future Society*.

cost functions by using a projected fast iterative shrinkage-thresholding (FISTA) algorithm. A part of the regularizations consists of an autoencoder ensuring that the solution is plausible. While this approach might be well suited for Deep Neural Networks, it might be less suited for standard ML models, where the regularization is not clear, and an autoencoder is not easily available, e.g. because the training set is too small. Furthermore, there do not exist theoretical guarantees of the result, in particular the sensitivity of the provided explanations with respect to the chosen regularization can be high.

In subsequent work [17], the authors extend CEM towards the model agnostic contrastive explanation method (MACEM) for computing contrastive explanations of an arbitrary (not necessarily differentiable) model. The modelling approach is somewhat similar to the one in [13]. MACEM uses FISTA and estimates the gradient in case of a fully black-box (not-differentiable) model. Furthermore, the authors also propose how to model categorical features.

The authors of [18] address model agnostic contrastive explanations, which are obtained based on locally trained decision trees which serve as a local surrogate of the observed model. Since this method needs to sample training points around a given data point, it is sensitive to the curse of dimensionality.

Most of the methods for computing contrastive explanations are somewhat model agnostic or are suitable for a "broader" class of models. As a consequence, it is not easily possible to provide guarantees on important properties such as uniqueness of the explanation, since no assumptions on the type of model are made. Further, the involved optimization technologies might be computationally demanding, and they often rely on iterative numeric methods such as general gradient-based optimization technologies. Here, we are interested in the question, how to efficiently compute contrastive explanations for specific models, which are popular in machine learning. For specific models, a general method might not be the most efficient one and specific formulations might provide particularly efficient alternatives, for which additional guarantees such as convexity and uniqueness hold. In this work we study how to exploit model specific structures for efficiently computing contrastive explanations of several standard ML models. To the best of our knowledge, this is the first work to address the question how to efficiently compute such model-specific contrastive explanations.

Our contributions: We make several contributions in this work:

1) In section II we address a conceptual issue, and we study how pertinent negatives are related to counterfactual explanations as discussed e.g. in [19]. We reduce the problem of computing a pertinent negative to the problem of computing a counterfactual explanation. For the latter, model-specific optimization schemes have recently been proposed in the work [20].
2) In section III we conceptualize computing pertinent positives and we propose a 2-phase algorithm for computing "high-quality" pertinent positives. In section III-C we develop mathematical programs (often even convex programs) for efficiently computing pertinent positives of many different standard ML models like linear/quadratic classifiers and learning vector quantization models. We also study how to compute plausible pertinent positives, and we also study special cases in which we can efficiently compute globally optimal pertinent positives.
3) We empirically evaluate our proposed methods in section III-F. For most settings, we obtain unique explanations.

Due to space constraints and for the purpose of better readability, we include all proofs and derivations to the appendix (section A).

II. Pertinent negatives as counterfactuals

A pertinent negative, as described in [13], specifies a "small and interpretable" perturbation $\vec{\delta}$ of the original sample $\vec{x}_{\text{orig}}$ that leads to a different prediction $y' \neq y_{\text{orig}}$, i.e. it contrasts the current output y_{orig} to another class y'. If we consider a small 1-norm as "small and interpretable", we can phrase the computation of a pertinent negative as the following optimization problem:

$$\min_{\vec{\delta}\,\in\,\mathbb{R}^d} \|\vec{\delta}\|_1 \qquad \text{s.t. } h(\vec{x}_{\text{orig}}+\vec{\delta}) = y' \neq y_{\text{orig}} \tag{1a}$$

where $h : \mathbb{R}^d \to \mathcal{Y}$ denotes the classifier whose prediction we want to explain. Here, the 1-norm accounts not only for a small change, but also sparsity as regards the number of features, which are changed.

The constrained optimization problem for computing a counterfactual explanation [12] as proposed by [20] is given as:

$$\min_{\vec{x}'\,\in\,\mathbb{R}^d} \theta(\vec{x}_{\text{orig}}, \vec{x}') \qquad \text{s.t. } h(\vec{x}') = y' \tag{2a}$$

where $\theta(\cdot)$ denotes a regularization (e.g. 1-norm), $\vec{x}'$ denotes the counterfactual and $y' \neq y_{\text{orig}}$ the requested target label.

We can turn Eq. (1) into Eq. (2) by setting $\vec{x}' = \vec{x}_{\text{orig}} + \vec{\delta}$ and choosing $\theta(\vec{x}_{\text{orig}}, \vec{x}') = \|\vec{x}_{\text{orig}} - \vec{x}'\|_1$. The appealing consequence of this is that we can reduce the problem of computing a pertinent negative to computing a counterfactual explanation for which several efficient methods already exists [12], [20]–[22]. The work [20], in particular, proposes convex formulations of the problem for a number of important ML models. The work [23] enriches this framework with efficient approximations of how to compute plausible counterfactuals with a guaranteed likelihood value, in order to distinguish those from adversarial examples, which correspond to artificial signals in particular for high dimensional data [24].

Note that the computation of pertinent negatives as counterfactual explanations perfectly fits the intuition of contrasting the given prediction y_{orig} against some other (predefined) prediction y' as discussed in the introduction of this work.

III. Pertinent positives

A. Modelling

In order to model the intuition of a pertinent positive, as described in [13], we have to consider several aspects:

- We want to "turn off" as many features as possible.
- For "turned on" features, the difference to the original feature values should be as small as possible.
- The pertinent positive must be still classified as y_{orig}.

We denote the final pertinent positive[1] $\vec{x}'$ as:

$$\vec{x}' = \vec{x}_{\text{orig}} - \vec{\delta} \tag{3}$$

where $\vec{\delta}$ denotes the perturbation (changes) that give rise to the pertinent positive. In order to improve readability of the subsequent formulas, we will sometimes substitute Eq. (3) and optimize over $\vec{x}'$ instead of $\vec{\delta}$ - we mean by this an optimization over $\vec{\delta}$ which implies $\vec{x}'$.

Mathematically, we formalize the fact that a feature is "turned off" by its value being identical to zero. Like the authors of [17] did, we can always subtract a constant $\vec{b}$ from the original sample $\vec{x}_{\text{orig}}$ to allow non-zero default values - i.e. $\vec{b}$ would denote the feature wise base/default values at which we consider a particular feature to be "turned off", in the sense that a feature does not deviate "much" from the default value (e.g. the expected value or a statistically robust estimation thereof). In the following, we assume $\vec{b} = 0$ for simplicity.

Considering all these aspects yields the following multi-objective optimization problem:

$$\min_{\vec{\delta}\,\in\,\mathbb{R}^d} \left| [\vec{x}_{\text{orig}} - \vec{x}']_{\mathcal{I}} \right| \quad \text{where } \vec{x}' = \vec{x}_{\text{orig}} - \vec{\delta} \tag{4a}$$

$$\min_{\mathcal{I}} \; |\mathcal{I}| \tag{4b}$$

$$\text{s.t.} \quad h(\vec{x}') = y_{\text{orig}} \tag{4c}$$

where $[\cdot]_{\mathcal{I}}$ denotes the selection operator on the set $\mathcal{I}$, whereby $\mathcal{I}$ denotes the set of all "turned on" features.[2] $\mathcal{I}$ is defined as follows:

$$\mathcal{I} = \Big\{ i : \big|(\vec{x}')_i\big| > \epsilon \Big\} \tag{5}$$

where $\epsilon \in \mathbb{R}_+$ denotes a tolerance threshold at which we consider a feature "to be turned on" - e.g. a strict choice would be $\epsilon = 0$.

The optimization problem Eq. (4) is difficult, since it is highly non-convex, it contains a discrete variable, and the two objectives Eq. (4a) and Eq. (4b) are in parts contradictory. Therefore, we propose a relaxation in the subsequent section. This relaxation allows us to efficiently compute pertinent positives of many standard ML models (we will turn this relaxation into a convex relaxation for many standard ML models) - we empirically evaluate our proposed relaxation in the experiments (see section III-F).

B. Relaxation by a 2-phase algorithm

For computing a pertinent positive Eq. (4), we have to ensure sparsity and closeness to the original sample. We propose to approximately solve Eq. (4) by a 2-phase algorithm where we separate the computational goals, sparsity and closeness, in two phases.

1) Sparsity: In order to achieve sparsity of the pertinent positive, we propose the following optimization for ensuring a sparse pertinent positive:

$$\min_{\vec{\delta}\,\in\,\mathbb{R}^d} \; \|\vec{x}_{\text{orig}} - \vec{\delta}\|_1 \tag{6a}$$

$$\text{s.t.} \quad h(\vec{x}_{\text{orig}} - \vec{\delta}) = y_{\text{orig}} \tag{6b}$$

Although the optimization problem Eq. (6) looks similar to the one proposed in [13], [17], a crucial difference is that Eq. (6) is a constrained optimization problem with a convex objective - this is what allows us (see section III-C) to derive convex programs for computing pertinent positives of many standard ML models. Sparsity is here enforced by the 1-norm, instead of the 0-norm. Furthermore, our formulation Eq. (6) allows to easily add additional constraints like box constraints or "freezing" some features, for meeting domain specific requirements (e.g. plausibility). Another consequence of our modelling is that we do not need any hyperparameters - note that the formulation in [13] uses several hyperparameters that have to be chosen. Since our formulation comes without any hyperparameters, the computation is easier. More importantly, by making use of convex optimization we can provide theoretical guarantees such as uniqueness or an exact statement of existence or non-existence of a solution.

2) Closeness: By solving the optimization problem Eq. (6) we obtain a sparse pertinent positive. As already discussed, while sparsity is in alignment with the intuition of a pertinent positive, it can happen that many features will be shrunken towards zero and thus be far away from the original features values - we will empirically observe this behavior in the experiments (section III-F) -, which contradicts the intuition of a pertinent positive. Therefore, we propose a second optimization step, enforcing closeness for the values, which are kept.

Also note that it can happen that the optimal solution of Eq. (6) is the zero vector $\vec{0}$; this holds if the zero vector is classified as the same class as the original sample - i.e. $h(\vec{0}) = y_{\text{orig}}$. In this case, all features would be "turned off".' We argue that in this case a pertinent positive might not make much sense because such an explanation would not be very informative for the user, and it is unclear how to break symmetries about which features are relevant in this case. Note that this might change when considering plausible pertinent positives instead of sparsest pertinent positives - see section III-E. We propose to reduce an explanation to the pertinent negative part, in this case, or to add additional semantic information, which indicates which features are relevant. As an example, one could avoid this issue by fixing some features to their original values or introducing box constraints

[1] It is debatable (and of course highly dependent on the use-case) whether the data point $\vec{x}'$ or the perturbation $\vec{\delta}$ is presented as the "explanation" to the user.

[2] The selection operator returns a vector whereby it only selects a subset of indices from the original vector as specified in the set $\mathcal{I}$.

```
Algorithm 1 Computation of a pertinent positive
Input: A labeled sample $(\vec{x}_{\text{orig}}, y_{\text{orig}})$
Output: A pertinent positive $\vec{x}'$
  Compute a pertinent positive $\vec{x}'$ by solving Eq. (6)
  Try to improve $\vec{x}'$ by solving Eq. (7)
```

that prevent a certain number of features of being "turned off". Such kind of constraints easily fit into the proposed optimization problems Eq. (6) and Eq. (7) and do not change the computational complexity of the problems.

Provided the first phase of the algorithm yields a reasonable and non-trivial solution $\{j : \mid |(\vec{x}')_i| \leq \epsilon\}$ for features which can be turned off, where $\vec{x}'$ is the solution from Eq. (6), we propose a second phase, where we minimize the distance of the remaining features to the original values, as follows:

$$\min_{\vec{x}' \,\in\, \mathbb{R}^d} \sum_{i \,\in\, \mathcal{I}} \left|(\vec{x}')_i - (\vec{x}_{\text{orig}})_i\right| \tag{7a}$$

$$\text{s.t.} \quad h(\vec{x}') = y_{\text{orig}} \tag{7b}$$

$$|(\vec{x}')_i| \leq \epsilon \quad \forall\, i \notin \mathcal{I} \tag{7c}$$

The final 2-phase algorithm is described as pseudo code in Algorithm 1 and is empirically evaluated in section III-F (Experiments). Interestingly, this two-step algorithm can be instantiated as efficient convex problems for many popular machine learning models, as we will show in the following.

C. Model specific programs

In the subsequent sections we study how the optimization problem Eq. (6) evolves for different standard ML models - in particular we reduce Eq. (6) to convex or "nearly convex" programs. Because the objectives Eq. (7a) and Eq. (6a) are both convex and independent of the model h, it is sufficient to work on Eq. (6) only - if we can turn Eq. (6) into a convex program (meaning we have to turn the constraint Eq. (7b) into a convex one), then the same holds for Eq. (7).

1) Linear models: A linear classifier $h : \mathbb{R}^d \to \mathcal{Y}$ can be written as follows:

$$h(\vec{x}) = \operatorname{sign}(\vec{w}^\top \vec{x} + b) \tag{8}$$

where we restrict our-self to a binary classifier - however, the idea (and everything that follows) can be generalized to multi-class problems. Popular instances of linear models are logistic regression, linear discriminant analysis (LDA) and linear support vector machine (linear-SVM).

Assuming $\mathcal{Y} = \{-1, 1\}$, we can rewrite the constraint Eq. (6b) as follows:

$$y_{\text{orig}} \vec{w}^\top \vec{\delta} - c + \epsilon \leq 0 \tag{9}$$

where ϵ denotes a small positive constant that ensures that the set of feasible solutions is closed (strict vs. non-strict inequality) and

$$c = y_{\text{orig}} \vec{w}^\top \vec{x}_{\text{orig}} + y_{\text{orig}} b \tag{10}$$

Note that Eq. (9) is linear in $\vec{\delta}$ and because the objectives Eq. (6a) and Eq. (7a) are linear, the optimization problems become linear programs which can be solved efficiently [25]. The derivation of Eq. (9) can be found in appendix A.

2) Quadratic models: A quadratic classifier $h : \mathbb{R}^d \to \mathcal{Y}$ can be written as follows:

$$h(\vec{x}) = \operatorname{sign}(\vec{x}^\top \mathbf{Q} \vec{x} + \vec{q}^\top \vec{x} + c) \tag{11}$$

where $\mathbf{Q} \in \mathcal{S}^d$ and again we restrict our-self to a binary classifier - again, the idea (and everything that follows) can be generalized to multi-class problems. Popular instances of quadratic models are quadratic discriminant analysis (QDA) and Gaussian Naive Bayes.

Again, if we assume $\mathcal{Y} = \{-1, 1\}$, we can rewrite the constraint Eq. (6b) as the following quadratic constraint:

$$\vec{\delta}^\top \tilde{\mathbf{Q}} \vec{\delta} + \vec{\delta}^\top \vec{z} + c' + \epsilon \leq 0 \tag{12}$$

where

$$\begin{aligned} \tilde{\mathbf{Q}} &= -y_{\text{orig}} \mathbf{Q} \qquad \vec{z} = 2 y_{\text{orig}} \vec{x}_{\text{orig}}^\top \mathbf{Q} \\ c' &= -y_{\text{orig}} \left(\vec{x}_{\text{orig}}^\top \mathbf{Q} \vec{x}_{\text{orig}} + \vec{q}^\top \vec{x}_{\text{orig}} + c \right) \end{aligned} \tag{13}$$

Since all we know about $\tilde{\mathbf{Q}}$ is that it is symmetric, Eq. (12) is in general non-convex. Solving non-convex quadratic programs is known to be NP-hard [25], [26]. However, we can rewrite Eq. (12) as a difference of two convex functions[3] and thus turn the whole program into a special instance of a difference of convex programming (DC) for which efficient approximation solvers exist - more details can be found in appendix B.

3) Learning vector quantization models: In learning vector quantization (LVQ) models [27] we compute a set of labeled prototypes $\{(\vec{\mathrm{p}}_i, o_i)\}$ from a training data set of labeled real-valued vectors - we refer to the i-th prototype as $\vec{\mathrm{p}}_i$ and the corresponding label as o_i. A new data point is classified according to the winner-takes-it-all scheme:

$$h : \vec{x} \mapsto o_i \qquad \text{s.t. } \vec{\mathrm{p}}_i = \underset{\vec{\mathrm{p}}_j}{\arg\min}\, \mathrm{d}(\vec{x}, \vec{\mathrm{p}}_j) \tag{14}$$

where $\mathrm{d}(\cdot)$ denotes a distance function. In vanilla LVQ, this is chosen globally as the squared Euclidean distance $\mathrm{d}(\vec{x}, \vec{\mathrm{p}}_j) = (\vec{x} - \vec{\mathrm{p}}_j)^\top \mathbb{I} (\vec{x} - \vec{\mathrm{p}}_j)$. There exist extensions to a global quadratic form $\mathrm{d}(\vec{x}, \vec{\mathrm{p}}_j) = (\vec{x} - \vec{\mathrm{p}}_j)^\top \mathbf{\Omega} (\vec{x} - \vec{\mathrm{p}}_j)$ with $\mathbf{\Omega} \in \mathcal{S}_+^d$, referred to as matrix-LVQ (GMLVQ) [28], or a prototype specific quadratic form $\mathrm{d}(\vec{x}, \vec{\mathrm{p}}_j) = (\vec{x} - \vec{\mathrm{p}}_j)^\top \mathbf{\Omega}_j (\vec{x} - \vec{\mathrm{p}}_j)$ with $\mathbf{\Omega}_j \in \mathcal{S}_+^d$, referred to as local-matrix LVQ (LGMLVQ) [29].

Similar to the algorithm for computing counterfactual explanations of LVQ models [30], the idea is to use a Divide-Conquer approach for computing a pertinent positive of a LVQ model Eq. (14). Because the LVQ model outputs the label of the closest prototype, we know that in order to get a specific prediction $y = y_{\text{orig}}$, the closest prototype must be one the prototypes labeled as y_{orig}. Therefore, we simply try

[3] Every symmetric matrix can be written as the difference of two s.psd. matrices.

Algorithm 2 Computing a pertinent positive of a LVQ model

```
Input: Labeled sample $(\vec{x}_{\text{orig}}, y_{\text{orig}})$ and the LVQ model
Output: Pertinent positive $\vec{x}'$
$\vec{x}' = \vec{0}$                                   ▷ Initialize dummy solution
$z = \infty$
for $\vec{\mathrm{p}}_i$ with $o_i = y_{\text{orig}}$ do     ▷ Try each prototype with a suitable label
    Solving Eq. (6) (substitute Eq. (6b) with Eq. (15)) yields a pertinent positive $\vec{x}'_*$
    if $\|\vec{x}'_*\|_1 < z$ then   ▷ Keep this pertinent positive if it is sparser than the currently "best" pertinent positive
        $z = \|\vec{x}'_*\|_1$
        $\vec{x}' = \vec{x}'_*$
    end if
end for
```

all possible prototypes (labeled as y_{orig}) and select the one that leads to the smallest objective Eq. (6a). For every suitable prototype $\vec{\mathrm{p}}_i$, we can rewrite the constraint Eq. (6b) as follows:

$$\vec{\delta}^\top \mathbf{A}_{ij}\vec{\delta} + \vec{\delta}^\top q_{ij} + c_{ij} + \epsilon \leq 0 \quad \forall\, j : o_j \neq y_{\text{orig}} \tag{15}$$

where

$$\begin{aligned} &\mathbf{A}_{ij} = \mathbf{\Omega}_i - \mathbf{\Omega}_j \quad \vec{q}_{ij} = 2\mathbf{\Omega}_j\left(\vec{x}_{\text{orig}} - \vec{\mathrm{p}}_j\right) - 2\mathbf{\Omega}_i\left(\vec{x}_{\text{orig}} - \vec{\mathrm{p}}_i\right) \\ &c_{ij} = \left(\vec{x}_{\text{orig}} - \vec{\mathrm{p}}_i\right)^\top \mathbf{\Omega}_i\left(\vec{x}_{\text{orig}} - \vec{\mathrm{p}}_i\right) - \\ &\qquad \left(\vec{x}_{\text{orig}} - \vec{\mathrm{p}}_j\right)^\top \mathbf{\Omega}_j\left(\vec{x}_{\text{orig}} - \vec{\mathrm{p}}_j\right) \end{aligned} \tag{16}$$

In case of GMLVQ, the constraints Eq. (15) become linear while in the case of LGMLVQ the constraints Eq. (15) become quadratic (but potentially non-convex). Because the objective Eq. (6a) is linear, Eq. (6) becomes a linear program in case of GMLVQ and a (non-convex) quadratic program in case of LGMLVQ. Again, while linear programs can be solved very efficiently [25], (non-convex) quadratic programs can not (unless they turn out to be convex quadratic programs) [25], [26]. Like in the case of quadratic classifiers, we can easily rewrite the constraint Eq. (15) as a difference of convex functions and then turn the whole program into a special instance of a DC for which good approximation solvers exist [26] - more details can be found in appendix C.

The resulting algorithm is summarized in Algorithm 2. Note that the `for` loop in Algorithm 2 can be easily parallelized because it does not matter when we compute the minimum.

D. Exact solutions for special cases

An alternative to the original modelling of a pertinent positive Eq. (4), is a stricter variant that do not allow any deviations for "turned on" features. Instead of Eq. (4), we propose the following similar modelling for computing a pertinent positive:

$$\min_{\mathcal{I}}\, |\mathcal{I}| \tag{17a}$$

$$\text{s.t. } h\Big(\left[\vec{x}_{\text{orig}}\right]_{\mathcal{I}}\Big) = y_{\text{orig}} \tag{17b}$$

Note that in contrast to Eq. (4), we require that all selected ("turned on") features are equal to their original values.

Although Eq. (17) is similar to Eq. (4), both modellings are not equivalent - however, a feasible solution of Eq. (17) is also feasible under Eq. (4).

In general, Eq. (17) can be interpreted as a feature selection problem which usually are computational difficult to solve exactly. However, in some cases (special instances of the classifier h) we can globally solve Eq. (17) efficiently - as we show in the next two subsections.

1) Linear model: We consider linear classifiers as defined in Eq. (8). We can rewrite Eq. (17) as follows:

$$\min_{\mathcal{I}}\, |\mathcal{I}| \qquad \text{s.t. } y_{\text{orig}}b + \sum_{i\,\in\,\mathcal{I}} z_i > 0 \tag{18}$$

where we defined

$$z_i = (\vec{w})_i(\vec{x}_{\text{orig}})_i y_{\text{orig}} \tag{19}$$

The new optimization problem Eq. (18) reduces the original problem Eq. (17) to a problem in which we want to find a minimal subset of real numbers z_i such that their sum is strictly greater than a given constant ($y_{\text{orig}}b$). We can find such a subset of numbers (features) by sorting all z_i in a descending order and then select the first k such that $y_{\text{orig}}b + \sum_i^k z_i > 0$. Finally, the construction of $\mathcal{I}$ follows immediately since each z_i corresponds uniquely to the i-th feature. Note that it might be the case that for some pairs $(\vec{x}_{\text{orig}}, y_{\text{orig}})$ no feasible solution exists - in such a case we have to fall back to the proposed 2-phase algorithm (see previous section). Also note that the globally optimal set $\mathcal{I}$ is not necessarily unique - however, the size $|\mathcal{I}|$ is unique.

2) Special quadratic model: We consider quadratic classifiers Eq. (11) where the matrix $\mathbf{Q}$ is a diagonal matrix - e.g. Gaussian Naive Bayes classifier:

$$\mathbf{Q} = \mathrm{diag}(\alpha_i) \quad \forall\, i : \alpha_i \in \mathbb{R} \tag{20}$$

Similar to the previous case of a linear classifier, we can rewrite the constraint Eq. (17b) as an independent of sum of weighted features - i.e. every summand uniquely corresponds to a feature:

$$z_i = \alpha_i(\vec{x}_{\text{orig}})_i^2 y_{\text{orig}} + (\vec{q})_i(\vec{x}_{\text{orig}})_i y_{\text{orig}} \tag{21}$$

Like in the case of the linear classifier, we can find a globally optimal solution by finding a minimal subset of indices i such that $y_{\text{orig}}b + \sum_i z_i > 0$. Again, we can do so by sorting all $\vec{z}_i$ in descending order and select the first k item such that $y_{\text{orig}}b + \sum_i^k z_i > 0$ - again, note that the global optimum is not necessarily unique and there might not even exist a feasible solution for every possible $\vec{x}_{\text{orig}}$ and y_{orig}.

E. Plausibility

So far, we ignore the aspect of plausibility - i.e. making sure that the pertinent positive is realistic and plausible in the data domain.

Here we propose to make use of a known density based method for computing plausible counterfactual explanations [23]. The authors of [23] propose to use a Gaussian Mixture Model (GMM) for estimating the density from a given training set $\hat{p}_{\text{GMM}}(\vec{x}) = \sum_{j=1}^{m} \pi_j \mathcal{N}(\vec{x} \mid \vec{\mu}_j, \mathbf{\Sigma}_j)$. Then they propose to use the following component wise approximation of $\hat{p}_{\text{GMM}}(\vec{x})$ as an additional quadratic convex constraint in the optimization problem for computing plausible counterfactual explanations:

$$(\vec{x} - \vec{\mu}_j)^\top \mathbf{\Sigma}_j^{-1} (\vec{x} - \vec{\mu}_j) + c_j \leq \delta' \tag{22}$$

where δ' denotes a density threshold that ensures that the solution $\vec{x}$ lies in a region of high density[4]. Since their proposed approximation is a component wise approximation, they get m different constraints of the form Eq. (22) and therefore have to solve the original optimization problem m times (each time with a different constraint Eq. (22)) - in the end they select the solution that minimizes the original objective (in their case the original objective is closeness to the original sample). Note that we can simply transfer this approach of plausibility to the setting of computing plausible pertinent positives because the approach [23] is completely independent of the original objective and other constraints - i.e. it is a general approach of ensuring that the solution of a mathematical program lies in a region of high density. The appealing benefit of using this approach is that we can guarantee that the resulting solution (pertinent positive) lies in a region of high density (i.e. it is plausible and realistic under the fitted GMM) although we give up closeness which we argue is not that important when it comes to plausibility.

In section III-F we empirically evaluate the quality of the plausible pertinent positives as computed by this approach and compare them with the closest pertinent positives as computed by our proposed 2-phase algorithm (see section III-B).

F. Experiments

We want to empirically verify that our proposed modelling yields pertinent positives that fit the intuition of a pertinent positive as discussed in the introduction. We therefore evaluate our proposed modelling and the derived mathematical programs on a set of different standard benchmark data sets. We compare the results of Eq. (6) with those of the 2-phase algorithm Algorithm 1. Since the convex programs are guaranteed to output valid pertinent positive, we would have to validate the outputs (check if it is a valid pertinent positive) of the non-convex programs only (e.g. DCs for quadratic and LGMLVQ models) - however, we can neglect this in our specific situation because we choose a specific solver that is guaranteed to output a feasible solution.

Evaluation measures: For the quantitative evaluation of the computed pertinent positives, we choose two scoring functions for assessing sparsity and closeness to the original sample. We evaluate sparsity of a pertinent positive $\vec{x}'$ with Eq. (23)[5] and closeness to the original sample $\vec{x}_{\text{orig}}$ with Eq. (24).

$$\|\vec{x}_{\text{orig}}\|_0 - \|\vec{x}'\|_0 \tag{23}$$

$$\sum_{i \in \mathcal{I}} \left|(\vec{x}')_i - (\vec{x}_{\text{orig}})_i\right| \tag{24}$$

Experimental setup: We run the experiments on four standard benchmark sets using logistic regression, a quadratic discriminant analysis (QDA) and GLVQ. We use the "Iris Plants Data Set" [31], the "Wine data set" [32], the "Ames Housing dataset" [33][6] and the "Breast Cancer Wisconsin (Diagnostic) Data Set" [34]. We compute a three-fold cross validation and compute a pertinent positive by only solving Eq. (6) and another one by using our proposed 2-phase algorithm (Algorithm 1). We standardize all data sets, use a regularization strength of 1.0 when estimating the covariance matrices in QDA, set the basis values to $\vec{b} = \vec{0}$ and set the threshold for "turned on" features Eq. (5) to $\epsilon = 0$ for all data sets. We report the mean sparsity Eq. (23) and the mean closeness Eq. (24) for each combination of model, method and data set (we also report the variance) - because the sparsity does not change when using the 2-phase algorithm instead of Eq. (6) only, we only report sparsity once. For the purpose of better observing the properties of non-trivial pertinent positives, we always exclude the class of the zero vector - as discussed in section III-B2, all samples from the class $h(\vec{0})$ would yield the sparsest and trivial pertinent positive $\vec{0}$ which makes them less suited for evaluating our proposed algorithms. In case of logistic regression, we also compute a globally optimal solution of a strict pertinent positive as modelled in section III-D. We compare the sparsity of these strict pertinent positives with the sparsity of the pertinent positives as computed by our proposed 2-phase algorithm. In addition, we compare the feature overlap between pertinent negatives and pertinent positives.

For the purpose of informative and useful explanations it is beneficial that the pertinent positive and the pertinent negative "share" as few features as possible - meaning that the overlap of "turned on" features in the pertinent positive and the perturbed features in a pertinent negative should be rather small. We argue that if the pertinent positive and the pertinent negative "share" many features they might not be that useful and informative[7] - if the overlap of features happens to be too large, one could add additional constraints to the optimization problems for manually including or excluding some features that finally result in a smaller overlap of features.

We compute the pertinent negatives by using a Python toolbox [35] for efficiently computing counterfactual explanations - we use the 1-norm as a regularization for enforcing sparsity. We also keep track of the F1-score to ensure that the classifiers learned a "somewhat reasonable" decision boundary - because all classifiers perform quite well, we do not report

[4] The definition of the other constants in Eq. (22) can be found in [23].

[5] We compare the sparseness of the original data point with the sparseness of the pertinent positive.

[6] We turn it into a binary classification problem by setting the target to 1 if the price is greater or equal to 160k$ and 0 otherwise. In addition, we select the following features: TotalBsmt, 1stFlr, 2ndFlr, GrLivA, WoodDeck, OpenP, 3SsnP, ScreenP and PoolA

[7] This depends of course a lot on the specific situation and use case.

TABLE I

Computation of contrastive explanations of different models on different standard benchmark data sets. We compute the sparsity Eq. (23), the closeness to the original sample Eq. (24) and the overlap of perturbed features in the pertinent negative vs. "turned on" features in the pertinent positive. We report the mean and variance (rounded to two decimal places) - for sparsity larger values are better whereas for closeness and the feature overlap smaller values are better. "Closeness+" denotes the closeness of the pertinent positive computed by the 2-phase algorithm (Algorithm 1) and "FeatOverlap" denotes the overlap of "turned on" features in the pertinent positive and the perturbed features in the pertinent negative.

Dataset	Scores	LogisticRegression	QDA	GLVQ
Iris	Sparsity	3.0(±0.0)	1.38(±0.3)	3.0(±0.0)
	Closeness	0.62(±0.06)	2.01(±0.58)	0.57(±0.14)
	Closeness+	0.01(±0.0)	0.11(±0.09)	0.14(±0.05)
	FeatOverlap	1.0(±0.0)	2.62(±0.3)	0.0(±0.0)
House prices	Sparsity	8.0(±0.0)	5.68(±1.2)	8.0(±0.0)
	Closeness	0.53(±0.0)	1.59(±0.36)	0.93(±0.28)
	Closeness+	0.0(±0.0)	0.43(±0.24)	0.37(±0.21)
	FeatOverlap	1.0(±0.0)	3.32(±1.2)	1.0(±0.0)
Breast cancer	Sparsity	29.0(±0.0)	17.38(±9.06)	29.0(±0.0)
	Closeness	0.89(±0.56)	9.42(±35.94)	2.59(±0.64)
	Closeness+	0.89(±0.56)	0.61(±0.63)	2.59(±0.66)
	FeatOverlap	1.0(±0.0)	12.62(±9.06)	0.04(±0.04)
Wine	Sparsity	12.0(±0.0)	7.7(±3.44)	11.45(±0.25)
	Closeness	0.61(±0.07)	3.54(±3.66)	1.05(±0.31)
	Closeness+	0.0(±0.0)	0.45(±0.42)	0.05(±0.01)
	FeatOverlap	1.0(±0.0)	5.3(±3.44)	1.1(±0.99)

the F1-scores in here and refer the interested reader to the published source code and protocols. We approximately solve the non-convex QPs using the convex-concave penalty (CCP) method [26]. Because the CCP method is guaranteed to output a feasible solution, we do not have to check if the pertinent positive is valid. Further details (including the raw protocols of the experiments) and the implementations itself is available on GitHub[8]. The results are shown in Table I whereby more details can be found in the raw protocols of the experiments that are available on GitHub.

Discussion of results: We observe that our proposed method is able to consistently compute sparse pertinent positives. Furthermore, we observe that our proposed 2-phase algorithm significantly increases the closeness of the pertinent positives to the original samples. Only in the case of GLVQ and logistic regression in combination with the breast cancer data set, the 2-phase algorithm is not able to improve on average upon Eq. (6) - we think that this might be an issue of unfavorable chosen hyperparameters[9] (we expect that changing the model would most likely show a difference). In addition, the large variances in the results of QDA for the breast cancer data set can be explained by some outliers. Also note that the mean sparsity is often just a little bit below the total number of features. This means that our proposed method was able to "turn off" many features which perfectly fits the intuition of a pertinent positive as discussed in the introduction. In case of logistic regression, we observe that the sparsity of the (globally optimal) strict pertinent positives with the sparsity of the pertinent positives as computed by our proposed 2-phase algorithm are equal for all data sets. We argue that this is a strong indicator for the powerfullness of our proposed 2-phase algorithm since it can reproduce a globally optimal strict pertinent positive although it is an approximation only. Finally, we observe that the overlap of "turned on" features in the pertinent positives and the perturbed features in the pertinent negatives is relatively small. This means that the pertinent positives and the pertinent negatives "share" only very few features in their explanations which makes them useful and informative in practice - as discussed previously, if both explanations would use (more or less) the same features, they would not be that informative to a user. However, please note that these findings are empirically only and might not necessarily generalize to other models and/or data sets.

Evaluating plausibility constraints: In order to demonstrate the effectiveness of the plausibility constraints (see section III-E), we compute and compare sparsest pertinent positives (as computed by our proposed 2-phase algorithm) and (approximately) closest plausible pertinent positives of the "Optical Recognition of Handwritten Digits Data Set" [36] under a softmax regression model - details on the hyper parameters can be found in the source code. In addition, we also compute a closest pertinent negative (we always choose $y' = 0$ as a target label) and compare it with the closest pertinent positive of the original class.

The results are shown in Fig. 1. We observe that the sparsest pertinent positives are by no way plausible (often very few features or no features at all are already sufficient for the prediction) but the closest pertinent positives under

[8] https://github.com/andreArtelt/contrastive_explanations

[9] Note that we use the same hyperparameters over all data sets.

plausibility constraints are plausible. We also observe that a closest pertinent negative and a sparsest pertinent positive are quite different from each other - which perfectly agrees with our observations of a low feature overlap in Table I.

Fig. 1. Samples from the digit data set. *First block:* Original samples. *Second block:* Closest pertinent negative where we always chose the same target label 0. *Third block:* Pertinent positives generated without any density/plausibility constraints. *Fourth block:* Pertinent positives generated with the proposed plausibility constraint. The corresponding labels are shown below each image.

Original samples

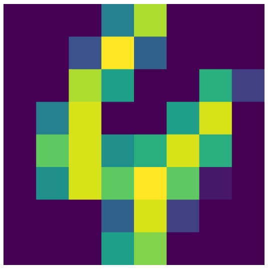

Fig. 2. Label: 4

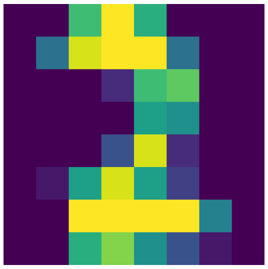

Fig. 3. Label: 2

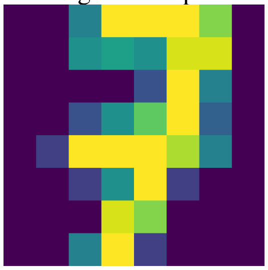

Fig. 4. Label: 7

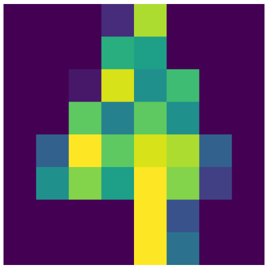

Fig. 5. Label: 4

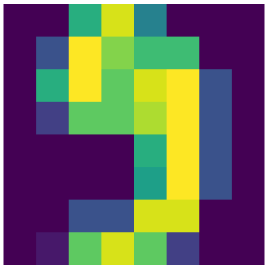

Fig. 6. Label: 9

Closest *pertinent negative* under a softmax regression model

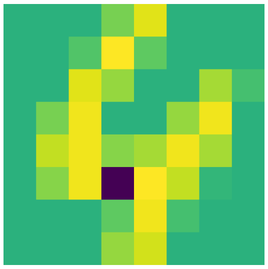

Fig. 7. Label: 0

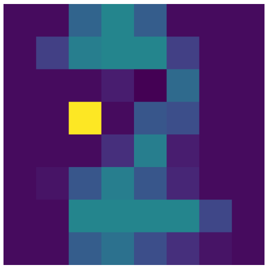

Fig. 8. Label: 0

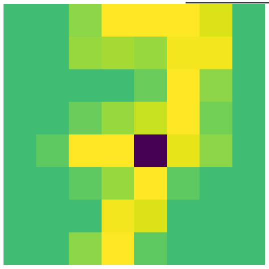

Fig. 9. Label: 0

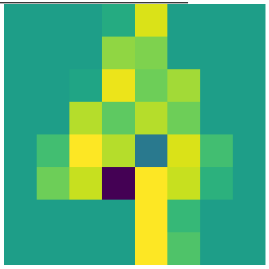

Fig. 10. Label: 0

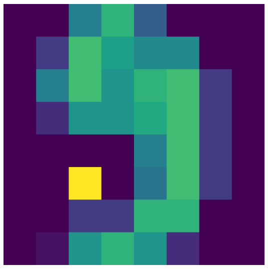

Fig. 11. Label: 0

Closest *pertinent positives* under a softmax regression model

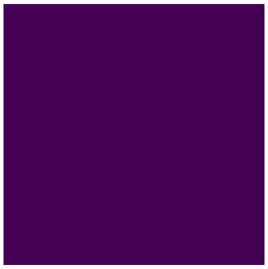

Fig. 12. Label: 4

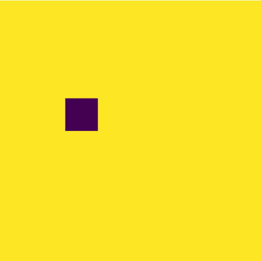

Fig. 13. Label: 2

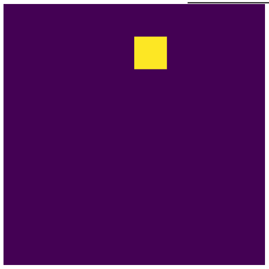

Fig. 14. Label: 7

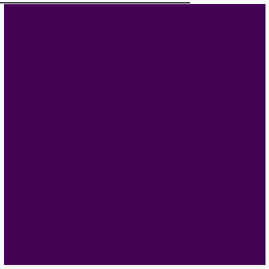

Fig. 15. Label: 4

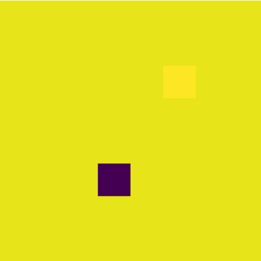

Fig. 16. Label: 9

Closest *plausible pertinent positive* under a softmax regression model

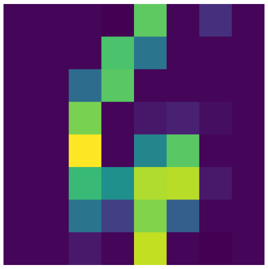

Fig. 17. Label: 4

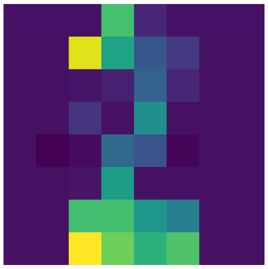

Fig. 18. Label: 2

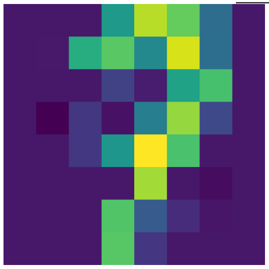

Fig. 19. Label: 7

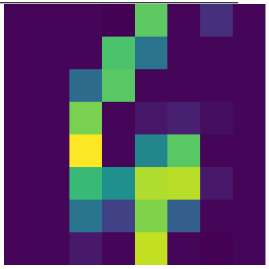

Fig. 20. Label: 4

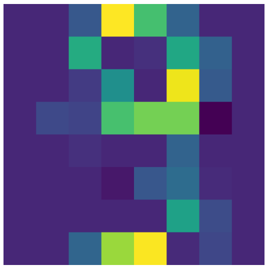

Fig. 21. Label: 9

IV. Discussion and Conclusion

In this work we extensively studied the computation of contrastive explanations that consists of a pertinent negative and a pertinent positive. We argued that computing a pertinent negative is equivalent to computing a counterfactual explanation - this reduction enables us to use methods from the counterfactual explanations literature for efficiently computing pertinent negatives. We also proposed to model pertinent positives as a constrained optimization problem and proposed upon that a 2-phase algorithm for computing qualitatively better pertinent positives. Building upon these, we derived mathematical programs for efficiently computing pertinent positives of many standard ML models. Furthermore, we proposed a stricter modelling of pertinent positives that allowed us to exactly efficiently compute pertinent positives. We also successfully applied ideas for computing plausible counterfactual explanations to the problem of computing plausible pertinent positives. Finally, we empirically evaluated our proposed methods on several standard benchmark data sets.

Appendix

A. Pertinent positives of linear models

We assume $\mathcal{Y} = \{-1, 1\}$ and $h(\vec{x}) = \operatorname{sign}\left(\vec{w}^\top\vec{x} + b\right)$. We then can rewrite the constraint Eq. (6b) as follows:

$$
\begin{aligned}
& h(\vec{x}_{\text{orig}} - \vec{\delta}) = y_{\text{orig}} \\
& \Leftrightarrow y_{\text{orig}}\left(h(\vec{x}_{\text{orig}} - \vec{\delta})\right) > 0 \\
& \Leftrightarrow y_{\text{orig}}\left(\vec{w}^\top\left(\vec{x}_{\text{orig}} - \vec{\delta}\right) + b\right) > 0 \\
& \Leftrightarrow y_{\text{orig}}\left(\vec{w}^\top\vec{x}_{\text{orig}} - \vec{w}^\top\vec{\delta} + b\right) > 0 \\
& \Leftrightarrow \underbrace{y_{\text{orig}}\vec{w}^\top\vec{x}_{\text{orig}} + y_{\text{orig}}b}_{\text{constant} := c} - y_{\text{orig}}\vec{w}^\top\vec{\delta} > 0 \\
& \Leftrightarrow y_{\text{orig}}\vec{w}^\top\vec{\delta} - c < 0
\end{aligned}
\tag{25}
$$

where we temporarily ignored the special case of $\operatorname{sign}(0)$.

Finally, we relax the strict inequality by adding a small positive number ϵ to the left side - by doing this we avoid that the resulting data points lies on the decision boundary (in this case the sign would be undefined):

$$
y_{\text{orig}}\vec{w}^\top\vec{\delta} - c + \epsilon \leq 0 \tag{26}
$$

Note that Eq. (26) is linear in $\vec{\delta}$ - thus the final optimization problems become linear programs (LPs) which can be solved very efficiently [25].

In case of a multi-class problem, we would get multiple constraints of the form Eq. (26) - however, since they are all linear, the final problems are still LPs.

B. Pertinent positives of quadratic models

We assume $\mathcal{Y} = \{-1, 1\}$ and $h(\vec{x}) = \operatorname{sign}\left(\vec{x}^\top\mathbf{Q}\vec{x} + \vec{q}^\top\vec{x} + c\right)$ with $\mathbf{Q} \in \mathcal{S}^d$. We then can rewrite the constraint Eq. (6b) as follows:

$$
\begin{aligned}
& h(\vec{x}_{\text{orig}} - \vec{\delta}) = y_{\text{orig}} \\
& \Leftrightarrow y_{\text{orig}}\, h(\vec{x}_{\text{orig}} - \vec{\delta}) > 0 \\
& \Leftrightarrow y_{\text{orig}}\Big(\left(\vec{x}_{\text{orig}} - \vec{\delta}\right)^\top \mathbf{Q}\left(\vec{x}_{\text{orig}} - \vec{\delta}\right) + \vec{q}^\top\left(\vec{x}_{\text{orig}} - \vec{\delta}\right) + c\Big) > 0 \\
& \Leftrightarrow y_{\text{orig}}\Big(\vec{x}_{\text{orig}}^\top\mathbf{Q}\vec{x}_{\text{orig}} - \vec{x}_{\text{orig}}^\top\mathbf{Q}\vec{\delta} - \\
& \qquad \vec{\delta}^\top\mathbf{Q}\vec{x}_{\text{orig}} + \vec{\delta}^\top\mathbf{W}\vec{\delta} + \vec{q}^\top\vec{x}_{\text{orig}} - \vec{q}^\top\vec{\delta} + c\Big) > 0 \\
& \Leftrightarrow \underbrace{y_{\text{orig}}\left(\vec{x}_{\text{orig}}^\top\mathbf{Q}\vec{x}_{\text{orig}} + \vec{q}^\top\vec{x}_{\text{orig}} + c\right)}_{\text{constant} := -c'} + \underbrace{-2y_{\text{orig}}\vec{x}_{\text{orig}}^\top\mathbf{Q}}_{\text{constant} := -\vec{z}^\top}\vec{\delta} + \\
& \quad y_{\text{orig}}\vec{\delta}^\top\mathbf{Q}\vec{\delta} > 0 \\
& \Leftrightarrow \vec{\delta}^\top\tilde{\mathbf{Q}}\vec{\delta} + \vec{\delta}^\top\vec{z} + c' < 0
\end{aligned}
\tag{27}
$$

where we defined

$$
\tilde{\mathbf{Q}} = -y_{\text{orig}}\mathbf{Q} \tag{28}
$$

Again, we relax the strict inequality by adding a small positive number ϵ to the left side:

$$
\vec{\delta}^\top\tilde{\mathbf{Q}}\vec{\delta} + \vec{\delta}^\top\vec{z} + c' + \epsilon \leq 0 \tag{29}
$$

A basic fact from linear algebra states that we can rewrite every real symmetric matrix as the difference of two s.psd. matrices. Furthermore, in case of QDA or Gaussian Naive Bayes such a decomposition appears naturally because in both cases the matrix $\mathbf{Q}$ is defined as the difference of two (s.psd.) covariance matrices. Assuming that we decompose $\tilde{\mathbf{Q}}$ as

$$
\tilde{\mathbf{Q}} = \tilde{\mathbf{Q}}_1 - \tilde{\mathbf{Q}}_2 \qquad \tilde{\mathbf{Q}}_1, \tilde{\mathbf{Q}}_2 \in \mathcal{S}_+^d \tag{30}
$$

we can rewrite Eq. (29) as follows:

$$
\underbrace{\vec{\delta}^\top\tilde{\mathbf{Q}}_1\vec{\delta} + \vec{\delta}^\top\vec{z} + c' + \epsilon}_{\text{convex in } \vec{\delta}} - \underbrace{\vec{\delta}^\top\tilde{\mathbf{Q}}_2\vec{\delta}}_{\text{convex in } \vec{\delta}} \leq 0 \tag{31}
$$

Clearly, Eq. (31) is now a difference of convex quadratic functions which turns the resulting optimization problem into a DC for which good approximatation solvers like the Suggest-and-Improve framework exist [26].

In case of a multi-class problem, we would get multiple constraints of the form Eq. (31) - however, since they are all of the same form, the final optimization problems are still DCs.

C. Pertinent positives of LVQ models

If the data point $\vec{x}_{\text{orig}} - \vec{\delta}$ is classified as y_{orig}, we know that the closest prototype must be one labeled as y_{orig}. Therefore, for each suitable prototype $\vec{\mathrm{p}}_i$) (that is $o_i = y_{\text{orig}}$), we get the following set of constraints:

$$
\mathrm{d}(\vec{x}_{\text{orig}} - \vec{\delta}, \vec{\mathrm{p}}_i) < \mathrm{d}(\vec{x}_{\text{orig}} - \vec{\delta}, \vec{\mathrm{p}}_j) \quad \forall\, j : o_j \neq y_{\text{orig}} \tag{32}
$$

After rearranging the terms and relaxing the strict inequality by adding a small positive ϵ, we get:

$$
\mathrm{d}(\vec{x}_{\text{orig}} - \vec{\delta}, \vec{\mathrm{p}}_i) - \mathrm{d}(\vec{x}_{\text{orig}} - \vec{\delta}, \vec{\mathrm{p}}_j) + \epsilon \leq 0 \quad \forall\, j : o_j \neq y_{\text{orig}} \tag{33}
$$

Fixing i and j and plugging the most general distance function $\mathrm{d}(\vec{x}, \vec{\mathrm{p}}_j) = (\vec{x} - \vec{\mathrm{p}}_j)^\top \mathbf{\Omega}_j (\vec{x} - \vec{\mathrm{p}}_j)$ with $\mathbf{\Omega}_j \in \mathcal{S}^d_+$ (LGMLVQ) into Eq. (33), yields:

$$
\begin{aligned}
&\mathrm{d}(\vec{x}_{\text{orig}} - \vec{\delta}, \vec{\mathrm{p}}_i) - \mathrm{d}(\vec{x}_{\text{orig}} - \vec{\delta}, \vec{\mathrm{p}}_j) + \epsilon \leq 0 \\
&\Leftrightarrow \left(\vec{x}_{\text{orig}} - \vec{\delta} - \vec{\mathrm{p}}_i\right)^\top \mathbf{\Omega}_i \left(\vec{x}_{\text{orig}} - \vec{\delta} - \vec{\mathrm{p}}_i\right) - \\
&\quad \left(\vec{x}_{\text{orig}} - \vec{\delta} - \vec{\mathrm{p}}_j\right)^\top \mathbf{\Omega}_j \left(\vec{x}_{\text{orig}} - \vec{\delta} - \vec{\mathrm{p}}_j\right) + \epsilon \leq 0 \\
&\Leftrightarrow \left(\vec{x}_{\text{orig}} - \vec{\mathrm{p}}_i\right)^\top \mathbf{\Omega}_i \left(\vec{x}_{\text{orig}} - \vec{\mathrm{p}}_i\right) - 2\left(\vec{x}_{\text{orig}} - \vec{\mathrm{p}}_i\right)^\top \mathbf{\Omega}_i \vec{\delta} + \\
&\quad \vec{\delta}^\top \mathbf{\Omega}_i \vec{\delta} - \left(\vec{x}_{\text{orig}} - \vec{\mathrm{p}}_j\right)^\top \mathbf{\Omega}_j \left(\vec{x}_{\text{orig}} - \vec{\mathrm{p}}_j\right) + \\
&\quad 2\left(\vec{x}_{\text{orig}} - \vec{\mathrm{p}}_j\right)^\top \mathbf{\Omega}_j \vec{\delta} - \vec{\delta}^\top \mathbf{\Omega}_j \vec{\delta} + \epsilon \leq 0 \\
&\Leftrightarrow \vec{\delta}^\top \left(\underbrace{\mathbf{\Omega}_i - \mathbf{\Omega}_j}_{:=\mathbf{A}_{ij}}\right) \vec{\delta} + \vec{\delta}^\top \underbrace{\left(2\mathbf{\Omega}_j \left(\vec{x}_{\text{orig}} - \vec{\mathrm{p}}_j\right) - 2\mathbf{\Omega}_i \left(\vec{x}_{\text{orig}} - \vec{\mathrm{p}}_i\right)\right)}_{\text{constant } :=\vec{q}_{ij}} \\
&\quad + \underbrace{\left(\vec{x}_{\text{orig}} - \vec{\mathrm{p}}_i\right)^\top \mathbf{\Omega}_i \left(\vec{x}_{\text{orig}} - \vec{\mathrm{p}}_i\right) - \left(\vec{x}_{\text{orig}} - \vec{\mathrm{p}}_j\right)^\top \mathbf{\Omega}_j \left(\vec{x}_{\text{orig}} - \vec{\mathrm{p}}_j\right)}_{\text{constant } :=c_{ij}} \\
&\quad + \epsilon \leq 0
\end{aligned}
\tag{34}
$$

Because all we can say about $\mathbf{A}_{ij}$ is that it is symmetric, the constraints Eq. (34) are quadratically non-convex. However, like we did in case of quadratic models (see appendix B), we can rewrite the constraints Eq. (34) as a difference of convex functions[10] and turn the whole optimization problem into a special case of a DC. Therefore, in case of LGMLVQ, the optimization problems are non-convex and can only be approximately solved (e.g. via a DC).

In case of G(M)LVQ, the distance matrices $\mathbf{\Omega}_i$ are always the same. Thus, the constraint Eq. (6b) becomes a set of linear constraints:

$$
\vec{\delta}^\top \vec{q}_{ij} + c_{ij} + \epsilon \leq 0 \quad \forall j : o_j \neq y_{\text{orig}} \tag{35}
$$

As a consequence, the resulting optimization problems become LPs which can be solved very efficiently [25].

References

[1] P. Stalidis, T. Semertzidis, and P. Daras, "Examining deep learning architectures for crime classification and prediction," vol. abs/1812.00602, 2018. [Online]. Available: http://arxiv.org/abs/1812.00602

[2] K. Waddell, "How algorithms can bring down minorities' credit scores," The Atlantic, 2016.

[3] "Karen hao," Technology Review, 2019. [Online]. Available: https://www.technologyreview.com/2019/01/29/137676/making-face-recognition-less-biased-doesnt-make-it-less-scary/

[4] E. parliament and council, "General data protection regulation: Regulation (eu) 2016/679 of the european parliament," https://eur-lex.europa.eu/eli/reg/2016/679/oj, 2016.

[5] C. Molnar, Interpretable Machine Learning, 2019, https://christophm.github.io/interpretable-ml-book/.

[6] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," ACM Comput. Surv., vol. 51, no. 5, pp. 93:1–93:42, Aug. 2018. [Online]. Available: http://doi.acm.org/10.1145/3236009

[7] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning," in 5th IEEE International Conference on Data Science and Advanced Analytics, DSAA 2018, Turin, Italy, October 1-3, 2018, 2018, pp. 80–89. [Online]. Available: https://doi.org/10.1109/DSAA.2018.00018

[8] W. Samek, T. Wiegand, and K. Müller, "Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models," CoRR, vol. abs/1708.08296, 2017. [Online]. Available: http://arxiv.org/abs/1708.08296

[9] E. Tjoa and C. Guan, "A survey on explainable artificial intelligence (XAI): towards medical XAI," CoRR, vol. abs/1907.07374, 2019. [Online]. Available: http://arxiv.org/abs/1907.07374

[10] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," arXiv preprint arXiv:1702.08608, 2017.

[11] A. Aamodt and E. Plaza., "Case-based reasoning: Foundational issues, methodological variations, and systemapproaches." AI communications, 1994.

[12] S. Wachter, B. D. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the GDPR," CoRR, vol. abs/1711.00399, 2017. [Online]. Available: http://arxiv.org/abs/1711.00399

[13] A. Dhurandhar, P. Chen, R. Luss, C. Tu, P. Ting, K. Shanmugam, and P. Das, "Explanations based on the missing: Towards contrastive explanations with pertinent negatives," in Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada, S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018, pp. 590–601.

[14] F. Doshi-Velez, M. Kortz, R. Budish, C. Bavitz, S. Gershman, D. O'Brien, S. Schieber, J. Waldo, D. Weinberger, and A. Wood, "Accountability of AI under the law: The role of explanation," CoRR, vol. abs/1711.01134, 2017. [Online]. Available: http://arxiv.org/abs/1711.01134

[15] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," CoRR, vol. abs/1706.07269, 2017. [Online]. Available: http://arxiv.org/abs/1706.07269

[16] P. Lipton, "Contrastive explanation and causal triangulation," Philosophy of Science, vol. 58, no. 4, pp. 687–697, 1991.

[17] A. Dhurandhar, T. Pedapati, A. Balakrishnan, P. Chen, K. Shanmugam, and R. Puri, "Model agnostic contrastive explanations for structured data," CoRR, vol. abs/1906.00117, 2019. [Online]. Available: http://arxiv.org/abs/1906.00117

[18] J. van der Waa, M. Robeer, J. van Diggelen, M. Brinkhuis, and M. A. Neerincx, "Contrastive explanations with local foil trees," CoRR, vol. abs/1806.07470, 2018. [Online]. Available: http://arxiv.org/abs/1806.07470

[19] R. Poyiadzi, K. Sokol, R. Santos-Rodriguez, T. D. Bie, and P. A. Flach, "FACE: feasible and actionable counterfactual explanations," CoRR, vol. abs/1909.09369, 2019. [Online]. Available: http://arxiv.org/abs/1909.09369

[20] A. Artelt and B. Hammer, "On the computation of counterfactual explanations - a survey," ArXiv, vol. abs/1911.07749, 2019.

[21] A. V. Looveren and J. Klaise, "Interpretable counterfactual explanations guided by prototypes," CoRR, vol. abs/1907.02584, 2019. [Online]. Available: http://arxiv.org/abs/1907.02584

[22] S. Sharma, J. Henderson, and J. Ghosh, "CERTIFAI: counterfactual explanations for robustness, transparency, interpretability, and fairness of artificial intelligence models," CoRR, vol. abs/1905.07857, 2019. [Online]. Available: http://arxiv.org/abs/1905.07857

[23] A. Artelt and B. Hammer, "Convex density constraints for computing plausible counterfactual explanations." 29th International Conference on Artificial Neural Networks (ICANN), 2020.

[24] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in International Conference on Learning Representations, 2015. [Online]. Available: http://arxiv.org/abs/1412.6572

[25] S. Boyd and L. Vandenberghe, Convex Optimization. New York, NY, USA: Cambridge University Press, 2004.

[26] J. Park and S. Boyd, "A cvxpy extension for handling nonconvex qcqp via suggest-and-improve framework," https://github.com/cvxgrp/qcqp, 2017.

[10] In fact Eq. (34) decomposes naturally into a difference of convex functions because the only non-convex part $\mathbf{A}_{ij}$ is equal to the difference of two s.psd. distance matrices.

[27] D. Nova and P. A. Estévez, "A review of learning vector quantization classifiers," Neural Comput. Appl., vol. 25, no. 3-4, pp. 511–524, Sep. 2014. [Online]. Available: https://doi.org/10.1007/s00521-013-1535-3

[28] P. Schneider, M. Biehl, and B. Hammer, "Adaptive relevance matrices in learning vector quantization," Neural Computation, vol. 21, no. 12, pp. 3532–3561, 2009, pMID: 19764875. [Online]. Available: https://doi.org/10.1162/neco.2009.11-08-908

[29] ——, "Distance learning in discriminative vector quantization," Neural Computation, vol. 21, no. 10, pp. 2942–2969, 2009, pMID: 19635012. [Online]. Available: https://doi.org/10.1162/neco.2009.10-08-892

[30] A. Artelt and B. Hammer, "Efficient computation of counterfactual explanations of lvq models." Proceedings of the 28th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN), 2020.

[31] R. A. Fisher, "The use of multiple measurements in taxonomic problems," Annual Eugenics, vol. 7 Part II, pp. 179–188, 1936.

[32] D. C. S. Aeberhard and O. de Vel, "Comparison of classifiers in high dimensional settings," Tech. Rep. no. 92-02, 1992.

[33] D. D. Cock, "Ames, iowa: Alternative to the boston housing data as an end of semester regression project," Journal of Statistics Education, vol. 19, no. 3, 2011. [Online]. Available: https://doi.org/10.1080/10691898.2011.11889627

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

[35] A. Artelt, "Ceml: Counterfactuals for explaining machine learning models - a python toolbox," https://www.github.com/andreArtelt/ceml, 2019 - 2020.

[36] E. Alpaydin and C. Kaynak, "Optical recognition of handwritten digits data set," https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits, 1998.