Efficient computation of counterfactual explanations of LVQ models

André Artelt[*] and Barbara Hammer

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

Abstract. With the increasing use of machine learning in practice and because of legal regulations like EU's GDPR, it becomes indispensable to be able to explain the prediction and behavior of machine learning models. An example of easy to understand explanations of AI models are counterfactual explanations. However, for many models it is still an open research problem how to efficiently compute counterfactual explanations.

We investigate how to efficiently compute counterfactual explanations of learning vector quantization models. In particular, we propose different types of convex and non-convex programs depending on the used learning vector quantization model.

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 [1–3]. In order to be accepted by the user and the society in general, it becomes important to be able to explain the output and behavior of ML models. Especially because of legal regulations like the EU regulation "General Data Protection Right" (GDPR) [4], that contains a "right to an explanation", it is nowadays indispensable to explain the output and behavior of artificial intelligence (AI) in a comprehensible way.

Recently the research community got attracted to explainability and transparency in machine learning [5–8]. There exist some methods for explaining and understanding ML models [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].

---

[*]corresponding author: aartelt@techfak.uni-bielefeld.de

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].

Model agnostic methods are universal in the sense that they are applicable to all models. However, usually model agnostic methods are computational expensive because they have to make many predictions using the model. Thus, model specific methods are relevant because they can be computed more efficiently. Therefore, it is desirable to have model specific techniques for computing explanations (e.g. contraining the architecture of a neural network [20]).

Learning vector quantization (LVQ) models [21] are prototype based models that represent data by a set of prototypes. LVQ models can be combined with metric learning and thereby increase the effectiveness of the model in case of few prototypes [22, 23]. Furthermore, LVQ models can be used for incremental learning [24] and life-long learning [25].

Therefore, it is highly relevant to be able to efficiently compute counterfactual explanations of LVQ models. In this paper we investigate how to efficiently compute counterfactual explanations of LVQ models by exploiting the special structure of LVQ models. More precisely, our contributions are:

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

- We empirically evaluate our methods and compare them with traditional methods.

The remainder of this paper is structured as follows: First, we briefly review counterfactual explanations (section 2.1) and learning vector quantization models (section 2.2). Then, in section 3 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 5). Furthermore, we empirically evaluate the efficiency of our methods. Finally, section 4 summarizes the results of this paper.

2 Review

2.1 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/dimensions of a given input such that the resulting data point (called counterfactual) has a different (specified) prediction than the original input. Using a counterfactual instance for explaining the prediction of the original input 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].

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.

Another disadvantage of counterfactuals is that there does not exist a general and efficient method for computing counterfactuals of categorical inputs - there are first attempts for bag-of-words models [26] but these methods work on a particular model in a particular domain only. However, recently CERTIFAI [27] was proposed. In CERTIFAI, a genetic algorithm is used for computing counterfactuals - a benefit of using a genetic algorithm is that we can deal with categorical variables and we can add domain knowledge about feature (interaction) constraints, whereas the downside is time complexity and complex hyperparameter tuning (e.g. size of the population, how to do a cross over, mutation, ...).

An alternative - but very similar in the spirit - to counterfactuals [19] is the Growing Spheres method [28]. 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.

Now, we first formally define the problem of finding counterfactuals in general form: Assume a prediction function h is given. Computing a counterfactual $\vec{x}' \in \mathbb{R}^d$ of a given input $\vec{x} \in \mathbb{R}^d$[1] can be interpreted as an optimization problem [19]:

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

where $\ell()$ denotes a loss function that penalizes deviation of the prediction $h(\vec{x}')$ from the requested prediction y'. $\theta()$ denotes a regularization that penalizes deviations from the original input $\vec{x}$. The hyperparameter C denotes the regularization strength.

Two common regularizations are the weighted Manhattan distance and the

---

[1]We restrict ourself to $\mathbb{R}^d$, but in theory one could use an arbitrary domain $\mathcal{X}$.

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 $\mathcal{D}$:

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

where $\tag{3}$

$$\mathrm{MAD}_j = \operatorname*{median}_{\vec{x} \in \mathcal{D}} \left(\left| (\vec{x})_j - \operatorname*{median}_{\vec{x} \in \mathcal{D}} \big((\vec{x})_j \big) \right| \right)$$

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[2] is defined as

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

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

Depending on the model and the choice of $\ell()$ and $\theta()$, 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, the best choice is to use a model and regularization specific optimizer. Unfortunately, there exit model specific optimizers for few ML models only.

2.2 Learning vector quantization

In learning vector quantization (LVQ) models [21] 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 label of the nearest prototype:

$$\begin{aligned} h(\vec{x}) &= o_i \\ \text{s.t. } &\min \mathrm{d}(\vec{x}, \vec{p}_i) \end{aligned} \tag{5}$$

where $\mathrm{d}()$ denotes a function for computing the distance between a data point and a prototype - usually this is the Euclidean distance[3]:

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

---

[2]also called Mahalanobis distance
[3]However, other distance functions are possible.

There exist LVQ models like GMLVQ, LGMLVQ [22], MRSLVQ and LMRSLVQ [23] that learn a custom (class or prototype specific) distance matrix $\mathbf{\Omega}_{\mathrm{p}}$ 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:

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

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

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

By this, we can guarantee that the matrix $\mathbf{\Omega}_{\mathrm{p}}$ is s.p.s.d., whereas the model only has to learn an arbitrary matrix $\tilde{\mathbf{\Omega}}_{\mathrm{p}}$ - 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 is irrelevant and we refer to the final model only.

3 Counterfactual explanations of LVQ models

We aim for an efficient explicit formulation how to find counterfactuals, given an input $\vec{x}$ and a specific LVQ model.

3.1 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 $o_i = y'$. 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 $o_i = y'$ and select the counterfactual $\vec{x}'$ yielding the smallest value of $\theta(\vec{x}', \vec{x})$:

$$\begin{aligned} &\underset{\vec{x}' \in \mathbb{R}^d}{\arg\min} \ \theta(\vec{x}', \vec{x}) \\ &\text{s.t.} \\ &\mathrm{d}(\vec{x}', \vec{\mathrm{p}}_i) < \mathrm{d}(\vec{x}', \vec{\mathrm{p}}_j) \quad \forall \vec{\mathrm{p}}_j \in \mathcal{P}(y') \end{aligned} \tag{9}$$

where $\mathcal{P}(y')$ denotes the set of all prototypes <u>not</u> labeled as y'. Note that the feasible region of Eq. 9 is always non-empty - the prototype $\vec{\mathrm{p}}_i$ is always a feasible solution.

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. Furthermore and in contrast to Eq. 1, we do not have any hyperparameters that need to be chosen.

In the subsequent sections we explore Eq. 9 for different regularizations $\theta()$

Algorithm 1 Computing a counterfactual of a LVQ model

Input: Original input $\vec{x}$, requested prediction y' 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 $o_i = y'$ **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**

---

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 5).

3.2 (Generalized matrix) LVQ

When using (generalized matrix) LVQ models - no class or prototype specific distance matrices - the optimization problem Eq. 9 becomes either a linear program (LP) or a convex quadratic program (QP).

When using the weighted Manhattan distance Eq. 2 as a regularization $\theta()$, Eq. 9 can be written as a LP:

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

$$\text{s.t.}$$
$$\mathbf{\Upsilon}\vec{x}' - \mathbf{\Upsilon}\vec{x} \le \vec{\beta}$$
$$-\mathbf{\Upsilon}\vec{x}' + \mathbf{\Upsilon}\vec{x} \le \vec{\beta} \tag{10}$$
$$\vec{x}'^\top \vec{q}_{ij} + r_{ij} < 0 \quad \forall \vec{p}_j \in \mathcal{P}(y')$$
$$\vec{\beta} \ge \vec{0}$$

where

$$\mathbf{\Upsilon} = \text{diag}(\alpha_j) \tag{11}$$

$$\vec{q}_{ij} = \frac{1}{2}\,\mathbf{\Omega}(\vec{p}_j - \vec{p}_i) \tag{12}$$

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

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

When using the Euclidean distance Eq. 4 as a regularization $\theta()$, Eq. 9 can be written as a convex quadratic program with linear constraints:

$$
\begin{aligned}
&\min_{\vec{x}' \in \mathbb{R}^d} \frac{1}{2}\vec{x}'^\top \vec{x}' - \vec{x}'^\top \vec{x} \\
&\text{s.t.} \\
&\vec{x}'^\top \vec{q}_{ij} + r_{ij} < 0 \quad \forall \vec{p}_j \in \mathcal{P}(y')
\end{aligned}
\tag{14}
$$

Note that q_{ij} and r_{ij} are the same as in Eq. 10.

Eq. 10 and Eq. 14 both contain strict inequalities $\vec{x}'^\top \vec{\gamma}_{ij} + c_{ij} < 0$. Unfortunately, strict inequalities are not allowed in linear/quadratic programming because the feasible region would become an open set. However, we could turn the $<$ into a $\leq$ by subtract a small number ϵ from the right side of the inequality. In practice, when implementing our methods, we found that we can safely replace all $<$ by $\leq$ without changing anything else - this might be because of the numerics (like round-off errors) of fixed size floating-point numbers.

Both a linear and a convex quadratic program can be solved efficiently [29]. Another benefit is that we can easily add additional linear 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.

3.3 (Localized generalized matrix) LVQ

In case of 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).

When using the weighted Manhattan distance Eq. 2 as a regularization $\theta()$, the optimization problem Eq. 9 becomes:

$$
\begin{aligned}
&\min_{\vec{x}', \vec{\beta} \in \mathbb{R}^d} \vec{1}^\top \vec{\beta} \\
&\text{s.t.} \\
&\mathbf{\Upsilon}\vec{x}' - \mathbf{\Upsilon}\vec{x} \leq \vec{\beta} \\
&-\mathbf{\Upsilon}\vec{x}' + \mathbf{\Upsilon}\vec{x} \leq \vec{\beta} \\
&\frac{1}{2}\vec{x}'^\top \mathbf{Q}_{ij}\vec{x}' + \vec{x}'^\top \vec{q}_{ij} + r_{ij} < 0 \quad \forall \vec{p}_j \in \mathcal{P}(y') \\
&\vec{\beta} \geq \vec{0}
\end{aligned}
\tag{15}
$$

where

$$
\mathbf{Q}_{ij} = \mathbf{\Omega}_i - \mathbf{\Omega}_j
\tag{16}
$$

$$
\vec{q}_{ij} = \frac{1}{2}\left(\mathbf{\Omega}_j \vec{p}_j - \mathbf{\Omega}_i \vec{p}_i\right)
\tag{17}
$$

$$r_{ij} = \frac{1}{2} \left(\vec{\mathrm{p}}_i^\top \boldsymbol{\Omega}_i \vec{\mathrm{p}}_i - \vec{\mathrm{p}}_j^\top \boldsymbol{\Omega}_j \vec{\mathrm{p}}_j \right) \tag{18}$$

When using the Euclidean distance Eq. 4 as a regularization $\theta()$, Eq. 9 can be written as:

$$
\begin{aligned}
&\min_{\vec{x}' \in \mathbb{R}^d} \frac{1}{2} \vec{x}'^\top \vec{x}' - \vec{x}'^\top \vec{x} \\
&\text{s.t.} \\
&\frac{1}{2} \vec{x}'^\top \mathbf{Q}_{ij} \vec{x}' + \vec{x}'^\top \vec{q}_{ij} + r_{ij} < 0 \quad \forall \vec{\mathrm{p}}_j \in \mathcal{P}(y')
\end{aligned}
\tag{19}
$$

Unfortunately, we can not make any statement about the definiteness of $\mathbf{Q}_{ij}$. Because $\mathbf{Q}_{ij}$ is the difference of two s.p.s.d. matrices, all we know is that it is a symmetric matrix. Therefore, Eq. 15 and Eq. 19 are both non-convex QCQPs and solving a non-convex QCQP is known to be NP-hard [30]. However, there exist methods like the Suggest-Improve framework [30] that can approximately solve a non-convex QCQP very efficiently - more details on how to apply this to Eq. 15 and Eq. 19 can be found in the appendix (section 5).

3.4 Experiments

In order to empirically confirm the efficiency of our proposed methods, we conducted some experiments:

We fitted GLVQ, GMLVQ and LGMLVQ models - we always used 3 prototypes per class - to the "Breast Cancer Wisconsin (Diagnostic) Data Set" [31] and the "Optical Recognition of Handwritten Digits Data Set" [32]. We used PCA to reduce the dimensionality of the "Optical Recognition of Handwritten Digits Data Set" to 10 and of the "Breast Cancer Wisconsin (Diagnostic) Data Set" to 5.

We implemented our proposed methods (Algorithm 1) for computing counterfactual explanations[4]. We used the SCS solver [33,34] for solving linear/quadratic programs and the Suggest-Improve framework [30] for approximatly solving non-convex QCQPs - we simply picked the target prototype as an initial feasible solution in the Suggest-step and used the penalty convex-concave procedure (CCP)[5] in the Improve-step [30, 35].
For comparison, we used the optimizer for computing counterfactual explanations of LVQ models as implemented in ceml [36] - the optimization problem Eq. 1, where the loss function is the distance to the nearest prototype with the requested label y', is minimized by the Downhill-Simplex algorithm. In all cases (including our methods), we used the Manhattan distance as a regularizer.

For each possible combination of model, data set and method, we did a 4-fold cross validation and recorded for each sample the computation time as well as the Manhattan distance between the counterfactual and the original data point.

---

[4] Available on GitHub `https://github.com/andreArtelt/efficient_computation_counterfactuals_lvq`

[5] As a subroutine, we used the MOSEK solver - `https://www.mosek.com`. We gratefully acknowledge the academic license provided by MOSEK ApS.

The results are listed in Table 1 and Table 2.

Data set	Breast cancer		Handwritten digits	
Model \Method	Black-box	Ours	Black-box	Ours
GLVQ	3.26	**1.96**	6.51	**3.99**
GMLVQ	2.71	**2.46**	21.34	**4.40**
LGMLVQ	2.00	**1.57**	8.12	**7.53**

Table 1: Mean <u>Manhattan distance</u> between the counterfactual and the original data point - best (smallest) values are **highlighted**.

Data set	Breast cancer		Handwritten digits	
Model \Method	Black-box	Ours	Black-box	Ours
GLVQ	0.95	**0.01**	3.15	**0.02**
GMLVQ	0.25	**0.01**	2.12	**0.02**
LGMLVQ	0.91	**0.65**	3.35	**2.09**

Table 2: Mean <u>computation time</u> in s - best (smallest) values are **highlighted**.

In all cases, our method yields counterfactuals that are closer to the original data point and is always (much) faster than optimizing the original cost function Eq. 1 by using the Downhill-Simplex algorithm.

All experiments were implemented in Python 3.6 [37] using the packages cvxpy [38] cvx-qcqp [39], sklearn-lvq [40], numpy [41], scipy [42], scikit-learn [43] and ceml [36].

4 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 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.

5 Appendix

5.1 Minimizing the Euclidean distance

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

$$
\begin{aligned}
\|\vec{x}' - \vec{x}\|_2^2 &= (\vec{x}' - \vec{x})^\top (\vec{x}' - \vec{x}) \\
&= \vec{x}'^\top \vec{x}' - \vec{x}'^\top \vec{x} - \vec{x}^\top \vec{x}' + \vec{x}^\top \vec{x} \\
&= \vec{x}'^\top \vec{x}' - 2\vec{x}\top\vec{x}' + \vec{x}^\top \vec{x}
\end{aligned}
\tag{20}
$$

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

$$
\begin{aligned}
&\min_{\vec{x}' \in \mathbb{R}^d} \|\vec{x}' - \vec{x}\|_2^2 \\
&\Leftrightarrow \\
&\min_{\vec{x}' \in \mathbb{R}^d} \frac{1}{2} \vec{x}'^\top \vec{x}' - \vec{x}^\top \vec{x}'
\end{aligned}
\tag{21}
$$

5.2 Minimizing the weighted Manhattan distance

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

$$
\begin{aligned}
&\min_{\vec{x}' \in \mathbb{R}^d} \sum_j \alpha_j \cdot |(\vec{x}')_j - (\vec{x})_j| \\
&\Leftrightarrow \\
&\min_{\vec{x}' \in \mathbb{R}^d, \beta \in \mathbb{R}} \beta \\
&\text{s.t.} \quad \sum_j \alpha_j \cdot |(\vec{x}')_j - (\vec{x})_j| \leq \beta \\
&\qquad \beta \geq 0
\end{aligned}
\tag{22}
$$

Next, we separate the dimensions:

$$
\begin{aligned}
&\min_{\vec{x}' \in \mathbb{R}^d, \beta \in \mathbb{R}} \beta \\
&\text{s.t.} \quad \sum_j \alpha_j \cdot |(\vec{x}')_j - (\vec{x})_j| \leq \beta \\
&\qquad \beta \geq 0 \\
&\Leftrightarrow \\
&\min_{\vec{x}', \vec{\beta} \in \mathbb{R}^d} \sum_j (\vec{\beta})_j \\
&\text{s.t.} \quad \alpha_j \cdot |(\vec{x}')_j - (\vec{x})_j| \leq (\vec{\beta})_j \quad \forall j \\
&\qquad (\vec{\beta})_j \geq 0 \quad \forall j
\end{aligned}
\tag{23}
$$

After that, we remove the absolute value function:

$$
\begin{aligned}
&\min_{\vec{x}', \vec{\beta} \in \mathbb{R}^d} \quad \sum_j (\vec{\beta})_j \\
&\text{s.t.} \quad \alpha_j \cdot |(\vec{x}')_j - (\vec{x})_j| \leq (\vec{\beta})_j \quad \forall j \\
&\qquad\quad (\vec{\beta})_j \geq 0 \quad \forall j \\
&\Leftrightarrow \\
&\min_{\vec{x}', \vec{\beta} \in \mathbb{R}^d} \quad \sum_j (\vec{\beta})_j \\
&\text{s.t.} \quad \alpha_j (\vec{x}')_j - \alpha_j (\vec{x})_j \leq (\vec{\beta})_j \quad \forall j \\
&\qquad\quad - \alpha_j (\vec{x}')_j + \alpha_j (\vec{x})_j \leq (\vec{\beta})_j \quad \forall j \\
&\qquad\quad (\vec{\beta})_j \geq 0 \quad \forall j
\end{aligned}
\tag{24}
$$

Finally, we rewrite everything in matrix-vector notation:

$$
\begin{aligned}
&\min_{\vec{x}', \vec{\beta} \in \mathbb{R}^d} \quad \vec{1}^\top \vec{\beta} \\
&\text{s.t.} \\
&\mathbf{\Upsilon} \vec{x}' - \mathbf{\Upsilon} \vec{x} \leq \vec{\beta} \\
&- \mathbf{\Upsilon} \vec{x}' + \mathbf{\Upsilon} \vec{x} \leq \vec{\beta} \\
&\vec{\beta} \geq \vec{0}
\end{aligned}
\tag{25}
$$

where

$$
\mathbf{\Upsilon} = \mathrm{diag}(\alpha_j)
\tag{26}
$$

5.3 Enforcing a specific prototype as the nearest neighbor

By using the following set of 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:

$$
\mathrm{d}(\vec{x}', \vec{p}_i) < \mathrm{d}(\vec{x}', \vec{p}_j) \quad \forall \vec{p}_j \in \mathcal{P}(y')
\tag{27}
$$

We consider a fixed pair of i and j:

$$\mathrm{d}(\vec{x}', \vec{\mathrm{p}}_i) < \mathrm{d}(\vec{x}', \vec{\mathrm{p}}_j)$$
$$\Leftrightarrow \|\vec{x}' - \vec{\mathrm{p}}_i\|^2_{\mathbf{\Omega}_i} < \|\vec{x}' - \vec{\mathrm{p}}_j\|^2_{\mathbf{\Omega}_j}$$
$$\Leftrightarrow (\vec{x}' - \vec{\mathrm{p}}_i)^\top \mathbf{\Omega}_i(\vec{x}' - \vec{\mathrm{p}}_i) < (\vec{x}' - \vec{\mathrm{p}}_j)^\top \mathbf{\Omega}_j(\vec{x}' - \vec{\mathrm{p}}_j)$$
$$\Leftrightarrow \vec{x}'^\top \mathbf{\Omega}_i \vec{x}' - 2\vec{x}'^\top \mathbf{\Omega}_i \vec{\mathrm{p}}_i + \vec{\mathrm{p}}_i^\top \mathbf{\Omega}_i \vec{\mathrm{p}}_i < \vec{x}'^\top \mathbf{\Omega}_j \vec{x}' - 2\vec{x}'^\top \mathbf{\Omega}_j \vec{\mathrm{p}}_j + \vec{\mathrm{p}}_j^\top \mathbf{\Omega}_i \vec{\mathrm{p}}_j$$
$$\Leftrightarrow \vec{x}'^\top \mathbf{\Omega}_i \vec{x}' - \vec{x}'^\top \mathbf{\Omega}_j \vec{x}' - 2\vec{x}'^\top \mathbf{\Omega}_i \vec{\mathrm{p}}_i + 2\vec{x}'^\top \mathbf{\Omega}_j \vec{\mathrm{p}}_j + \vec{\mathrm{p}}_i^\top \mathbf{\Omega}_i \vec{\mathrm{p}}_i - \vec{\mathrm{p}}_j^\top \mathbf{\Omega}_i \vec{\mathrm{p}}_j < 0 \quad (28)$$
$$\Leftrightarrow \vec{x}'^\top (\mathbf{\Omega}_i - \mathbf{\Omega}_j)\vec{x}' + \vec{x}'^\top (-2\mathbf{\Omega}_i \vec{\mathrm{p}}_i + 2\mathbf{\Omega}_j \vec{\mathrm{p}}_j) + (\vec{\mathrm{p}}_i^\top \mathbf{\Omega}_i \vec{\mathrm{p}}_i - \vec{\mathrm{p}}_j^\top \mathbf{\Omega}_i \vec{\mathrm{p}}_j)$$
$$\Leftrightarrow \frac{1}{2}\vec{x}'^\top (\mathbf{\Omega}_i - \mathbf{\Omega}_j)\vec{x}' + \frac{1}{2}\vec{x}'^\top (\mathbf{\Omega}_j \vec{\mathrm{p}}_j - \mathbf{\Omega}_i \vec{\mathrm{p}}_i) + \frac{1}{2}(\vec{\mathrm{p}}_i^\top \mathbf{\Omega}_i \vec{\mathrm{p}}_i - \vec{\mathrm{p}}_j^\top \mathbf{\Omega}_i \vec{\mathrm{p}}_j) < 0$$
$$\Leftrightarrow \frac{1}{2}\vec{x}'^\top \mathbf{Q}_{ij}\vec{x}' + \vec{x}'^\top \vec{q}_{ij} + r_{ij} < 0$$

where

$$\mathbf{Q}_{ij} = \mathbf{\Omega}_i - \mathbf{\Omega}_j \tag{29}$$

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

$$r_{ij} = \frac{1}{2}\left(\vec{\mathrm{p}}_i \top \mathbf{\Omega}_i \vec{\mathrm{p}}_i - \vec{\mathrm{p}}_j \top \mathbf{\Omega}_j \vec{\mathrm{p}}_j\right) \tag{31}$$

If we only have one global distance matrix $\mathbf{\Omega}$, we find that $\mathbf{Q}_{ij} = \mathbf{0}$ and the inequality Eq. 28 simplifies:

$$\mathrm{d}(\vec{x}, \vec{\mathrm{p}}_i) < \mathrm{d}(\vec{x}, \vec{\mathrm{p}}_j)$$
$$\Leftrightarrow \vec{x}'^\top \vec{q}_{ij} + r_{ij} < 0 \tag{32}$$

where

$$\vec{q}_{ij} = \frac{1}{2}\mathbf{\Omega}\left(\vec{\mathrm{p}}_j - \vec{\mathrm{p}}_i\right) \tag{33}$$

$$r_{ij} = \frac{1}{2}\left(\vec{\mathrm{p}}_i^\top \mathbf{\Omega} \vec{\mathrm{p}}_i - \vec{\mathrm{p}}_j^\top \mathbf{\Omega} \vec{\mathrm{p}}_j\right) \tag{34}$$

If we do not use a custom distance matrix, we have $\mathbf{\Omega} = \mathbb{I}$ and Eq. 28 becomes:

$$\mathrm{d}(\vec{x}, \vec{\mathrm{p}}_i) < \mathrm{d}(\vec{x}, \vec{\mathrm{p}}_j)$$
$$\Leftrightarrow \vec{x}'^\top \vec{q}_{ij} + r_{ij} < 0 \tag{35}$$

where

$$\vec{q}_{ij} = \frac{1}{2}\left(\vec{\mathrm{p}}_j - \vec{\mathrm{p}}_i\right) \tag{36}$$

$$r_{ij} = \frac{1}{2}\left(\vec{\mathrm{p}}_i^\top \vec{\mathrm{p}}_i - \vec{\mathrm{p}}_j^\top \vec{\mathrm{p}}_j\right) \tag{37}$$

5.4 Approximately solving the non-convex QCQP

Recall the non-convex quadratic constraint from Eq. 28:

$$\frac{1}{2}\vec{x}'^{\top}\mathbf{Q}_{ij}\vec{x}' + \vec{x}'^{\top}\vec{q}_{ij} + r_{ij} < 0 \tag{38}$$

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

$$\mathbf{Q}_{ij} = \mathbf{\Omega}_i - \mathbf{\Omega}_j \tag{39}$$

By making use of Eq. 39, we can rewrite Eq. 38 as:

$$\frac{1}{2}\vec{x}'^{\top}\mathbf{\Omega}_i\vec{x}' + \vec{x}'^{\top}\vec{q}_{ij} + r_{ij} - \frac{1}{2}\vec{x}'^{\top}\mathbf{\Omega}_j\vec{x}' < 0$$
$$\Leftrightarrow f(\vec{x}') - g(\vec{x}') < 0 \tag{40}$$

where

$$f(\vec{x}') = \frac{1}{2}\vec{x}'^{\top}\mathbf{\Omega}_i\vec{x}' + \vec{x}'^{\top}\vec{q}_{ij} + r_{ij} \tag{41}$$

$$g(\vec{x}') = \frac{1}{2}\vec{x}'^{\top}\mathbf{\Omega}_j\vec{x}' \tag{42}$$

Under the assumption that our regularization function $\theta()$ is a convex function[6], we can rewrite a generic version of the non-convex QCQP (Eq. 15 and Eq. 19) as follows:

$$\min_{\vec{x}' \in \mathbb{R}^d} \; \theta(\vec{x}', \vec{x})$$
$$\text{s.t.} \tag{43}$$
$$f(\vec{x}') - g(\vec{x}') \leq 0$$

Note that we relaxed the strict inequality. Because $\mathbf{\Omega}_i$ and $\mathbf{\Omega}_j$ are s.p.s.d. matrices, we know that $f(\vec{x}')$ and $g(\vec{x}')$ are convex functions. Therefore, Eq. 43 is a difference-of-convex program (DCP).

This allows us to use the penalty convex-concave procedure (CCP) [30] for computing an approximate solution of Eq. 43 that is equivalent to the original non-convex QCQPs Eq. 15 and Eq. 19. For using the penalty CCP, we need the first order Taylor approximation of $g(\vec{x}')$ around a current point $\vec{x}_k$:

$$\begin{aligned}
\hat{g}(\vec{x}')_{\vec{x}_k} &= g(\vec{x}_k) + (\nabla_{\vec{x}'} g)(\vec{x}_k)^{\top}(\vec{x}' - \vec{x}_k) \\
&= \frac{1}{2}\vec{x}_k^{\top}\mathbf{\Omega}_j\vec{x}_k + (\mathbf{\Omega}_j\vec{x}_k)^{\top}(\vec{x}' - \vec{x}_k) \\
&= (\mathbf{\Omega}_j\vec{x}_k)^{\top}\vec{x}' + \frac{1}{2}\vec{x}_k^{\top}\mathbf{\Omega}_j\vec{x}_k - (\mathbf{\Omega}_j\vec{x}_k)^{\top}\vec{x}_k \\
&= \vec{\rho}_{jk}^{\top}\vec{x}' + \tilde{r}_{jk}
\end{aligned} \tag{44}$$

where

$$\vec{\rho}_{jk} = \mathbf{\Omega}_j\vec{x}_k \tag{45}$$

$$\tilde{r}_{jk} = -\frac{1}{2}\vec{x}_k^{\top}\mathbf{\Omega}_j\vec{x}_k \tag{46}$$

---

[6]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. Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (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] Zebin Yang, Aijun Zhang, and Agus Sudjianto. Enhancing explainability of neural networks through architecture constraints. CoRR, abs/1901.03838, 2019.

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

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

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

[24] Ye Xu, Shen Furao, Osamu Hasegawa, and Jinxi Zhao. An online incremental learning vector quantization. In Advances in Knowledge Discovery and Data Mining, 13th Pacific-Asia Conference, PAKDD 2009, Bangkok, Thailand, April 27-30, 2009, Proceedings, pages 1046–1053, 2009.

[25] Stephan Kirstein, Heiko Wersing, Horst-Michael Gross, and Edgar Körner. A lifelong 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] David Martens and Foster Provost. Explaining data-driven document classifications. MIS Q., 38(1):73–100, March 2014.

[27] 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.

[28] 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.

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

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

[31] 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.

[32] 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.

[33] B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd. Conic optimization via operator splitting and homogeneous self-dual embedding. Journal of Optimization Theory and Applications, 169(3):1042–1068, June 2016.

[34] B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd. SCS: Splitting conic solver, version 2.1.1. https://github.com/cvxgrp/scs, November 2017.

[35] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. Found. Trends Mach. Learn., 3(1):1–122, January 2011.

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

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

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

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

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

[41] 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.

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

[43] 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.