Artificial Intelligence 336 (2024) 104218

Contents lists available at ScienceDirect

Artificial Intelligence

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

Interval abstractions for robust counterfactual explanations

Junqi Jiang [*], Francesco Leofante, Antonio Rago, Francesca Toni

Department of Computing, Imperial College London, 180 Queen's Gate, London, SW7 2AZ, United Kingdom

ARTICLE INFO

Keywords:
Explainable AI
Counterfactual explanations
Algorithmic recourse
Robustness of explanations

ABSTRACT

Counterfactual Explanations (CEs) have emerged as a major paradigm in explainable AI research, providing recourse recommendations for users affected by the decisions of machine learning models. However, CEs found by existing methods often become invalid when slight changes occur in the parameters of the model they were generated for. The literature lacks a way to provide exhaustive robustness guarantees for CEs under model changes, in that existing methods to improve CEs' robustness are mostly heuristic, and the robustness performances are evaluated empirically using only a limited number of retrained models. To bridge this gap, we propose a novel interval abstraction technique for parametric machine learning models, which allows us to obtain provable robustness guarantees for CEs under a possibly infinite set of plausible model changes Δ. Based on this idea, we formalise a robustness notion for CEs, which we call Δ-robustness, in both binary and multi-class classification settings. We present procedures to verify Δ-robustness based on Mixed Integer Linear Programming, using which we further propose algorithms to generate CEs that are Δ-robust. In an extensive empirical study involving neural networks and logistic regression models, we demonstrate the practical applicability of our approach. We discuss two strategies for determining the appropriate hyperparameters in our method, and we quantitatively benchmark CEs generated by eleven methods, highlighting the effectiveness of our algorithms in finding robust CEs.

1. Introduction

As the field of explainable AI (XAI) has matured, *counterfactual explanations* (CEs) have risen to prominence as one of the dominant post-hoc methods for explaining the outputs of AI models (see [1,2] for overviews). For a given input to a model, a CE essentially presents a user with a minimally modified input which results in a different output from the model, thus revealing how a different outcome could be achieved if the proposed changes were to be enforced. CEs have been shown to improve human understanding and trust [3], as they provide information about alternative possibilities that humans can use to build rich mental representations [4]. Initial approaches to generate CEs [5,6] were optimised for *validity*, i.e. the ability of a CE to correctly change the output, and *proximity*, with respect to some distance measure between the original and modified inputs. Since then, additional metrics have been

* Corresponding author.
E-mail addresses: junqi.jiang@imperial.ac.uk (J. Jiang), f.leofante@imperial.ac.uk (F. Leofante), a.rago@imperial.ac.uk (A. Rago), f.toni@imperial.ac.uk (F. Toni).

https://doi.org/10.1016/j.artint.2024.104218
Received 18 April 2024; Received in revised form 20 August 2024; Accepted 28 August 2024

Available online 2 September 2024

proposed (see [1] for an overview), such as *diversity* [7], i.e. how widely CEs differ from one another, and *plausibility* [8], i.e. whether the CEs lie within the data distribution.

A further metric which is receiving increasing attention of late is *robustness* [9], i.e. how the validity of a CE is affected by changes in the scenario for which the CE was initially generated. In this paper, we focus on robustness to slight changes in the AI model parameters induced by, for example, retraining [10–16]. This form of robustness is of critical importance in practice. For illustration, consider a mortgage applicant who was rejected by a model and received a CE demonstrating changes they could make to their situation in order to have their application accepted. If retraining occurs while the applicant makes those changes, without robustness, their modified case may still result in a rejected application, leaving the mortgage provider liable due to their conflicting statements. This is especially concerning when CEs are optimised for proximity, as they are likely to be close to the model's decision boundary and thus at high risk of being invalid if small changes in this boundary occur [10].

Methods recently introduced to target this problem typically induce robustness to model changes either by using dedicated continuous optimisation procedures [14,10], or by applying post-hoc refinements to candidate CEs found by non-robust CE generation methods [12,13,17]. However, due to their approximate nature, these methods are unable to provide formal robustness guarantees, which are advocated as being vital towards achieving trustworthy AI [18].

In this work, we fill this gap by presenting a method which provides formal robustness guarantees for CEs under model shifts. Specifically, we target plausible model shifts [10] that can be encoded by a set Δ of norm-bounded perturbations to the parameters of machine learning models. Given this setting, we propose a novel *interval abstraction* technique, which over-approximates the output ranges of parametric machine learning models (including neural networks and logistic regressions) when subject to the model shifts encoded in Δ. This technique is inspired by the abstraction method presented in [19], originally proposed for estimating neural networks' outputs by grouping weight edges into weight intervals. We show that using our interval abstraction, a precise notion of robustness for CEs under Δ can be stated and formally verified. In the following, we refer to such notion of robustness as the Δ-robustness of CEs. Unlike most previous robust CE methods which only apply to binary classification, our focus on computing output ranges allows our method to also work on multi-class classification.

Using interval abstractions, we propose two procedures to generate provably robust CEs: an iterative algorithm augmenting existing CE generation methods, and a sound and complete Robust Nearest-neighbour Counterfactual Explanations (RNCE) method. Since this work uses Mixed Integer Linear Programming (MILP) to practically test Δ-robustness, the ensuing presentation will only focus on machine learning models whose forward pass can be encoded into a MILP program.

Finally, in an extensive empirical study, we demonstrate the effectiveness of our CE generation algorithms in a benchmarking study against seven existing methods. Notably, one configuration of our iterative algorithm finds Δ-robust CEs with the lowest costs among all the robust baselines, and our new RNCE algorithm achieves perfectly robust results while finding CEs close to the data manifold.

The paper is structured as follows. In Section 2 we cover the related work, and in Section 3 we introduce background notions on computing CEs. Sections 4, 5, and 6 introduce a formalisation of Δ-robustness and a MILP encoding for testing it. Then, the two CEs generation algorithms are presented in Section 7, which are extensively evaluated through experiments in Section 8. We conclude in Section 9 with future research directions. The core contributions of our work are summarised as follows:

- We propose a novel interval abstraction method to test Δ-robustness, formally verifying whether a CE is robust against plausible model changes.
- Our method explicitly characterises robustness for CEs in multi-class classification, and, to the best of our knowledge, is the first to do so.
- We present a principled workflow demonstrating the usefulness of Δ-robustness for evaluating and generating robust CEs in practice.
- We introduce an iterative algorithm and the RNCE algorithm to generate provably robust CEs, which are demonstrated to have superior performances against seven baselines.

This paper builds upon our previous work [15] with significant extensions. Specifically, Section 4 extends the corresponding section in [15] to account for different parametric machine learning models in addition to feed-forward neural networks. We present in-depth discussions and relaxations for the soundness of Δ (Definition 9), and formalisation for multi-class classifications are included with an empirical study (Sections 5, 8.6). Section 6 formalises testing procedures for Δ-robustness in terms of MILP programs, which were only briefly mentioned in Appendix B of our previous work. Section 7 significantly extends the algorithm proposed in [15], which could fail to find provably robust CEs. We additionally propose a new algorithm, RNCE, (Algorithm 2), which is guaranteed to return provably robust CEs, while also addressing plausibility, another desirable property of CEs. In Section 8, we propose and comprehensively investigate two strategies to find the optimal hyperparameters in our approach, which is not presented in the previous work. Further, the empirical study additionally includes benchmarking of the runtime performance of our approach, new results obtained for logistic regression models, and four more CE generation baselines, two of which generate robust CEs, giving a more thorough experimental evaluation of both our approach and the robustness research landscape in general. Finally, throughout the paper, we have added discussion and examples to give more intuition on the introduced concepts, as well as a more in-depth view of the existing literature.

2. Related work

2.1. Counterfactual explanations

Various methods for generating CEs in classification tasks have been proposed throughout the recent surge in XAI research, often optimising for one or more metrics characterising desirable properties of CEs. Tolomei et al. [5] focused on tree-based classifiers and evaluated the CEs' validity, whereas Wachter et al. [6] formulated the CE search problem for differentiable models as a gradient-based optimisation problem and evaluated also CEs' proximity (we refer their method as *GCE* in Section 8). These two metrics remain a prominent research focus, e.g. Mohammadi et al. [20] treat CE generation in neural networks as a constrained optimisation problem such that formal validity and proximity guarantees can be given. Various works have considered plausibility, e.g. Brughmans et al. [21] find dataset points which are naturally on the data manifold as CEs (we refer to their method as *NNCE* in Section 8). Meanwhile, variational auto-encoders have been used to generate plausible[1] CEs [8,22,23]. Actionability ensures that the CEs only coherently change the mutable features. This is usually dealt with by customising constraints in the optimisation process of finding CEs [24]. Mothilal et al. [7] and Dandl et al. [25] build optimisation frameworks for the diversity of the generated CEs. Another line of research focuses on building links between CEs and the causality literature, formulating the problem of finding CEs as intervention operations in causal frameworks [26,27].

Methods for generating CEs have also been defined for other classifiers, e.g. Ustun et al. [24] consider different types of linear classification models, Albini et al. [28] focus on different forms of Bayesian classifier, and Kanamori et al. [29] target logistic regression and random forest classifiers. Outside the scope of tabular data classification, studies have also investigated CEs for, e.g., graph data tasks [30], visual tasks [31], time series prediction tasks [32], etc. We refer to [1,2] for recent overviews.

2.2. Robustness of counterfactual explanations

In this work, we consider the robustness of CEs against model changes. When using traditional methods (i.e. methods that focus on the properties introduced in Section 2.1) to find CEs for some classification models, the resulting CEs are highly unlikely to remain valid when the model parameters are updated [33,10]. As illustrated in Section 1, this could cause issues for both the users receiving the CEs and the providers of the explanations.

Many research studies have been conducted to tackle this problem, aiming at generating high-quality, robust CEs. Upadhyay et al. [10] adopt a gradient-based robust optimisation approach to generate CEs that are robust to model parameter changes. A similar gradient-based approach is taken by [14,11,34] under probabilistic frameworks where model changes are expressed by probability distributions associated with ambiguity sets. Ferrario and Loi [35] proposes a retraining procedure using counterfactual data augmentation to mitigate the invalidation of previously generated non-robust CEs, while Guo et al. [36] introduce a robust training framework which jointly optimises the accuracy of neural networks and the robustness of CEs. Another line of work [12,13,17] places more focus on designing heuristics aimed at increasing the model confidence (predicted class probability) to induce more robust CEs. These heuristics are then used as part of certain search-based refining processes to improve the robustness of CEs found by any base CEs generation methods. Differently from previous works, we work target exhaustive robustness guarantees for CEs against norm-bounded model parameter changes. To practically verify and to compute provably robust CEs, we adopt MILP-based approaches. Though some studies introduce methods which provide empirical measures of CEs' robustness [11,13,17], no existing approach is able to give the strong formal guarantees which our method affords, to the best of our knowledge.

Other forms of robustness of CEs have also been studied in the literature. Robustness against input perturbations requires that the CEs generation method not produce drastically different CEs for very similar inputs [37,38]. Robustness against changes in the training dataset, especially those that resulted in data deletion requests, aims to ensure the CEs' validity under the consequently retrained models [39,40]. The methods in [41–43] focus on the implications of the inconsistencies of CEs under predictive multiplicity, where multiple comparable models exist for the same task while assigning conflicting labels for the same inputs. Finally, users might only implement the recourse indicated by CEs to an approximate level, instead of achieving the prescribed feature values exactly. It is therefore desirable that CEs stay valid when subject to such noisy execution [44–48]. The study of these forms of robustness is outside the scope of this paper as our focus is on model changes, we refer to [49,9] for recent surveys.

2.3. Robustness and verification in machine learning

The robustness problem has been extensively studied in the machine learning literature. Studies in this area are usually concerned with proving the consistency of neural network predictions when various types of small perturbations occur in the input [50,51] or in the model parameters [52,53]. One popular approach to check the robustness of learning models such as neural networks relies on abstraction techniques. These methods derive over-approximations of the neural networks' output ranges, which are then integrated into the training loop to ensure provable robustness guarantees against small perturbations [54–58]. Notably, interval bound propagation-based methods have been found effective, using interval arithmetic to propagate the perturbations in the input space to the output layer [55,56]. Tighter bounds are further obtained by symbolic interval propagation [57,58]. Finally, the most relevant work to our study is the Interval Neural Networks (INNs) proposed by Prabhakar and Afzal [19]. INNs also use interval arithmetic to

[1] "Plausible" has been used to describe both the property of CEs and the form of model changes; the specific meanings are clear from the context.

estimate the model output ranges, but with a focus on aggregating adjacent weights to simplifying the neural network architecture for easier verification. By solving MILP programs, over-approximations of model output ranges can be obtained. Differently from previous work, we propose to use INNs to obtain a novel abstraction technique that represents the robustness property of CEs under a pre-defined set of plausible model changes.

3. Background

Notation. We use $[k]$ to denote the set $\{1,\dots,k\}$, for $k \in \mathbb{Z}^+$. Given a vector $x \in \mathbb{R}^n$ we use x_i to denote the i-th component. Similarly, for a matrix $W \in \mathbb{R}^n \times \mathbb{R}^m$, we use $W_{i,j}$ to denote the i,j-th element and $\mathbf{vec}(W)$ to refer to W's vectorisation $\mathbf{vec}(W) = [W_{1,1},\dots,W_{n,1},W_{1,2},\dots,W_{n,2},\dots,W_{n,m}]$. Finally, $\mathbb{I}(\mathbb{R})$ denotes the set of all closed intervals over $\mathbb{R}$.

Classification models. A classification model is a parametric model characterised by a set of equations over a parameter space $\Theta \subseteq \mathbb{R}^d$, for some $d > 0$. We use $\mathcal{M}_\Theta$ to denote the family of classifiers spanning Θ and $\mathcal{M}_\theta$ to refer to a specific concretisation obtained for some $\theta \in \Theta$. The latter is typically obtained by training on a set of labelled inputs. Then, for any unlabelled input $x \in \mathcal{X}$, $\mathcal{M}_\theta$ can be used to infer (predict) its label.

Example 1. Consider an input $x \in \mathcal{X}$ and assume $\mathcal{M}_\Theta$ implements a *logistic regression classifier*, characterised by the following equation:

$$\mathcal{M}_\Theta(x) = \sigma\left(\sum_{i=1}^{d-1} \theta_i x_i + \theta_d\right)$$

where σ is a logistic function defined as usual. Then, $\theta = [\theta_1,\dots,\theta_d]$.

Example 2. Consider an input $x \in \mathcal{X}$ and assume $\mathcal{M}_\Theta$ implements a *fully connected neural network* with k hidden layers, characterised by the following equations:

- $V^{(0)} = x$;
- $V^{(i)} = \phi(W^{(i)} \cdot V^{(i-1)} + B^{(i)})$ for $i \in [k]$, where $W^{(i)}$ and $B^{(i)}$ are weights and biases, respectively, associated with layer i, and ϕ is an activation function applied element-wise;
- $\mathcal{M}_\Theta(x) = \sigma(V^{(k+1)}) = \sigma(W^{(k+1)} \cdot V^{(k)} + B^{(k+1)})$, where σ is a logistic function defined as usual.

Then, $\theta = [\mathbf{vec}(W^{(1)})\ \mathbf{vec}(B^{(1)}) \dots \mathbf{vec}(W^{(k+1)})\ \mathbf{vec}(B^{(k+1)})]$.

We now define the classification outcome of $\mathcal{M}_\theta$; while we focus on binary classification tasks for legibility, i.e. with label $c \in \{0,1\}$, the ensuing definitions can be generalised to any classification setting.

Definition 1. Given input $x \in \mathcal{X}$ and model $\mathcal{M}_\theta$, we say that $\mathcal{M}_\theta$ *classifies* x *as* 1 if $\mathcal{M}_\theta(x) \geq 0.5$, and otherwise x is classified as 0.

In the following, we abuse notation and use $\mathcal{M}_\theta(x) = 1$ (respectively $\mathcal{M}_\theta(x) = 0$) to denote when x is classified as 1 (respectively 0). Note that classes 0 and 1 are often assumed to be the undesirable and the desirable classes for the user [10,20,13].

Counterfactual explanations. Consider a classification model $\mathcal{M}_\theta$ trained to solve a binary classification problem. Assume an input x is given for which $\mathcal{M}_\theta(x) = 0$. Intuitively, a counterfactual explanation is a new input x' which is somehow similar to x, e.g. in terms of some specified distance between features values, and for which $\mathcal{M}_\theta(x') = 1$. Formally, existing methods in the literature may be understood to compute counterfactuals as follows.

Definition 2. Consider an input $x \in \mathcal{X}$ and a (binary) classification model $\mathcal{M}_\theta$ s.t. $\mathcal{M}_\theta(x) = 0$. Given a distance metric $d : \mathcal{X}\times\mathcal{X} \to \mathbb{R}^+$, a *counterfactual explanation* is any x' such that:

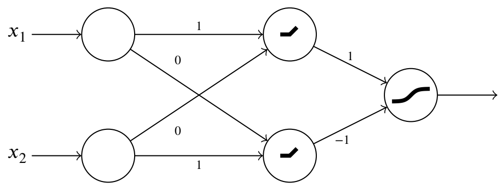
$$\arg\min_{x' \in \mathcal{X}} \quad d(x, x') \tag{1a}$$

$$\text{subject to} \quad \mathcal{M}_\theta(x') = 1 \tag{1b}$$

A counterfactual explanation thus corresponds to the closest input x' (Eq. (1a)) belonging to the original input space that makes the classification flip (Eq. (1b)). Eq. (1a) and (1b) are typically referred to as *proximity* and *validity* properties, respectively, of counterfactual explanations. A common choice for the distance metric d is the normalised L_1 distance [6] for sparse changes in the CEs, which we also adopt here. Under this choice of d, we note that when $\mathcal{M}_\theta$ is a piece-wise linear model, an exact solution to Eqs. (1a-b) can be computed with MILP – see, e.g., [20].

Example 3. (Continuing from Example 1.) Assume a logistic regression classification model $\mathcal{M}_\theta(x) = \sigma(-x_1 + x_2)$ for any input $x = [x_1, x_2]$, where σ is the standard sigmoid function. For a concrete input $x = [0.7, 0.5]$, we have $\mathcal{M}_\theta(x) = 0$. A possible counterfactual explanation for x may be $x' = [0.7, 0.7]$, for which $\mathcal{M}_\theta(x') = 1$.

Example 4. (Continuing from Example 2.) Consider the fully-connected feed-forward neural network $\mathcal{M}_\theta$ below, where weights are as indicated in the diagram, biases are zero. Hidden layers use ReLU activations, whereas the output node uses a sigmoid function. The network receives a two-dimensional input $x = [x_1, x_2]$ and produces output $\mathcal{M}_\theta(x)$.



The symbolic expressions for the output is $\mathcal{M}_\theta(x) = \sigma(\max(0, x_1) - \max(0, x_2))$, where σ denotes a sigmoid function with the usual meaning. Given a concrete input $x = [1, 2]$, we have $\mathcal{M}_\theta(x) = 0$. A possible counterfactual explanation may be $x' = [2.1, 2]$, for which $\mathcal{M}_\theta(x') = 1$.

4. Interval abstractions and robustness for binary classification

In this section, we introduce a novel interval abstraction technique inspired by INNs to reason about the provable robustness guarantees of CEs, under the possibly infinite family of classification models obtained by applying a pre-defined set of plausible model changes, Δ. We formalise the novel notion of Δ-robustness, which, once satisfied, will ensure that the validity of CEs is not compromised by any model parameter change encoded in Δ.

4.1. Plausible model changes Δ

First, in this section, we formalise the type of model changes central to our method. We begin by defining a notion of distance between two concretisations of a parametric classifier.

Definition 3. Let $\mathcal{M}_\theta$ and $\mathcal{M}_{\theta'}$ be two concretisations of a parametric classification model $\mathcal{M}_\Theta$. For $0 \le p \le \infty$, the *p-distance between $\mathcal{M}_\theta$ and $\mathcal{M}_{\theta'}$* is defined as:

$$d_p(\mathcal{M}_\theta, \mathcal{M}_{\theta'}) = \|\theta - \theta'\|_p.$$

Example 5. Consider two models $\mathcal{M}_\theta = \sigma(-x_1 + x_2)$ and $\mathcal{M}_{\theta'} = \sigma(0.8 \cdot x_1 + x_2)$. Assume $p = \infty$. Then, their p-distance is $d_p(\mathcal{M}_\theta, \mathcal{M}_{\theta'}) = \max_i |\theta_i - \theta'_i| = 1.8$.

Intuitively p-distance compares $\mathcal{M}_\theta$ and $\mathcal{M}_{\theta'}$ in terms of their parameters and computes the distance between them as the p-norm of the difference of their parameter vectors. Using this notion, we next characterise a model shift as follows.

Definition 4. Given $0 \le p \le \infty$, a *model shift* is a function S mapping a classification model $\mathcal{M}_\theta$ into another $\mathcal{M}_{\theta'} = S(\mathcal{M}_\theta)$ such that:

- $\mathcal{M}_\theta$ and $\mathcal{M}_{\theta'}$ are concretisations of the same parameterised family $\mathcal{M}_\Theta$;
- $d_p(\mathcal{M}_\theta, \mathcal{M}_{\theta'}) > 0$.

Model shifts are typically observed in real-world applications when a model is regularly retrained to incorporate new data. In such cases, models are likely to see only small changes at each update. In the same spirit as [10], we capture this as follows.

Definition 5. Given a classification model $\mathcal{M}_\theta$, a threshold $\delta \in \mathbb{R}_{>0}$ and $0 \le p \le \infty$, the *set of plausible model shifts* is defined as:

$$\Delta = \{S \mid d_p(\mathcal{M}_\theta, S(\mathcal{M}_\theta)) \le \delta\}.$$

When a set of plausible model shifts Δ is applied on $\mathcal{M}_\theta$, the resulting maximum change on each model parameter in the original model is conservatively upper-bounded:

Lemma 1. *Consider a classification model $\mathcal{M}_\theta$ and a set of plausible model shifts Δ with threshold δ and $0 \le p \le \infty$. Then, $\forall \mathcal{M}_{\theta'} = S(\mathcal{M}_\theta)$, $S \in \Delta$, and $\forall \theta_i$, θ'_i, $i \in [d]$, we have $\theta'_i \in [\theta_i - \delta, \theta_i + \delta]$.*

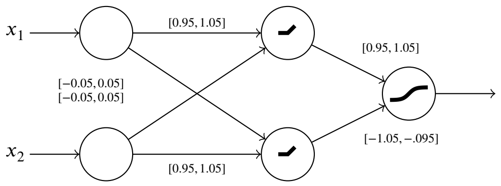


Fig. 1. Illustration of Example 7.

Proof. Combining Definition 5 with Definition 3, we obtain:

$$\left(\sum_{i=1}^{d} |\theta_i - \theta'_i|^p\right)^{\frac{1}{p}} \leq \delta$$

We raise both sides to the power of p:

$$\sum_{i=1}^{d} \left(|\theta_i - \theta'_i|^p\right) \leq \delta^p$$

where the inequality is preserved as both sides are always positive. We now observe this inequation bounds each addend from above, i.e.,

$$|\theta_i - \theta'_i|^p \leq \delta^p$$

Solving the inequation for each addend we obtain $\theta'_i \in [\theta_i - \delta, \theta_i + \delta]$, which gives a conservative upper-bound on the maximum change that can be applied to each parameter in $\mathcal{M}_{\theta'}$. □

4.2. Interval abstractions

To guarantee robustness to the plausible model changes Δ, methods are needed to compactly represent and reason about the behaviour of a CE under the potentially infinite family of models originated by applying each $S \in \Delta$ to $\mathcal{M}_\theta$. In the following, we introduce an abstraction framework that can be used to this end.

Definition 6. Consider a classification model $\mathcal{M}_\theta$ with $\theta = [\theta_1, \dots, \theta_d]$. Given a set of plausible model shifts Δ, we define the *interval abstraction of* $\mathcal{M}_\theta$ *under* Δ as $\mathcal{I}_{(\theta,\Delta)} : \mathcal{X} \to \mathbb{I}(\mathbb{R})$ such that:

- $\mathcal{M}_\theta$ and $\mathcal{I}_{(\theta,\Delta)}$ have the same model architecture;
- $\mathcal{I}_{(\theta,\Delta)}$ is parameterised by an interval-valued vector $\boldsymbol{\theta} = [\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_d]$;
- $\boldsymbol{\theta}_i$, for $i \in [d]$, encodes the range of possible changes induced by the application of any $S \in \Delta$ to $\mathcal{M}_\theta$ such that $\boldsymbol{\theta}_i = [\theta_i - \delta, \theta_i + \delta]$, where δ is the maximum shift obtainable as per Definition 5.

Lemma 2. $\mathcal{I}_{(\theta,\Delta)}$ *over-approximates the set of models* $\mathcal{M}_{\theta'}$ *that can be obtained from* $\mathcal{M}_\theta$ *via* Δ.

Proof. Lemma 2 states that $\mathcal{I}_{(\theta,\Delta)}$ captures all models $\mathcal{M}_{\theta'}$ that can be obtained from $\mathcal{M}_\theta$ applying a $S \in \Delta$, and possibly more models that violate the plausibility constraint (Definition 5, the p-distances are upper-bounded by δ). The former can be seen by observing that each parameter θ_i in $\mathcal{I}_{(\theta,\Delta)}$ is initialised in such a way to contain all possible parameterisations obtainable starting from the initial value θ_i and perturbing it up to $\pm\delta$ (Lemma 1). As a result, $\mathcal{I}_{(\theta,\Delta)}$ captures all shifted models by construction. $\mathcal{I}_{(\theta,\Delta)}$ also captures more concrete models for which the p-distance is greater than δ, as the bounds used in the interval abstraction, $\boldsymbol{\theta}_i = [\theta_i - \delta, \theta_i + \delta]$, are conservative. □

The following two examples demonstrate the interval abstraction in the context of different classification models.

Example 6. Continuing from Example 3. Consider the same model $\mathcal{M}_\theta$ and assume a set of plausible model shifts $\Delta = \{S \mid d_\infty(\mathcal{M}_\theta, S(\mathcal{M}_\theta)) \leq 0.1\}$. The interval abstraction $\mathcal{I}_{(\theta,\Delta)}$ is defined by the model equation $\mathcal{I}_{(\theta,\Delta)}(x) = \sigma\left(\boldsymbol{\theta}_1 x_1 + \boldsymbol{\theta}_2 x_2\right)$, where $\boldsymbol{\theta}_1 = [-1 - 0.1, -1 + 0.1]$ and $\boldsymbol{\theta}_2 = [1 - 0.1, 1 + 0.1]$. Then, $\boldsymbol{\theta} = [\boldsymbol{\theta}_1, \boldsymbol{\theta}_2]$.

Example 7. Continuing from Example 4. Consider the same model $\mathcal{M}_\theta$ and assume a set of plausible model shifts $\Delta = \{S \mid d_\infty(\mathcal{M}_\theta, S(\mathcal{M}_\theta)) \leq 0.05\}$. The resulting interval abstraction $\mathcal{I}_{(\theta,\Delta)}$ is represented in Fig. 1, where the symbolic expression of the output interval is $\sigma([0.95, 1.05] \cdot \max(0, [0.95, 1.05] \cdot x_1 + [-0.05, 0.05] \cdot x_2) + [-1.05, -0.95] \cdot \max(0, [-0.05, 0.05] \cdot x_1 + [0.95, 1.05] \cdot x_2))$.

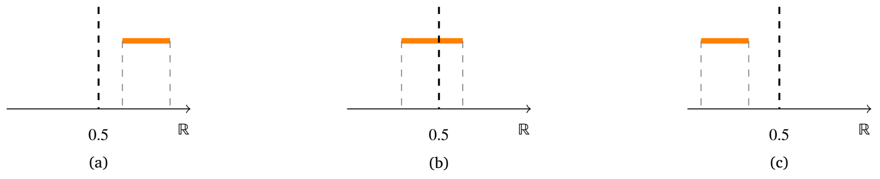


Fig. 2. Visual representation of Definition 7. In (a), $\mathcal{I}_{(\theta,\Delta)}$ classifies an input as 1 because the output range for that input is always greater than 0.5. In (b), the output range includes value 0.5 therefore the classification result is undefined. In (c), in a similar manner to the way in which the input in (a) is classified as 1, the input is classified as 0.

Given an input, our interval abstraction maps it to an output interval, constructed by taking the lowest and the highest values from all the possible classification outcomes by any shifted model $\mathcal{M}_{\theta'} \in \Delta$. This output interval therefore over-approximates the outputs obtainable from any shifted models in Δ. The classification semantics of the interval abstraction thus departs from Definition 1 and needs to be generalised to account for this new behaviour.

Definition 7. Let $\mathcal{I}_{(\theta,\Delta)}$ be the interval abstraction of a classification model $\mathcal{M}_\theta$. Given an input $x \in \mathcal{X}$, let $[l, u]$ be the output interval obtained by applying $\mathcal{I}_{(\theta,\Delta)}$ to x. We say that $\mathcal{I}_{(\theta,\Delta)}$ *classifies* x *as* 1, if $l \geq 0.5$, 0 if $u < 0.5$, and *undefined* otherwise.

As in Section 3, we will slightly abuse notation and use $\mathcal{I}_{(\theta,\Delta)}(x) = 1$ (respectively, $\mathcal{I}_{(\theta,\Delta)}(x) = 0$) to denote the case where the abstraction classifies an input as 1 (respectively, 0). A visual representation of this interval-based classification semantics is given in Fig. 2. In this work, we are interested in the conservative worst-case robustness of CEs. Therefore, we leave for future work (Section 9) the explorations of the undefined case, as in Fig. 2 (b).

4.3. Δ-robustness

Using the interval abstraction as a tool to represent infinite families of shifted models, we now present Δ-robustness, a notion of robustness to model changes which is central to this contribution. First, we define the conditions within which the robustness of a counterfactual can be assessed by introducing a notion of soundness as follows.

Definition 8. Consider an input $x \in \mathcal{X}$ and a model $\mathcal{M}_\theta$ such that $\mathcal{M}_\theta(x) = 0$. Let $\mathcal{I}_{(\theta,\Delta)}$ be the interval abstraction of $\mathcal{M}_\theta$ for a set of plausible model shifts Δ. We say that Δ *is sound for* x iff $\mathcal{I}_{(\theta,\Delta)}(x) = 0$.

In other words, soundness requires that shifts in Δ do not alter the class predicted for the original input x. This is a safety requirement that we introduce to ensure consistency in the predictions produced by the interval abstraction.

We then can reason about the robustness properties defined as follows.

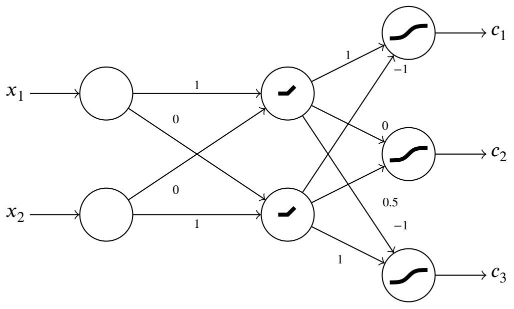
Definition 9. Consider an input $x \in \mathcal{X}$ and a model $\mathcal{M}_\theta$ such that $\mathcal{M}_\theta(x) = 0$. Let $\mathcal{I}_{(\theta,\Delta)}$ be the interval abstraction of $\mathcal{M}_\theta$ for a set of plausible model shifts Δ. We say that a counterfactual explanation x' is:

- Δ*-robust* iff $\mathcal{I}_{(\theta,\Delta)}(x') = 1$ and
- *strictly* Δ*-robust* iff it is Δ-robust & Δ is sound for x.

The soundness of Δ ensures the theoretical correctness of Δ-robustness. We point out that, in practice, this requirement may be relaxed such that for any input x labelled as class 0 by the original model, one can find a CE x' that is classified as class 1 by all the model shifts in Δ. In this case, strictly speaking, x' cannot be said to be *valid* for Δ because it is possible that for some $\mathcal{M}_i$ induced by Δ, $\mathcal{M}_i(x) = \mathcal{M}_i(x') = 1$. However, in many applications soundness with respect to Δ is not required (as are the setups in e.g. [10,13]), in which case computing Δ-robust CEs may be more practical than computing those which are strictly Δ-robust.

We conclude with an example summarising the main concepts presented in this section.

Example 8. (Continuing from Examples 3 and 6). We observe that $\mathcal{I}_{(\theta,\Delta)}(x)$ produces an output interval $[0.42, 0.48]$, i.e. $\mathcal{I}_{(\theta,\Delta)}(x) = 0$. Since $\mathcal{M}_\theta(x) = 0$, we can conclude that the set of model shifts Δ is sound. We then check whether the previously computed counterfactual explanation $x' = [0.7, 0.7]$ is Δ-robust. Running x' through $\mathcal{I}_{(\theta,\Delta)}$ we obtain an output interval $[0.45, 0.53]$, i.e. $\mathcal{I}_{(\theta,\Delta)}(x) = 0$. Therefore, the counterfactual is not Δ-robust. Assume a new counterfactual is obtained as $x'' = [0.7, 0.86]$. The output interval produced by $\mathcal{I}_{(\theta,\Delta)}$ is now $[0.5, 0.58]$, i.e. $\mathcal{I}_{(\theta,\Delta)}(x'') = 1$. Thus, the counterfactual x'' is strictly Δ-robust.



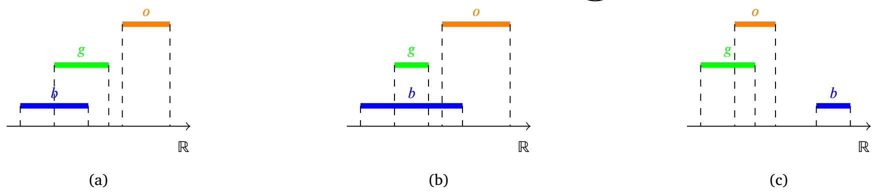


Fig. 3. Visual representation of Definition 12 instantiated with three colour-coded classes {'orange', 'green', 'blue'}. In (a), $\mathcal{I}_{(\theta,\Delta)}$ classifies an input as o since the output range for that class is always greater than those of any other classes. In (b), the output ranges for o overlap with b, the classification result is therefore undefined. In (c), in a similar manner to the way in which the input in (a) is classified as o, $\mathcal{I}_{(\theta,\Delta)}$ classifies an input as b. (For interpretation of the colours in the figure(s), the reader is referred to the web version of this article.)

5. Interval abstractions and robustness for multi-class classification

Next, we introduce the notion of Δ-robustness in multi-class classification settings. Unlike the binary case, where it is usually assumed that class 0 (unwanted class) is the original prediction result for the input and class 1 is the desirable target class for the CE to achieve, in the multi-class settings, the classes need to be explicitly specified [25,7]. Given a set of labels $\{1,\ldots,\ell\}$, we generalise Definitions 1 and 2 as follows.

Definition 10. Given input $x \in \mathcal{X}$ and model $\mathcal{M}_\theta$, we say that $\mathcal{M}_\theta$ *classifies* x *as* $c \in \{1,\ldots,\ell\}$ if $\mathcal{M}_\theta(x)_c \geq \mathcal{M}_\theta(x)_{c'}$, for all $c' \in \{1,\ldots,\ell\}$ such that $c' \neq c$.

Definition 11. Consider an input $x \in \mathcal{X}$ and a classification model $\mathcal{M}_\theta$ s.t. $\mathcal{M}_\theta(x) = c \in \{1,\ldots,\ell\}$. Assume a desirable target class $c' \in \{1,\ldots,\ell\}$ such that $c' \neq c$ and given a distance metric $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$, a *counterfactual explanation* for class c' is any x' such that:

$$\begin{aligned} &\arg\min_{x' \in \mathcal{X}} \quad d(x,x') \\ &\text{subject to} \quad \mathcal{M}_\theta(x') = c' \end{aligned}$$

Example 9 illustrates Definition 11 on a toy example where the classifier is a neural network.

Example 9. Consider the fully-connected feed-forward neural network $\mathcal{M}_\theta$ below, where weights are as indicated in the diagram, biases are zero. Hidden layers use ReLU activations, whereas the output layer uses a softmax function. Output classes are $\{c_1, c_2, c_3\}$.

The symbolic expressions for the output is $\mathcal{M}_\theta(x) = Softmax([\max(0,x_1) - \max(0,x_2), 0.5\max(0,x_2), \max(0,x_2) - \max(0,x_1)])$. Given a concrete input $x = [2,2]$, we have $\mathcal{M}_\theta(x) = c_2$. A possible counterfactual explanation for class c_1 may be $x' = [3,1]$, for which $\mathcal{M}_\theta(x') = c_1$.

Then, the classification semantics of an interval abstraction for multiclass problems can be obtained by extending Definition 7 as follows.

Definition 12. Let $\mathcal{I}_{(\theta,\Delta)}$ be the interval abstraction of a classification model $\mathcal{M}_\theta$. Given an input $x \in \mathcal{X}$, let $[l_i, u_i]$ be the output interval obtained for each $i \in \{1,\ldots,\ell\}$ by applying $\mathcal{I}_{(\theta,\Delta)}$ to x. We say that $\mathcal{I}_{(\theta,\Delta)}$ *classifies* x *as class* c, if $l_c \geq u_{c'}$ for all $c' \in \{1,\ldots,\ell\} \setminus \{c\}$. Otherwise we say the classification result from $\mathcal{I}_{(\theta,\Delta)}$ is *undefined* if $\forall c' \in \{1,\ldots,\ell\}$, $\exists c'' \in \{1,\ldots,\ell\} \setminus \{c'\}$ such that $u_{c''} \geq l_{c'}$.

Fig. 3 intuitively illustrates the classification semantics of interval abstractions in multi-class settings. Next, we formalise Δ-robustness for CEs.

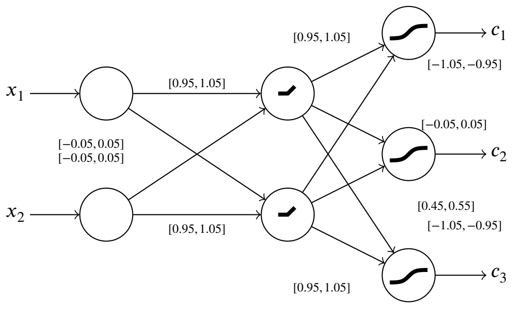


Fig. 4. Illustration for Example 10.

Definition 13. Consider an input $x \in \mathcal{X}$ and a model $\mathcal{M}_\theta$ such that $\mathcal{M}_\theta(x) = c \in \{1, \dots, \ell\}$. Let $\mathcal{I}_{(\theta,\Delta)}$ be the interval abstraction of $\mathcal{M}_\theta$ for a set of plausible model shifts Δ. We say that Δ *is sound for* x iff $\mathcal{I}_{(\theta,\Delta)}(x) = c$.

Definition 14. Consider an input $x \in \mathcal{X}$ and a model $\mathcal{M}_\theta$ such that $\mathcal{M}_\theta(x) = c \in \{1, \dots, \ell\}$. Let $\mathcal{I}_{(\theta,\Delta)}$ be the interval abstraction of $\mathcal{M}_\theta$ for a set of plausible model shifts Δ. Assume a desirable target class $c' \in \{1, \dots, \ell\}$ such that $c' \neq c$, we say that a counterfactual explanation x' is:

- Δ*-robust* for class c' iff $\mathcal{I}_{(\theta,\Delta)}(x') = c'$ and
- *strictly* Δ*-robust* for class c' iff it is Δ-robust & Δ is sound for x.

The definition of Δ-robustness transfers to the multi-class semantics by estimating lower and upper bounds for each output class and then checking the robustness property. In Example 10, we instantiate an interval abstraction for a neural network and show how the output intervals can be calculated, thus how Δ-robustness can be examined. We introduce a more general solution for testing Δ-robustness in the next section.

Example 10. Continuing from Example 9. Consider the same model $\mathcal{M}_\theta$ and assume a set of plausible model shifts $\Delta = \{S \mid d_\infty(\mathcal{M}_\theta, S(\mathcal{M}_\theta)) \leq 0.05\}$. The resulting interval abstraction $\mathcal{I}_{(\theta,\Delta)}$ is defined as in Fig. 4.

The symbolic expression of the output interval is obtained by applying softmax activation on a vector of intervals $[\mathcal{M}_\theta(x)_{c_1}, \mathcal{M}_\theta(x)_{c_2}, \mathcal{M}_\theta(x)_{c_3}]$ where each entry is the pre-softmax output interval for output classes $\{c_1, c_2, c_3\}$, then sorting the corresponding class interval for each class. We denote the node interval for the hidden nodes as $\mathcal{M}_\theta(x)_{h_1} = \max(0, [0.95, 1.05] \cdot x_1 + [-0.05, 0.05] \cdot x_2)$ and $\mathcal{M}_\theta(x)_{h_2} = \max(0, [-0.05, 0.05] \cdot x_1 + [0.95, 1.05] \cdot x_2)$. Then, the output node intervals can be expressed as $\mathcal{M}_\theta(x)_{c_1} = [0.95, 1.05] \cdot \mathcal{M}_\theta(x)_{h_1} + [-1.05, -0.95] \cdot \mathcal{M}_\theta(x)_{h_2}$, $\mathcal{M}_\theta(x)_{c_2} = [-0.05, 0.05] \cdot \mathcal{M}_\theta(x)_{h_1} + [0.45, 0.55] \cdot \mathcal{M}_\theta(x)_{h_2}$, $\mathcal{M}_\theta(x)_{c_3} = [-1.05, -0.95] \cdot \mathcal{M}_\theta(x)_{h_1} + [0.95, 1.05] \cdot \mathcal{M}_\theta(x)_{h_2}$.

For the input $x = [2, 2]$, we observe that $\mathcal{I}_{(\theta,\Delta)}(x)$ produces output intervals $[0.114, 0.322], [0.356, 0.51], [0.114, 0.322]$ respectively for each class, therefore the lower bound of class output node c_2 (0.356) is greater than the upper bound of the other two classes (0.322), i.e. $\mathcal{I}_{(\theta,\Delta)}(x) = c_2$. Since $\mathcal{M}_\theta(x) = c_2$, we can conclude that the set of model shifts Δ is sound for c_2. We then check whether the previously computed counterfactual explanation $x' = [3, 1]$ is Δ-robust. Running x' through $\mathcal{I}_{(\theta,\Delta)}$ we obtain output intervals $[0.617, 0.91], [0.08, 0.345], [0.01, 0.038]$, i.e. $\mathcal{I}_{(\theta,\Delta)}(x) = c_1$. Therefore, the counterfactual is Δ-robust for class c_1. Furthermore, this CE is strictly Δ-robust.

6. Computing Δ-robustness with MILP

To determine whether a CE x' is Δ-robust, we are interested in the output ranges of the interval abstraction $\mathcal{I}_{(\theta,\Delta)}$ when x' is passed in. Prabhakar and Afzal [19] proposed an approach to compute the output ranges of an INN, which we adapt in this section to compute the reachable intervals for the output node in our interval abstraction $\mathcal{I}_{(\theta,\Delta)}$. The output range estimation problem can be encoded in MILP; in the following, we instantiate the encoding for fully connected neural networks with k hidden layers (as first introduced in Example 2) with ReLU activation functions. The encoding introduces:

- a real variable $v_j^{(0)}$ for $j \in [|V^{(0)}|]$, used to model the input of $\mathcal{I}_{(\theta,\Delta)}$;
- a real variable $v_j^{(i)}$ to model the value of each hidden and output node $V^{(i)}$, for $i \in [k+1]$ and $j \in [|V^{(i)}|]$;
- a binary variable $\xi_j^{(i)}$ to model the activation state of each node in $V^{(i)}$, for $i \in [k]$ and $j \in [|V^{(i)}|]$.

Then, for each layer index $i \in [k]$ and neuron index $j \in [|V^{(i)}|]$, the following set of constraints are asserted:

$$C_j^{(i)} = \Big\{ v_j^{(i)} \geq 0, v_j^{(i)} \leq M(1 - \xi_j^{(i)}),$$

$$v_j^{(i)} \leq \sum_{l=1}^{|V^{(i-1)}|} (W_{j,l}^{(i)} + \delta) v_l^{(i-1)} + (B_j^{(i)} + \delta) + M\xi_j^{(i)},$$

$$v_j^{(i)} \geq \sum_{l=1}^{|V^{(i-1)}|} (W_{j,l}^{(i)} - \delta) v_l^{(i-1)} + (B_j^{(i)} - \delta) \Big\} \tag{3}$$

where M is a sufficiently large constant, and δ is the magnitude of model shifts in Δ. Each $C_j^{(i)}$ uses the standard big-M formulation to encode the ReLU activation [59] and estimate the lower and upper bounds of nodes in the INN.

Then, constraints pertaining to the output layer $k+1$ are asserted for each class $j \in |V^{(k+1)}|$.

$$\begin{aligned} C_j^{(k+1)} = \Big\{ v_j^{(k+1)} &\leq \sum_{l=1}^{|V^{(k)}|} (W_{j,l}^{(k+1)} + \delta) v_l^{(k)} + (B_j^{(k+1)} + \delta), \\ v_j^{(k+1)} &\geq \sum_{l=1}^{|V^{(k)}|} (W_{j,l}^{(k+1)} - \delta) v_l^{(k)} + (B_j^{(k+1)} - \delta) \Big\} \end{aligned} \tag{4}$$

For more details about the encoding and its properties, we refer to the original work [19]. Note that, instead of directly computing the model output, $\mathcal{M}_\Theta(x) = \sigma(V^{(k+1)})$, our MILP program analyses the interval of the final-layer node values before applying the final sigmoid or softmax function, e.g. $V^{(k+1)} = W^{(k+1)} \cdot V^{(k)} + B^{(k+1)}$. The model output interval can be subsequently obtained by applying these final activation functions.

For the case of binary classification (e.g. as in Example 4), testing Δ-robustness of a CE amounts to solving one optimisation problem as shown in the following.

Definition 15. Consider an input $x \in \mathcal{X}$ and a fully connected neural network $\mathcal{M}_\theta$ with k hidden layers, and $\mathcal{M}_\theta(x) = 0$. Let $\mathcal{I}_{(\theta,\Delta)}$ be the interval abstraction of $\mathcal{M}_\theta$ for a set of plausible model shifts Δ. Let ${v_1^{(k+1)}}_{lb}$ be the solution of the optimisation problem

$$\begin{aligned} &\min_{v,\xi} && v_1^{(k+1)} \\ &\text{subject to} && C_j^{(i)},\ i \in [k+1], j \in [|V^{(i)}|] \end{aligned}$$

We say that a counterfactual explanation x' is Δ-*robust* if ${v_1^{(k+1)}}_{lb} \geq 0$.

Indeed, if for a CE x' the lower bound of the output node satisfies ${v_1^{(k+1)}}_{lb} \geq 0$, then after the final sigmoid function, $\sigma({v_1^{(k+1)}}_{lb}) \geq 0.5$, meaning that $\mathcal{I}_{(\theta,\Delta)}(x') = 1$, x' is thus Δ-robust.

For multi-class classification (e.g. as in Example 9), the exact output range for each class $j \in |V^{(k+1)}|$ can be computed by solving two optimisation problems that minimise, respectively maximise, variable $v_j^{(k+1)}$ subject to constraints (3)-(4). Referring to Definitions 12 and 14, we are interested in comparing the lower bound of the desirable target class output interval with the upper bound of output intervals from other classes. Therefore, testing Δ-robustness in multi-class classification amounts to solving $|V^{(k+1)}|$ optimisation problems.

Definition 16. Consider an input $x \in \mathcal{X}$ and a fully connected neural network $\mathcal{M}_\theta$ with k hidden layers with $\mathcal{M}_\theta(x) = c \in \{1, \dots, \ell\}$ and the desirable target class $c' \in \{1, \dots, \ell\}$ such that $c' \neq c$. Let $\mathcal{I}_{(\theta,\Delta)}$ be the interval abstraction of $\mathcal{M}_\theta$ for a set of plausible model shifts Δ. Let ${v_{c'}^{(k+1)}}_{lb}$ be the solution of the optimisation problem

$$\begin{aligned} &\min_{v,\xi} && v_{c'}^{(k+1)} \\ &\text{subject to} && C_j^{(i)},\ i \in [k+1], j \in [|V^{(i)}|] \end{aligned}$$

For each class $c'' \in \{1, \dots, \ell\}$ such that $c'' \neq c'$, let ${v_{c''}^{(k+1)}}_{ub}$ be the solution of the optimisation problems

$$\begin{aligned} &\max_{v,\xi} && v_{c''}^{(k+1)} \\ &\text{subject to} && C_j^{(i)},\ i \in [k+1], j \in [|V^{(i)}|] \end{aligned}$$

We say that a counterfactual explanation x' is Δ-*robust* for class c' if ${v_{c'}^{(k+1)}}_{lb} \geq {v_{c''}^{(k+1)}}_{ub}$.

Similar to the binary case, if a CE passes the above Δ-robustness test, then after applying the final softmax function, the predicted class probability of the desirable target class will always be greater than that of the other classes.

Algorithm 1 Iterative algorithm embedding Δ-robustness tests.

```
Require: Classifier $\mathcal{M}_\theta$, input $x$ such that $\mathcal{M}_\theta(x) = 0$,
 1:      set of plausible model shifts $\Delta$ and threshold $t$
 2: Step 1: build encoding for interval abstraction $\mathcal{I}_{(\mathcal{M},\Delta)}$.
 3: Step 2 (Optional): check the soundness of $\Delta$
 4: while iteration number < $t$ do
 5:     Step 3: compute CE $x'$ for $x$ and $\mathcal{M}_\theta$ using base method
 6:     if $\mathcal{I}_{(\theta,\Delta)}(x') = 1$ then
 7:         return $x'$
 8:     else
 9:         Step 4: increase allowed distance of next CFX
10:         Step 5: increase iteration number
11: return the last $x'$ found
```

In practice, with white-box access to the classifier, the above optimisation problems can be conveniently encoded using any off-the-shelf optimisation solver. In this work we use the Gurobi engine.[2]

7. Algorithms

We now show how the Δ-robustness tests introduced above can be leveraged to generate CEs with formal robustness guarantees.

7.1. *Embedding Δ-robustness tests in existing algorithms*

We propose an approach (Algorithm 1) that can be applied on top of any CE generation algorithms which explicitly parameterise the tradeoff between validity and cost. For example, if the method is optimising a loss function containing a validity loss term and a cost loss term with a tradeoff hyperparameter (similar to [6]), then modifying the hyperparameter to allow more costly CE with better validity (higher class probability) could lead to more robust CEs. Such heuristics, identified as necessary conditions for more robust CEs, are discussed in several recent studies [12,13,17,39].

The algorithm proceeds as follows. First, an interval abstraction is constructed for the classifier $\mathcal{M}_\theta$ and targeted plausible model change Δ; the latter is then optionally checked for soundness (Definition 9) using the same MILP programs for Δ-robustness tests, depending on whether we are aiming at finding strict or non-strict robust CEs. Then, the algorithm performs a Δ-robustness test for the CE generated by the base method. If the test passes, then the algorithm terminates and returns the solution. Otherwise, the search continues iteratively, at each step the hyperparameter of the base method is modified such that CEs of increasing distance can be found. These steps are repeated until a threshold number of iterations t is reached. As a result, the algorithm is incomplete, as it may report that no Δ-robust CE can be found within t steps (while one may exist for larger t). In such cases, the last CE found before the algorithm terminates is returned, assuming the CE found at each iteration is more robust than the previous iteration. As we will see in Section 8, we have identified a configuration of our iterative algorithm with a MILP-based method [20] as the base CE generation method, which empirically overcomes the above limitations and is always able to find Δ-robust CEs.

7.2. *Robust Nearest-neighbour Counterfactual Explanations (RNCE)*

We also propose a robust and plausible CE algorithm shown in Algorithm 2, which is complete under mild assumptions. After some initialisation steps, an interval abstraction is constructed for the model $\mathcal{M}_\theta$ and set Δ in Step 1 (Algorithm 2, 6). Similarly to Algorithm 1, the checking for soundness of Δ step is optional. The algorithm then moves on to Step 2 where the dataset $\mathcal{D}$ used to train $\mathcal{M}_\theta$ is traversed to identify potential CEs for x and filter out unsuitable inputs as described in Algorithm 3. In a nutshell, Algorithm 3 iterates through $\mathcal{D}$ and picks instances that satisfy the counterfactual requirement, parameterised according to a robustness criterion specified by the user via the `robustInit` parameter. When `robustInit` is `T` (`True`), the interval abstraction is used to check, for every instance in the dataset, whether it satisfies Δ-robustness (Definition 9) prior to adding it to $\mathcal{S}$ (Algorithm 3, 9). Alternatively, when setting `robustInit` to `F` (`False`), instances are added to the set of candidate CEs $\mathcal{S}$ as long as their predicted label differs from that of x (Algorithm 3, 8). In the latter case, the Δ-robustness guarantees are postponed to Step 4.

Once the set $\mathcal{S}$ of potential candidates is obtained from $\mathcal{D}$ (Algorithm 2, 8), we fit a k-d tree $\mathcal{T}$ to efficiently store spatial information about them (Step 3). The algorithm then enters its final step, where the best CE is selected from $\mathcal{S}$ as described in Algorithm 4 and returned to the user (Algorithm 2, 10). First, $\mathcal{T}$ is queried to obtain the closest robust NNCE x'_{nn}. When Algorithm 3 is instantiated with `robustInit` $=$ `T`, the first nearest neighbour returned from $\mathcal{T}$ is guaranteed to be robust. Otherwise, several queries might be needed to obtain an instance from $\mathcal{T}$ that also satisfies Δ-robustness (Algorithm 4, 4-6).[3] Thus, instantiations of different `robustInit` settings help to find the nearest point satisfying Δ-robustness. As the nearest neighbour may not be optimal in terms of proximity to the original instance x, further optimisation steps can be triggered by setting the parameter `optimal` to `T`. This starts a line search to find the closest robust CE to x (Algorithm 4, 7-10). The CE computed by Algorithm 4 is then returned to the user.

[2] https://www.gurobi.com/solutions/gurobi-optimizer/.

[3] For clarity, Algorithm 4 describes the latter case; the robustness check in omitted in the code when `robustInit` is `T`.

Algorithm 2 RNCE: main routine.

```
Require: input $x$, model $\mathcal{M}_\theta$,
 1:     training dataset $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$,
 2:     (sound) set of plausible model shifts $\Delta$,
 3:     parameters robustInit, optimal $\in \{$F, T$\}$
 4: Init: $c \leftarrow \mathcal{M}_\theta(x)$; $x' \leftarrow \emptyset$
 5: Step 1: build encoding for interval abstraction $\mathcal{I}_{(\mathcal{M},\Delta)}$
 6: Step 2: select candidate counterfactuals
 7: $\mathcal{S} \leftarrow$ getCandidates($x, \mathcal{M}_\theta, \mathcal{D}, \mathcal{I}_{(\theta,\Delta)}$, robustInit)
 8: Step 3: fit k-d tree $\mathcal{T}$ on $\mathcal{S}$
 9: Step 4: $x' \leftarrow$ getRobustCE($x, \mathcal{M}_\theta, \mathcal{I}_{(\theta,\Delta)}, \mathcal{T}$, optimal)
10: return $x'$
```

Algorithm 3 RNCE: `getCandidates`.

```
Require: input $x$, model $\mathcal{M}_\theta$,
 1:     dataset $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$
 2:     interval abstraction $\mathcal{I}_{(\theta,\Delta)}$
 3:     parameter robustInit $\in \{$F, T$\}$
 4: Init: $c \leftarrow \mathcal{M}_\theta(x)$, $\mathcal{S} \leftarrow \{\}$
 5: Step 1: find candidate subset of the training dataset
 6: for $(x_i, y_i) \in \mathcal{D}$ do
 7:     if robustInit is F then
 8:         Add $(x_i, y_i)$ to $\mathcal{S}$ if $\mathcal{M}_\theta(x_i) = 1 - c$
 9:     else Add $(x_i, y_i)$ to $\mathcal{S}$ if $\mathcal{I}_{(\theta,\Delta)}(x_i) = 1 - c$
10: return $\mathcal{S}$
```

Algorithm 4 RNCE: `getRobustCE`.

```
Require: input $x$, model $\mathcal{M}_\theta$, interval abstraction $\mathcal{I}_{(\theta,\Delta)}$,
 1:     k-d tree $\mathcal{T}$, parameter optimal $\in \{$F, T$\}$
 2: Init: $c \leftarrow \mathcal{M}_\theta(x)$; $x' \leftarrow \emptyset$; $a \leftarrow 1$; $s \leftarrow 0.05$
 3: Step 1: find robust CE
 4: while $\mathcal{T}$.queryNextNeighbour($x$) is not $\emptyset$ do
 5:     $x'_{nn} \leftarrow \mathcal{T}$.queryNextNeighbour($x$)
 6:     if $\mathcal{I}_{(\theta,\Delta)}(x'_{nn}) = 1 - c$ then  $x' \leftarrow x'_{nn}$
 7: if optimal is T then
 8:     while $a > 0$ do
 9:         $x'_{line} \leftarrow ax' + (1 - a)x$;  $a \leftarrow a - s$
10:         if $\mathcal{I}_{(\theta,\Delta)}(x'_{line}) = 1 - c$ then  $x' \leftarrow x'_{line}$
11: return $x'$
```

We stress that when `robustInit=T` and `optimal=F`, once $\mathcal{T}$ has been fitted, the CE generation time (Algorithm 2, 10) for any input is $O(log(|\mathcal{D}|))$, and $\mathcal{T}$ can be used to query CEs for any number of inputs efficiently. When the number of inputs requiring CEs is far smaller than $|\mathcal{D}|$, `robustInit=F` may result in faster computation than `robustInit=T` (including time for obtaining $\mathcal{T}$), see Appendix A for more details.

Remark 1. RNCE is sound.

Soundness of the procedure is guaranteed by construction, as solutions can only be chosen among the set of instances for which the classification label flips (see Algorithm 3). This is also true when RNCE is instantiated with `optimal` set to `T`, as an additional check is performed in line 10 of Algorithm 4 to ensure that only valid CEs are returned.

Remark 2. RNCE is complete if there exists an $(x', y') \in \mathcal{D}$ such that $\mathcal{I}_{(\theta,\Delta)}(x') \neq \mathcal{M}_\theta(x)$.

Completeness of the procedure is conditioned on the existence of at least one Δ-robust instance x' in $\mathcal{D}$ whose label differs from that of the original input x. Completeness is not affected by the configurations of parameters in RNCE. When `robustInit` is `T`, Algorithm 3 is guaranteed to identify such input as a candidate CE and add it to set $\mathcal{S}$; when `robustInit` is `F`, the multiple queries of the next nearest neighbour and the following Δ-robustness test (Algorithm 4, lines 4-6) will also identify a feasible result. For the parameter `optimal`, Algorithm 4 will always return one among x' or an optimised version of it (in terms of distance) for which robustness is still guaranteed (Algorithm 4, 10). Our experimental analysis in Section 8 shows that our approach is always able to find Δ-robust CEs.

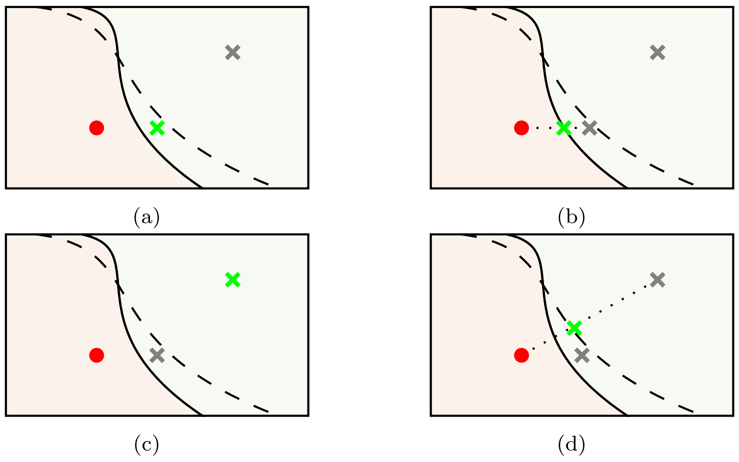


Fig. 5. RNCE behaviours when `optimal` = F, $\Delta = \emptyset$ (5a); `optimal` = T, $\Delta = \emptyset$ (5b); `optimal` = F, $\Delta \neq \emptyset$ (5c); `optimal` = T, $\Delta \neq \emptyset$ (5d). Each figure depicts the CE that is chosen (green cross) among a set of candidate CEs (grey crosses) for a given input (red circle) and model $\mathcal{M}_\theta$. The continuous curved line represents the decision boundary of $\mathcal{M}_\theta$; the dashed line represents a possible change in the decision boundary under Δ. Figs. 5a and 5b show configurations under which RNCE may return CEs that are not Δ-robust, whereas Figs. 5c and 5d depict robust ones.

Table 1
Dataset and classifier details.

dataset	data points	attributes	NN accuracy	LR accuracy
adult	48832	13	.847±.006	.828±.004
compas	6172	7	.844±.010	.833±.011
gmc	115527	10	.860±.001	.852±.002
heloc	9871	21	.728±.015	.725±.013

Remark 3. RNCE is equivalent to a plain NNCE algorithm if `optimal` is F and $\Delta = \emptyset$.

RNCE collapses to a plain NNCE algorithm if the set of plausible model shifts Δ is an empty set, thus $\mathcal{I}_{(\theta,\Delta)}$ will be equivalent to the original model, $\mathcal{M}_\theta$. Then, the choice of the parameter `robustInit` makes no impact as lines 8 and 9 of Algorithm 3 become identical (the multiple queries specified in Algorithm 4, 4-6 are also not needed). Note that the line search controlled by `optimal` can also be performed when $\Delta = \emptyset$.

Fig. 5 shows a pictorial representation of the different behaviours that can be obtained from RNCE based on the configuration of the parameter `optimal` and whether Δ is empty. When $\Delta = \emptyset$ and `optimal` is F (Fig. 5a), the algorithm behaves like a standard algorithm producing NNCEs and returns the closest counterfactual instance. However, this CE may not be robust. Analogous results may be obtained when $\Delta = \emptyset$ and `optimal` is T (Fig. 5b). Conversely, when $\Delta \neq \emptyset$ (Figs. 5c and 5d), RNCE will only operate on robust instances as candidates, thus guaranteeing the CEs' Δ-robustness.

8. Experiments

In this section, we demonstrate through experiments how Δ-robustness can be used in practice. We first described the experimental setup used for our experiments, and present two strategies to find realistic magnitudes in the plausible model changes given a dataset and the corresponding original classifier. Then, using these magnitudes, we construct interval abstractions to test the robustness of existing CE generation methods, showing a lack of Δ-robustness in state-of-the-art methods. Finally, we benchmark the proposed algorithms to compute Δ-robust CEs with evaluation metrics for proximity, plausibility, and robustness, showing the effectiveness of our methods.

Additionally, we demonstrate the applicability of our methods to multi-class classification tasks, while most existing methods focus on obtaining robust CEs only for binary classifiers. Finally, we present experiments aimed at benchmarking the runtime performance of our procedures and compare them with existing approaches.

The code for our implementation and experiments is available at:

https://github.com/junqi-jiang/interval-abstractions.

8.1. Setup and evaluation metrics

We experiment on four popular tabular datasets for benchmarking performances of CE generation algorithms, *adult* income dataset [60], *compas* recidivism dataset [61], give me some credit (*gmc*) dataset [62], Home equity line of credit (*heloc*) dataset [63], from the CARLA library [64]. All datasets contain min-max scaled continuous features, one-hot encoded binary discrete features, and two output classes. Class 0 is the unwanted class while class 1 is the target class for CEs. We train neural networks with two hidden layers and 6 to 20 neurons in each layer, and logistic regression models (used in Section 8.4) on the datasets. Table 1 reports the dataset sizes, the number of attributes of the datasets, and the 5-fold cross-validation accuracy of the classifiers obtained.

We randomly split each dataset $\mathcal{D}$ into two halves, $\mathcal{D}_1$ and $\mathcal{D}_2$, each including a training and a test set. We use $\mathcal{D}_1$ to train the classifiers as stated above, and we call these original models $\mathcal{M}_1$. $\mathcal{D}_2$ is used to simulate scenarios with incoming data after $\mathcal{M}_1$ are deployed, explained in the next sections.

8.2. Identifying δ values

Recall from Definition 5 that δ upper-bounds the magnitude of the model shifts in Δ, as measured by the p-distance (Definition 3). When practically using Δ-robustness, δ, values, regarded as the hyperparameter in our method, need to be first determined. We now propose two realistic strategies for obtaining δ values, depending on how the underlying classifier is retrained with new data in the application.

Incremental retraining. This refers to the case where $\mathcal{M}_1$ is periodically fine-tuned by gradient descent on some portions of $\mathcal{D}_2$ with a small number of iterations. In this scenario, depending on how many data points in $\mathcal{D}_2$ are used for retraining, the magnitude of parameter changes could be small. Therefore, the δ values can be directly estimated by calculating the p-distance between $\mathcal{M}_1$ and the updated classifiers. By observing the p-distances when retraining on various portions of $\mathcal{D}_2$, the model developers could potentially link the estimated δ values to time intervals in real-world applications, depending on the frequency at which new data are collected.

Complete or leave-one-out retraining. Complete retraining refers to a scenario where a model is retrained with the same hyperparameter setting as $\mathcal{M}_1$, using the concatenation of $\mathcal{D}_1$ and $\mathcal{D}_2$. *Leave-one-out retraining* concerns obtaining a new model using a subset of $\mathcal{D}_1$ with 1% of data points removed. As mentioned in [17], when retraining from scratch using the concatenation of $\mathcal{D}_1$ and $\mathcal{D}_2$, it becomes unrealistic to upper-bound the weights and biases differences in classifiers as the p-distance can be arbitrarily large. In this case, we can treat δ as a hyperparameter and empirically find the minimum value which can ensure robustness on a validation set. Similarly to the standard train-validation-test split for evaluating the accuracy of machine learning models, we propose to use a held-out validation set to estimate δ values which lead to sufficient robustness under such retraining scenarios. The procedure can be summarised as follows:

1. Retrain from scratch some new classifiers using $\mathcal{D}_1$ and $\mathcal{D}_2$,[4] set initial δ value to a sufficiently small value;
2. Generate Δ-robust CEs using the current δ;
3. Evaluate the percentage of the explanations which are valid under all the retrained models;
4. Examine the above empirical validity, if it has not reached 100% then increase the δ value and repeat steps 2-3. Choose the smallest δ value which results in 100% validity.

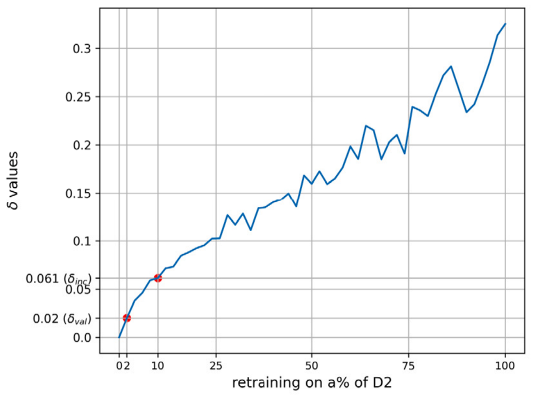
The termination condition in step 4 balances the robustness-cost tradeoff. Once the empirical validity on multiple retrained models reaches 100% for the validation set, increasing δ further will negatively affect the proximity evaluations. It is also expected that when finding Δ-robust CEs with the same δ value for the test set, a similar level of robustness can be observed.

We report the δ value results using both strategies in Fig. 6, referred to as δ_{inc} (incremental retraining) and δ_{val} (validation set). Specifically, for the first scenario, we record and plot δ values as the average ∞-distances (in the experiments, we use $p = \infty$) between $\mathcal{M}_1$ and five incrementally retrained[5] models using increasing sizes of the retraining dataset (a% of $\mathcal{D}_2$). As can be observed, δ values increase with slight fluctuations as incrementally retraining on more data. The magnitudes are classifier- and dataset-dependent, though in our setting we obtain values from 0 to about 0.3. The δ_{inc} values are obtained by recording the δ values when retraining on 10% of $\mathcal{D}_2$, and use them as one robustness target in the next experiments.

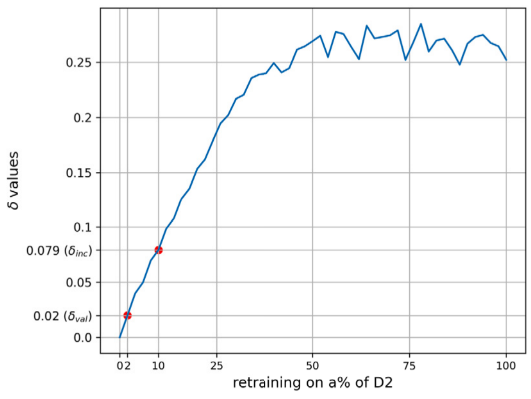
For the second scenario, we use five completely retrained and five leave-one-out retrained classifiers as the new models in Step 1. We use RNCE-`FF` to find Δ-robust CEs in Step 2. We record the obtained δ_{val} as another robustness target, and we highlight these values in the same plots in Fig. 6. After matching their magnitudes to the ones obtained for incremental retraining, we identify that these δ_{val} are much smaller than δ_{inc} and they correspond to the magnitudes of retraining on only 2% of $\mathcal{D}_2$. The fact that being Δ-robust against a very small magnitude results in 100% empirical robustness (validity) in a validation set confirms the conservative nature of Δ-robustness, as it guarantees the CE's robustness against all possible model shifts entailed by Δ (Lemma 2).

[4] Complete retraining is possible in an experimental environment. In practice, however, if $\mathcal{D}_2$ is not available at the time of robust CE generation, leave-one-out retraining on $\mathcal{D}_1$ could be a viable option in step 1.

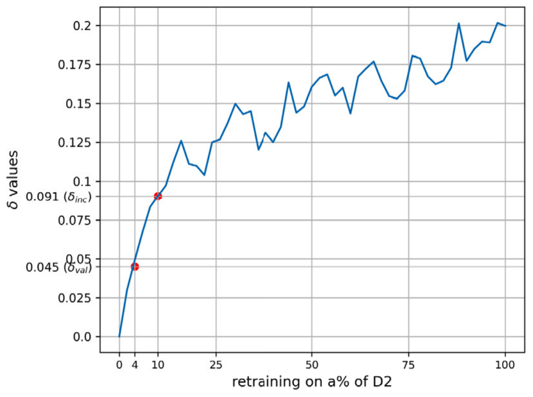
[5] We re-implemented the partial_fit function in Scikit-learn library: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html%23sklearn.neural_network.MLPClassifier.partial_fit.



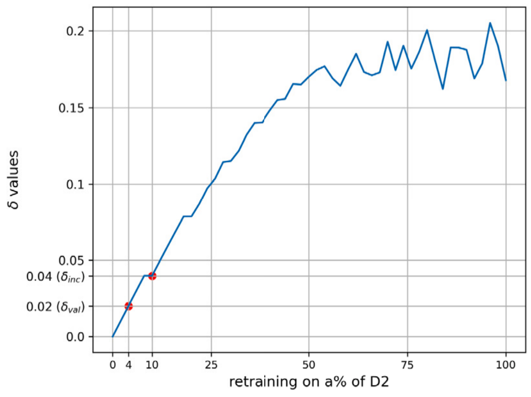
(a) adult dataset



(b) compas dataset



(c) gmc dataset



(d) heloc dataset

Fig. 6. δ values obtained by retraining on increasing portions of D_2. The red dots highlight the δ_{val} and δ_{inc} values found by the two strategies described in Section 8.2.

8.3. *Verifying Δ-robustness*

In this experiment, we demonstrate how our MILP procedure can be used as an evaluation tool to examine the robustness of CEs generated by SOTA algorithms. We consider three traditional non-robust baselines, namely a gradient-descent based method *GCE* similar to [6], a plausible method using gradient descent *Proto* [23], and a MILP-based method (referred to as MCE) inspired by [20]. We also include *ROAR* [10], a SOTA framework specifically designed to generate robust CEs, focusing on robustness against the same notion of plausible model changes.[6] For our algorithms, using the iterative algorithm (Algorithm 1) proposed in Section 7, we devise robustified versions of the non-robust baselines, which we call *GCE-R*, *PROTO-R*, and *MCE-R*. We also include our RNCE-`FF` algorithm (Algorithm 2, `robustInit=False`, `optimal=False`) to show the guaranteed robustness results of this method.

For each dataset, we randomly select 50 test inputs for which we use the above baselines to generate CEs. We evaluate their robustness against model shifts of magnitudes up to δ_{inc} using Δ-*validity*, the percentage of test inputs whose CEs are Δ-robust. For all robust methods, their targeted Δ values are instantiated with δ_{inc}.

Figs. 7 (a-d) report the results of our analysis for the four datasets. As we can observe, all methods generate CEs that tend to be valid for the original model. For the non-robust baselines, Δ-validities soon drop to value 0 as small model shifts are applied, revealing a lack of robustness for these baselines. ROAR exhibits a higher degree of Δ-robustness, as expected. However, its heuristic nature does not allow to reason about all possible shifts in Δ, which affects the Δ-robustness of CFXs as δ grows larger. Also, the fact that it uses a local surrogate model to approximate the behaviour of neural networks could negatively affect the results. For compas and gmc datasets, its Δ-robustness stays lower than 100%.

For the two gradient-based non-robust baselines, our robustified versions (GCE-R, PROTO-R) successfully improved their robustness against small model shifts with lower δ values, however, the Δ-robustness tends to drop (drastically for adult dataset) when it increases near δ_{inc}. This is likely due to the vulnerability to local optimum solutions for the gradient-descent algorithms. The MILP-based method MCE-R which gives more exact solutions for the problem always finds robust CEs with 100% Δ-validity. With proven robustness guarantees, our RNCE algorithm also finds CEs that are 100% robust.

[6] Differently to our previous work [15], we use the implementation of ROAR in CARLA library [64] which allows more comprehensive hyperparameter tuning.

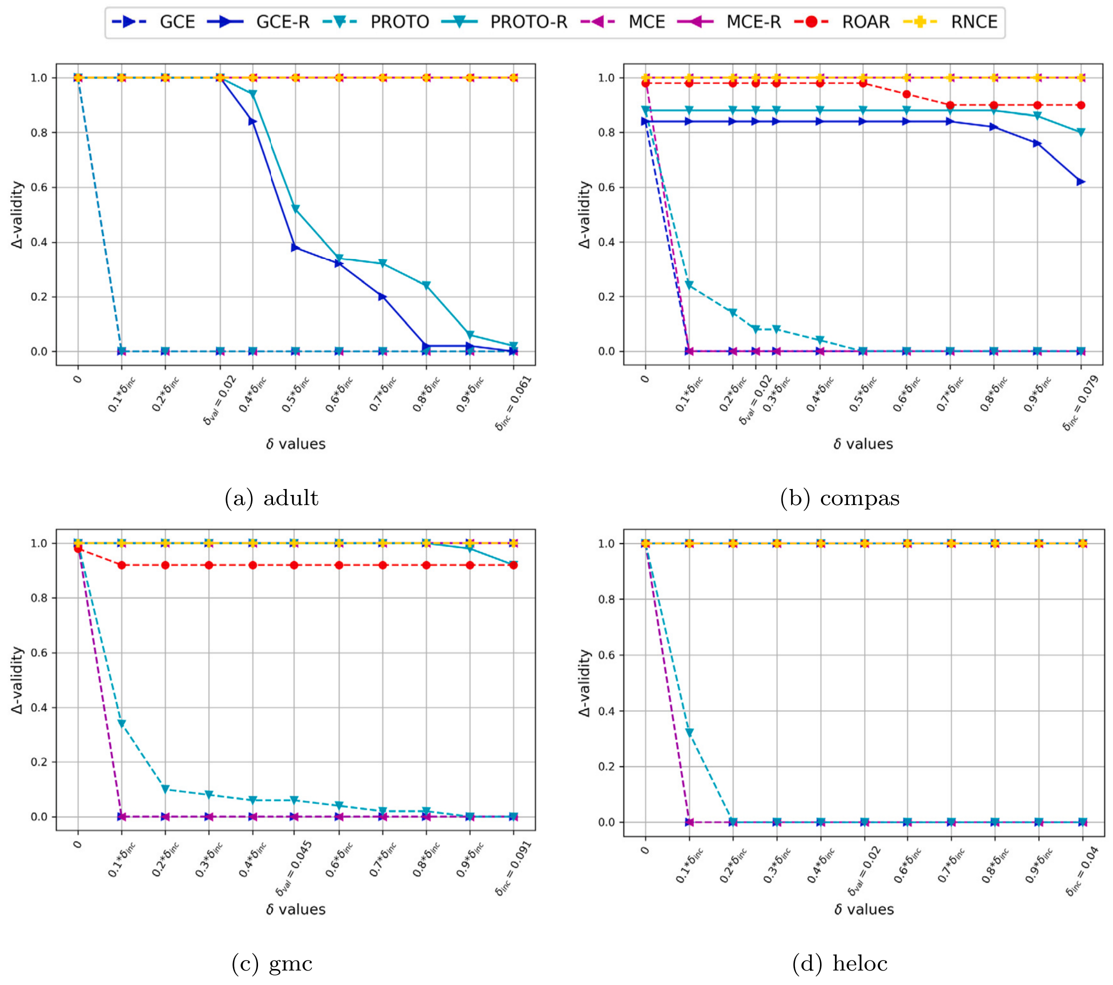


Fig. 7. Evaluation of Δ-validity (against increasing δ values) of the CEs found by state-of-the-art methods (GCE, PROTO, MCE, ROAR) and by our methods (GCE-R, PROTO-R, MCE-R, RNCE).

8.4. *Generating Δ-robust CEs*

Next, we rigorously benchmark the performance of our CE generation methods against various robust and non-robust baselines. Apart from the CE methods used in Section 8.3, we additionally include NNCE [21], *RBR* [14], and *ST-CE* [17]. We refer the reader to Section 2.2 for their details.[7] We also instantiate RNCE-FT as one of our methods. The properties of all compared methods are summarised in Table 2.

For each dataset, we randomly select 20 test points from the test set to generate CEs using each method. We repeat the process five times with different random seeds and report the mean and standard deviation of the results. The CEs are evaluated against three aspects using the standard metrics in the literature. For proximity, we calculate the average ℓ_1 *distance* between the test input and its CE, which captures both closeness of CEs and sparsity of changes [6]. For plausibility, we report the average local outlier factor score *lof* [65] which quantifies the local data density. A lof score close to value 1 indicates an inlier; the more it deviates from 1, the less plausible the CE. For robustness, we report validity after retraining *vr*, i.e. the percentage of CEs correctly classified to class 1, under 15 retrained classifiers using respectively complete retraining, leave-one-out retraining, and incremental retraining (with 10% new data). We use the same δ_{val} and δ_{inc} obtained from Section 8.2 to instantiate Δ with different model shift magnitudes and report the respective Δ-robustness, termed $v\Delta_{val}$ and $v\Delta_{inc}$.

Table 3 reports the mean results for neural network classifiers of the benchmarking study. See Appendix B for the standard deviation results and the evaluations for logistic regression classifiers. Next, we analyse the results by their properties, and for our methods, we first consider the results when targeting δ_{val}.

[7] We only included methods which either have open-source implementations or could be re-implemented without requiring excessive effort.

Table 2
Properties (validity, proximity, plausibility, robustness) addressed by baselines and whether the methods apply to neural network (NN) or logistic regression (LR), indicated by ✓.

Method	Properties				Classifiers	
	Validity	Proximity	Plausibility	Robustness	NN	LR
GCE [6]	✓	✓			✓	
PROTO [23]	✓	✓	✓		✓	
MCE [20]	✓	✓			✓	
NNCE [21]	✓	✓	✓		✓	✓
RBR [14]	✓	✓	✓	✓	✓	
ROAR [10]	✓	✓		✓	✓	✓
ST-CE [17]	✓	✓	✓	✓	✓	
GCE-R (ours)	✓	✓		✓	✓	
PROTO-R (ours)	✓	✓	✓	✓	✓	
MCE-R (ours)	✓	✓		✓	✓	
RNCE (ours)	✓	✓	✓	✓	✓	✓

Table 3
Quantitative evaluation of the compared CE generation methods on neural networks. The ↑ (↓) following each metric indicates the higher (lower) the value, the better. Methods are separated by horizontal lines, indicating non-robust baselines, robust baselines, and our methods with robustness target Δ instantiated by δ_{val} and δ_{inc}, respectively.

	vr↑	vΔ_{val}↑	vΔ_{inc}↑	ℓ_1↓	lof↓	vr↑	vΔ_{val}↑	vΔ_{inc}↑	ℓ_1↓	lof↓
	adult					compas				
GCE	51%	0%	0%	.016	1.29	26.5%	0%	0%	.039	3.05
PROTO	61.1%	1%	0%	.011	1.44	50.7%	6.6%	0%	.144	1.66
MCE	48.5%	0%	0%	.009	1.41	25.6%	0%	0%	.019	1.79
NNCE	76.1%	2%	2%	.032	1.34	43.3%	8%	0%	.028	1.30
ROAR	100%	100%	94.8%	.877	12.5	100%	100%	94.7%	.388	8.44
RBR	90.1%	0%	0%	.025	1.33	98.3%	38%	0%	.038	1.53
ST-CE	98.7%	20%	4%	.046	1.27	99.9%	74%	0%	.039	1.23
	target $\delta_{val} = 0.02$					target $\delta_{val} = 0.02$				
GCE-R	100%	100%	0%	.048	1.47	86.9%	87%	0%	.055	3.04
PROTO-R	100%	69%	0%	.042	1.68	91%	91%	0%	.049	1.74
MCE-R	100%	100%	0%	.021	1.65	99.8%	100%	0%	.035	1.68
RNCE-FF	100%	100%	4%	.057	1.32	100%	100%	0%	.039	1.26
RNCE-FT	100%	100%	0%	.049	1.28	100%	100%	0%	.037	1.33
	target $\delta_{inc} = 0.061$					target $\delta_{inc} = 0.079$				
GCE-R	100%	100%	0%	.051	1.65	87%	87%	67%	.109	3.62
PROTO-R	100%	100%	9%	.072	2.36	91%	91%	84%	.108	1.99
MCE-R	100%	100%	100%	.051	2.91	100%	100%	100%	.096	2.81
RNCE-FF	100%	100%	100%	.122	2.78	100%	100%	100%	.088	1.11
RNCE-FT	100%	100%	100%	.095	2.70	100%	100%	100%	.088	1.11
	gmc					heloc				
GCE	75.8%	0%	0%	.022	1.52	20.5%	0%	0%	.019	1.18
PROTO	89.1%	1%	0%	.023	1.36	39.5%	0%	0%	.024	1.16
MCE	63.3%	0%	0%	.016	1.40	22%	0%	0%	.014	1.40
NNCE	88.9%	22%	1%	.029	1.23	35.9%	0%	0%	.053	1.05
ROAR	99.3%	98%	98%	.199	23.1	100%	100%	100%	.454	6.94
RBR	100%	62%	0%	.034	1.55	58.7%	0%	0%	.038	1.08
ST-CE	100%	92%	6%	.041	1.10	100%	40%	0%	.078	1.04
	target $\delta_{val} = 0.02$					target $\delta_{val} = 0.02$				
GCE-R	100%	100%	0%	.032	1.79	100%	100%	0%	.049	1.32
PROTO-R	100%	100%	0%	.036	1.45	100%	100%	11%	.079	1.62
MCE-R	100%	100%	0%	.022	1.48	100%	100%	0%	.031	1.94
RNCE-FF	100%	100%	7%	.040	1.07	100%	100%	0%	.083	1.04
RNCE-FT	100%	100%	0%	.035	1.36	100%	100%	0%	.080	1.04
	target $\delta_{inc} = 0.091$					target $\delta_{inc} = 0.04$				
GCE-R	100%	100%	100%	.053	3.43	100%	100%	100%	.109	2.07
PROTO-R	100%	100%	100%	.118	2.49	100%	100%	100%	.163	2.41
MCE-R	100%	100%	100%	.032	1.86	100%	100%	100%	.049	3.04
RNCE-FF	100%	100%	100%	.084	1.22	100%	100%	100%	.150	1.13
RNCE-FT	100%	100%	100%	.084	1.22	100%	100%	100%	.150	1.13

Our methods produce the most robust CEs. Our RNCE algorithm (both configurations) generates the most robust CEs among the compared methods, showing 100% vr and 100% targeted Δ-validity. For the robustified methods using Algorithm 2, MCE-R is the most robust, finding perfectly Δ-robust and 100% empirically robust CEs in most experiments. As discussed in Section 8.3, the limited search space might be the cause of the reduced robustness for GCE-R and PROTO-R in two datasets. As a result, when compared with the robust baselines, RNCE and MCE-R both give better robustness than ROAR, ST-CE, and RBR. All robust methods have better robustness performances than the non-robust baselines, as expected.

Cost-robustness tradeoff. This tradeoff has been discussed in several other studies [10,66], which we have also empirically observed. The non-robust baselines always find the most proximal CEs (lowest $\ell 1$ costs). Apart from them, the next best proximity results were obtained by MCE-R (when targeting the smaller δ_{val}) among all the robust methods. Considering that MCE-R also finds near-perfectly Δ-robust CEs, it can be concluded that MCE-R effectively balances the cost-robustness tradeoff. RBR also finds CEs with low costs, but their method is not as robust. Similar remarks can be made for ST-CE as this method moves more towards the robustness end of the tradeoff. Our methods GCE-R, PROTO-R, RNCE demonstrate similar $\ell 1$ costs. Setting `optimal=True` in our RNCE algorithm slightly improves the proximity, as can be seen when comparing RNCE-`FF` and RNCE-`FT`. ROAR results in CEs with high $\ell 1$ cost, and the method has been identified as being overly costly [34] due to their gradient-based robust optimisation procedure.

Plausibility results. ST-CE has the best lof scores among all the methods, followed by RNCE which yields comparable results. This is because these two methods select CEs from in-manifold dataset points and are thus unlikely to return outliers. By inherently addressing data density estimation, RBR also finds plausible CEs. The plausibility performances of our three robustified methods and ROAR are not as good due to less regulated search spaces or lack of plausibility constraints, with ROAR having the worst lof results.

The effects of robustification. For Algorithm 1, robustifying GCE, PROTO, and MCE resulted in improved empirical and Δ-robustness, but this negatively affected the proximity and plausibility results. Targeting larger-magnitude plausible model shifts (instantiated with δ_{inc}) pushes these tradeoffs further. For RNCE, similar trends can be identified when compared with NNCE, but in most cases, the lof score improves. When targeting δ_{inc} instead of δ_{val}, the cost also increases together with robustness, but plausibility stays comparable.

Concluding remarks. From the analysis above, we can conclude that MCE-R achieved the best robustness-cost tradeoff, finding the most proximal CEs among the robust baselines while showing near-perfect robustness results, outperforming all robust baselines. RNCE, on the other hand, finds CEs with even stronger robustness guarantees and great plausibility, at slightly higher costs. However, less costly methods than RNCE are not as robust.

8.5. Computation time analysis

In this section, we discuss the computation time required by each method to obtain the results in Section 8.4. The average computation times for generating CEs for 20 test points are presented in Table 4.

While model complexity impacts all methods, the runtimes of gradient- and MILP-based optimisation methods that are not wrapped in an iterative approach (GCE, PROTO, MCE, RBR) are mostly sensitive to the number of attributes in the dataset, and those iteratively computed for robustness (ROAR, GCE-R, PROTO-R, MCE-R) are also easily affected by their hyperparameters concerning robustness. The methods that require traversing (part of) the training dataset (NNCE, RNCE, ST-CE) are sensitive to the number of data points in each dataset.

MCE-R is the fastest among the methods we devised, showing better runtime than ROAR and RBR when targeting a smaller validation δ, and producing close CEs with robustness guarantees. On the other hand, our PROTO-R method is the slowest, while it also fails to give the best CEs in terms of the desired properties (Table 3). Similar remarks can be made for GCE-R, which has comparable runtime as RBR. These three robustified methods (GCE-R, PROTO-R, MCE-R) largely inherit the runtime of their respective base method (GCE, PROTO, MCE). For RNCE, the number of attributes and model complexity largely impact the time required to compute the solution for a single MILP program, while the number of dataset points and targeted δ value jointly impact the number of MILP programs that need to be checked. Therefore, the RNCE runtime varies to a large extent in our experiments. On logistic regression classifiers, our approach is faster than the other robust baseline, ROAR. On neural networks, for a smaller validation δ, the runtimes are in the same order of magnitude as ROAR and are much faster than RBR, but they deteriorate when switching to a larger δ_{inc}.

We conclude that our best performing methods, MCE-R and RNCE, when targeting the δ values found by the validation set method, are able to find better-quality and more robust CEs than the robust baselines while demonstrating comparable runtimes. When trying to achieve robustness guarantees for higher δ values, worse runtimes can be observed. This result is expected, given that proving robustness with respect to model changes has been shown to be an NP-complete problem [67].

8.6. Multi-class classification

Next, we demonstrate the applicability of our RNCE method to multi-class classification settings, which is not supported by any robust CE generation baselines listed in Table 2. We use the Iris dataset [68,69], a small-scale dataset for three-class classification tasks, and the California Housing dataset [70], in which we transform the regression labels into three classes. Both datasets are available in sklearn [71]. We trained two neural network models with cross validation accuracy of 0.93 and 0.70, respectively. Again, we randomly select (five times with different random seeds) 20 test instances from the test sets which are classified to class 0, and we specify a desired label 2 for CEs. Following the same experimental protocol described in Section 8.4, we use the validation

Table 4
Computation time (in seconds) of the compared methods on logistic regression models and neural networks. The adult dataset has 48832 data points with 13 attributes, the compas dataset has 6172 data points with 7 attributes, the gmc dataset has 115527 data points with 10 attributes, the heloc dataset has 9871 data points and 21 attributes.

Model	Method	adult	compas	gmc	heloc
Logistic Regression	NNCE	0.039	0.024	0.636	0.003
	ROAR	11.70	11.49	4.958	13.71
	RNCE-FF δ_{val}	0.410	0.195	0.755	27.71
	RNCE-FF δ_{val}	1.554	1.035	1.722	29.75
	RNCE-FF δ_{inc}	0.627	0.948	0.754	1.534
	RNCE-FF δ_{inc}	1.779	2.061	1.681	3.522
Neural Network	GCE	27.23	29.06	25.15	25.08
	PROTO	604.3	555.8	542.2	607.6
	MCE	0.934	0.340	0.163	1.221
	NNCE	0.095	0.018	0.514	0.039
	ROAR	20.03	2.535	1.740	2.007
	RBR	187.2	118.4	112.2	110.5
	ST-CE	0.579	1.29	0.347	1.419
	GCE-R δ_{val}	101.7	56.97	86.32	67.33
	PROTO-R δ_{val}	1602	756.9	690.5	965.2
	MCE-R δ_{val}	18.77	2.364	1.281	13.05
	RNCE-FF δ_{val}	15.64	4.125	3.447	32.76
	RNCE-FT δ_{val}	27.96	6.744	5.307	43.50
	GCE-R δ_{inc}	155.2	117.08	116.01	120.9
	PROTO-R δ_{inc}	1920	1824	1625	1469
	MCE-R δ_{inc}	37.45	9.152	2.771	19.43
	RNCE-FF δ_{inc}	218.3	48.07	163.2	670.6
	RNCE-FT δ_{inc}	229.9	51.51	165.5	681.6

Table 5
Quantitative evaluation of the compared methods on neural networks in multi-class classification tasks. The evaluation metrics are the same as in Table 3.

Dataset	Method	vr↑	vΔ_{val}↑	ℓ_1↓	lof↓
iris $\delta_{val} = 0.015$	NNCE	75%±.004	0%±.0	.393±.002	1.48±.026
	RNCE-FF	100%±.0	100%±.0	.438±.003	1.50±.002
	RNCE-FT	100%±.0	100%±.0	.438±.003	1.50±.002
housing $\delta_{val} = 0.040$	NNCE	18.1%±.035	0%±.0	.032±.003	1.10±.020
	RNCE-FF	100%±.0	100%±.0	.053±.004	1.33±.037
	RNCE-FT	100%±.0	100%±.0	.052±.004	1.31±.035

set strategy to identify δ values as our targeted plausible model changes magnitude, and we report the same evaluation metrics for NNCE and RNCE.

Table 5 presents the results we obtained. Similarly to the results for binary classification, the CEs computed by the NNCE method are not robust against realistically retrained models (indicated by the low **vr**). Measured by the Δ-validity, they also fail to satisfy the Δ-robustness tests for multi-class classifications. With a slight tradeoff with ℓ_1 costs and lof scores, our two configurations of RNCE are both able to find perfectly robust CEs in this study. This demonstrates that the Δ-robustness notion can be used to evaluate provable robustness guarantees for CEs, and our proposed algorithms succeed at finding more robust CEs in the multi-class classification setting.

9. Discussion and future work

Despite the recent advances in achieving robustness against model changes for CEs, state-of-the-art lack rigorous robustness guarantees on the CEs they produce. By introducing a formal robustness notion, Δ-robustness, and a novel interval abstraction technique, we provided the first method in the literature to obtain CEs with certified robustness guarantees. We showed how such Δ-robustness can be practically tested in binary and multi-class classification settings by solving optimisation problems via MILP. Furthermore, we proposed two algorithms to generate CEs that are provably Δ-robust. To demonstrate how our methods can be used in practice, two strategies for identifying the appropriate hyperparameters in our method have been investigated, linking to three model retraining strategies. We then presented an extensive empirical evaluation involving eleven algorithms for generating CEs, including seven algorithms specifically designed to generate robust explanations. Our results show that our MCE-R algorithm finds CEs with the lowest costs along with the best robustness results, and our RNCE outperforms existing approaches and is able to generate CEs that are both provably robust and plausible, achieving a balance in the robustness-cost and plausibility-cost tradeoffs. We also compared the runtimes and observed that our best-performing methods have comparable runtimes with the robust baselines when targeting small δ values. We see these outcomes as important contributions towards complementing existing formal approaches for XAI and making them applicable in practice.

Despite the positive results obtained here, we also identified a number of limitations of our approach. Firstly, white-box access to both the training dataset and the classifiers is required. As a result, our approach is primarily aimed at expert users, e.g. model developers. Secondly, our method only works for parametric machine learning classifiers whose forward pass can be precisely represented by a MILP program. We have considered two examples in this paper, but other models could be considered. Thirdly, the computation time for our method, when targeting high δ values on complex models and large datasets, can be high as that for providing exact robustness guarantees for CEs under model changes, which is an NP-complete problem [67]. Nevertheless, in this work we demonstrated that our method leads to better-quality CEs than existing methods.

This work opens up several promising avenues for future work. One such direction is to investigate relaxed forms of robustness, e.g., when the output intervals in $\mathcal{I}_{(\theta,\Delta)}$ for different classes overlap, with similar interval abstraction techniques. In this work, we considered the deterministic robustness guarantees, aiming at ensuring CEs' validity against even worst-case model parameter perturbations encoded by Δ. However, we observed that this notion is conservative in that the worst-case perturbations might not always occur in realistic model retraining. One potential way to overcome this is to determine the hyperparameter δ which controls the magnitude of model changes in Δ using a validation set, as illustrated in Section 8.2. However, that requires additional computational efforts. Therefore, probabilistic relaxations of our Δ-robustness notion allowing more fine-grained analyses would be valuable.

Another possibility is to investigate the Δ-robustness of CEs in a causal setting. CEs which conform to some structural causal models are usually within data distribution, and they reflect the true causes of predictions, making causality a desirable property [26,27]. It has also been found that plausible CEs are likely to be more robust against model shifts [41]. Therefore, there could be intrinsic links between causality and robustness. Dominguez-Olmedo et al. [44] have proposed a framework to compute CEs robust to noisy executions under a causal setting. It would be interesting to also investigate the interplay of Δ-robustness and causality to facilitate the development of higher-quality CEs.

Finally, it would be possible to apply similar interval abstraction techniques to provide robustness guarantees for other forms of CE robustness. In our Δ-robustness tests, a fixed-value CE is fed into the interval-valued abstraction of a classification model. Intuitively, if instead an interval-valued CE is passed as input into a fixed-valued classification model, it would be similar to reasoning about the robustness of CEs against noisy executions. Going further, it would be interesting to investigate training techniques incorporating Δ-robustness notions to train models which can produce robust CEs by plain gradient-based optimisation methods.

CRediT authorship contribution statement

Junqi Jiang: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Francesco Leofante:** Writing – review & editing, Writing – original draft, Validation, Project administration, Formal analysis, Conceptualization. **Antonio Rago:** Writing – review & editing, Validation. **Francesca Toni:** Writing – review & editing, Validation, Supervision.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Junqi Jiang, Antonio Rago, Francesca Toni reports financial support was provided by JP Morgan Chase Bank. Antonio Rago, Francesca Toni reports financial support was provided by European Research Council. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

I have shared the link to my code and data used

Acknowledgements

Jiang, Rago and Toni were partially funded by J.P. Morgan and by the Royal Academy of Engineering under the Research Chairs and Senior Research Fellowships scheme. Leofante was funded by Imperial College London through under the Imperial College Research Fellowship scheme. Rago and Toni were partially funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 101020934). Any views or opinions expressed herein are solely those of the authors listed.

Table A.6
RNCE computation time (seconds) of generating robust CEs for 100 randomly selected inputs using neural networks (NN) or logistic regressions (LR), when configured with different `robustInit` settings.

configuration	adult		compas		gmc		heloc	
	NN	LR	NN	LR	NN	LR	NN	LR
RNCE-FF	7.69	0.47	11.47	4.34	4.33	0.41	55.83	0.49
RNCE-TF incl. time for obtaining $\mathcal{T}$	59.55	3.84	20.21	1.77	268.35	43.20	44.23	3.17
RNCE-TF excl. time for obtaining $\mathcal{T}$	0.002	0.002	0.001	0.001	0.008	0.007	0.001	0.002

Table A.7
RNCE computation time (seconds) of generating robust CEs for all data points in the training dataset that are classified to class 0 using neural networks (NN) or logistic regressions (LR), when configured with different `robustInit` settings. The numbers following the dataset names represent the approximate number of inputs for each dataset.

Configuration	adult (15500)		compas (200)		gmc (700)		heloc (2000)	
	NN	LR	NN	LR	NN	LR	NN	LR
RNCE-FF	1306	59.55	27.16	5.63	42.44	1.22	1052	9.79
RNCE-TF incl. time for obtaining $\mathcal{T}$	59.75	4.15	20.21	1.77	268.42	43.21	44.24	3.19
RNCE-TF excl. time for obtaining $\mathcal{T}$	0.21	0.31	0.001	0.001	0.08	0.02	0.01	0.02

Appendix A. Computation time of RNCE and the impact of `robustInit`

Since querying for the nearest neighbour from a k-d tree is in logarithm time complexity wrt the tree size, the computation time bottleneck of RNCE is the total time for the more complex Δ-robustness tests. Therefore, the computation time of RNCE mainly depends on two factors, the time required for each Δ-robustness test, and the number of Δ-robustness tests performed in total. For the former factor, each Δ-robustness test is a MILP program whose problem size, affected by the number of attributes in the dataset and the architecture of the classifier in our setting, determines its computation time. The latter factor, however, is directly controlled by the algorithm parameter, `robustInit`. When `robustInit=F`, the number of Δ-robustness tests is the product of the number of inputs for which CEs are generated and the average number of times querying for the next nearest neighbour (Alg.3, lines 4-6) before reaching a Δ-robust CE, which is further affected by the value of δ constructing Δ. When `robustInit=T`, the number of Δ-robustness tests is upper-bounded by the dataset size.

We empirically compare the computation time when `robustInit=T` with `robustInit=F`. Specifically, since the line search controlled by the parameter `optimal` (Algorithm 4, lines 7-10) is independent of the impact of `robustInit`, we report results for `optimal=F`, i.e., RNCE-FF and RNCE-TF, under two different settings, in Tables A.6 and A.7.

The results empirically support our analysis. The computation times of both RNCE-TF and RNCE-FF for neural networks are much higher than those for logistic regressions due to the fact that neural networks are more complex in terms of model architecture. The computation time of RNCE-FF tends to be lower than RNCE-TF when the number of inputs is smaller because fewer Δ-robustness tests are required. When `robustInit=F`, the time is almost proportional to the number of inputs. When `robustInit=T`, the computation times are almost identical regardless of the number of inputs because building the k-d tree $\mathcal{T}$ is more time-consuming than querying for CEs. Our RNCE-TF excl. time for obtaining $\mathcal{T}$ results show that, when `robustInit=T`, after obtaining $\mathcal{T}$, the time needed for querying CEs is almost negligible, since for every input, the first query will be the closest Δ-robust CE.

Appendix B. Full experiment results

Standard deviation results accompanying Table 3 are presented in Table B.8.

For linear regression models, we find δ_{inc} and δ_{val} using the same strategy for neural networks as stated in Section 8.2. We quantitatively compare the CEs found by NNCE, ROAR, and two configurations of our algorithm RNCE using the same evaluation metrics introduced in Section 8.4. Tables B.9 and B.10 report the mean and standard deviation results for logistic regression classifiers.

Slightly different from the neural network results, the δ_{inc} values, found by incrementally retraining on 10% of $\mathcal{D}_2$, can be smaller than the δ_{val}, and sometimes insufficient to induce 100% empirical robustness (indicated by vr). Comparing RNCE with baselines, similar to the results for neural networks in Section 8.4, RNCE produces more robust CEs than the NNCE method, and is less costly and more plausible than ROAR. The impact of changing `optimal` to `True` is more obvious in this set of experiments, with ℓ_1 costs decreasing to a greater extent and plausibility remaining comparable to RNCE-FF.

Table B.8
Standard deviation results of the quantitative evaluation of the compared CE generation methods on neural networks.

	vr	$v\Delta_{val}$	$v\Delta_{inc}$	ℓ_1	lof	vr	$v\Delta_{val}$	$v\Delta_{inc}$	ℓ_1	lof
	adult					compas				
GCE	.223	0	0	.003	.036	.011	0	0	.007	.697
PROTO	.178	.020	0	.002	.089	.172	.055	0	.062	.192
MCE	.201	0	0	.001	.063	.126	0	0	.006	.047
NNCE	.139	.024	.024	.004	.114	.123	.024	0	.003	.057
ROAR	0	0	.065	.208	4.187	0	0	.033	.033	.574
RBR	.092	0	0	.002	.051	.008	.024	0	.003	.170
ST-CE	.014	.071	.020	.008	.087	.003	.128	0	.003	.057
	target $\delta_{val} = 0.02$					target $\delta_{val} = 0.02$				
GCE-R	0	0	0	.007	.079	.041	.040	0	.006	.664
PROTO-R	0	.203	0	.010	.119	.037	.037	0	.008	.299
MCE-R	0	0	0	.001	.099	.002	0	0	.003	.161
RNCE-FF	0	0	.020	.005	.021	0	0	0	.003	.062
RNCE-FT	0	0	0	.003	.045	0	0	0	.003	.060
	target $\delta_{inc} = 0.061$					target $\delta_{inc} = 0.079$				
GCE-R	0	0	0	.008	.130	.040	.040	.051	.005	.565
PROTO-R	0	0	.091	.053	.356	.037	.037	.058	.009	.244
MCE-R	0	0	0	.005	.054	0	0	0	.004	.235
RNCE-FF	0	0	0	.006	.168	0	0	0	.004	.086
RNCE-FT	0	0	0	.011	.085	0	0	0	.004	.086
	gmc					heloc				
GCE	.003	0	0	.005	.064	.042	0	0	.002	.045
PROTO	.012	.020	0	.004	.057	.052	0	0	.002	.036
MCE	.032	0	0	.003	.103	.053	0	0	.001	.034
NNCE	.027	.068	.020	.003	.031	.053	0	0	.003	.011
ROAR	.015	.025	.025	.014	9.69	0	0	0	.010	.164
RBR	0	.117	0	.004	.105	.070	0	0	.003	.022
ST-CE	0	.087	.058	.005	.068	0	.152	0	.004	.016
	target $\delta_{val} = 0.02$					target $\delta_{val} = 0.02$				
GCE-R	0	0	0	.006	.019	0	0	0	.003	.043
PROTO-R	0	0	0	.004	.019	0	0	.073	.002	.061
MCE-R	0	0	0	.003	.103	.002	0	0	.001	.043
RNCE-FF	0	0	.087	.004	.033	0	0	0	.002	.019
RNCE-FT	0	0	0	.004	.157	0	0	0	.003	.019
	target $\delta_{inc} = 0.091$					target $\delta_{inc} = 0.04$				
GCE-R	0	0	0	.006	.021	0	0	0	.010	.114
PROTO-R	0	0	0	.011	.014	0	0	0	.003	.025
MCE-R	0	0	0	.003	.100	0	0	0	.002	.055
RNCE-FF	0	0	0	.007	.036	0	0	0	.003	.022
RNCE-FT	0	0	0	.007	.036	0	0	0	.003	.022

Table B.9
Quantitative evaluation of the compared CE generation methods on logistic regression models. The evaluation metrics are the same as in Table 3.

	vr↑	vΔ_{val}↑	vΔ_{inc}↑	ℓ_1↓	lof↓	vr↑	vΔ_{val}↑	vΔ_{inc}↑	ℓ_1↓	lof↓
	adult					compas				
NNCE	97.9%	45%	42%	.078	1.99	73.6%	1%	49%	.034	1.32
ROAR	100%	100%	100%	.265	1.69	100%	100%	96.8%	.220	2.59
	target $\delta_{val} = 0.04$					target $\delta_{val} = 0.08$				
RNCE-FF	100%	100%	70%	.085	2.21	100%	100%	100%	.045	1.22
RNCE-FT	100%	100%	16%	.060	1.71	100%	100%	100%	.043	1.30
	target $\delta_{inc} = 0.063$					target $\delta_{inc} = 0.01$				
RNCE-FF	100%	100%	100%	.087	2.18	86.1%	1%	100%	.035	1.30
RNCE-FT	100%	100%	100%	.064	1.74	78.1%	0%	100%	.033	1.34
	gmc					heloc				
NNCE	92.7%	57%	51%	.035	1.21	77.5%	5%	49%	.057	1.06
ROAR	100%	100%	100%	.119	2.76	100%	100%	100%	.089	1.42
	target $\delta_{val} = 0.06$					target $\delta_{val} = 0.07$				
RNCE-FF	100%	100%	92%	.043	1.23	100%	100%	100%	.070	1.06
RNCE-FT	100%	100%	35%	.034	1.23	100%	100%	0%	.066	1.05
	target $\delta_{inc} = 0.079$					target $\delta_{inc} = 0.019$				
RNCE-FF	100%	100%	100%	.043	1.17	98.8%	10%	100%	.061	1.06
RNCE-FT	100%	100%	100%	.035	1.23	96.7%	0%	100%	.054	1.06

Table B.10
Standard deviation results of the quantitative evaluation of the compared CE generation methods on logistic regression models.

	vr	vΔ_{val}	vΔ_{inc}	ℓ_1	lof	vr	vΔ_{val}	vΔ_{inc}	ℓ_1	lof
	adult					compas				
NNCE	.035	.130	.144	.017	.188	.025	.020	.037	.003	.068
ROAR	0	0	0	.028	.020	0	0	0	.002	.151
	target $\delta_{val} = 0.04$					target $\delta_{val} = 0.08$				
RNCE-FF	0	0	.122	.016	.251	0	0	0	.003	.032
RNCE-FT	0	0	.092	.010	.070	0	0	0	.002	.060
	target $\delta_{inc} = 0.063$					target $\delta_{inc} = 0.01$				
RNCE-FF	0	0	0	.015	.273	.015	.020	0	.003	.068
RNCE-FT	0	0	0	.009	.104	.002	0	0	.003	.093
	gmc					heloc				
NNCE	.028	.129	.159	.003	.021	.023	.045	.102	.003	.013
ROAR	0	0	0	.010	.225	0	0	0	.003	.038
	target $\delta_{val} = 0.06$					target $\delta_{val} = 0.07$				
RNCE-FF	0	0	.112	.004	.055	0	0	0	.003	.021
RNCE-FT	0	0	.055	.004	.048	0	0	0	.004	.019
	target $\delta_{inc} = 0.079$					target $\delta_{inc} = 0.019$				
RNCE-FF	0	0	0	.003	.039	.010	.008	0	.003	.016
RNCE-FT	0	0	0	.004	.053	.012	0	0	.004	.015

References

[1] R. Guidotti, Counterfactual explanations and how to find them: literature review and benchmarking, Data Min. Knowl. Discov. (2022) 1–55.
[2] A. Karimi, G. Barthe, B. Schölkopf, I. Valera, A survey of algorithmic recourse: contrastive explanations and consequential recommendations, ACM Comput. Surv. 55 (2023) 95:1–95:29.
[3] T. Miller, Explanation in artificial intelligence: insights from the social sciences, Artif. Intell. 267 (2019) 1–38.
[4] L. Celar, R.M.J. Byrne, How people reason with counterfactual and causal explanations for AI decisions in familiar and unfamiliar domains, Mem. Cogn. 51 (2023) 1481–1496.
[5] G. Tolomei, F. Silvestri, A. Haines, M. Lalmas, Interpretable predictions of tree-based ensembles via actionable feature tweaking, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017, pp. 465–474.
[6] S. Wachter, B.D. Mittelstadt, C. Russell, Counterfactual explanations without opening the black box: automated decisions and the GDPR, Harv. J. Law Technol. 31 (2017) 841.

[7] R.K. Mothilal, A. Sharma, C. Tan, Explaining machine learning classifiers through diverse counterfactual explanations, in: FAT* 2020: ACM Conference on Fairness, Accountability, and Transparency, 2020, pp. 607–617.
[8] A. Dhurandhar, P. Chen, R. Luss, C. Tu, P. Ting, K. Shanmugam, P. Das, Explanations based on the missing: towards contrastive explanations with pertinent negatives, Adv. Neural Inf. Process. Syst. 31 (2018) 590–601, NeurIPS.
[9] J. Jiang, F. Leofante, A. Rago, F. Toni, Robust counterfactual explanations in machine learning: a survey, in: Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24, 2024, pp. 8086–8094, Survey Track.
[10] S. Upadhyay, S. Joshi, H. Lakkaraju, Towards robust and reliable algorithmic recourse, Adv. Neural Inf. Process. Syst. 34 (2021) 16926–16937, NeurIPS.
[11] N. Bui, D. Nguyen, V.A. Nguyen, Counterfactual plans under distributional ambiguity, in: The 10th International Conference on Learning Representations, ICLR, 2022.
[12] E. Black, Z. Wang, M. Fredrikson, Consistent counterfactuals for deep models, in: The 10th International Conference on Learning Representations, ICLR, 2022.
[13] S. Dutta, J. Long, S. Mishra, C. Tilli, D. Magazzeni, Robust counterfactual explanations for tree-based ensembles, in: Proceedings of the 39th International Conference on Machine Learning, in: PMLR, vol. 162, ICML, 2022, pp. 5742–5756.
[14] T.H. Nguyen, N. Bui, D. Nguyen, M. Yue, V.A. Nguyen, Robust bayesian recourse, in: Proceedings of the 38th Conference on Uncertainty in Artificial Intelligence, in: PMLR, vol. 180, 2022, pp. 1498–1508.
[15] J. Jiang, F. Leofante, A. Rago, F. Toni, Formalising the robustness of counterfactual explanations for neural networks, in: The 37th AAAI Conference on Artificial Intelligence, AAAI, 2023, pp. 14901–14909.
[16] J. Jiang, J. Lan, F. Leofante, A. Rago, F. Toni, Provably robust and plausible counterfactual explanations for neural networks via robust optimisation, in: Proceedings of the 15th Asian Conference on Machine Learning, in: PMLR, vol. 222, ACML, 2024, pp. 582–597.
[17] F. Hamman, E. Noorani, S. Mishra, D. Magazzeni, S. Dutta, Robust counterfactual explanations for neural networks with probabilistic guarantees, in: Proceedings of the 40th International Conference on Machine Learning, in: PMLR, vol. 202, ICML, 2023, pp. 12351–12367.
[18] J. Marques-Silva, A. Ignatiev, Delivering trustworthy AI through formal XAI, in: The 36th AAAI Conference on Artificial Intelligence, 2022, pp. 12342–12350.
[19] P. Prabhakar, Z.R. Afzal, Abstraction based output range analysis for neural networks, Adv. Neural Inf. Process. Syst. 32 (2019) 15762–15772, NeurIPS.
[20] K. Mohammadi, A. Karimi, G. Barthe, I. Valera, Scaling guarantees for nearest counterfactual explanations, in: AIES'21: AAAI/ACM Conference on AI, Ethics, and Society, 2021, pp. 177–187.
[21] D. Brughmans, P. Leyman, D. Martens, Nice: an algorithm for nearest instance counterfactual explanations, Data Min. Knowl. Discov. (2023) 1–39.
[22] M. Pawelczyk, K. Broelemann, G. Kasneci, Learning model-agnostic counterfactual explanations for tabular data, in: WWW'20: The Web Conference 2020, 2020, pp. 3126–3132.
[23] A. Van Looveren, J. Klaise, Interpretable counterfactual explanations guided by prototypes, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, 2021, pp. 650–665.
[24] B. Ustun, A. Spangher, Y. Liu, Actionable recourse in linear classification, in: FAT* 2019: ACM Conference on Fairness, Accountability, and Transparency, 2019, pp. 10–19.
[25] S. Dandl, C. Molnar, M. Binder, B. Bischl, Multi-objective counterfactual explanations, in: 16th International Conference on Parallel Problem Solving from Nature, 2020, pp. 448–469.
[26] A. Karimi, B.J. von Kügelgen, B. Schölkopf, I. Valera, Algorithmic recourse under imperfect causal knowledge: a probabilistic approach, Adv. Neural Inf. Process. Syst. 33 (2020), NeurIPS.
[27] A. Karimi, B. Schölkopf, I. Valera, Algorithmic recourse: from counterfactual explanations to interventions, in: FAccT 2021: ACM Conference on Fairness, Accountability, and Transparency, 2021, pp. 353–362.
[28] E. Albini, A. Rago, P. Baroni, F. Toni, Relation-based counterfactual explanations for bayesian network classifiers, in: Proceedings of the 29th International Joint Conference on AI, IJCAI, 2020, pp. 451–457.
[29] K. Kanamori, T. Takagi, K. Kobayashi, H. Arimura, DACE: distribution-aware counterfactual explanation by mixed-integer linear optimization, in: Proceedings of the 29th International Joint Conference on Artificial Intelligence, IJCAI, 2020, pp. 2855–2862.
[30] M. Bajaj, L. Chu, Z.Y. Xue, J. Pei, L. Wang, P.C. Lam, Y. Zhang, Robust counterfactual explanations on graph neural networks, Adv. Neural Inf. Process. Syst. 34 (2021) 5644–5655, NeurIPS.
[31] M. Augustin, V. Boreiko, F. Croce, M. Hein, Diffusion Visual Counterfactual Explanations, Adv. Neural Inf. Process. Syst. 35 (2022), NeurIPS.
[32] E. Delaney, D. Greene, M.T. Keane, Instance-based counterfactual explanations for time series classification, in: The 29th International Conference on Case-Based Reasoning, in: Lecture Notes in Computer Science, vol. 12877, 2021, pp. 32–47.
[33] K. Rawal, E. Kamar, H. Lakkaraju, Algorithmic recourse in the wild: understanding the impact of data and model shifts, preprint, arXiv:2012.11788, 2020.
[34] D. Nguyen, N. Bui, V.A. Nguyen, Distributionally robust recourse action, in: The 11th International Conference on Learning Representations, ICLR, 2023.
[35] A. Ferrario, M. Loi, The robustness of counterfactual explanations over time, IEEE Access 10 (2022) 82736–82750.
[36] H. Guo, F. Jia, J. Chen, A.C. Squicciarini, A. Yadav, Rocoursenet: robust training of a prediction aware recourse model, in: Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, CIKM, 2023, pp. 619–628.
[37] D. Slack, A. Hilgard, H. Lakkaraju, S. Singh, Counterfactual explanations can be manipulated, Adv. Neural Inf. Process. Syst. 34 (2021) 62–75, NeurIPS.
[38] F. Leofante, N. Potyka, Promoting counterfactual robustness through diversity, in: The 38th AAAI Conference on Artificial Intelligence, AAAI, 2024, pp. 21322–21330, in press.
[39] S. Krishna, J. Ma, H. Lakkaraju, Towards bridging the gaps between the right to explanation and the right to be forgotten, in: Proceedings of the 40th International Conference on Machine Learning, in: PMLR, vol. 202, ICML, 2023, pp. 17808–17826.
[40] M. Pawelczyk, T. Leemann, A. Biega, G. Kasneci, On the trade-off between actionable explanations and the right to be forgotten, in: The 11th International Conference on Learning Representations, ICLR, 2023.
[41] M. Pawelczyk, K. Broelemann, G. Kasneci, On counterfactual explanations under predictive multiplicity, in: Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence, in: PMLR, vol. 124, UAI, 2020, pp. 809–818.
[42] F. Leofante, E. Botoeva, V. Rajani, Counterfactual explanations and model multiplicity: a relational verification view, in: Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning, KR, 2023, pp. 763–768.
[43] J. Jiang, F. Leofante, A. Rago, F. Toni, Recourse under model multiplicity via argumentative ensembling, in: Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems, AAMAS, 2024, pp. 954–963.
[44] R. Dominguez-Olmedo, A. Karimi, B. Schölkopf, On the adversarial robustness of causal algorithmic recourse, in: Proceedings of the 39th International Conference on Machine Learning, ICML 2022, in: PMLR, vol. 162, 2022, pp. 5324–5342.
[45] F. Leofante, A. Lomuscio, Robust explanations for human-neural multi-agent systems with formal verification, in: Proceedings of the 20th European Conference on Multi-Agent Systems, in: Lecture Notes in Computer Science, vol. 14282, EUMAS, 2023, pp. 244–262.
[46] F. Leofante, A. Lomuscio, Towards robust contrastive explanations for human-neural multi-agent systems, in: Proceedings of the 22nd International Conference on Autonomous Agents and Multiagent Systems, AAMAS, 2023, pp. 2343–2345.
[47] M. Virgolin, S. Fracaros, On the robustness of sparse counterfactual explanations to adverse perturbations, Artif. Intell. 316 (2023) 103840.
[48] D. Maragno, J. Kurtz, T.E. Röber, R. Goedhart, Ş.I. Birbil, D. den Hertog, Finding regions of counterfactual explanations via robust optimization, INFORMS J. Comput. (2024).
[49] S. Mishra, S. Dutta, J. Long, D. Magazzeni, A survey on the robustness of feature importance and counterfactual explanations, preprint, arXiv:2111.00358, 2021.

[50] N. Carlini, D.A. Wagner, Towards evaluating the robustness of neural networks, in: IEEE Symposium on Security and Privacy, SP, 2017, pp. 39–57.
[51] T. Weng, H. Zhang, P. Chen, J. Yi, D. Su, Y. Gao, C. Hsieh, L. Daniel, Evaluating the robustness of neural networks: an extreme value theory approach, in: The 6th International Conference on Learning Representations, ICLR, 2018.
[52] T. Weng, P. Zhao, S. Liu, P. Chen, X. Lin, L. Daniel, Towards certificated model robustness against weight perturbations, in: The 34th AAAI Conference on Artificial Intelligence, AAAI, 2020, pp. 6356–6363.
[53] Y. Tsai, C. Hsu, C. Yu, P. Chen, Formalizing generalization and adversarial robustness of neural networks to weight perturbations, Adv. Neural Inf. Process. Syst. 34 (2021) 19692–19704, NeurIPS.
[54] E. Wong, J.Z. Kolter, Provable defenses against adversarial examples via the convex outer adversarial polytope, in: Proceedings of the 35th International Conference on Machine Learning, in: PMLR, vol. 80, ICML, 2018, pp. 5283–5292.
[55] M. Mirman, T. Gehr, M.T. Vechev, Differentiable abstract interpretation for provably robust neural networks, in: Proceedings of the 35th International Conference on Machine Learning, in: PMLR, vol. 80, ICML, 2018, pp. 3575–3583.
[56] S. Gowal, K. Dvijotham, R. Stanforth, R. Bunel, C. Qin, J. Uesato, R. Arandjelovic, T.A. Mann, P. Kohli, Scalable verified training for provably robust image classification, in: IEEE/CVF International Conference on Computer Vision, ICCV, 2019, pp. 4841–4850.
[57] H. Zhang, H. Chen, C. Xiao, S. Gowal, R. Stanforth, B. Li, D.S. Boning, C. Hsieh, Towards stable and efficient training of verifiably robust neural networks, in: The 8th International Conference on Learning Representations, ICLR, 2020.
[58] P. Henriksen, A. Lomuscio, Robust training of neural networks against bias field perturbations, in: B. Williams, Y. Chen, J. Neville (Eds.), The 37th AAAI Conference on Artificial Intelligence, AAAI, 2023, pp. 14865–14873.
[59] A. Lomuscio, L. Maganti, An approach to reachability analysis for feed-forward relu neural networks, preprint, arXiv:1706.07351, 2017.
[60] D. Dua, C. Graff, et al., Uci Machine Learning Repository, 2017.
[61] S.M.J. Angwin, J. Larson, L. Kirchner, Machine Bias: There's Software Used Across the Country to Predict Future Criminals. and It's Biased Against Blacks, 2016.
[62] W. Cukierski, Give Me Some Credit, 2011.
[63] FICO, Explainable Machine Learning Challenge, 2018.
[64] M. Pawelczyk, S. Bielawski, J. van den Heuvel, T. Richter, G. Kasneci, CARLA: a python library to benchmark algorithmic recourse and counterfactual explanation algorithms, in: NeurIPS Datasets and Benchmarks Track, 2021.
[65] M.M. Breunig, H. Kriegel, R.T. Ng, J. Sander, LOF: identifying density-based local outliers, in: The 2000 ACM SIGMOD International Conference on Management of Data, 2000, pp. 93–104.
[66] M. Pawelczyk, T. Datta, J. van den Heuvel, G. Kasneci, H. Lakkaraju, Probabilistically robust recourse: navigating the trade-offs between costs and robustness in algorithmic recourse, in: The 11th International Conference on Learning Representations, ICLR, 2023.
[67] L. Marzari, F. Leofante, F. Cicalese, A. Farinelli, Rigorous probabilistic guarantees for robust counterfactual explanations, in: Proceedings of the 27th European Conference on Artificial Intelligence (ECAI), 2024.
[68] E. Anderson, The species problem in iris, Ann. Missouri Botan. Garden 23 (1936) 457–509.
[69] R.A. Fisher, The use of multiple measurements in taxonomic problems, Ann. Eugen. 7 (1936) 179–188.
[70] R.K. Pace, R. Barry, Sparse spatial autoregressions, Stat. Probab. Lett. 33 (1997) 291–297.
[71] 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, E. Duchesnay, Scikit-learn: machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.