Efficient computation of counterfactual explanations of LVQ models

André Artelt[*] and Barbara Hammer [†]

CITEC - Cognitive Interaction Technology
Bielefeld University - Faculty of Technology
Inspiration 1, 33619 Bielefeld - Germany

Abstract. The increasing use of machine learning in practice and legal regulations like EU's GDPR cause the necessity to be able to explain the prediction and behavior of machine learning models. A prominent example of particularly intuitive explanations of AI models in the context of decision making are counterfactual explanations. Yet, it is still an open research problem how to efficiently compute counterfactual explanations for many models. We investigate how to efficiently compute counterfactual explanations for an important class of models, prototype-based classifiers such as learning vector quantization models. In particular, we derive specific convex and non-convex programs depending on the used metric.

1 Introduction

Due to the recent advances in machine learning (ML), ML models are being more and more used in practice and applied to real-world scenarios for decision making [1–3]. Essential demands for user acceptance as well as legal regulations like the EU's "General Data Protection Right" (GDPR) [4], that contains a "right to an explanation", make it indispensable to explain the output and behavior of ML models in a comprehensible way.

As a consequence, many research approaches focused on the question how to realize explainability and transparency in machine learning in recent years [5–8]. There exist diverse methods for explaining ML models [6, 9]. One family of methods are model-agnostic methods [6,10]. Model-agnostic methods are flexible in the sense that they are not tailored to a particular model or representation. This makes model-agnostic methods (in theory) applicable to many different types of ML models. In particular, "truly" model-agnostic methods do not need access to the training data or model internals. It is sufficient to have an interface for passing data points to the model and obtaining the output/predictions of the model - the underlying model is viewed as a black-box.

Examples of model-agnostic methods are feature interaction methods [11], feature importance methods [12], partial dependency plots [13] and local methods that approximates the model locally by an explainable model (e.g. a decision tree) [14,15]. These methods explain the models by using features as vocabulary.

A different class of model-agnostic explanations are example-based explanations where a prediction or behavior is explained by a (set of) data points [16]. Instances of example-based explanations are prototypes & criticisms [17] and influential instances [18]. Another instance of example-based explanations are counterfactual explanations [19]. A counterfactual explanation is a change of the original input that leads to a different (specific) prediction/behavior of the ML model - *what has to be different in order to change the prediction of the model?* Such an explanation is considered to be fairly intuitive, human-friendly and useful because it tells people what to do in order to achieve a desired outcome [9, 19].

Existing counterfactual explanations are mostly model agnostic methods, which are universally applicable but computationally expensive [15, 20, 21]. In this work, we address the question how to efficiently compute counterfactuals for a popular class of models by referring to its specific structure.

Prototype based models such as learning vector quantization (LVQ) represent data by a set of representative samples [22]. LVQ models can be combined with metric learning and thereby increase the effectiveness of the model in case of few prototypes [23, 24]. Furthermore, LVQ models can be used in many settings like life-long learning [25].

Here, we will consider the question how to efficiently compute counterfactual explanations of prototype-based classifiers, in particular LVQ models. By exploiting the special structure of such models, we are able to

1. propose model- and regularization-dependent methods for efficiently computing counterfactual explanations of LVQ models,

2. and empirically demonstrate the efficiency of the modeling as regards speed-up as well as required amount of change in comparison to standard techniques.

Further, the framework enables a straightforward incorporation of domain knowledge, which can be phrased as additional constraints. Such domain knowledge could be used for generating more plausible and feasible counterfactual explanations.

The remainder of this paper is structured as follows: First, we briefly review counterfactual explanations (section 2) and learning vector quantization models (section 3). Then, in section 4 we introduce our convex and non-convex programming framework for efficiently computing counterfactual explanations of different types of LVQ models - note that all derivations can be found in the appendix (section 6). Furthermore, we empirically evaluate the efficiency of our methods. Finally, section 5 summarizes the results of this paper.

2 Counterfactual explanations

Counterfactual explanations [19] (often just called counterfactuals) are an instance of example-based explanations [16]. Other instances of example-based

explanations [9] are influential instances [18] and prototypes & criticisms [17].

A counterfactual states a change to some features of a given input such that the resulting data point (called counterfactual) has a different (specified) prediction than the original input. The rational is considered to be intuitive, human-friendly and useful because it tells practitioners which minimum changes can lead to a desired outcome [9, 19].

A classical use case of counterfactual explanations is loan application [3, 9]: *Imagine you applied for a credit at a bank. Unfortunately, the bank rejects your application. Now, you would like to know why. In particular, you would like to know what would have to be different so that your application would have been accepted. A possible explanation might be that you would have been accepted if you would earn 500$ more per month and if you would not have a second credit card.*

Although counterfactuals constitute very intuitive explanation mechanisms, there do exist a couple of problems.

One problem is that there often exist more than one counterfactual - this is called *Rashomon effect* [9]. If there are more than one possible explanation (counterfactual), it is not clear which one should be selected.

In our opinion, the main issue is feasibility and plausibility - in the sense that the counterfactual data point should be valid and plausible in the given domain. There exist first approaches like [26, 27] but in general it is still an open research question how to efficiently compute feasible and plausible counterfactuals.

An alternative - but very similar in the spirit - to counterfactuals [19] is the Growing Spheres method [21]. However, this method suffers from the curse of dimensionality because it has to draw samples from the input space, which can become difficult if the input space is high-dimensional.

Definition 1 (Counterfactual explanation). *Assume a prediction function h is given. Computing a counterfactual $\vec{x}' \in \mathbb{R}^d$ for a given input $\vec{x} \in \mathbb{R}^d$ is phrased as an optimization problem [19]:*

$$\underset{\vec{x}' \in \mathbb{R}^d}{\arg\min} \ \ell\left(h(\vec{x}'), y^c\right) + C \cdot \theta(\vec{x}', \vec{x}) \tag{1}$$

where $\ell(\cdot)$ denotes the loss function, y^c the requested prediction, and $\theta(\cdot)$ a penalty term for deviations of $\vec{x}'$ from the original input $\vec{x}$. $C > 0$ denotes the regularization strength.

Two common regularizations are the weighted Manhattan distance and the generalized L2 distance.

The weighted Manhattan distance is defined as

$$\theta(\vec{x}', \vec{x}) = \sum_j \alpha_j \cdot |(\vec{x})_j - (\vec{x}')_j| \tag{2}$$

where $\alpha_j > 0$ denote the feature wise weights. A popular choice [19] for α_j is the inverse median absolute deviation of the j-th feature median in the training

data set $\mathcal{D}$:

$$\alpha_j = \frac{1}{\text{MAD}_j}$$

$$\text{where } \text{MAD}_j = \underset{\vec{x} \in \mathcal{D}}{\text{median}} \left(\left| (\vec{x})_j - \underset{\vec{x} \in \mathcal{D}}{\text{median}} \big((\vec{x})_j \big) \right| \right) \tag{3}$$

The benefit of this choice is that it takes the (potentially) different variability of the features into account. However, because we need access to the training data set $\mathcal{D}$, this regularization is not a truly model-agnostic method - it is not usable if we only have access to a prediction interface of a black-box model.

The generalized L2 distance[1] is defined as

$$\theta(\vec{x}', \vec{x}) = \|\vec{x} - \vec{x}'\|^2_{\mathbf{\Omega}} = (\vec{x} - \vec{x}')^\top \mathbf{\Omega} (\vec{x} - \vec{x}') \tag{4}$$

where $\mathbf{\Omega}$ denotes a symmetric positive semi-definite (s.psd) matrix. Note that the L2 distance can be recovered by setting $\mathbf{\Omega} = \mathbb{I}$. The generalized L2 distance can be interpreted as the Euclidean distance in a linearly distorted space.

Depending on the model and the choice of $\ell(\cdot)$ and $\theta(\cdot)$, the final optimization problem might be differentiable or not. If it is differentiable, we can use a gradient-based optimization algorithm like conjugate gradients, gradient descent or (L-)BFGS. Otherwise, we have to use a black-box optimization algorithm like Downhill-Simplex (also called Nelder-Mead) or Powell. However, using a model and regularization specific optimization technique has the best chances to enable a simplification and computationally efficient solution. Unfortunately, there exit model specific optimizers for counterfactual explanations for few ML models only.

3 Learning vector quantization

In learning vector quantization (LVQ) models [22] we compute a set of labeled prototypes $\{(\vec{p}_i, o_i)\}$ from a training data set of labeled real-valued vectors - we refer to the i-th prototype as $\vec{p}_i$ and the corresponding label as o_i. A new data point is classified according to the winner-takes-it-all scheme:

$$h(\vec{x}) = o_i \quad \text{s.t.} \ \vec{p}_i = \underset{\vec{p}_j}{\arg\min} \ \text{d}(\vec{x}, \vec{p}_j) \tag{5}$$

where $\text{d}(\cdot)$ denotes a function for computing the distance between a data point and a prototype - in vanillas LVQ, this is independent chosen globally as the squared Euclidean distance:

$$\text{d}(\vec{x}, \vec{p}) = (\vec{x} - \vec{p})^\top \mathbb{I} (\vec{x} - \vec{p}) \tag{6}$$

There exist matrix-LVQ models like GMLVQ, LGMLVQ [23], MRSLVQ and LMRSLVQ [24] that learn a custom (class or prototype specific) distance matrix

---

[1]also called Mahalanobis distance

$\boldsymbol{\Omega}_{\mathrm{p}}$ that is used instead of the identity $\mathbb{I}$ when computing the distance between a data point and a prototype. This gives rise to the generalized L2 distance Eq. (4):

$$\mathrm{d}(\vec{x}, \vec{\mathrm{p}}) = (\vec{x} - \vec{\mathrm{p}})^{\top} \boldsymbol{\Omega}_{\mathrm{p}} (\vec{x} - \vec{\mathrm{p}}) \tag{7}$$

Because $\boldsymbol{\Omega}_{\mathrm{p}}$ must be a s.psd matrix, instead of learning $\boldsymbol{\Omega}_{\mathrm{p}}$ directly, these LVQ variants learn a matrix $\tilde{\boldsymbol{\Omega}}_{\mathrm{p}}$ and compute the final distance matrix as:

$$\boldsymbol{\Omega}_{\mathrm{p}} = \tilde{\boldsymbol{\Omega}}_{\mathrm{p}}^{\top} \tilde{\boldsymbol{\Omega}}_{\mathrm{p}} \tag{8}$$

By this, we can guarantee that the matrix $\boldsymbol{\Omega}_{\mathrm{p}}$ is s.psd, whereas the model only has to learn an arbitrary matrix $\tilde{\boldsymbol{\Omega}}_{\mathrm{p}}$ - which is much easier than making some restriction on the definiteness. Training usually takes place by optimizing suitable cost functions as regards prototype locations and metric parameters. For counterfactual reasoning, the specific training method [23, 24] is irrelevant and we refer to the final model only.

4 Counterfactual explanations of LVQ models

We aim for an efficient explicit formulation how to find counterfactuals, for diverse LVQ models.

4.1 General approach

Because a LVQ model assigns the label of the nearest prototype to a given input, we know that the nearest prototype of a counterfactual $\vec{x}'$ must be a prototype $\vec{\mathrm{p}}_i$ with $o_i = y^c$. In order to compute a counterfactual $\vec{x}'$ of a given input $\vec{x}$, it is sufficient to solve the following optimization problem for each prototype $\vec{\mathrm{p}}_i$ with $o_i = y^c$ and select the counterfactual $\vec{x}'$ yielding the smallest value of $\theta(\vec{x}', \vec{x})$:

$$\underset{\vec{x}' \in \mathbb{R}^d}{\arg\min} \ \theta(\vec{x}', \vec{x}) \tag{9a}$$

$$\text{s.t.} \ \mathrm{d}(\vec{x}', \vec{\mathrm{p}}_i) + \epsilon \leq \mathrm{d}(\vec{x}', \vec{\mathrm{p}}_j) \quad \forall \vec{\mathrm{p}}_j \in \mathcal{P}(y^c) \tag{9b}$$

where $\mathcal{P}(y^c)$ denotes the set of all prototypes <u>not</u> labeled as y^c and $\epsilon > 0$ is a small value preventing that the counterfactual lies exactly on the decision boundary. Note that the feasible region of Eq. (9) is always non-empty - the prototype $\vec{\mathrm{p}}_i$ is always a feasible solution. Furthermore, in contrast to Eq. (1), the formulation does not include hyperparameters.

The pseudocode for computing a counterfactual of a LVQ model is described in Algorithm 1. Note that the `for` loop in line 3 of Algorithm 1 can be easily parallelized.

For the weighted Manhattan distance as a regularization $\theta(\cdot)$, the objective Eq. (9a) becomes linear in $\vec{x}'$, where $\boldsymbol{\Upsilon}$ is the diagonal matrix with entries α_j and $\vec{\beta}$ is an auxiliary variable that can be discarded afterwards:

$$\begin{aligned} \underset{\vec{x}', \vec{\beta} \in \mathbb{R}^d}{\min} \ & \vec{1}^{\top} \vec{\beta} \\ \text{s.t.} \ & \boldsymbol{\Upsilon} \vec{x}' - \boldsymbol{\Upsilon} \vec{x} \leq \vec{\beta}, \quad -\boldsymbol{\Upsilon} \vec{x}' + \boldsymbol{\Upsilon} \vec{x} \leq \vec{\beta}, \quad \vec{\beta} \geq \vec{0} \end{aligned} \tag{10}$$

Algorithm 1 Computing a counterfactual of a LVQ model

Input: Original input $\vec{x}$, requested prediction y^c of the counterfactual, the LVQ model

Output: Counterfactual $\vec{x}'$

  1: $\vec{x}' = \vec{0}$ ▷ Initialize dummy solution
  2: $z = \infty$
  3: **for** $\vec{p}_i$ with $o_i = y^c$ **do**   ▷ Try each prototype with the correct label
  4:      Solving Eq. (9) yields a counterfactual $\vec{x}'_*$
  5:      **if** $\theta(\vec{x}'_*, \vec{x}) < z$ **then**   ▷ Keep this counterfactual if it deviates less from the original input then the currently "best" counterfactual
  6:           $z = \theta(\vec{x}'_*, \vec{x})$
  7:           $\vec{x}' = \vec{x}'_*$
  8:      **end if**
  9: **end for**

For the Euclidean distance Eq. (4) as regularization $\theta(\cdot)$, the objective Eq. (9a) can be written in a convex quadratic form:

$$\min_{\vec{x}' \in \mathbb{R}^d} \frac{1}{2} \vec{x}'^\top \vec{x}' - \vec{x}'^\top \vec{x} \tag{11}$$

In the subsequel we explore Eq. (9) for different regularizations $\theta(\cdot)$ and LVQ models, and investigate how to solve it efficiently. Note that for the purpose of better readability and due to space constraints, we put all derivations in the appendix (section 6).

4.2 (Generalized matrix) LVQ

When using (generalized matrix) LVQ models - no class or prototype specific distance matrices - the optimization problem Eq. (9) becomes a linear program (LP) when using the weighted Manhattan distance as a regularizer, and a convex quadratic program (QP) problem when using the Euclidean distance. Both programs can be solved efficiently and (up to equivalence) uniquely [28]. More precisely, the constraints Eq. (9b) can be written as a set of linear inequality constraints:

$$\vec{x}'^\top \vec{q}_{ij} + r_{ij} + \epsilon \leq 0 \quad \forall \vec{p}_j \in \mathcal{P}(y^c) \tag{12}$$

where

$$\vec{q}_{ij} = \mathbf{\Omega}(\vec{p}_j - \vec{p}_i) \qquad r_{ij} = \frac{1}{2}\left(\vec{p}_i^\top \mathbf{\Omega}\, \vec{p}_i - \vec{p}_j^\top \mathbf{\Omega}\, \vec{p}_j\right) \tag{13}$$

We set $\mathbf{\Omega} = \mathbb{I}$ if the LVQ model uses the Euclidean distance.

    Another benefit is that we can easily add additional convex quadratic constraints like box constraints, restricting the value range of linear interaction of features or freezing some features - these features are not allowed to be different in the counterfactual. We can use such additional constraints for computing a more plausible and feasible counterfactual.

4.3 (Localized generalized matrix) LVQ

In case of local matrix-LVQ models that learn a class or prototype specific distance matrix $\mathbf{\Omega}_{\mathrm{p}}$, the optimization problem Eq. (9) becomes a quadratically constrained quadratic program (QCQP) for Manhattan and Euclidean regularization, since the constraints Eq. (9b) become a set of quadratic constraints:

$$\frac{1}{2}\vec{x}'^{\top}\mathbf{Q}_{ij}\vec{x}' + \vec{x}'^{\top}\vec{q}_{ij} + r_{ij} + \epsilon \leq 0 \quad \forall \vec{\mathrm{p}}_j \in \mathcal{P}(y^c) \tag{14}$$

where

$$\mathbf{Q}_{ij} = \mathbf{\Omega}_i - \mathbf{\Omega}_j \tag{15}$$

Unfortunately, we can not make any statement about the definiteness of $\mathbf{Q}_{ij}$. Because $\mathbf{Q}_{ij}$ is the difference of two s.psd matrices, all we know is that it is a symmetric matrix. Therefore, Eq. (14) yields a non-convex QCQP and solving a non-convex QCQP is known to be NP-hard [29]. However, there exist methods like the Suggest-Improve framework [29] that can approximately solve a non-convex QCQP very efficiently - more details on how to apply this to Eq. (14) can be found in the appendix (section 6).

4.4 Experiments

We empirically confirm the efficiency of our proposed methods in comparison to black-box mechanisms by means of the following experiments: We use GLVQ, GMLVQ and LGMLVQ models with 3 prototypes per class for the "Breast Cancer Wisconsin (Diagnostic) Data Set" [30], the "Optical Recognition of Handwritten Digits Data Set" [31] and the "Ames Housing dataset" [32]. Thereby, we use PCA-preprocessing [2] to reduce the dimensionality of the digit data set to 10 and of the breast cancer data set to 5. We standardize the house data set and turned it into a binary classification problem[3] by setting the target to 1 if the price is greater or equal to 160k\$ and 0 otherwise. We use the Suggest-Improve framework [29] for solving the non-convex QCQPs, where we pick the target prototype as initial solution in the Suggest-step and we use the penalty convex-concave procedure (CCP) in the Improve-step [29]. For comparison, we use the optimizer for computing counterfactual explanations of LVQ models as implemented in ceml [33] - where the distance to the nearest prototype with the requested label y^c is minimized by Downhill-Simplex search or CMA-ES.

We report results for the Manhattan distance as regularizer - we used the Manhattan distance for enforcing a sparse solution. For each possible combination of model, data set and method, a 4-fold cross validation is conducted and

---

[2]This is for the purpose of better stability and better semantic meaning, since in the original domain already a small perturbation is sufficient for changing the class, since adversarial attacks exist even for linear functions in high dimensions if feature correlations are neglected. Since PCA can be approximately inverted, counterfactuals in PCA space can be lifted to the original data space.

[3]In addition, we select the following features: TotalBsmt, 1stFlr, 2ndFlr, GrLivA, WoodDeck, OpenP, 3SsnP, ScreenP and PoolA - When computing counterfactuals, we fix the last five features.

Data set	Breast cancer			Handwritten digits			House prices		
Method	DS	CMA	Ours	DS	CMA	Ours	DS	CMA	Ours
GLVQ	3.26	3.28	**1.96**	6.51	6.53	**3.99**	3.81	3.85	**3.32**
GMLVQ	2.71	6.49	**2.46**	21.34	11.63	**4.40**	5.06	8.63	**3.78**
LGMLVQ	*2.00*	*1.61*	**1.57**	*8.12*	*7.88*	**7.53**	*12.74*	*12.59*	**8.20**

Table 1: Mean Manhattan distance between the counterfactual and the original data point - best (smallest) values are **highlighted**. For LGMLVQ with DS or CMA-ES (marked italic), in 5% to 60% of the cases no solution was found.

the mean distance is reported. The results are listed in Table 1. In all cases, our method yields counterfactuals that are closer to the original data point than the one found by minimizing the original cost function Eq. (1) with Downhill-Simplex search (DS) or CMA-ES. In addition, our method is between 1.5 and 158.0 faster in comparison to DS/CMA-ES method. Furthermore, Downhill-Simplex and CMA-ES did not always find a counterfactual when dealing with LGMLVQ models. We would like to remark that our formulation can easily be extended by linear/quadratic constraints which can incorporate prior knowledge such as a maximum possible change of specific input features - see Table 2 for an example. Such extensions do not change the form of the optimization problem hence its complexity.

Data point	TotalBsmt	1stFlr	2ndFlr	GrLivA	Label
Original	0	1120	468	1588	1
Counterfactual	0	366	1824	2225	0
Constrained Counterfactual	373	1454	1454	3125	0

Table 2: House prices, we obtain a "plausible" counterfactual by adding constrains, here the constraint "2ndFlr $\leq$ 1stFlr" is added.

The implementation of our proposed method for computing counterfactual explanations is available online[4]. All experiments were implemented in Python 3.6 [34] using the packages cvxpy [35] cvx-qcqp [36], sklearn-lvq [37], numpy [38], scipy [39], scikit-learn [40] and ceml [33].

5 Conclusion

We proposed, and empirically evaluated, model- and regularization-dependent convex and non-convex programs for efficiently computing counterfactual explanations of LVQ models. We found that in many cases we get either a set of linear or convex quadratic programs which both can be solved efficiently. Only in the case of localized matrix-LVQ models we have to solve a set of non-convex quadratically constrained quadratic programs - we found that they can be efficiently approximately solved by using the Suggest-Improve framework.

---

[4]https://github.com/andreArtelt/efficient_computation_counterfactuals_lvq

6 Appendix

6.1 Minimizing the Euclidean distance

First, we expand the Euclidean distance Eq. (4):

$$
\begin{aligned}
\|\vec{x}' - \vec{x}\|_2^2 &= (\vec{x}' - \vec{x})^\top (\vec{x}' - \vec{x}) \\
&= \vec{x}'^\top \vec{x}' - \vec{x}'^\top \vec{x} - \vec{x}^\top \vec{x}' + \vec{x}^\top \vec{x} \\
&= \vec{x}'^\top \vec{x}' - 2\vec{x}^\top \vec{x}' + \vec{x}^\top \vec{x}
\end{aligned}
\tag{16}
$$

Next, we note that that we can drop the constant $\vec{x}^\top \vec{x}$ when optimizing with respect to $\vec{x}'$:

$$
\begin{aligned}
&\min_{\vec{x}' \in \mathbb{R}^d} \|\vec{x}' - \vec{x}\|_2^2 \\
&\Leftrightarrow \\
&\min_{\vec{x}' \in \mathbb{R}^d} \frac{1}{2} \vec{x}'^\top \vec{x}' - \vec{x}^\top \vec{x}'
\end{aligned}
\tag{17}
$$

6.2 Minimizing the weighted Manhattan distance

First, we transform the problem of minimizing the weighted Manhattan distance Eq. (2) into epigraph form:

$$
\begin{aligned}
&\min_{\vec{x}' \in \mathbb{R}^d} \sum_j \alpha_j \cdot |(\vec{x}')_j - (\vec{x})_j| \\
&\Leftrightarrow \\
&\min_{\vec{x}' \in \mathbb{R}^d, \beta \in \mathbb{R}} \beta \\
&\text{s.t.} \quad \sum_j \alpha_j \cdot |(\vec{x}')_j - (\vec{x})_j| \leq \beta \\
&\qquad \beta \geq 0
\end{aligned}
\tag{18}
$$

Next, we separate the dimensions:

$$
\begin{aligned}
&\min_{\vec{x}' \in \mathbb{R}^d, \beta \in \mathbb{R}} \beta \\
&\quad \text{s.t.} \quad \sum_j \alpha_j \cdot |(\vec{x}')_j - (\vec{x})_j| \leq \beta \\
&\qquad\quad \beta \geq 0 \\
&\Leftrightarrow \\
&\min_{\vec{x}', \vec{\beta} \in \mathbb{R}^d} \sum_j (\vec{\beta})_j \\
&\quad \text{s.t.} \quad \alpha_j \cdot |(\vec{x}')_j - (\vec{x})_j| \leq (\vec{\beta})_j \quad \forall j \\
&\qquad\quad (\vec{\beta})_j \geq 0 \quad \forall j
\end{aligned}
\tag{19}
$$

After that, we remove the absolute value function:

$$\min_{\vec{x}',\vec{\beta} \in \mathbb{R}^d} \sum_j (\vec{\beta})_j$$

$$\text{s.t.} \quad \alpha_j \cdot |(\vec{x}')_j - (\vec{x})_j| \leq (\vec{\beta})_j \quad \forall j$$

$$(\vec{\beta})_j \geq 0 \quad \forall j$$

$$\Leftrightarrow \tag{20}$$

$$\min_{\vec{x}',\vec{\beta} \in \mathbb{R}^d} \sum_j (\vec{\beta})_j$$

$$\text{s.t.} \quad \alpha_j (\vec{x}')_j - \alpha_j (\vec{x})_j \leq (\vec{\beta})_j \quad \forall j$$

$$- \alpha_j (\vec{x}')_j + \alpha_j (\vec{x})_j \leq (\vec{\beta})_j \quad \forall j$$

$$(\vec{\beta})_j \geq 0 \quad \forall j$$

Finally, we rewrite everything in matrix-vector notation:

$$\min_{\vec{x}',\vec{\beta} \in \mathbb{R}^d} \vec{1}^\top \vec{\beta}$$

$$\text{s.t.}$$

$$\boldsymbol{\Upsilon}\vec{x}' - \boldsymbol{\Upsilon}\vec{x} \leq \vec{\beta} \tag{21}$$

$$- \boldsymbol{\Upsilon}\vec{x}' + \boldsymbol{\Upsilon}\vec{x} \leq \vec{\beta}$$

$$\vec{\beta} \geq \vec{0}$$

where

$$\boldsymbol{\Upsilon} = \text{diag}(\alpha_j) \tag{22}$$

6.3 Enforcing a specific prototype as the nearest neighbor

By using the following set of strict inequalities, we can force the prototype $\vec{p}_i$ to be the nearest neighbor of the counterfactual $\vec{x}'$ - which would cause $\vec{x}'$ to be classified as o_i:

$$\text{d}(\vec{x}', \vec{p}_i) < \text{d}(\vec{x}', \vec{p}_j) \quad \forall \vec{p}_j \in \mathcal{P}(y^c) \tag{23}$$

We consider a fixed pair of i and j:

$$\mathrm{d}(\vec{x}', \vec{\mathrm{p}}_i) < \mathrm{d}(\vec{x}', \vec{\mathrm{p}}_j)$$
$$\Leftrightarrow \|\vec{x}' - \vec{\mathrm{p}}_i\|_{\mathbf{\Omega}_i}^2 < \|\vec{x}' - \vec{\mathrm{p}}_j\|_{\mathbf{\Omega}_j}^2$$
$$\Leftrightarrow (\vec{x}' - \vec{\mathrm{p}}_i)^\top \mathbf{\Omega}_i (\vec{x}' - \vec{\mathrm{p}}_i) < (\vec{x}' - \vec{\mathrm{p}}_j)^\top \mathbf{\Omega}_j (\vec{x}' - \vec{\mathrm{p}}_j)$$
$$\Leftrightarrow \vec{x}'^\top \mathbf{\Omega}_i \vec{x}' - 2\vec{x}'^\top \mathbf{\Omega}_i \vec{\mathrm{p}}_i + \vec{\mathrm{p}}_i^\top \mathbf{\Omega}_i \vec{\mathrm{p}}_i < \vec{x}'^\top \mathbf{\Omega}_j \vec{x}' - 2\vec{x}'^\top \mathbf{\Omega}_j \vec{\mathrm{p}}_j + \vec{\mathrm{p}}_j^\top \mathbf{\Omega}_i \vec{\mathrm{p}}_j$$
$$\Leftrightarrow \vec{x}'^\top \mathbf{\Omega}_i \vec{x}' - \vec{x}'^\top \mathbf{\Omega}_j \vec{x}' - 2\vec{x}'^\top \mathbf{\Omega}_i \vec{\mathrm{p}}_i + 2\vec{x}'^\top \mathbf{\Omega}_j \vec{\mathrm{p}}_j + \vec{\mathrm{p}}_i^\top \mathbf{\Omega}_i \vec{\mathrm{p}}_i - \vec{\mathrm{p}}_j^\top \mathbf{\Omega}_i \vec{\mathrm{p}}_j < 0 \quad (24)$$
$$\Leftrightarrow \vec{x}'^\top (\mathbf{\Omega}_i - \mathbf{\Omega}_j) \vec{x}' + \vec{x}'^\top (-2\mathbf{\Omega}_i \vec{\mathrm{p}}_i + 2\mathbf{\Omega}_j \vec{\mathrm{p}}_j) + (\vec{\mathrm{p}}_i^\top \mathbf{\Omega}_i \vec{\mathrm{p}}_i - \vec{\mathrm{p}}_j^\top \mathbf{\Omega}_i \vec{\mathrm{p}}_j) < 0$$
$$\Leftrightarrow \frac{1}{2}\vec{x}'^\top (\mathbf{\Omega}_i - \mathbf{\Omega}_j) \vec{x}' + \vec{x}'^\top (\mathbf{\Omega}_j \vec{\mathrm{p}}_j - \mathbf{\Omega}_i \vec{\mathrm{p}}_i) + \frac{1}{2}(\vec{\mathrm{p}}_i^\top \mathbf{\Omega}_i \vec{\mathrm{p}}_i - \vec{\mathrm{p}}_j^\top \mathbf{\Omega}_i \vec{\mathrm{p}}_j) < 0$$
$$\Leftrightarrow \frac{1}{2}\vec{x}'^\top \mathbf{Q}_{ij} \vec{x}' + \vec{x}'^\top \vec{q}_{ij} + r_{ij} < 0$$

where

$$\mathbf{Q}_{ij} = \mathbf{\Omega}_i - \mathbf{\Omega}_j \tag{25}$$

$$\vec{q}_{ij} = \mathbf{\Omega}_j \vec{\mathrm{p}}_j - \mathbf{\Omega}_i \vec{\mathrm{p}}_i \tag{26}$$

$$r_{ij} = \frac{1}{2}\left(\vec{\mathrm{p}}_i \top \mathbf{\Omega}_i \vec{\mathrm{p}}_i - \vec{\mathrm{p}}_j \top \mathbf{\Omega}_j \vec{\mathrm{p}}_j\right) \tag{27}$$

If we only have a global distance matrix $\mathbf{\Omega}$, we find that $\mathbf{Q}_{ij} = \mathbf{0}$ and the inequality Eq. (24) simplifies:

$$\begin{aligned} &\mathrm{d}(\vec{x}, \vec{\mathrm{p}}_i) < \mathrm{d}(\vec{x}, \vec{\mathrm{p}}_j) \\ &\Leftrightarrow \vec{x}'^\top \vec{q}_{ij} + r_{ij} < 0 \end{aligned} \tag{28}$$

where

$$\vec{q}_{ij} = \mathbf{\Omega}\left(\vec{\mathrm{p}}_j - \vec{\mathrm{p}}_i\right) \tag{29}$$

$$r_{ij} = \frac{1}{2}\left(\vec{\mathrm{p}}_i^\top \mathbf{\Omega} \vec{\mathrm{p}}_i - \vec{\mathrm{p}}_j^\top \mathbf{\Omega} \vec{\mathrm{p}}_j\right) \tag{30}$$

If we do not use a custom distance matrix, we have $\mathbf{\Omega} = \mathbb{I}$ and Eq. (24) becomes:

$$\begin{aligned} &\mathrm{d}(\vec{x}, \vec{\mathrm{p}}_i) < \mathrm{d}(\vec{x}, \vec{\mathrm{p}}_j) \\ &\Leftrightarrow \vec{x}'^\top \vec{q}_{ij} + r_{ij} < 0 \end{aligned} \tag{31}$$

where

$$\vec{q}_{ij} = \vec{\mathrm{p}}_j - \vec{\mathrm{p}}_i \tag{32}$$

$$r_{ij} = \frac{1}{2}\left(\vec{\mathrm{p}}_i^\top \vec{\mathrm{p}}_i - \vec{\mathrm{p}}_j^\top \vec{\mathrm{p}}_j\right) \tag{33}$$

6.4 Approximately solving the non-convex QCQP

Recall the non-convex quadratic constraint from Eq. (24):

$$\frac{1}{2}\vec{x}'^{\top}\mathbf{Q}_{ij}\vec{x}' + \vec{x}'^{\top}\vec{q}_{ij} + r_{ij} + \epsilon \leq 0 \tag{34}$$

where the matrix $\mathbf{Q}_{ij}$ is defined as the difference of two s.psd matrices:

$$\mathbf{Q}_{ij} = \mathbf{\Omega}_i - \mathbf{\Omega}_j \tag{35}$$

By making use of Eq. (35), we can rewrite Eq. (34) as:

$$\frac{1}{2}\vec{x}'^{\top}\mathbf{\Omega}_i\vec{x}' + \vec{x}'^{\top}\vec{q}_{ij} + r_{ij} + \epsilon - \frac{1}{2}\vec{x}'^{\top}\mathbf{\Omega}_j\vec{x}' \leq 0$$
$$\Leftrightarrow f(\vec{x}') - g(\vec{x}') \leq 0 \tag{36}$$

where

$$f(\vec{x}') = \frac{1}{2}\vec{x}'^{\top}\mathbf{\Omega}_i\vec{x}' + \vec{x}'^{\top}\vec{q}_{ij} + r_{ij} + \epsilon \tag{37}$$

$$g(\vec{x}') = \frac{1}{2}\vec{x}'^{\top}\mathbf{\Omega}_j\vec{x}' \tag{38}$$

Under the assumption that the regularization function $\theta(\cdot)$ is a convex function[5], we can rewrite a generic version of the non-convex QCQP (induced by Eq. (14)) as follows:

$$\min_{\vec{x}' \in \mathbb{R}^d} \theta(\vec{x}', \vec{x})$$
$$\text{s.t.} \tag{39}$$
$$f(\vec{x}') - g(\vec{x}') \leq 0$$

Because $\mathbf{\Omega}_i$ and $\mathbf{\Omega}_j$ are s.psd matrices, we know that $f(\vec{x}')$ and $g(\vec{x}')$ are convex functions. Therefore Eq. (39) is a difference-of-convex program (DCP).

This allows us to use the penalty convex-concave procedure (CCP) [29] for computing an approximate solution of Eq. (39) that is equivalent to the original non-convex QCQP induced by Eq. (14). For using the penalty CCP, we need the first order Taylor approximation of $g(\vec{x}')$ around a current point $\vec{x}_k$:

$$\hat{g}(\vec{x}')_{\vec{x}_k} = g(\vec{x}_k) + (\nabla_{\vec{x}'} g)(\vec{x}_k)^{\top}(\vec{x}' - \vec{x}_k)$$
$$= \frac{1}{2}\vec{x}_k^{\top}\mathbf{\Omega}_j\vec{x}_k + (\mathbf{\Omega}_j\vec{x}_k)^{\top}(\vec{x}' - \vec{x}_k)$$
$$= (\mathbf{\Omega}_j\vec{x}_k)^{\top}\vec{x}' + \frac{1}{2}\vec{x}_k^{\top}\mathbf{\Omega}_j\vec{x}_k - (\mathbf{\Omega}_j\vec{x}_k)^{\top}\vec{x}_k \tag{40}$$
$$= \vec{\rho}_{jk}^{\top}\vec{x}' + \tilde{r}_{jk}$$

where

$$\vec{\rho}_{jk} = \mathbf{\Omega}_j\vec{x}_k \tag{41}$$

$$\tilde{r}_{jk} = -\frac{1}{2}\vec{x}_k^{\top}\mathbf{\Omega}_j\vec{x}_k \tag{42}$$

---

[5]The weighted Manhattan distance and the Euclidean distance are convex functions!

References

[1] Sharad Goel, Justin M. Rao, and Ravi Shroff. Precinct or prejudice? understanding racial disparities in new york city's stop-and-frisk policy. 2016.

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

[3] Amir E. Khandani, Adlar J. Kim, and Andrew Lo. Consumer credit-risk models via machine-learning algorithms. Journal of Banking & Finance, 34(11):2767–2787, 2010.

[4] European parliament and council. General data protection regulation. `https://eur-lex.europa.eu/eli/reg/2016/679/oj`, 2016.

[5] Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana 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, pages 80–89, 2018.

[6] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. ACM Comput. Surv., 51(5):93:1–93:42, August 2018.

[7] Erico Tjoa and Cuntai Guan. A survey on explainable artificial intelligence (XAI): towards medical XAI. CoRR, abs/1907.07374, 2019.

[8] Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. CoRR, abs/1708.08296, 2017.

[9] Christoph Molnar. Interpretable Machine Learning. 2019. `https://christophm.github.io/interpretable-ml-book/`.

[10] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Model-agnostic interpretability of machine learning. In ICML Workshop on Human Interpretability in Machine Learning (WHI), 2016.

[11] Brandon M. Greenwell, Bradley C. Boehmke, and Andrew J. McCarthy. A simple and effective model-based variable importance measure. CoRR, abs/1805.04755, 2018.

[12] Aaron Fisher, Cynthia Rudin, and Francesca Dominici. All Models are Wrong but many are Useful: Variable Importance for Black-Box, Proprietary, or Misspecified Prediction Models, using Model Class Reliance. arXiv e-prints, page arXiv:1801.01489, Jan 2018.

[13] Qingyuan Zhao and Trevor Hastie. Causal interpretations of black-box models. Journal of Business & Economic Statistics, 0(ja):1–19, 2019.

[14] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, pages 1135–1144, New York, NY, USA, 2016. ACM.

[15] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Dino Pedreschi, Franco Turini, and Fosca Giannotti. Local rule-based explanations of black box decision systems. CoRR, abs/1805.10820, 2018.

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

[17] Been Kim, Oluwasanmi Koyejo, and Rajiv Khanna. Examples are not enough, learn to criticize! criticism for interpretability. In Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain, pages 2280–2288, 2016.

[18] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017, pages 1885–1894, 2017.

[19] Sandra Wachter, Brent D. Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. CoRR, abs/1711.00399, 2017.

[20] Shubham Sharma, Jette Henderson, and Joydeep Ghosh. CERTIFAI: counterfactual explanations for robustness, transparency, interpretability, and fairness of artificial intelligence models. CoRR, abs/1905.07857, 2019.

[21] Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. Comparison-based inverse classification for interpretability in machine learning. In Information Processing and Management of Uncertainty in Knowledge-Based Systems. Theory and Foundations - 17th International Conference, IPMU 2018, Cádiz, Spain, June 11-15, 2018, Proceedings, Part I, pages 100–111, 2018.

[22] David Nova and Pablo A. Estévez. A review of learning vector quantization classifiers. Neural Comput. Appl., 25(3-4):511–524, September 2014.

[23] Petra Schneider, Michael Biehl, and Barbara Hammer. Adaptive relevance matrices in learning vector quantization. Neural Computation, 21(12):3532–3561, 2009. PMID: 19764875.

[24] Petra Schneider, Michael Biehl, and Barbara Hammer. Distance learning in discriminative vector quantization. Neural Computation, 21(10):2942–2969, 2009. PMID: 19635012.

[25] Stephan Kirstein, Heiko Wersing, Horst-Michael Gross, and Edgar Körner. 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, 04 2012.

[26] Arnaud Van Looveren and Janis Klaise. Interpretable counterfactual explanations guided by prototypes. CoRR, abs/1907.02584, 2019.

[27] Rafael Poyiadzi, Kacper Sokol, Raúl Santos-Rodriguez, Tijl De Bie, and Peter A. Flach. FACE: feasible and actionable counterfactual explanations. CoRR, abs/1909.09369, 2019.

[28] Stephen Boyd and Lieven Vandenberghe. Convex Optimization. Cambridge University Press, New York, NY, USA, 2004.

[29] Jaehyun Park and Stephen Boyd. General heuristics for nonconvex quadratically constrained quadratic programming. arXiv preprint arXiv:1703.07870, 2017.

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

[31] 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.

[32] Dean De Cock. Ames, iowa: Alternative to the boston housing data as an end of semester regression project. Journal of Statistics Education, 19(3), 2011.

[33] André Artelt. Ceml: Counterfactuals for explaining machine learning models - a python toolbox. `https://www.github.com/andreArtelt/ceml`, 2019.

[34] Guido Van Rossum and Fred L Drake Jr. Python tutorial. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.

[35] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. Journal of Machine Learning Research, 17(83):1–5, 2016.

[36] Jaehyun Park and Stephen Boyd. A cvxpy extension for handling nonconvex qcqp via suggest-and-improve framework. `https://github.com/cvxgrp/qcqp`, 2017.

[37] Joris Jensen. sklearn-lvq. `https://github.com/MrNuggelz/sklearn-lvq`, 2017.

[38] Stéfan van der Walt, S. Chris Colbert, and Gaël Varoquaux. The numpy array: A structure for efficient numerical computation. Computing in Science and Engineering, 13(2):22–30, 2011.

[39] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–.

[40] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830, 2011.