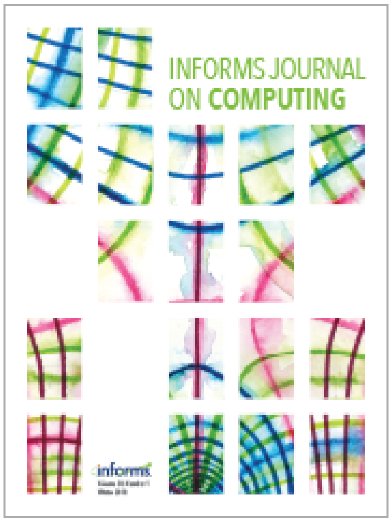


INFORMS Journal on Computing

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Finding Regions of Counterfactual Explanations via Robust Optimization

Donato Maragno, Jannis Kurtz, Tabea E. Röber, Rob Goedhart, Ş. İlker Birbil, Dick den Hertog

To cite this article:
Donato Maragno, Jannis Kurtz, Tabea E. Röber, Rob Goedhart, Ş. İlker Birbil, Dick den Hertog (2024) Finding Regions of Counterfactual Explanations via Robust Optimization. INFORMS Journal on Computing

Published online in Articles in Advance 22 Feb 2024

. https://doi.org/10.1287/ijoc.2023.0153 https://pubsonline.informs.org/journal/ijoc

INFORMS JOURNAL ON COMPUTING
Articles in Advance, pp. 1–19
ISSN 1091-9856 (print), ISSN 1526-5528 (online)

Finding Regions of Counterfactual Explanations via Robust Optimization

Donato Maragno,[a],* Jannis Kurtz,[a] Tabea E. Röber,[a] Rob Goedhart,[a] Ş. İlker Birbil,[a] Dick den Hertog[a]

[a] Amsterdam Business School, University of Amsterdam, 1018 TV Amsterdam, Netherlands

*Corresponding author

Contact: d.maragno@uva.nl, https://orcid.org/0000-0001-5857-8807 (DM); j.kurtz@uva.nl, https://orcid.org/0000-0003-1570-7044 (JK); t.e.rober@uva.nl, https://orcid.org/0009-0005-7089-5924 (TER); r.goedhart2@uva.nl, https://orcid.org/0000-0001-9966-0284 (RG); s.i.birbil@uva.nl, https://orcid.org/0000-0001-7472-7032 (ŞİB); d.denhertog@uva.nl, https://orcid.org/0000-0002-1829-855X (DdH)

Received: May 16, 2023
Revised: October 23, 2023; January 12, 2024; January 16, 2024
Accepted: January 17, 2024
Published Online in Articles in Advance: February 22, 2024

https://doi.org/10.1287/ijoc.2023.0153

Abstract. Counterfactual explanations (CEs) play an important role in detecting bias and improving the explainability of data-driven classification models. A CE is a minimal perturbed data point for which the decision of the model changes. Most of the existing methods can only provide one CE, which may not be achievable for the user. In this work, we derive an iterative method to calculate robust CEs (i.e., CEs that remain valid even after the features are slightly perturbed). To this end, our method provides a whole region of CEs, allowing the user to choose a suitable recourse to obtain a desired outcome. We use algorithmic ideas from robust optimization and prove convergence results for the most common machine learning methods, including decision trees, tree ensembles, and neural networks. Our experiments show that our method can efficiently generate globally optimal robust CEs for a variety of common data sets and classification models.

History: Accepted by Andrea Lodi, Area Editor for Design & Analysis of Algorithms—Discrete.
Funding: This work was supported by the Nederlandse Organisatie voor Wetenschappelijk Onderzoek [Grant OCENW.GROOT.2019.015, Optimization for and with Machine Learning (OPTIMAL)].
Supplemental Material: The online appendix is available at https://doi.org/10.1287/ijoc.2023.0153.

Keywords: counterfactual explanation • explainable AI • machine learning • robust optimization

1. Introduction

Counterfactual explanations (CEs), also known as algorithmic recourse, are becoming increasingly popular as a way to explain the decisions made by black-box machine learning (ML) models. Given a factual instance for which we want to derive an explanation, we search for a counterfactual feature combination describing the minimum change in the feature space that will lead to a flipped model prediction. For example, for a person with a rejected loan application, the CE could be "if the *annual salary* would increase to $50,000, then the *loan application* would be approved." This method enables a form of user agency and is, therefore, particularly attractive in consequential decision making, where the user is directly and indirectly impacted by the outcome of the ML model.

The first optimization-based approach to generate CEs was proposed by Wachter et al. (2018). Given a trained classifier $h : \mathcal{X} \to [0,1]$ and a *factual instance* $\hat{x} \in \mathcal{X}$, the aim is to find a *counterfactual* $\tilde{x} \in \mathcal{X}$ that has the shortest distance to $\hat{x}$ and has the opposite target. The problem to obtain $\tilde{x}$ can be formulated as

$$\underset{x \in \mathcal{X}}{\text{minimize}} \; d(\hat{x}, x) \tag{1}$$

$$\text{subject to } h(x) \geq \tau, \tag{2}$$

where $d(\cdot,\cdot)$ is a distance function, often chosen to be the ℓ_1-norm or the ℓ_2-norm, and $\tau \in [0,1]$ is a given threshold parameter for the classification decision.

Others have built on this work and proposed approaches that generate CEs with increased practical value, primarily by adding constraints to ensure actionability of the proposed changes and generating CEs that are close to the data manifold (Mahajan et al. 2019, Russell 2019, Ustun et al. 2019, Mothilal et al. 2020, Maragno et al. 2022). Nonetheless, the user agency provided by these methods remains theoretical; the generated CEs are exact point solutions that may remain difficult, if not impossible, to implement in practice. A minimal change to the proposed CE could fail to flip the model's prediction, especially because the CEs are close to the decision boundary because of minimizing the distance between $\hat{x}$ and $\tilde{x}$. As a solution, prior work suggests generating several CEs to increase the likelihood of generating at least one attainable solution. Approaches generating several CEs typically require solving the optimization problem multiple times (Russell 2019, Karimi et al. 2020, Mothilal et al. 2020, Kanamori et al. 2021), which might heavily affect the optimization time when the number of explanations is large. Maragno et al. (2022) suggest using incumbent solutions, however, this does not allow us to control the quality of suboptimal solutions. On top of that, the added practical value may be unconvincing; each of the CEs is still sensitive to arbitrarily small changes in the actions implemented by the user (Dominguez-Olmedo et al. 2022, Pawelczyk et al. 2022, Virgolin and Fracaros 2023).

This problem has been acknowledged in prior work and falls under the discussion of robustness in CEs. In the literature, the concept of robustness in CEs has different meanings: (1) robustness to input perturbations (Artelt et al. 2021, Slack et al. 2021) and generating explanations for a group of individuals (Carrizosa et al. 2021), (2) robustness to model changes (Rawal et al. 2020, Black et al. 2021, Upadhyay et al. 2021, Bui et al. 2022, Dutta et al. 2022, Ferrario and Loi 2022, Forel et al. 2022), (3) robustness to hyperparameter selection (Dandl et al. 2020), and (4) robustness to recourse (Dominguez-Olmedo et al. 2022, Pawelczyk et al. 2022, Virgolin and Fracaros 2023). The latter perspective, albeit very user centered, has so far received only a little attention. Our work focuses on the latter definition of robustness in CEs, specifically the idea of robustness to recourse. This means that a counterfactual solution should remain valid even if small changes are made to the implemented recourse action. In other words, we aim to define regions of counterfactual solutions that allow the user to choose any point within that region to flip the model prediction. This extends the idea of offering several explanations for the user to choose from such that not only the suggested point is a counterfactual but also, every point in the defined region. Returning to the example we used, a final explanation may be "if the *annual salary* would increase to anywhere between $50,000 and $54,200, then the *loan application* would be approved." Although existing research has tackled this problem, their solutions are not comprehensive and have room for further improvements. In the remainder of this section, we will explore the related prior work and present our own contributions to this field.

Pawelczyk et al. (2022) introduce the notion of recourse invalidation rate, which amounts to the proportion of recourse that does not lead to the desired model prediction (i.e., that is invalid). They model the noise around a counterfactual data point with a Gaussian distribution and suggest an approach that ensures that the invalidation rate within a specified neighborhood around the counterfactual data point is no larger than a target recourse invalidation rate. However, their work provides a heuristic solution using a gradient-based approach, which makes it not directly applicable to decision tree (DT) models. Additionally, it only provides a probabilistic robustness guarantee. Dominguez-Olmedo et al. (2022) introduce an approach where the optimal solution is surrounded by an uncertainty set such that every point in the set is a feasible solution. They also model causality between (perturbed) features to obtain a more informative neighborhood of similar points. Given a structural causal model, they model such perturbations as additive interventions on the factual instance features. The authors design an iterative approach that works only for differentiable classifiers and does not guarantee that the generated recourse actions are adversarially robust. Virgolin and Fracaros (2023) incorporate the possibility of additional intervention to contrast perturbations in their search for CEs. They make a distinction between the features that could be changed and those that should be kept as they are, and they introduce the concept of C-setbacks, a subset of perturbations in changeable features that work against the user. Rather than seeking CEs that are not invalidated by C-setbacks, they seek CEs for which the additional intervention cost to overcome the setback is minimal. Perturbations to features that should be kept as they are according to a CE are orthogonal to the direction of the counterfactual, and Virgolin and Fracaros (2023) approximate a robustness score for such features. A drawback of this method is that it is only applicable in situations where additional intervention is possible and not in situations where (e.g., because of time limitations) only a single recourse is possible.

Our work addresses robustness to recourse by utilizing a robust optimization approach to generate regions of CEs. For a given factual instance, our method generates a CE that is robust to small perturbations. This gives the user more flexibility in implementing the recourse and reduces the risk of invalidating it. In this work, we consider numerical features, ensuring that small perturbations do not affect the recourse validity, whereas categorical features are treated as immutable based on user preferences. Additionally, the generated CEs are optimal in terms of their objective distance to the factual instance. The proposed algorithm is proven to converge, ensuring that the optimal solution is reached. This is different from prior work that provides only heuristic algorithms that are not provably able to find the optimal (i.e., closest) counterfactual point with a certain robustness guarantee (e.g., Dominguez-Olmedo et al. 2022, Pawelczyk et al. 2022). Unlike prior research in this area, our approach is able to provide deterministic robustness guarantees for the CEs generated. Furthermore, our method does not require differentiability of the underlying ML model and is applicable to the tree-based models, which to the best of our knowledge, has not been done before.

In summary, we make the following contributions.

- We propose an iterative algorithm that effectively finds global optimal robust CEs for trained decision trees, ensembles of trees, and neural networks.

- We prove the convergence of the algorithm for the considered trained models.
- We demonstrate the power of our algorithm on several data sets and different ML models. We empirically evaluate its convergence performance and compare the robustness as well as the validity of the generated CEs with the prior work in the literature.
- We release an open-source software called RCE to make the proposed algorithm easily accessible to practitioners. Our software is available in a dedicated repository[1] through which all our results can be reproduced.

2. Robust Counterfactual Explanations

We consider binary classification problems (i.e., we have a trained classifier $h : \mathcal{X} \to [0,1]$ that assigns a value between zero and one to each data point in the data space $\mathcal{X} \subseteq \mathbb{R}^n$). A point $x \in \mathcal{X}$ is then predicted to correspond to class +1 if $h(x) \geq \tau$ and to class -1 otherwise. Here, $\tau \in [0,1]$ is a given threshold parameter, which is often chosen to be $\tau = 0.5$. Given a factual instance $\hat{x} \in \mathcal{X}$, which is predicted to be in class -1 (i.e., $h(\hat{x}) < \tau$), the robust CE problem is defined as

$$\underset{x \in \mathcal{X}}{\text{minimize}} \;\; d(x, \hat{x}) \tag{3}$$

$$\text{subject to} \;\; h(x+s) \geq \tau, \quad \forall s \in \mathcal{S}, \tag{4}$$

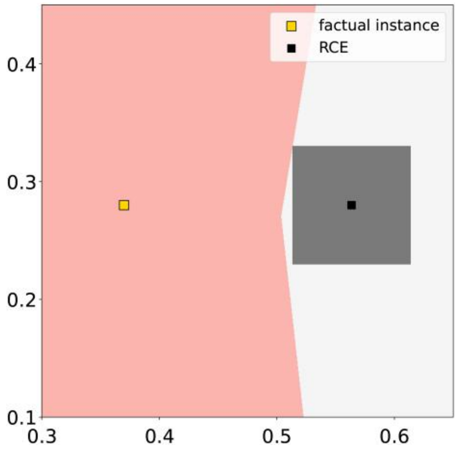
where the $d(x, \hat{x})$ represents a distance function (e.g., induced by the ℓ_1-, ℓ_2-, or ℓ_∞-norm) and $\mathcal{S} \subset \mathbb{R}^n$ is a given uncertainty set. The idea of the problem is to find a point that is as close as possible to the factual instance $\hat{x}$ such that for all perturbations $s \in \mathcal{S}$, the corresponding point $x + s$ is classified as +1, which is enforced by Constraints (4); see Figure 1. This results in a large set of counterfactual explanations.

We consider uncertainty sets of the type

$$\mathcal{S} = \{s \in \mathbb{R}^n \,|\, \|s\| \leq \rho\}, \tag{5}$$

where $\|\cdot\|$ is a given norm. Popular choices are the ℓ_∞-norm, resulting in a box with upper and lower bounds on features, or the ℓ_2-norm, resulting in a circular uncertainty set. We refer to Ben-Tal et al. (2009) for a discussion of uncertainty sets. From the user perspective, choosing the ℓ_∞-norm has a practical advantage because the region $\mathcal{S}$ is a box (i.e., we obtain an interval for each attribute of $\hat{x}$). Each attribute can be changed in its corresponding interval independently, resulting in a counterfactual explanation. Hence, the user can easily detect if there exists a CE in the region that can be practically reached. The choice of the robustness budget ρ greatly depends on the specific domain. Larger values of ρ are associated with CEs that are more robust but can have a larger distance to the factual instance. However, our approach outlined in the subsequent sections is designed to minimize the proximity to the factual instance for a given robustness parameter ρ. If the perturbation applied to the CE adheres to a known distribution, it becomes feasible to determine ρ in a way that offers a probabilistic guarantee of CE robustness. We refer to Online Appendix A for a comprehensive guide on how to determine the appropriate value for ρ.

Figure 1. (Color online) Robust CE for a Neural Network and Using a Box Uncertainty Set



Note. All points in the red region are classified as -1, and all points in the white region are classified as +1.

2.1. Comparison Against Model Robustness

There are several works that study the robustness of counterfactual explanations regarding changes in the parameters of the trained machine learning model (see, e.g., Rawal et al. 2020, Black et al. 2021, Upadhyay et al. 2021, Bui et al. 2022, Dutta et al. 2022, Ferrario and Loi 2022, Forel et al. 2022). Model changes can appear frequently in real-world applications (e.g., a model used for classification is retrained with new data or a different hyperparameter setup). In these cases, the model parameters can change, and a counterfactual explanation for the old model can become invalid for the retrained model.

Consider a classifier $h_{\hat{\omega}}$, where $\hat{\omega}$ is the vector of model parameters that was determined during the training process. In the case of a neural network, this vector contains all weights of the neural network, or in the case of a linear classifier, ω contains all weight parameters of the linear hyperplane separating the two classes.

Translated into the robust optimization setting we study in this work, a model-robust CE is a point that remains a CE for all parameter values $\omega \in \Omega$, where $\Omega = \{\omega : \|\omega - \hat{\omega}\| \leq \rho_{mod}\}$ is a given uncertainty set that contains all possible model parameters that have distance at most ρ_{mod} to the original weights of the classifier. In other words, if x^{CE} is a counterfactual point for the original classifier (i.e., $h_{\hat{\omega}}(x^{CE}) \geq \tau$), then for each classifier h_ω where $\omega \in \Omega$, this must also be the case (i.e., $h_\omega(x^{CE}) \geq \tau$). Note that defining model robustness for decision tree models is much more elaborate because a decision tree is not only defined by the parameters of its split hyperplanes but also, defined by the tree structure that can change after retraining the model. So, defining Ω is not as straightforward as it is for linear models.

However, a natural question arises. Are the two concepts, model robustness and recourse robustness, equivalent? In this case, concepts from both fields could profit from each other. Unfortunately, this is not the case, which we show in the following example.

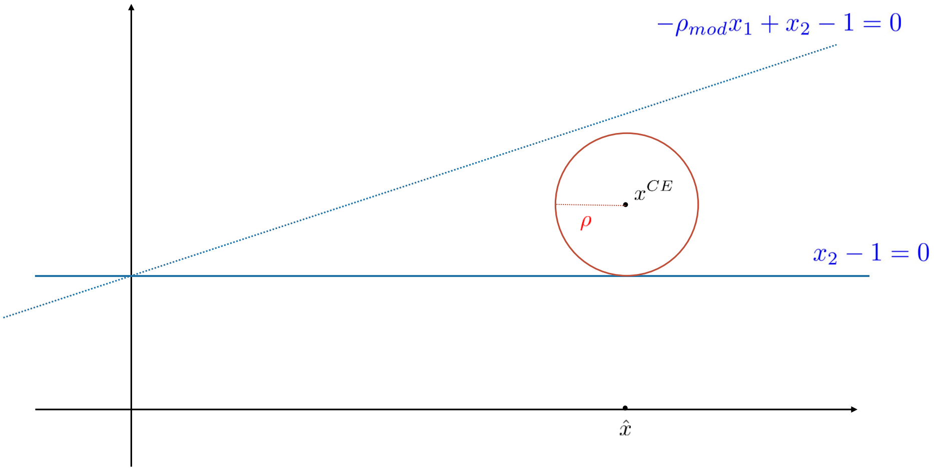
Consider the linear classifier given by the hyperplane $0x_1 + x_2 - 1 = 0$ (i.e., every point $x \in \mathbb{R}^2$ where $x_2 - 1 \geq 0$ is classified as one, and all others are classified as zero). Then, for every $\rho, \rho_{mod} > 0$, there exists a recourse-robust CE with radius ρ that is not model robust with radius ρ_{mod}. The construction works as follows. Let $\rho, \rho_{mod} > 0$, and define the factual instance $\hat{x} = (\rho/2\rho_{mod}, 0)$. Then, the closest recourse-robust CE for radius ρ is $x^{CE} = (\rho/2\rho_{mod}, 1 + \rho)$ for all relevant norms used in (5); see Figure 2. Now, consider the ρ_{mod}-perturbed hyperplane $-\rho_{mod}x_1 + x_2 - 1 = 0$. For x^{CE}, it holds that

$$-\rho_{mod}x_1 + x_2 - 1 = -\rho_{mod}\frac{\rho}{2\rho_{mod}} + 1 + \rho - 1 = -\frac{1}{2}\rho < 0.$$

Hence, x^{CE} is not a counterfactual point for the perturbed model.

A similar setup can be used to show that there exist points that are model robust but not recourse robust. Consider the classifier $w_1x_1 + w_2x_2 = 0$, where $w_1 = 0$ and $w_2 = 1$. We assume that only the parameters w_1, w_2 are allowed to change. Let $\rho, \rho_{mod} > 0$. Define the point $\tilde{x}^{CE} = (0, 0)$, which is classified as one. Clearly, this point is model robust for radius ρ_{mod} because for every change of w_1, w_2, it holds that $w_1x_1 + w_2x_2 = 0$. However, the point is not recourse robust regarding radius ρ because the perturbed point $\tilde{x} = (0, -\varepsilon)$ for $\varepsilon < \rho$ is classified as zero.

Figure 2. (Color online) Example in Section 2.1



Although the latter examples show that the equivalence of both robustness types does not hold, there can be special cases of models where both concepts are related. However, we place this interesting analysis on our future research agenda.

2.2. Algorithm

We note that the model in (3)–(4) has infinitely many constraints. One approach often used in robust optimization is to rewrite Constraints (4) as

$$\min_{s\in\mathcal{S}} \; h(\boldsymbol{x}+\boldsymbol{s}) \geq \tau$$

and dualize the optimization problem on the left-hand side. This leads to a problem with a finite number of constraints. Unfortunately, strong duality is required to perform this reformulation, which does not hold for most classifiers h involving nonconvexity or integer variables.[2] In the latter case, we can use an alternative method popular in robust optimization where the constraints are generated iteratively. This iterative approach to solve Problems (3)–(4) is known as the *adversarial approach*. The approach was intensively used for robust optimization problems; see Bienstock and Özbay (2008) and Mutapcic and Boyd (2009). In Bertsimas et al. (2016), the adversarial approach was compared with the classical robust reformulation.

The idea of the approach is to consider a relaxed version of the model, where only a finite subset of scenarios $\mathcal{Z} \subset \mathcal{S}$ is considered:

$$\begin{aligned} &\underset{x\in\mathcal{X}}{\text{minimize}} \quad d(\boldsymbol{x},\hat{\boldsymbol{x}}) && \text{(MP)} \\ &\text{subject to} \quad h(\boldsymbol{x}+\boldsymbol{s}) \geq \tau, \quad \forall \boldsymbol{s} \in \mathcal{Z}. && (6) \end{aligned}$$

This problem is called the *master problem* (MP), and it only has a finite number of constraints. Note that the optimal value of (MP) is a lower bound of the optimal value of (3)–(4). However, an optimal solution $\boldsymbol{x}^*$ of (MP) is not necessarily feasible for the original problem because there may exist a scenario in $\mathcal{S}$ that is not contained in $\mathcal{Z}$ for which the solution is not feasible. More precisely, it may be that there exists an $\boldsymbol{s} \in \mathcal{S}$ such that $h(\boldsymbol{x}^* + \boldsymbol{s}) < \tau$, and hence, $\boldsymbol{x}^*$ is not a robust counterfactual. In this case, we want to find such a scenario $\boldsymbol{s}^*$ that makes solution $\boldsymbol{x}^*$ infeasible. This can be done by solving the following so-called *adversarial problem* (AP):

$$\max_{s\in\mathcal{S}} \tau - h(\boldsymbol{x}^* + \boldsymbol{s}). \qquad \text{(AP)}$$

The idea is to find a scenario $\boldsymbol{s}^* \in \mathcal{S}$ such that the prediction of classifier h for point $\boldsymbol{x}^* + \boldsymbol{s}^*$ is -1 (i.e., $\tau - h(\boldsymbol{x}^* + \boldsymbol{s}^*) > 0$). If we can find such a scenario and add it to the set $\mathcal{Z}$ in the MP, then $\boldsymbol{x}^*$ cannot be feasible anymore for (MP). To find the scenario with the largest impact, we maximize the constraint violation in the objective function in (AP). If the optimal value of (AP) is positive, then $\boldsymbol{x}^* + \boldsymbol{s}^*$ is classified as -1, the optimal solution $\boldsymbol{s}^*$ is added to $\mathcal{Z}$, and we calculate a solution $\boldsymbol{x}^*$ of the updated (MP). We iterate until no violating scenario can be found: that is, until the optimal value of (AP) is smaller than or equal to zero. Note that in this case, $h(\boldsymbol{x}^* + \boldsymbol{s}) \geq \tau$ holds for all $\boldsymbol{s} \in \mathcal{S}$, which means that $\boldsymbol{x}^*$ is a robust counterfactual. Algorithm 1 shows the steps of our approach, and Figure 3 shows its iterative behavior. Each time a new scenario $\boldsymbol{s}$ is found by solving the AP, it is added to the uncertainty set $\mathcal{Z}$, and the new solution $\boldsymbol{x}^*$ moves to also be feasible for the new scenario. This is repeated until no scenario can be found anymore (i.e., until the full box lies in the correct region). Note that instead of checking for a positive optimal value of (AP), we use an accuracy parameter $\varepsilon > 0$ in Algorithm 1. In this case, we can guarantee the convergence of our algorithm using the following result.

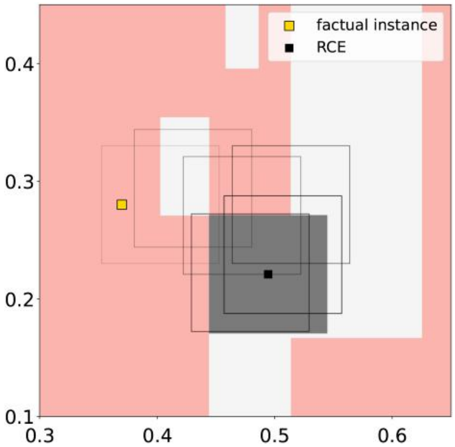
Algorithm 1 (Adversarial Algorithm)

```
Input: $\mathcal{S}$, $\hat{\boldsymbol{x}}$, $\varepsilon > 0$
$\mathcal{Z} = \{\boldsymbol{0}\}$
repeat
    $\boldsymbol{x}^* \leftarrow$ Solve (MP) with $\mathcal{Z}, \hat{\boldsymbol{x}}$
    $\boldsymbol{s}^*, \text{opt} \leftarrow$ Solve (AP) with $\boldsymbol{x}^*, \mathcal{S}$
    $\mathcal{Z} \leftarrow \mathcal{Z} \cup \{\boldsymbol{s}^*\}$
until opt $\leq \varepsilon$
Return: $x^*$
```

Theorem 1 (Mutapcic and Boyd 2009). *If $\mathcal{X}$ is bounded and if h is a Lipschitz-continuous function (i.e., there exists an $L > 0$ such that*

$$|h(\boldsymbol{x}_1) - h(\boldsymbol{x}_2)| \leq L\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|$$

Figure 3. (Color online) Iterations of Algorithm 1 to Find the Optimal Robust CE for a Decision Tree



Notes. For each (MP) solution, we show the uncertainty box around it. As long as the box overlaps with the red region, a new scenario can be found, and the solution moves in the next iteration.

for all $\boldsymbol{x_1}, \boldsymbol{x_2} \in \mathcal{X}$*), then for any tolerance parameter value* $\epsilon > 0$*, Algorithm* 1 *terminates after a finite number of steps with a solution* $\boldsymbol{x}^*$ *such that*

$$h(\boldsymbol{x}^* + \boldsymbol{s}) \geq \tau - \varepsilon$$

for all $\boldsymbol{s} \in \mathcal{S}$.

Indeed, without Lipschitz continuity, the convergence of Algorithm 1 cannot be ensured. We elaborate on this necessity in the following example, for which Algorithm 1 does not terminate in a finite number of steps.

Example 1. Consider a classifier $h : \mathbb{R}^2 \rightarrow [0,1]$ with $h(\boldsymbol{x}) = 0$ if $x_2 > 1/2$ and $h(\boldsymbol{x}) = 1$ otherwise. The threshold is $\tau = 0.5$ (i.e., a point is classified as 1 if $x_2 \leq 1/2$ and as -1 otherwise). The factual instance is $\hat{\boldsymbol{z}} = (0,2)$, which is classified as -1. Furthermore, the uncertainty set is given as $\mathcal{S} = \{\boldsymbol{s} \in \mathbb{R}^2 : \|\boldsymbol{s}\|_\infty \leq 1\}$. We can warm start Algorithm 1 with the (MP) solution $\boldsymbol{x}^1 = (0,0)$. Now, assume that in iteration i, the optimal solution returned by (AP) is $\boldsymbol{s}^i = \left(1, 1/2 + \sum_{j=1}^{i} (1/4)^j\right)$. Note that in the first iteration, $\boldsymbol{s}^1 = (1, 3/4)$ lies on the boundary of $\mathcal{S}$ and $\boldsymbol{x}^1 + \boldsymbol{s}^1$ is classified as -1 (i.e., it is an optimal solution of (AP)). We are looking now for the closest point $\boldsymbol{x}^2$ to $\hat{\boldsymbol{x}}$ such that $\boldsymbol{x}^2 + \boldsymbol{s}^1$ is classified as one; that is, it has a second component of at most $1/2$. This is the point $\boldsymbol{x}^2 = (0, -1/4)$, which must be the optimal solution of (MP). Note that $\boldsymbol{s}^2$ is again on the boundary of $\mathcal{S}$ and that $\boldsymbol{x}^2 + \boldsymbol{s}^2 = (1, 1/2 + 1/8)$ is classified as -1. Hence, $\boldsymbol{s}^2$ is an optimal solution of (AP). We can conclude inductively that the optimal solution of (MP) in iteration i is $\boldsymbol{x}^i = \left(0, -\sum_{j=1}^{i} (1/4)^j\right)$ and that $\boldsymbol{s}^i$ is an optimal solution of (AP) in iteration i. Note that the latter is true because the value of h is constant in the negative region, and hence, each point in the uncertainty set is an optimal solution of (AP). If the latter solutions are returned by (AP), then the sequence of solutions $\boldsymbol{x}^i$ converges to the point $\bar{\boldsymbol{x}} = (0, -1/3)$, which follows from the limit of the geometric series. However, $\bar{\boldsymbol{x}}$ is not a robust CE regarding S because for instance, $\bar{\boldsymbol{x}} + (0,1) = (0, 2/3)$ is classified as -1. Consequently, Algorithm 1 never terminates. This example is illustrated in Figure 4.

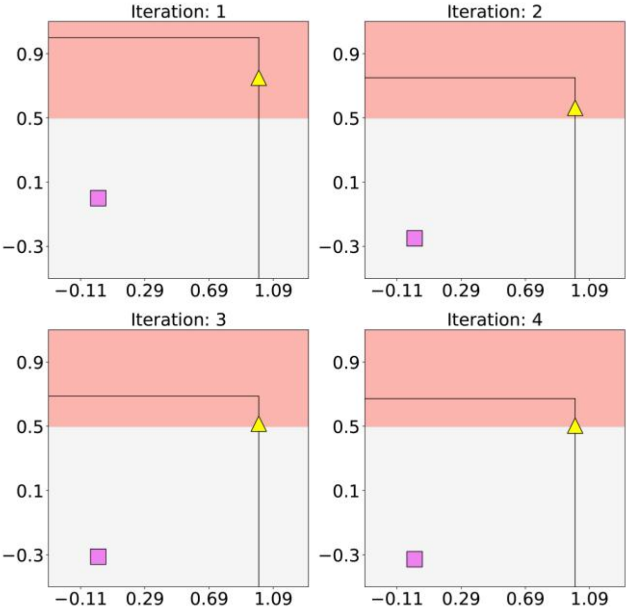
One difficulty is modeling the constraints of the form $h(\boldsymbol{x} + \boldsymbol{s}) \geq \tau$ for different trained classifiers. For decision trees, ensembles of decision trees, and neural networks, these constraints can be modeled by mixed-integer linear constraints as we present in the following section. Another difficulty is that h is discontinuous for decision trees and ensembles of decision trees. To handle these models, we have to find Lipschitz-continuous extensions of h with equivalent predictions to guarantee convergence of Algorithm 1.

3. Trained Models

The main contribution of this section is modeling MP and AP for different types of classifiers h by mixed-integer programming formulations. Furthermore, to prove convergence, we need to assure that all studied classifiers h are Lipschitz continuous, which is not the case for tree-based models. Hence, we introduce a Lipschitz-continuous classifier for tree-based models and derive the MP and AP for it.

We give reformulations for (MP) and (AP) for decision trees, tree ensembles, and neural networks that satisfy the conditions needed in Theorem 1 for convergence.

Figure 4. (Color online) Illustration of the Iterations of Algorithm 1 for the Problem in Example 1



Notes. The purple squares represent the current (MP) solution, whereas the yellow triangles represent the solution of the adversarial problem. The distance of the adversarial solutions to the decision boundary is $\left(\frac{1}{4}\right)^i$ in iteration *i*.

3.1. Decision Trees

A DT partitions the data samples into distinct *leaves* through a series of *feature splits*. A split at node *j* is performed by a hyperplane $\tilde{\boldsymbol{a}}^\top \boldsymbol{x} = \tilde{b}$. We assume that $\tilde{\boldsymbol{a}}$ can have multiple nonzero elements, in which we have the hyperplane split setting; if there is only one nonzero element, this creates an orthogonal (single-feature) split. Formally, each leaf $\mathcal{L}^i$ of a decision tree is defined by a set of (strict) inequalities

$$\mathcal{L}^i = \{\boldsymbol{x} \in X : \boldsymbol{a}^\top \boldsymbol{x} \le b,\ \boldsymbol{\alpha}^\top \boldsymbol{x} < \beta; (\boldsymbol{a}, b) \in \mathcal{N}^i_\le, (\boldsymbol{\alpha}, \beta) \in \mathcal{N}^i_<\},$$

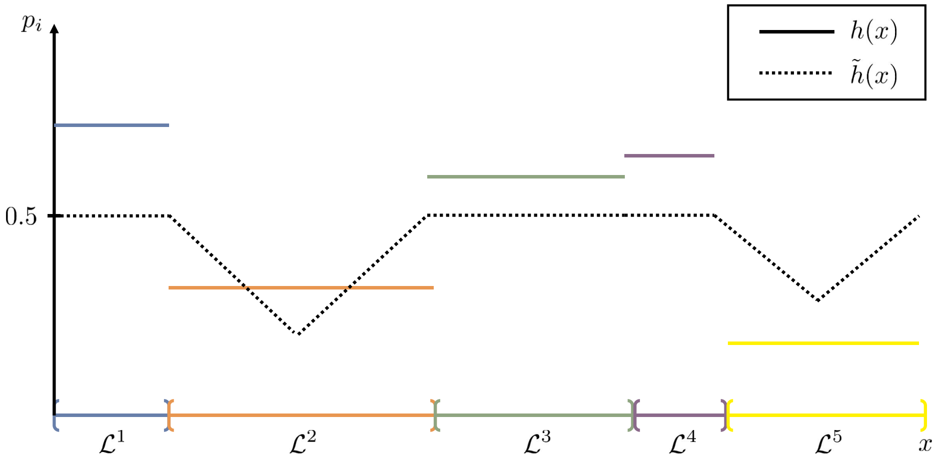
where $\mathcal{N}^i_\le$ and $\mathcal{N}^i_<$ contain all split parameters of the leaf for the corresponding inequality type. For ease of notation in the following, we do not distinguish between strict and nonstrict inequalities and define $\mathcal{N}^i = N^i_\le \cup \mathcal{N}^i_<$. Furthermore, it holds that $\mathbb{R}^n = \cup_{i \in L} \mathcal{L}^i$, where L is the index set of all leaves of the tree. Each leaf i is assigned a weight $p_i \in [0,1]$, which is usually determined by the fraction of training data of class 1 inside the leaf. The classifier is a piecewise constant function h, where $h(\boldsymbol{x}) = p_i$ if and only if $\boldsymbol{x}$ is contained in leaf i. Because h is a discontinuous step function, it is not Lipschitz continuous. To achieve convergence of our algorithm, we have to find a Lipschitz-continuous function assigning the same classes to each data point as h. To this end, we define the function

$$\tilde{h}(\boldsymbol{x}) = \begin{cases} \tau, & \boldsymbol{x} \in \mathcal{L}_i, p_i \ge \tau; \\ \tau - \min\limits_{(\boldsymbol{a}, b) \in \mathcal{N}^i} b - \boldsymbol{a}^\top \boldsymbol{x}, & \boldsymbol{x} \in \mathcal{L}_i, p_i < \tau. \end{cases}$$

We choose this function to have a constant value of τ for all leaves with $p_i \ge \tau$, whereas for a point $\boldsymbol{x}$ in one of the other leaves, we subtract from τ the minimum slack value of the point over all leaf-defining constraints. Because the minimum slack on the boundary of the leaves is zero, $\tilde{h}$ is a continuous function, and it holds $\tilde{h}(\boldsymbol{x}) < \tau$ in the interior of the latter leaves. Note that the value of h decreases if a point is farther away from the boundary of the leaf. Figure 5 illustrates this construction on a one-dimensional feature space. Theoretically, other function classes than piecewise linear functions could be used to connect the leaf values as long as these functions are Lipschitz continuous on each leaf region. However, this would lead to nonlinear problem formulations for the adversarial problem, which would increase the computational effort of our method.

Unfortunately, because of imposed continuity, the predictions on the boundaries of the leaves can be different than the original predictions of h. We show in the following lemma that $\tilde{h}$ is Lipschitz continuous and except on the leaf boundaries, that the same class is assigned to each data point as it is done by the original classifier h.

Figure 5. (Color online) Example of the Functions h and $\tilde{h}$ in a One-Dimensional Feature Space



Lemma 1. *The function $\tilde{h}$ is Lipschitz continuous on $\mathcal{X}$ and* $\text{int}(\{\boldsymbol{x} : \tilde{h}(\boldsymbol{x}) \geq \tau\}) \subseteq \{\boldsymbol{x} : h(\boldsymbol{x}) \geq \tau\} \subseteq \{\boldsymbol{x} : \tilde{h}(\boldsymbol{x}) \geq \tau\}$, *where int$(\cdot)$ denotes the interior of the set.*

Proof. We first show that $\tilde{h}$ is Lipschitz continuous. To this end, let $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$. Consider the following three cases.

Case 1. Both points are contained in a leaf with prediction 1 (i.e., $\boldsymbol{x} \in \mathcal{L}_i$ and $\boldsymbol{y} \in \mathcal{L}_{i'}$ with $p_i, p_{i'} \geq \tau$). In this case, we have

$$|\tilde{h}(\boldsymbol{x}) - \tilde{h}(\boldsymbol{y})| = |\tau - \tau| = 0 \leq \|\boldsymbol{x} - \boldsymbol{y}\|. \tag{7}$$

Case 2. Point $\boldsymbol{x}$ is in a leaf with prediction 1, and point $\boldsymbol{y}$ is in a leaf with prediction -1 (i.e., $\boldsymbol{x} \in \mathcal{L}_i$ and $\boldsymbol{y} \in \mathcal{L}_{i'}$ with $p_i \geq \tau$ and $p_{i'} < \tau$). Because $\boldsymbol{x}$ is not contained in $\mathcal{L}_{i'}$, there must be split parameters $(\boldsymbol{a}_*, b_*)$ (without loss of generality, we assume that it is related to a nonstrict inequality) such that $\boldsymbol{a}_*^\top \boldsymbol{y} \leq b_*$ and $\boldsymbol{a}_*^\top \boldsymbol{x} > b_*$. It holds that $\tilde{h}(\boldsymbol{x}) = \tau \geq \tilde{h}(\boldsymbol{y})$, and we obtain

$$\begin{aligned}
|\tilde{h}(\boldsymbol{x}) - \tilde{h}(\boldsymbol{y})| &= \tau - \left(\tau - \min_{(\boldsymbol{a}, b) \in \mathcal{N}^{i'}} b - \boldsymbol{a}^\top \boldsymbol{y}\right) && (8)\\
&= \min_{(\boldsymbol{a}, b) \in \mathcal{N}^{i'}} b - \boldsymbol{a}^\top \boldsymbol{y} && (9)\\
&\leq b_* - \boldsymbol{a}_*^\top \boldsymbol{y} && (10)\\
&< b_* - \boldsymbol{a}_*^\top \boldsymbol{y} + \boldsymbol{a}_*^\top \boldsymbol{x} - b_* && (11)\\
&= \boldsymbol{a}_*^\top (\boldsymbol{x} - \boldsymbol{y}) && (12)\\
&\leq \|\boldsymbol{a}_*\| \|\boldsymbol{x} - \boldsymbol{y}\|, && (13)
\end{aligned}$$

where the first inequality follows from $(\boldsymbol{a}_*, b_*) \in \mathcal{N}^{i'}$, the second inequality follows from $\boldsymbol{a}_*^\top \boldsymbol{x} > b_*$, and for the last inequality, we apply the Cauchy–Schwarz inequality.

Case 3. Both points are contained in a leaf with prediction -1 (i.e., $\boldsymbol{x} \in \mathcal{L}_i$ and $\boldsymbol{y} \in \mathcal{L}_{i'}$ with $p_i, p_{i'} < \tau$). First, assume that $i \neq i'$. Without loss of generality, we also assume that $\tilde{h}(\boldsymbol{x}) \geq \tilde{h}(\boldsymbol{y})$. In this case, we have

$$|\tilde{h}(\boldsymbol{x}) - \tilde{h}(\boldsymbol{y})| \leq \tau - \left(\tau - \min_{(\boldsymbol{a}, b) \in \mathcal{N}^{i'}} b - \boldsymbol{a}^\top \boldsymbol{y}\right), \tag{14}$$

which follows from $\tilde{h}(\boldsymbol{x}) \leq \tau$ for all $\boldsymbol{x} \in \mathcal{X}$. We can prove Lipschitz continuity in this case by following the same steps as in Case 2. When $i = i'$, we designate $(\boldsymbol{a}_*, b_*)$ as the parameters that attain the minimum in

$$\tau - \min_{(\boldsymbol{a}, b) \in \mathcal{N}^{i}} b - \boldsymbol{a}^\top \boldsymbol{x}. \tag{15}$$

Then, we have

$$
\begin{aligned}
|\tilde{h}(\boldsymbol{x}) - \tilde{h}(\boldsymbol{y})| &= \tau - b_* + \boldsymbol{a}_*^\top \boldsymbol{x} - \left(\tau - \min_{(\boldsymbol{a},b)\in\mathcal{N}^{i'}} b - \boldsymbol{a}^\top \boldsymbol{y}\right) && (16)\\
&\le -b_* + \boldsymbol{a}_*^\top \boldsymbol{x} + b_* - \boldsymbol{a}_*^\top \boldsymbol{y} && (17)\\
&= \boldsymbol{a}_*^\top(\boldsymbol{x} - \boldsymbol{y}) && (18)\\
&\le \|\boldsymbol{a}_*\| \|\boldsymbol{x} - \boldsymbol{y}\|, && (19)
\end{aligned}
$$

where we use $(\boldsymbol{a}_*, b_*) \in \mathcal{N}^{i'}$ for the first inequality and the Cauchy–Schwarz inequality for the last one.

Following the three cases, we show that $\tilde{h}$ is Lipschitz continuous with Lipschitz constant $L = \max_{i\in\mathcal{L}} \max_{(\boldsymbol{a},b)\in\mathcal{N}^i} \|\boldsymbol{a}\|$.

We now show the second part of the result. First, assume for $\boldsymbol{x}$ that $h(\boldsymbol{x}) \ge \tau$. This implies that $\boldsymbol{x}$ is contained in a leaf $\mathcal{L}_i$ with $p_i \ge \tau$, and hence, $\tilde{h}(\boldsymbol{x}) = \tau$ showing the second inclusion. For the first inclusion, let $\boldsymbol{x}$ now be a point in the interior of the set $\{\boldsymbol{x} : \tilde{h}(\boldsymbol{x}) \ge \tau\}$. Assume the contrary of the statement (i.e., it is contained in a leaf $\mathcal{L}_i$ with $p_i < \tau$). We can assume that the leaf is full dimensional because otherwise, the interior is empty. Then, by definition of $\tilde{h}$, it must hold that

$$\tau - \min_{(\boldsymbol{a},b)\in\mathcal{N}^i} b - \boldsymbol{a}^\top \boldsymbol{x} \ge \tau. \tag{20}$$

That is, there is a $(\boldsymbol{a}, b) \in \mathcal{N}^i$ such that $\boldsymbol{a}^\top \boldsymbol{x} = b$. Because the leaf is a full-dimensional polyhedron, there exists a $\overline{\delta} > 0$ and a direction $\boldsymbol{v}$ such that $\boldsymbol{a}^\top(\boldsymbol{x} + \delta \boldsymbol{v}) < b$ for all $0 < \delta < \overline{\delta}$ and all $(\boldsymbol{a}, b) \in \mathcal{N}^i$. Consequently, $\tilde{h}(\boldsymbol{x} + \delta\boldsymbol{v}) < \tau$ for all $0 < \delta < \overline{\delta}$. This implies that $\boldsymbol{x}$ cannot be in the interior of the set $\{\boldsymbol{x} : \tilde{h}(\boldsymbol{x}) \ge \tau\}$, which is a contradiction. Thus, $\boldsymbol{x}$ must be contained in a leaf with $p_i \ge \tau$, which proves the result. □

We can now derive the formulations for (MP) and (AP) for our tree model. By using Lemma 1, we can use h instead of $\tilde{h}$ to model Constraints (6) in (MP). Then, we can adapt the decision tree formulation proposed by Maragno et al. (2022) and reformulate Constraint (6) of (MP) as

$$
\begin{aligned}
&\boldsymbol{a}^\top(\boldsymbol{x} + \boldsymbol{s}) - M(1 - l_i(\boldsymbol{s})) \le b, i \in \mathcal{L}, (\boldsymbol{a}, b) \in \mathcal{N}^i_\le, \boldsymbol{s} \in \mathcal{Z}, && (21)\\
&\boldsymbol{a}^\top(\boldsymbol{x} + \boldsymbol{s}) - M(1 - l_i(\boldsymbol{s})) < b, i \in \mathcal{L}, (\boldsymbol{a}, b) \in \mathcal{N}^i_<, \boldsymbol{s} \in \mathcal{Z}, && (22)\\
&\sum_{i\in\mathcal{L}} l_i(\boldsymbol{s}) = 1, \quad \boldsymbol{s} \in \mathcal{Z}, && (23)\\
&\sum_{i\in\mathcal{L}} l_i(\boldsymbol{s}) p_i \ge \tau, \quad \boldsymbol{s} \in \mathcal{Z}, && (24)\\
&l_i(\boldsymbol{s}) \in \{0, 1\}, \quad i \in \mathcal{L}, \boldsymbol{s} \in \mathcal{Z}, && (25)
\end{aligned}
$$

where M is a predefined large-enough constant. The variables $l_i(\boldsymbol{s})$ are binary variables associated with the corresponding leaf i and scenario $\boldsymbol{s}$, where $l_i(\boldsymbol{s}) = 1$ if solution $\boldsymbol{x} + \boldsymbol{s}$ is contained in leaf i. Constraints (23) ensure that each scenario gets assigned to exactly one leaf. Constraints (21) and (22) ensure that only if leaf i is selected for scenario $\boldsymbol{s}$ (i.e., $l_i(\boldsymbol{s}) = 1$), then $\boldsymbol{x} + \boldsymbol{s}$ has to fulfill the corresponding constraints of $\mathcal{L}_i$, whereas the constraints for all other leaves can be violated, which is ensured by the big-M value. Note that in our computational experiments, we use an $\tilde{\varepsilon}$-accuracy parameter to reformulate the strict inequalities as nonstrict inequalities. Finally, Constraints (24) ensure that the chosen leaf has a weight p_i that is greater than or equal to the threshold τ. We can remove all variables and constraints of the problem related to leaves with $p_i < \tau$ together with Constraint (24) because only leaves that correspond to label +1 can be chosen to obtain a feasible solution.

Using the Lipschitz-continuous function $\tilde{h}$, (AP) can be reformulated as

$$\tau + \max_{\boldsymbol{s}\in\mathcal{S}} -\tilde{h}(\boldsymbol{x}^* + \boldsymbol{s}). \tag{26}$$

Optimizing $-\tilde{h}(\boldsymbol{x}^* + \boldsymbol{s})$ over $\mathcal{S}$ is equivalent to iterating over all leaves $\mathcal{L}_i$ with $p_i < \tau$ and maximizing the same function over the corresponding leaf. The problem is formulated as

$$
\begin{aligned}
\text{maximize} - \tau + &\min_{(\boldsymbol{a},b)\in\mathcal{N}^i} \{b - \boldsymbol{a}^\top(\boldsymbol{x}^* + \boldsymbol{s})\} && (27)\\
\text{subject to}\ \ &\boldsymbol{a}^\top(\boldsymbol{x}^* + \boldsymbol{s}) \le b, \quad (\boldsymbol{a}, b) \in \mathcal{N}^i_\le, && (28)\\
&\boldsymbol{a}^\top(\boldsymbol{x}^* + \boldsymbol{s}) < b, \quad (\boldsymbol{a}, b) \in \mathcal{N}^i_<, && (29)\\
&\boldsymbol{s} \in \mathcal{S} && (30)
\end{aligned}
$$

for each such leaf. Using a level-set transformation and substituting the latter problem in (26) lead to

$$\begin{aligned} \text{maximize} \quad & \alpha && (31)\\ \text{subject to} \quad & \alpha \le w_{(\boldsymbol{a},b)}, \quad (\boldsymbol{a},b) \in \mathcal{N}^i, && (32)\\ & \boldsymbol{a}^\top(\boldsymbol{x}^* + \boldsymbol{s}) + w_{(\boldsymbol{a},b)} \le b, \quad (\boldsymbol{a},b) \in \mathcal{N}^i_{\le}, && (33)\\ & \boldsymbol{a}^\top(\boldsymbol{x}^* + \boldsymbol{s}) + w_{(\boldsymbol{a},b)} < b, \quad (\boldsymbol{a},b) \in \mathcal{N}^i_{<}, && (34)\\ & \boldsymbol{s} \in \mathcal{S}, \boldsymbol{w} \ge 0, && (35) \end{aligned}$$

which is equivalent to maximizing the minimum slacks of the constraints corresponding to the leaves. Geometrically this means that we try to find a perturbation $\boldsymbol{s}$ such that $\boldsymbol{x}^* + \boldsymbol{s}$ is as deep as possible in one of the negative leaves; see Figure 6. Note that Problems (31) are continuous optimization problems that can be solved efficiently by state-of-the-art solvers, such as Gurobi (Gurobi Optimization LLC 2022) or CPLEX (Cplex II 2009).

3.1.1. Heuristic Variant. Using Algorithm 1 can be computationally demanding because it requires solving (MP) and (AP) many times in an iterative manner. An alternative and more efficient approach can be conducted, where we try to find a CE $\boldsymbol{x}^*$ that is robust only regarding one leaf of the tree. More precisely, this means that $\boldsymbol{x}^* + \boldsymbol{s}$ is contained in the same leaf for all $\boldsymbol{s} \in \mathcal{S}$. This is an approximation because for each scenario $\boldsymbol{s}$, the point $\boldsymbol{x}^* + \boldsymbol{s}$ could be contained in a different neighboring leaf, leading to better CEs. Hence, the solutions of the latter approach may be nonoptimal. When restricting to one leaf, we can iterate over all possible leaves $\mathcal{L}_i$ with $p_i < \tau$ and solve the resulting (MP):

$$\begin{aligned} \text{minimize} \quad & d(\boldsymbol{x}, \hat{\boldsymbol{x}}) && (36)\\ \text{subject to} \quad & \boldsymbol{a}^\top \boldsymbol{x} + \rho \|\boldsymbol{a}\|^* \le b, \quad (\boldsymbol{a},b) \in \mathcal{N}^i_{\le}, && (37)\\ & \boldsymbol{a}^\top \boldsymbol{x} + \rho \|\boldsymbol{a}\|^* < b, \quad (\boldsymbol{a},b) \in \mathcal{N}^i_{<}, && (38)\\ & \boldsymbol{x} \in \mathcal{X}, && (39) \end{aligned}$$

and we choose the solution $\boldsymbol{x}^*$ for the leaf that yields the best optimal value. Note that in that case, we do not need binary assignment variables anymore because we only consider one leaf for (MP). Alternatively, we can obtain the same result modeling the entire decision tree using auxiliary binary variables, one for each leaf i with $p_i \ge \tau$. In Figure 7, we present the computation time and the distance of the calculated CE to the factual instance using both the heuristic and the (exact) adversarial algorithm. The results show that the heuristic approach outperforms the exact method in terms of speed, and its computation time remains unaffected by the robustness budget ρ. However, it is noteworthy that the CEs generated through the heuristic method have a larger distance to the factual instance, where the difference to the optimal distance provided by our exact algorithm increases with increasing ρ; see Figure 8.

3.2. Tree Ensembles

In the case of tree ensembles, like random forest (RF) and gradient-boosting machines (GBMs), we model the validity constraints by formulating each base learner separately. Assume that we obtain K base learners. Because each base learner is a decision tree, we can use the construction of Constraints (21)–(25) and apply them to all base learners

Figure 6. (Color online) Slack Values of a Solution Regarding the Leaf-Defining Constraints Where the Minimum Slack Is Maximized

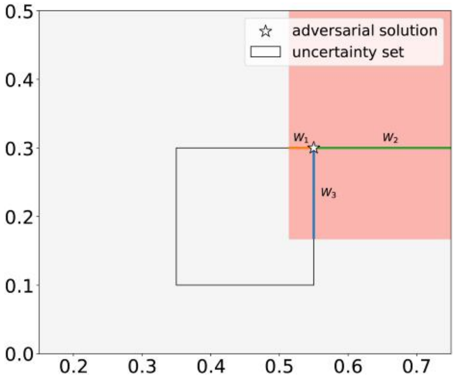
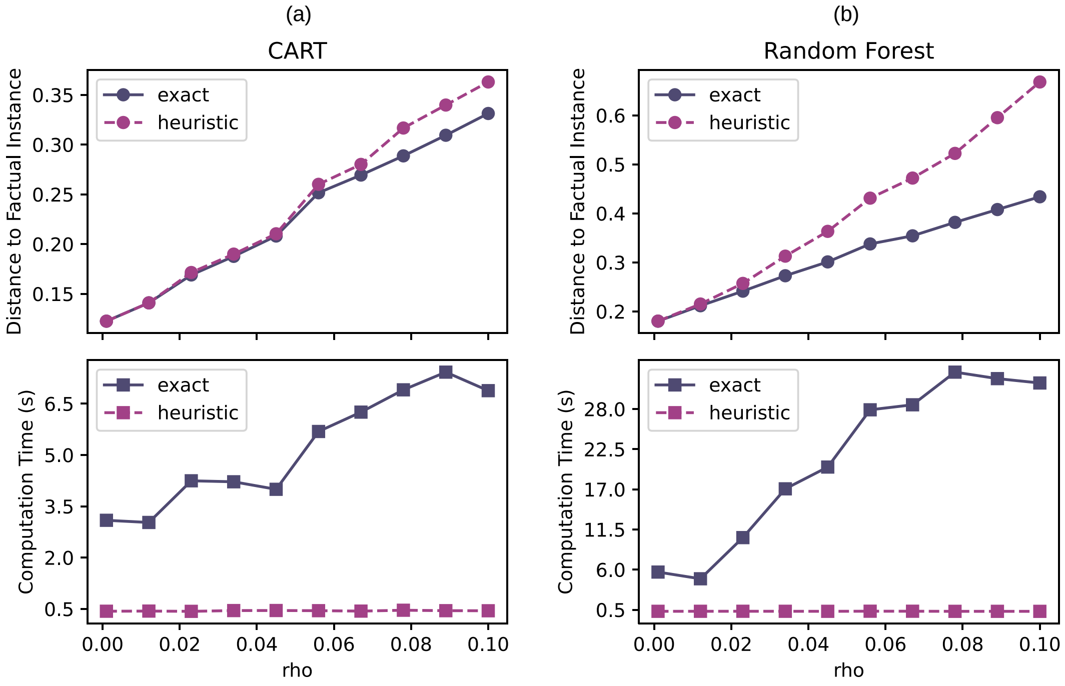


Figure 7. (Color online) Comparison of Counterfactual Explanations Generated Using the Heuristic Method and the Adversarial Algorithm (Exact Method) in Terms of Computation Time and Distance Between the Factual Instance and the Counterfactual Explanations



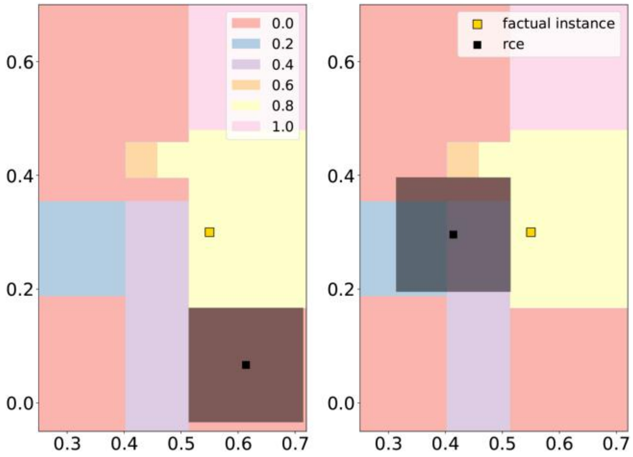
Notes. Results are shown for (a) decision trees trained by CART and (b) random forests. The results are obtained using the Diabetes data set and are averaged over 10 distinct factual instances. (a) DT with maximum depth of five. (b) RF with 10 DTs.

separately. Then, we add the constraints to the master problem, where each base learner k gets a separate copy $l_i(\boldsymbol{s})(k)$ of the assignment variables and has its own set of node inequalities given by $\boldsymbol{a}(\boldsymbol{k})$ and $b(k)$. Additionally, we have to replace Constraint (24) by

$$\frac{\sum_{k=1}^{K}\sum_{i\in\mathcal{L}} l_i(\boldsymbol{s})(k)p_i(k)}{K} \geq \tau, \tag{40}$$

where $p_i(k)$ is the weight of leaf i in base learner k. This constraint forces the average prediction value of the tree ensemble to be larger than or equal to τ. Note that to model a majority vote, we can use $p_i(k) \in \{0,1\}$. Because a

Figure 8. (Color online) (Left Panel) The CE of the Heuristic Approach Where the Whole Set Is Restricted to Be Contained in One Leaf



Note. (Right panel) An optimal CE where the uncertainty set can overlap over different leaves of a decision tree.

random forest is equivalent to a decision tree, the same methodology for (AP) can be used as in Section 3.1. Note that for classical DTs, we may iterate over all leaves and solve Problem (31). However, deriving the polyhedral descriptions of all leaves for an ensemble of trees is very time consuming. Instead, (AP) can be reformulated as

$$\begin{aligned}
&\text{maximize} \quad \alpha && (41)\\
&\text{subject to} \quad \alpha \le w_{(\boldsymbol{a}(\boldsymbol{k}),b(k))}, \quad (\boldsymbol{a},b) \in \mathcal{N}^l(k), \;\; \forall k \in [K], && (42)\\
&\qquad \boldsymbol{a}(\boldsymbol{k})^\top(\boldsymbol{x}^* + \boldsymbol{s}) + w_{(\boldsymbol{a}(\boldsymbol{k}),b(k))} \le b(k), \quad (\boldsymbol{a}(\boldsymbol{k}),b(k)) \in \mathcal{N}^i_{\le}(k), \;\; \forall k \in [K], && (43)\\
&\qquad \boldsymbol{a}(\boldsymbol{k})^\top(\boldsymbol{x}^* + \boldsymbol{s}) + w_{(\boldsymbol{a}(\boldsymbol{k}),b(k))} < b(k), \quad (\boldsymbol{a}(\boldsymbol{k}),b(k)) \in \mathcal{N}^i_{<}(k), \;\; \forall k \in [K], && (44)\\
&\qquad \boldsymbol{s} \in \mathcal{S}, \; \boldsymbol{w} \ge 0, && (45)
\end{aligned}$$

where $\mathcal{N}^i_{\le}(k), \mathcal{N}^i_{<}(k)$ contain the split parameters $(\boldsymbol{a}(\boldsymbol{k}), b(k))$ of the nodes of tree k as defined in Section 3.1, and we use $[K]$ to denote the set of the first K positive integers: that is, $[K] = \{1, \dots, K\}$.

Finally, note that because the classifier of an ensemble of trees is equivalent to a classical decision tree classifier, the convergence analysis presented in Section 3.1 also holds for the ensemble case.

3.3. Neural Networks

In the case of neural networks, convergence of Algorithm 1 is immediately guaranteed when we consider ReLU activation functions. More precisely, the evaluation function $h : \mathcal{X} \longrightarrow [0,1]$ of a trained neural network with rectified linear unit (ReLU) activation functions is Lipschitz continuous because it is a concatenation of Lipschitz-continuous functions; see Online Appendix C for a formal proof.

Moreover, neural networks with ReLU activation functions (NN) can be represented by mixed-integer programming formulations (Grimstad and Andersson 2019, Anderson et al. 2020), and we adopt the formulation proposed by Fischetti and Jo (2018). The ReLU operator of a neuron in layer l is given by

$$v_i^l = \max\left\{0, \beta_{i0}^l + \sum_{j \in N^{l-1}} \beta_{ij}^l v_j^{l-1}\right\}, \tag{46}$$

where $\boldsymbol{\beta}_i^l$ is the coefficient vector for neuron i in layer l, β_{i0}^l is the bias value, and v_j^{l-1} is the output of neuron j of layer $l-1$. Note that in our model, the input of the neural network can be a data point $\boldsymbol{x}$ perturbed by a scenario $\boldsymbol{s}$ (i.e., all variables depend on the perturbation $\boldsymbol{s}$). The input in the first layer is $\boldsymbol{v}^0(\boldsymbol{s}) = \boldsymbol{x} + \boldsymbol{s}$, and the output of layer l is denoted as $v_j^l(\boldsymbol{s})$. The ReLU operator (46) can then be linearly reformulated as

$$\begin{aligned}
&v_i^l(\boldsymbol{s}) \ge \beta_{i0}^l + \sum_{j \in \mathcal{N}^{l-1}} \beta_{ij}^l v_j^{l-1,s}, \quad \boldsymbol{s} \in \mathcal{Z}, && (47)\\
&v_i^l(\boldsymbol{s}) \le \beta_{i0}^l + \sum_{j \in \mathcal{N}^{l-1}} \beta_{ij}^l v_j^{l-1,s} + M_{LB}(1 - l_i^l(\boldsymbol{s})), \quad \boldsymbol{s} \in \mathcal{Z}, && (48)\\
&v_i^l(\boldsymbol{s}) \le M_{UB} l_i^l(\boldsymbol{s}), \; \boldsymbol{s} \in \mathcal{Z}, && (49)\\
&v_i^l(\boldsymbol{s}) \ge 0, \quad \boldsymbol{s} \in \mathcal{Z}, && (50)\\
&l_i^l(\boldsymbol{s}) \in \{0,1\}, \quad \boldsymbol{s} \in \mathcal{Z}, && (51)
\end{aligned}$$

where M_{LB} and M_{UB} are big-M values.

The following is a complete formulation of the master problem (MP) in the case of neural networks with ReLU activation functions:

$$\begin{aligned}
&\underset{x \in \mathcal{X}}{\text{minimize}} \quad d(\boldsymbol{x}, \hat{\boldsymbol{x}}) && (52)\\
&\text{subject to} \quad \sum_{j \in \mathcal{N}^L} \beta_j^L v_j^{L-1}(\boldsymbol{s}) \ge \tau, \quad \boldsymbol{s} \in \mathcal{Z}, && (53)\\
&\qquad v_i^l(\boldsymbol{s}) \ge \beta_{i0}^l + \sum_{j \in \mathcal{N}^{l-1}} \beta_{ij}^l v_j^{l-1}(\boldsymbol{s}), \quad \boldsymbol{s} \in \mathcal{Z}, \; i \in \mathcal{N}^l, \;\; \forall l \in [L], && (54)\\
&\qquad v_i^l(\boldsymbol{s}) \le \beta_{i0}^l + \sum_{j \in \mathcal{N}^{l-1}} \beta_{ij}^l v_j^{l-1}(\boldsymbol{s}) + M_{LB}(1 - l_i^l(\boldsymbol{s})), \quad \boldsymbol{s} \in \mathcal{Z}, \; i \in \mathcal{N}^l, \\
&\qquad\qquad \forall l \in [L], && (55)\\
&\qquad v_i^l(\boldsymbol{s}) \le M_{UB} l_i^l(\boldsymbol{s}), \; \boldsymbol{s} \in \mathcal{Z}, \; i \in \mathcal{N}^l, \;\; \forall l \in [L], && (56)
\end{aligned}$$

$$v_i^0(\boldsymbol{s}) = x_i + s_i, \quad \boldsymbol{s} \in \mathcal{Z}, \quad \forall i \in [n], \tag{57}$$

$$v_i^l(\boldsymbol{s}) \geq 0, \quad \boldsymbol{s} \in \mathcal{Z}, \ i \in \mathcal{N}^l, \quad \forall l \in [L], \tag{58}$$

$$l_i^l(\boldsymbol{s}) \in \{0,1\}, \quad \boldsymbol{s} \in \mathcal{Z}, \ i \in \mathcal{N}^l, \quad \forall l \in [L], \tag{59}$$

where L represents the number of layers with $[L] = \{1, \ldots, L\}$ and $\mathcal{N}^l$ is the set of neurons in layer l. The first $L-1$ layers are activated by an ReLU function except for the output layer, which consists of a single node that is a linear combination of the node values in layer $L-1$. The variable $v_i^l(\boldsymbol{s})$ is the output of the activation function in node i, layer l, and scenario $\boldsymbol{s}$.

Likewise, the adversarial Problem (AP) is formulated as

$$\underset{\boldsymbol{s}\in\mathcal{S}}{\text{maximize}} \quad \tau - \sum_{j\in\mathcal{N}^L} \beta_j^L v_j^{L-1}, \tag{60}$$

$$\text{subject to} \quad v_i^l \geq \beta_{i0}^l + \sum_{j\in\mathcal{N}^{l-1}} \beta_{ij}^l v_j^{l-1}, \quad i \in \mathcal{N}^l, \quad \forall l \in [L], \tag{61}$$

$$v_i^l \leq \beta_{i0}^l + \sum_{j\in\mathcal{N}^{l-1}} \beta_{ij}^l v_j^{l-1} + M_{LB}(1 - l_i^l), \quad i \in \mathcal{N}^l, \quad \forall l \in [L], \tag{62}$$

$$v_i^l \leq M_{UB} l_i^l, \ i \in \mathcal{N}^l, \quad \forall l \in [L], \tag{63}$$

$$v_i^0 = s_i + x_i^*, \quad i = 1, \ldots, n, \tag{64}$$

$$v_i^l \geq 0, \quad i \in \mathcal{N}^l, \quad \forall l \in [L], \tag{65}$$

$$l_i^l \in \{0,1\}, \quad i \in \mathcal{N}^l, \quad \forall l \in [L], \tag{66}$$

$$\boldsymbol{s} \in \mathcal{S}, \tag{67}$$

where $\boldsymbol{x}^*$ is the counterfactual solution of (MP).

4. Experiments

In this section, we aim to illustrate the effectiveness of our method by conducting empirical experiments on various data sets. The mixed-integer optimization formulations of the ML models used in our experiments are based on Maragno et al. (2023). In our experiments, we consistently set ϵ to 1 e-7 and utilize a big-M value of 1e3 for decision trees and tree ensembles, whereas for neural networks, we employ a big-M value of 100. The experiments were conducted on a computer with an Apple M1 Pro processor and 16 GB of RAM. For reproducibility, our open-source implementation can be found in our repository.[3] It is important to note that, to the best of our knowledge, the present work is the first approach that generates a region of CEs for a range of different models involving nondifferentiable models. Therefore, a comparison with prior work is only possible for the case of neural networks with ReLU activation functions. In the last part of the experiments, we compare our method against the one proposed by Dominguez-Olmedo et al. (2022) in terms of CE validity and robustness.

In the first part of the experiments, we analyze our method using three well-known data sets: Banknote Authentication, Diabetes, and Ionosphere (Dua and Graff 2017). Before training the ML models, we scaled each feature to be between zero and one. None of the data sets contain categorical features, which otherwise, would have been considered immutable or fixed according to the user's preference. We apply our algorithm to generate robust CEs for 20 factual instances randomly selected from the data set. We use ℓ_∞-norm as the uncertainty set with a radius (ρ) of 0.01 or 0.05. The distance function adopted is the ℓ_1-norm, which can be linearly expressed within the optimization model. For each instance, we use a time limit of 1,000 seconds. Although less practical from a user's perspective, we also report the results using ℓ_2-norm as an uncertainty set in Table 2. The data sets used in our analysis do not require any additional constraints, such as *actionability*, *sparsity*, or *data manifold closeness*. However, it is important to note that these types of constraints can be added to our master problem when needed by using constraints like the ones proposed in Maragno et al. (2022). The accuracy of each trained ML model is provided in Online Appendix D.

In Table 1, we show (from left to right) the type of ML model and model-specific hyperparameters, and for each data set, we show (from left to right) the average computation time (in seconds), the number of iterations performed by the algorithm, and the number of instances when the algorithm hits the time limit without providing an optimal solution. For the computation time and the number of iterations, we show the standard error values in parentheses. For the early stops, we show the (average) maximum radius of the uncertainty set, which is feasible for the generated counterfactual solutions in each iteration of the algorithm. The latter value gives a measure for the robustness of the returned solution. As for hyperparameters, we report the maximum tree depth for DT, the number of generated trees

Table 1. Generation of Robust CEs for 20 Factual Instances Using ℓ_∞-Norm as the Uncertainty Set

		Banknote authentication			Diabetes			Ionosphere		
		4 features			8 features			34 features		
Model	Specifications	Comp. time (s)	# iterations	# early stops	Comp. time (s)	# iterations	# early stops	Comp. time (s)	# iterations	# early stops
$\rho = 0.01$										
Linear	ElasticNet	0.25 (0.01)	—	—	0.23 (0.01)	—	—	0.25 (0.01)	—	—
DT	max depth: 3	1.53 (0.04)	1.00 (0.00)	—	1.66 (0.07)	1.20 (0.09)	—	1.66 (0.07)	1.10 (0.07)	—
	max depth: 5	1.86 (0.09)	1.10 (0.07)	—	2.29 (0.11)	1.20 (0.09)	—	1.96 (0.08)	1.10 (0.07)	—
	max depth: 10	3.90 (0.88)	2.00 (0.58)	—	5.77 (0.48)	1.40 (0.13)	—	2.86 (0.16)	1.20 (0.09)	—
RF[a]	# est.: 5	3.50 (0.36)	1.75 (0.22)	—	5.89 (1.98)	2.60 (0.83)	—	3.79 (0.24)	1.70 (0.13)	—
	# est.: 10	6.74 (1.03)	2.60 (0.40)	—	7.20 (0.94)	2.35 (0.29)	—	8.76 (0.89)	2.85 (0.33)	—
	# est.: 20	21.21 (4.61)	4.55 (0.99)	—	33.55 (7.48)	6.15 (0.79)	—	22.33 (2.83)	4.40 (0.51)	—
	# est.: 50	115.79 (34.35)	7.80 (1.65)	—	110.24 (32.43)	6.47 (1.39)	3 ($\bar{\rho}$ = 0.007)	137.26 (33.37)	8.20 (1.20)	—
	# est.: 100	214.38 (65.07)	6.44 (1.31)	2 ($\bar{\rho}$ = 0.009)	274.09 (71.93)	8.87 (1.49)	5 ($\bar{\rho}$ = 0.004)	285.62 (95.57)	8.27 (2.02)	9 ($\bar{\rho}$ = 0.004)
GBM[b]	# est.: 5	2.70 (0.22)	1.20 (0.14)	—	2.76 (0.22)	1.85 (0.17)	—	2.37 (0.15)	1.60 (0.13)	—
	# est.: 10	3.20 (0.30)	1.45 (0.15)	—	2.72 (0.29)	1.50 (0.24)	—	4.35 (0.44)	2.75 (0.30)	—
	# est.: 20	5.94 (0.50)	2.60 (0.23)	—	4.25 (0.45)	2.15 (0.28)	—	9.01 (1.11)	3.85 (0.50)	—
	# est.: 50	18.38 (1.62)	4.05 (0.35)	—	24.60 (8.21)	5.85 (1.41)	—	81.33 (28.39)	8.90 (1.36)	—
	# est.: 100	87.11 (26.24)	7.28 (0.77)	2 ($\bar{\rho}$ = 0.006)	164.32 (42.12)	11.63 (2.00)	1 ($\bar{\rho}$ = 0.004)	137.98 (22.74)	10.33 (0.97)	2 ($\bar{\rho}$ = 0.007)
NN	(10,)	1.63 (0.04)	1.00 (0.00)	—	1.55 (0.06)	1.00 (0.00)	—	2.22 (0.19)	1.80 (0.21)	—
	(10, 10, 10)	2.98 (0.15)	1.15 (0.08)	—	2.40 (0.12)	1.15 (0.08)	—	13.01 (3.57)	2.30 (0.40)	—
	(50,)	2.60 (0.14)	1.00 (0.00)	—	2.09 (0.12)	1.05 (0.05)	—	5.75 (0.75)	1.20 (0.12)	—
	(100,)	3.53 (0.15)	1.00 (0.00)	—	4.36 (0.62)	1.10 (0.07)	—	61.31 (33.11)	1.50 (0.22)	10 ($\bar{\rho}$ = 0.000)
$\rho = 0.05$										
Linear	ElasticNet	0.13 (0.00)	—	—	0.15 (0.01)	—	—	0.14 (0.00)	—	—
DT	max depth: 3	1.00 (0.06)	1.70 (0.15)	—	1.09 (0.07)	1.60 (0.15)	—	1.02 (0.07)	1.30 (0.15)	—
	max depth: 5	1.18 (0.11)	1.80 (0.22)	—	1.63 (0.13)	2.05 (0.22)	—	1.33 (0.10)	1.60 (0.18)	—
	max depth: 10	2.22 (0.41)	2.95 (0.61)	—	8.84 (1.49)	4.45 (0.59)	—	3.68 (2.14)	2.95 (1.49)	—
RF[a]	# est.: 5	2.85 (0.47)	3.25 (0.54)	—	4.29 (0.73)	5.25 (0.86)	—	2.27 (0.19)	2.35 (0.23)	—
	# est.: 10	13.51 (3.03)	8.95 (1.60)	—	14.71 (3.53)	8.35 (1.24)	—	7.41 (1.22)	5.15 (0.67)	—
	# est.: 20	13.86 (3.58)	5.53 (0.92)	1 ($\bar{\rho}$ = 0.041)	89.00 (28.17)	14.00 (2.28)	2 ($\bar{\rho}$ = 0.045)	47.44 (26.35)	9.50 (2.52)	—
	# est.: 50	101.21 (24.90)	11.37 (1.53)	1 ($\bar{\rho}$ = 0.048)	303.27 (93.95)	16.73 (2.55)	9 ($\bar{\rho}$ = 0.034)	307.72 (72.52)	19.31 (3.15)	7 ($\bar{\rho}$ = 0.044)
	# est.: 100	156.28 (33.21)	8.70 (1.02)	—	453.67 (111.27)	15.43 (2.19)	13 ($\bar{\rho}$ = 0.032)	156.45 (116.11)	5.75 (2.29)	16 ($\bar{\rho}$ = 0.029)
GBM[b]	# est.: 5	1.57 (0.15)	1.75 (0.24)	—	1.50 (0.10)	2.05 (0.20)	—	1.97 (0.32)	2.65 (0.50)	—
	# est.: 10	4.84 (0.58)	3.55 (0.46)	—	8.87 (4.44)	8.55 (3.21)	—	11.76 (6.51)	8.85 (3.27)	—
	# est.: 20	12.72 (2.22)	8.28 (0.85)	2 ($\bar{\rho}$ = 0.038)	41.81 (17.86)	17.05 (4.37)	1 ($\bar{\rho}$ = 0.025)	19.20 (6.32)	9.45 (1.80)	—
	# est.: 50	73.23 (27.00)	13.76 (1.82)	3 ($\bar{\rho}$ = 0.039)	223.87 (125.98)	19.86 (4.23)	13 ($\bar{\rho}$ = 0.027)	139.18 (56.80)	16.31 (3.03)	7 ($\bar{\rho}$ = 0.023)
	# est.: 100	274.22 (51.40)	17.25 (1.57)	4 ($\bar{\rho}$ = 0.040)	—	—	20 ($\bar{\rho}$ = 0.022)	537.13 (295.04)	15.00 (2.08)	17 ($\bar{\rho}$ = 0.022)
NN	(10,)	0.90 (0.01)	1.00 (0.00)	—	1.00 (0.04)	1.15 (0.08)	—	2.96 (0.23)	3.00 (0.25)	—
	(10, 10, 10)	1.48 (0.03)	1.00 (0.00)	—	2.02 (0.26)	1.65 (0.21)	—	229.69 (104.80)	4.90 (0.67)	10 ($\bar{\rho}$ = 0.039)
	(50,)	1.39 (0.06)	1.00 (0.00)	—	1.75 (0.13)	1.35 (0.11)	—	19.06 (6.63)	2.37 (0.24)	1 ($\bar{\rho}$ = 0.049)
	(100,)	1.83 (0.05)	1.00 (0.00)	—	5.88 (1.19)	1.80 (0.16)	—	289.62 (125.61)	2.70 (0.30)	10 ($\bar{\rho}$ = 0.000)

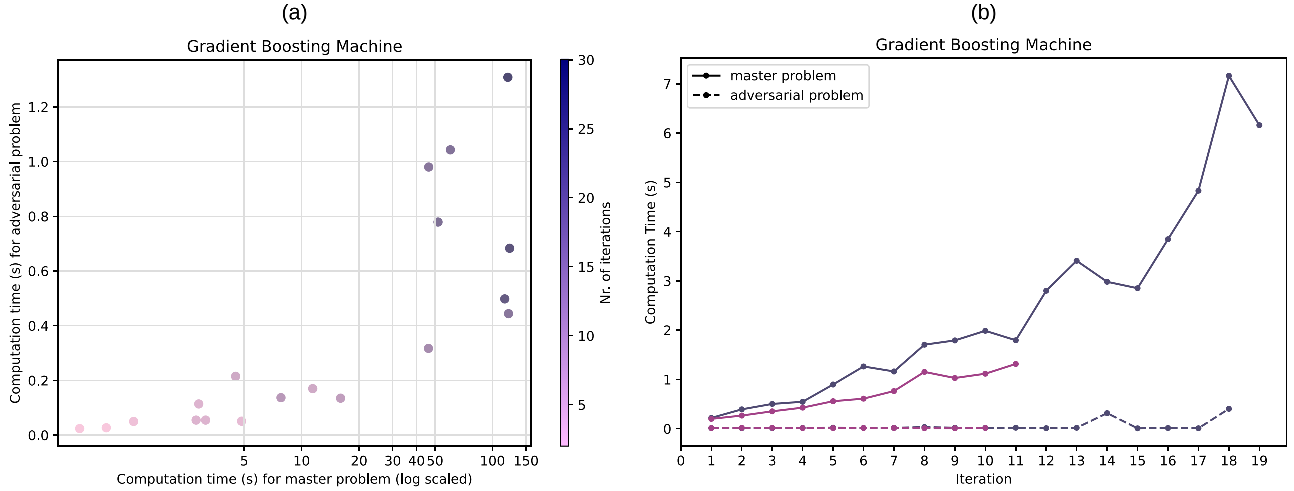
[a] Maximum depth of each decision tree equal to three.
[b] Maximum depth of each decision tree equal to two.

Table 2. Generation of Robust CEs for 20 Factual Instances Using ℓ_2-Norm as the Uncertainty Set

		Banknote authentication			Diabetes			Ionosphere		
		4 features			8 features			34 features		
Model	Specifications	Comp. time (s)	# iterations	# early stops	Comp. time (s)	# iterations	# early stops	Comp. time (s)	# iterations	# early stops
$\rho = 0.01$										
Linear	ElasticNet	0.24 (0.01)	—	—	0.24 (0.01)	—	—	0.26 (0.01)	—	—
DT	max depth: 3	2.09 (0.11)	1.45 (0.11)	—	2.18 (0.13)	1.70 (0.15)	—	5.54 (0.38)	1.20 (0.09)	—
	max depth: 5	2.64 (0.14)	1.53 (0.12)	1 ($\bar{\rho}$ = 0.000)	4.21 (0.39)	2.00 (0.23)	—	10.17 (0.78)	1.56 (0.16)	4 ($\bar{\rho}$ = 0.000)
	max depth: 10	4.61 (0.88)	2.70 (0.58)	—	14.17 (2.64)	2.63 (0.50)	1 ($\bar{\rho}$ = 0.000)	21.75 (2.58)	2.06 (0.26)	2 ($\bar{\rho}$ = 0.000)
RF[a]	# est.: 5	5.20 (0.62)	2.35 (0.34)	—	7.96 (1.65)	3.17 (0.55)	2 ($\bar{\rho}$ = 0.004)	69.94 (10.77)	4.41 (0.54)	3 ($\bar{\rho}$ = 0.006)
	# est.: 10	10.83 (2.68)	3.41 (0.78)	3 ($\bar{\rho}$ = 0.006)	15.20 (3.17)	4.20 (0.68)	—	237.03 (41.12)	5.50 (0.70)	4 ($\bar{\rho}$ = 0.003)
	# est.: 20	22.59 (7.10)	4.15 (1.22)	7 ($\bar{\rho}$ = 0.004)	104.42 (27.35)	9.81 (1.78)	4 ($\bar{\rho}$ = 0.007)	370.21 (41.84)	7.33 (0.73)	5 ($\bar{\rho}$ = 0.004)
	# est.: 50	61.93 (14.29)	3.89 (1.22)	11 ($\bar{\rho}$ = 0.004)	137.37 (39.96)	6.50 (1.51)	10 ($\bar{\rho}$ = 0.004)	570.88 (111.94)	9.70 (1.61)	10 ($\bar{\rho}$ = 0.001)
	# est.: 100	103.33 (31.75)	3.20 (0.89)	10 ($\bar{\rho}$ = 0.004)	177.47 (41.01)	3.80 (0.86)	15 ($\bar{\rho}$ = 0.002)	531.13 (121.00)	6.17 (0.98)	14 ($\bar{\rho}$ = 0.001)
GBM[b]	# est.: 5	4.50 (0.50)	2.50 (0.34)	—	4.67 (0.47)	2.45 (0.26)	—	11.47 (1.48)	4.42 (0.66)	8 ($\bar{\rho}$ = 0.002)
	# est.: 10	6.87 (0.88)	3.15 (0.42)	—	6.11 (1.04)	2.70 (0.48)	—	42.68 (6.67)	6.88 (0.86)	3 ($\bar{\rho}$ = 0.001)
	# est.: 20	14.02 (1.09)	4.40 (0.34)	—	10.97 (1.45)	3.65 (0.49)	—	58.16 (7.01)	9.62 (1.00)	4 ($\bar{\rho}$ = 0.004)
	# est.: 50	34.03 (3.15)	5.21 (0.44)	1 ($\bar{\rho}$ = 0.010)	76.04 (33.09)	9.35 (2.21)	—	192.70 (42.52)	14.77 (2.47)	7 ($\bar{\rho}$ = 0.006)
	# est.: 100	133.53 (24.40)	8.50 (0.98)	6 ($\bar{\rho}$ = 0.007)	201.13 (77.68)	12.08 (2.92)	8 ($\bar{\rho}$ = 0.005)	415.96 (80.09)	17.29 (3.36)	13 ($\bar{\rho}$ = 0.003)
NN	(10,)	1.64 (0.09)	1.00 (0.00)	—	1.73 (0.05)	1.00 (0.00)	—	4.73 (0.57)	1.27 (0.19)	9 ($\bar{\rho}$ = 0.004)
	(10, 10, 10)	2.54 (0.15)	1.00 (0.00)	—	2.04 (0.10)	1.00 (0.00)	6 ($\bar{\rho}$ = 0.000)	282.24 (193.65)	1.50 (0.50)	18 ($\bar{\rho}$ = 0.003)
	(50,)	2.42 (0.14)	1.00 (0.00)	—	2.00 (0.08)	1.00 (0.00)	1 ($\bar{\rho}$ = 0.000)	48.13 (12.76)	2.40 (0.51)	15 ($\bar{\rho}$ = 0.008)
	(100,)	3.57 (0.14)	1.00 (0.00)	—	5.28 (0.83)	1.15 (0.11)	—	352.64 (153.96)	1.09 (0.09)	9 ($\bar{\rho}$ = 0.000)
$\rho = 0.05$										
Linear	ElasticNet	0.15 (0.00)	—	—	0.34 (0.02)	—	—	0.26 (0.01)	—	—
DT	max depth: 3	1.72 (0.17)	2.25 (0.19)	—	6.71 (0.65)	2.40 (0.24)	—	4.40 (1.35)	2.60 (0.90)	—
	max depth: 5	2.69 (0.52)	3.95 (0.74)	1 ($\bar{\rho}$ = 0.026)	20.28 (2.88)	5.00 (0.68)	1 ($\bar{\rho}$ = 0.000)	5.91 (1.09)	2.26 (0.40)	1 ($\bar{\rho}$ = 0.000)
	max depth: 10	4.44 (0.80)	4.95 (0.93)	—	178.35 (34.48)	9.11 (1.30)	1 ($\bar{\rho}$ = 0.045)	20.91 (10.20)	5.05 (1.51)	—
RF[a]	# est.: 5	4.49 (0.64)	4.40 (0.64)	—	117.95 (28.96)	9.63 (1.82)	1 ($\bar{\rho}$ = 0.049)	28.79 (2.76)	6.35 (0.51)	—
	# est.: 10	12.39 (2.57)	6.82 (1.14)	3 ($\bar{\rho}$ = 0.048)	200.39 (69.46)	12.92 (2.93)	7 ($\bar{\rho}$ = 0.043)	232.63 (36.46)	10.79 (1.36)	1 ($\bar{\rho}$ = 0.042)
	# est.: 20	51.29 (18.03)	10.17 (2.32)	8 ($\bar{\rho}$ = 0.049)	149.39 (46.57)	12.75 (2.78)	12 ($\bar{\rho}$ = 0.042)	450.46 (68.06)	8.50 (1.06)	10 ($\bar{\rho}$ = 0.027)
	# est.: 50	93.18 (51.21)	7.78 (2.17)	11 ($\bar{\rho}$ = 0.048)	560.78 (—)	6.00 (—)	19 ($\bar{\rho}$ = 0.031)	510.69 (341.34)	7.50 (3.50)	18 ($\bar{\rho}$ = 0.021)
	# est.: 100	108.47 (29.76)	5.25 (0.80)	8 ($\bar{\rho}$ = 0.047)	868.10 (—)	4.00 (—)	19 ($\bar{\rho}$ = 0.024)	495.03 (268.88)	9.00 (2.08)	17 ($\bar{\rho}$ = 0.014)
GBM[b]	# est.: 5	2.67 (0.33)	3.55 (0.47)	—	8.89 (1.29)	3.68 (0.53)	1 ($\bar{\rho}$ = 0.000)	11.38 (1.83)	5.74 (0.68)	1 ($\bar{\rho}$ = 0.029)
	# est.: 10	5.77 (0.45)	5.55 (0.44))	—	46.46 (14.33)	12.00 (3.12)	—	72.15 (15.31)	15.19 (2.62)	4 ($\bar{\rho}$ = 0.005)
	# est.: 20	23.89 (3.55)	10.41 (1.00)	3 ($\bar{\rho}$ = 0.046)	153.43 (43.11)	18.76 (3.73)	3 ($\bar{\rho}$ = 0.036)	156.17 (19.16)	16.42 (1.71)	8 ($\bar{\rho}$ = 0.005)
	# est.: 50	123.75 (51.10)	18.14 (4.47)	6 ($\bar{\rho}$ = 0.044)	243.24 (80.72)	24.50 (6.31)	14 ($\bar{\rho}$ = 0.029)	894.23 (100.35)	32.50 (6.50)	18 ($\bar{\rho}$ = 0.013)
	# est.: 100	389.06 (124.83)	20.25 (3.32)	12 ($\bar{\rho}$ = 0.044)	— (—)	— (—)	20 ($\bar{\rho}$ = 0.020)	— (—)	— (—)	20 ($\bar{\rho}$ = 0.010)
NN	(10,)	0.91 (0.00)	1.00 (0.00)	—	4.13 (0.13)	1.00 (0.00)	—	36.34 (5.22)	1.68 (0.19)	1 ($\bar{\rho}$ = 0.000)
	(10, 10, 10)	1.45 (0.03)	1.00 (0.00)	—	8.02 (0.86)	1.31 (0.18)	4 ($\bar{\rho}$ = 0.024)	328.70 (72.59)	2.38 (0.35)	4 ($\bar{\rho}$ = 0.012)
	(50,)	1.36 (0.03)	1.00 (0.00)	—	11.27 (1.10)	1.50 (0.15)	—	57.07 (7.28)	1.22 (0.10)	2 ($\bar{\rho}$ = 0.000)
	(100,)	2.26 (0.04)	1.00 (0.00)	—	26.82 (3.39)	1.30 (0.13)	—	111.85 (26.18)	1.44 (0.24)	11 ($\bar{\rho}$ = 0.001)

[a] Maximum depth of each decision tree equal to three.
[b] Maximum depth of each decision tree equal to two.

Figure 9. (Color online) Visual Analysis of the Computation Time Required to Solve the MP and AP



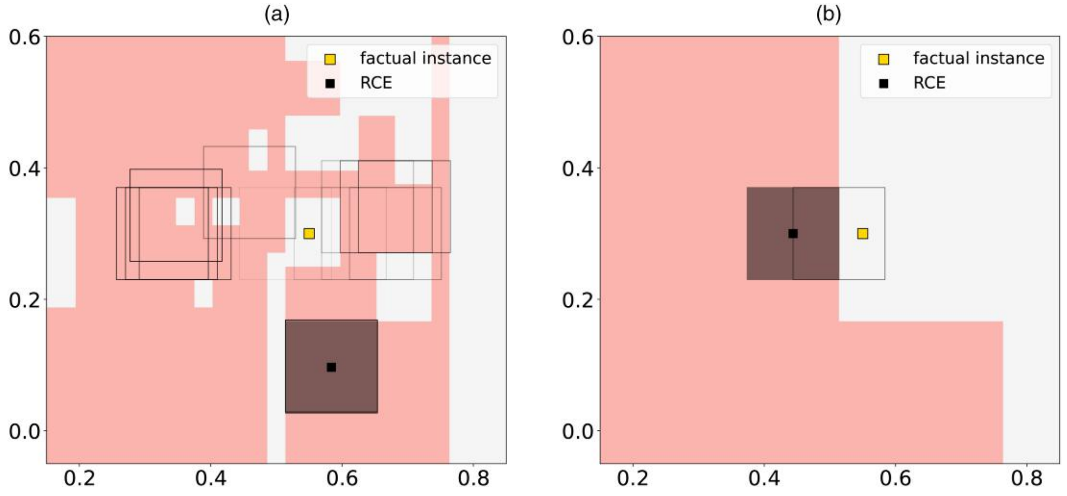
Notes. The trained model is a GBM with 100 estimators. In panel (a), we show the computation times for MP and AP summed over all iterations for each of 20 instances. In panel (b), we have visualized the computation times in each iteration of two selected instances. (a) Overall computation time for 20 instances. (b) Computation time per iteration for two instances.

for RF and GBM, and the depth of each layer in the NN. The results indicate that the computation time and the number of iterations increase primarily because of the complexity of the ML models rather than the number of features in the data sets. We further inspect the computation time by looking at the time needed to solve the MP and AP at each iteration. To do so, we use the Diabetes data set and train a GBM with 100 estimators. In Figure 9(a), we plot the overall computation time of the AP versus the overall computation time of the MP for 20 instances. In Figure 9(b), we inspect the time required per iteration for two of those instances. We can see that the time required to solve the AP remains small and stable, whereas the time required to solve the MP increases exponentially with each iteration (i.e., as more scenarios are added).

Decision trees tend to overfit as their depth increases. In Figure 10, we show iterations for a decision tree with maximum depths of 10 (Figure 10(a)) and 3 (Figure 10(b)). We observe that the more complex model substantially increases the number of iterations and hence, the computation time.

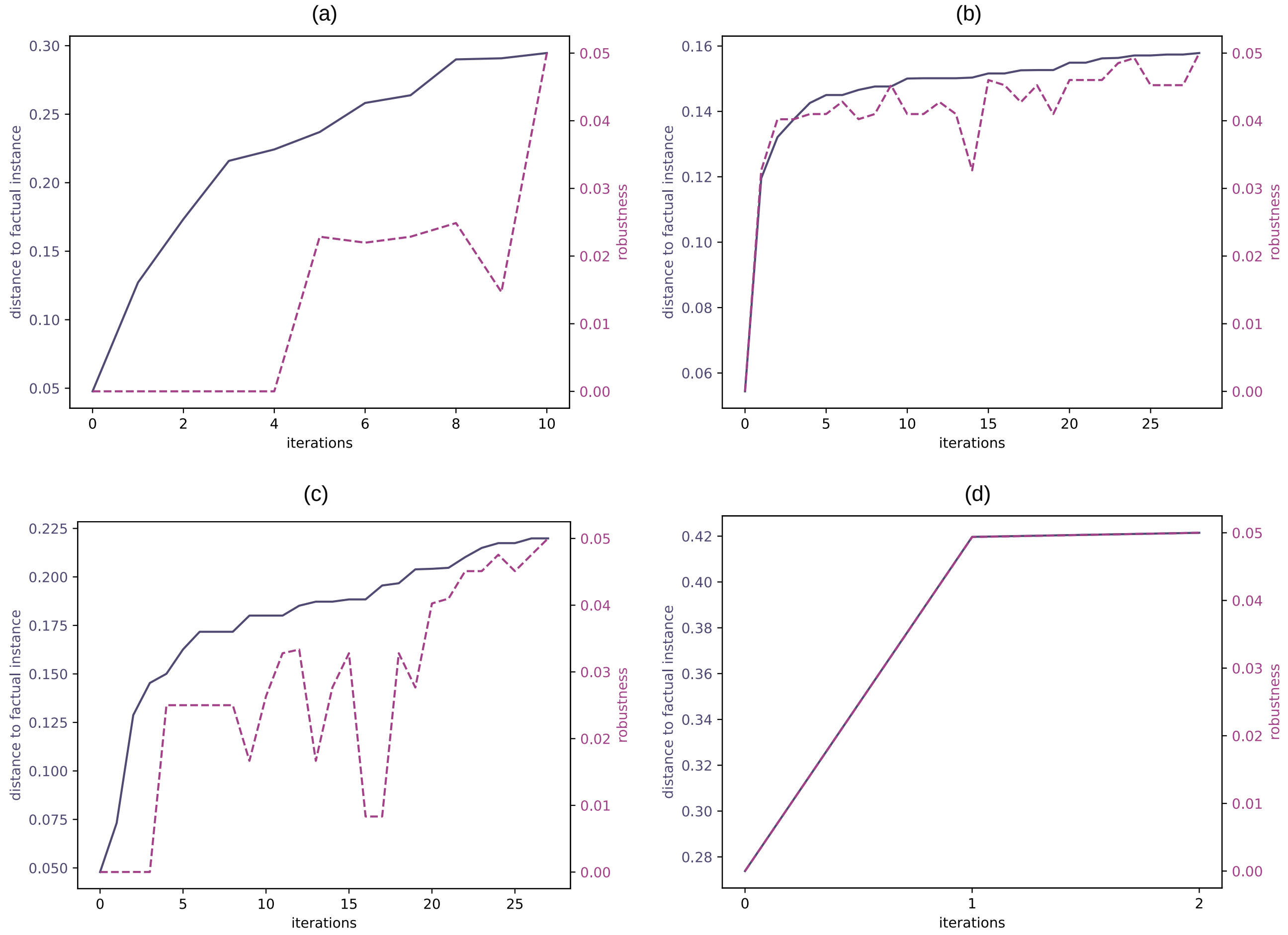
When the time limit is reached, we can still provide a list of solutions generated by solving the (MP) at each iteration. Each one of these solutions comes with information on the distance to the actual instance and the radius of

Figure 10. (Color online) Comparison of Decision Trees with Different Maximum Depths



Notes. The decision tree in panel (b) has a maximum depth of 3, whereas the one in panel (a) has a maximum depth of 10 and is overfitted. The deeper decision tree leads to a higher number of iterations needed to find a robust solution, as shown by the larger number of boxes/solutions generated before converging. (a) Decision tree with maximum depth = 10. (b) Decision tree with maximum depth = 3.

Figure 11. (Color online) Convergence Plots for (a) DT with Maximum Depth 10, (b) RF with 50 Trees, (c) GBM with 50 Trees, and (d) NN with One Hidden Layer Containing 100 Nodes



Notes. The models are trained on the Diabetes data set, and the uncertainty set is the ℓ_∞-norm. (a) Decision tree. (b) Random forest. (c) Gradient-boosting machines. (d) Neural network.

the uncertainty set for which the model predictions still belong to the desired class for all perturbations. Therefore, the decision makers still have the chance to select CEs from a region, albeit slightly smaller than intended. Overall, the algorithm converges relatively quickly, but our experiments show that it converges slower when using ℓ_2-norm as uncertainty set; see Table 2. Furthermore, as expected, for a smaller radius of $\rho = 0.01$, we reach the global optimum faster. Figure 11 shows the convergence behavior of various predictive models trained on the Diabetes data set and evaluated on a specific instance. Although the distance to the factual instance (solid lines) follows a monotonic increasing trend, the robustness (dashed lines) exhibits both peaks and troughs. This behavior can be attributed to the objective function of (MP), which seeks to minimize the distance between the CE and the factual instance. With each iteration of our algorithm, a new scenario/constraint is introduced to (MP), resulting in a worse objective value (higher) but not necessarily an improvement in robustness.

In the latter part of the experiment, we train a neural network with one hidden layer containing 50 nodes using the same three data sets. For the Ionosphere data set, we also trained another neural network with one hidden layer containing 10 nodes. To generate robust counterfactuals, we used the ℓ_2-norm uncertainty set and tested our algorithm against the one proposed by Dominguez-Olmedo et al. (2022) on 10 instances. In Table 3, we report the percentage of time that each algorithm was able to find a counterfactual that was at least ρ distant from the decision boundary (robustness) and the percentage of time that the counterfactual was valid (validity). We test ρ values of 0.1 and 0.2 for each data set. Our approach consistently generated counterfactual explanations that were optimal in terms of distance to the factual instance and valid. In the Diabetes data set, our algorithm reached the time limit without finding a counterfactual with robustness of at least 0.2 in 20% of cases. Nevertheless, the generated counterfactuals were still valid, with a ρ value smaller than 0.2. In contrast, the algorithm proposed by Dominguez-Olmedo et al. (2022) often

Table 3. Comparison Between Dominguez-Olmedo et al. (2022) and Our Algorithm Regarding the Percentage of Time That Each Algorithm Was Able to Find a Counterfactual at Least P Distant from the Decision Boundary and the Percentage of Time That the Counterfactual Explanations Were Valid and Therefore, Resulting in a Flip in the Prediction

	Dominguez-Olmedo et al. (2022)		Our algorithm	
ρ	Robustness, %	Validity, %	Robustness, %	Validity, %
Banknote NN(50)				
0.1	80	100	100	100
0.2	0	100	100	100
Diabetes NN(50)				
0.1	60	100	100	100
0.2	0	100	80	100
Ionosphere NN(10)				
0.1	70	90	100	100
0.2	40	90	100	100
Ionosphere NN(50)				
0.1	50	100	100	100
0.2	30	100	100	100

returned solutions that were not entirely robust with respect to ρ, particularly with the increase in ρ and the complexity of the predictive model. Even more concerning, their algorithm generated invalid solutions in 10% of the cases for the IONOSPHERE data set. Note that our experiments only cover neural network models because the method in Dominguez-Olmedo et al. (2022) cannot be used for nondifferentiable predictive models.

5. Discussion and Future Work

In this paper, we propose a robust optimization approach for generating regions of CEs for tree-based models and neural networks. We have also shown theoretically that our approach converges. This result has also been supported by empirical studies on different data sets. Our experiments demonstrate that our approach is able to generate explanations efficiently on a variety of data sets and ML models. Our results indicate that the proposed method scales well with the number of features and that the main computational challenge lies in solving the master problem as it becomes larger with every iteration. Overall, our results suggest that our proposed approach is a promising method for generating robust CEs for a variety of machine learning models.

As part of our future work, we plan to investigate methods for speeding up the calculations of the master problem. One approach involves exploring more efficient formulations of predictive models, such as the one proposed by Parmentier and Vidal (2021) for tree-based models. We also plan to focus on tightening the big-M formulations in the future. For instance, this can be achieved for tree-based models by solving a maximum violation problem per node.

We also aim to evaluate the user perception of the robust CEs generated by our approach and investigate how the choice of the uncertainty set affects the quality of the solution. Another research direction is the implementation of categorical and immutable features into the model. In the current version, categorical features are considered immutable, following user preferences. Adhering to user preferences by keeping some feature values constant, whether they are categorical or numerical, offers a practical strategy for generating sparse counterfactual explanations. This approach assumes that only a subset of the features (the ones that are not considered immutable) is susceptible to perturbations. Considering mutable categorical features would require a reformulation of the uncertainty set to account for the different feature types. This would lead to nonconvex, discrete, and lower-dimensional uncertainty sets. However, this change would only affect the adversarial problem, whereas the master problem remains the same.

Endnotes

[1] See https://github.com/donato-maragno/robust-CE.

[2] See Online Appendix B for the well-known dual approach applied to linear models.

[3] See https://github.com/donato-maragno/robust-CE.

References

Anderson R, Huchette J, Ma W, Tjandraatmadja C, Vielma JP (2020) Strong mixed-integer programming formulations for trained neural networks. *Math. Programming* 183(1–2):3–39.

Artelt A, Vaquet V, Velioglu R, Hinder F, Brinkrolf J, Schilling M, Hammer B (2021) Evaluating robustness of counterfactual explanations. *2021 IEEE Sympos. Ser. Comput. Intelligence (SSCI)* (IEEE, Piscataway, NJ), 01–09.

Ben-Tal A, El Ghaoui L, Nemirovski A (2009) *Robust Optimization* (Princeton University Press, Princeton, NJ).

Bertsimas D, Dunning I, Lubin M (2016) Reformulation vs. cutting-planes for robust optimization: A computational study. *Comput. Management Sci.* 13:195–217.

Bienstock D, Özbay N (2008) Computing robust basestock levels. *Discrete Optim.* 5(2):389–414.

Black E, Wang Z, Fredrikson M, Datta A (2021) Consistent counterfactuals for deep models. Preprint, submitted October 6, https://arxiv.org/abs/2110.03109.

Bui N, Nguyen D, Nguyen VA (2022) Counterfactual plans under distributional ambiguity. Preprint, submitted April 10, https://arxiv.org/abs/2201.12487.

Carrizosa E, Ramírez-Ayerbe J, Morales DR (2021) Generating collective counterfactual explanations in score-based classification via mathematical optimization. *Expert Systems Appl.* 238(D):121954.

Cplex II (2009) V12. 1: User's manual for CPLEX. *International Business Machines Corporation* 46(53):157. Accessed February 12, 2024, https://public.dhe.ibm.com/software/websphere/ilog/docs/optimization/cplex/ps_usrmancplex.pdf.

Dandl S, Molnar C, Binder M, Bischl B (2020) Multi-objective counterfactual explanations. Bäck T, Preuss M, Deutz A, Wang H, Doerr C, Emmerich M, Trautmann H, eds. *Parallel Problem Solving from Nature—PPSN XVI* (Springer International Publishing, Cham, Switzerland), 448–469.

Dominguez-Olmedo R, Karimi AH, Schölkopf B (2022) On the adversarial robustness of causal algorithmic recourse. Chaudhuri K, Jegelka S, Song L, Szepesvari C, Niu G, Sabato S, eds. *Proc. 39th Internat. Conf. Machine Learn.*, Proceedings of Machine Learning Research (PMLR, New York), 5324–5342.

Dua D, Graff C (2017) UCI machine learning repository. Accessed February 14, 2024, http://archive.ics.uci.edu.

Dutta S, Long J, Mishra S, Tilli C, Magazzeni D (2022) Robust counterfactual explanations for tree-based ensembles. Chaudhuri K, Jegelka S, Song L, Szepesvari C, Niu G, Sabato S, eds. *Proc. 39th Internat. Conf. Machine Learn.*, vol. 162 (PMLR, New York), 5742–5756.

Ferrario A, Loi M (2022) The robustness of counterfactual explanations over time. *IEEE Access* 10:82736–82750.

Fischetti M, Jo J (2018) Deep neural networks and mixed integer linear optimization. *Constraints* 23(3):296–309.

Forel A, Parmentier A, Vidal T (2022) Robust counterfactual explanations for random forests. Preprint, submitted, May 27, https://arxiv.org/abs/2205.14116.

Grimstad B, Andersson H (2019) ReLU networks as surrogate models in mixed-integer linear programs. *Comput. Chemical Engrg.* 131:106580.

Gurobi Optimization LLC (2022) Gurobi optimizer reference manual. Accessed February 14, 2024, https://www.gurobi.com.

Kanamori K, Takagi T, Kobayashi K, Ike Y, Uemura K, Arimura H (2021) Ordered counterfactual explanation by mixed-integer linear optimization. *Proc. Conf. AAAI Artificial Intelligence* 35(13):11564–11574.

Karimi AH, Barthe G, Balle B, Valera I (2020) Model-agnostic counterfactual explanations for consequential decisions. Chiappa S, Calandra R, eds. *Proc. Twenty Third Internat. Conf. Artificial Intelligence Statist.*, vol. 108 (PMLR), 895–905.

Mahajan D, Tan C, Sharma A (2019) Preserving causal constraints in counterfactual explanations for machine learning classifiers. Preprint, submitted December 6, https://arxiv.org/abs/1912.03277.

Maragno D, Röber TE, Birbil I (2022) Counterfactual explanations using optimization with constraint learning. Preprint December 14, https://arxiv.org/abs/2209.10997.

Maragno D, Wiberg H, Bertsimas D, Birbil SI, den Hertog D, Fajemisin A (2023) Mixed-integer optimization with constraint learning. *Oper. Res.*, epub ahead of print December 1, https://doi.org/10.1287/opre.2021.0707.

Mothilal RK, Sharma A, Tan C (2020) Explaining machine learning classifiers through diverse counterfactual explanations. *Proc. 2020 Conf. Fairness Accountability Transparency* (Association for Computing Machinery, New York), 607–617.

Mutapcic A, Boyd S (2009) Cutting-set methods for robust convex optimization with pessimizing oracles. *Optim. Methods Software* 24(3):381–406.

Parmentier A, Vidal T (2021) Optimal counterfactual explanations in tree ensembles. Meila M, Zhang T, eds. *Proc. 38th Internat. Conf. Machine Learn.*, vol. 139 (PMLR, New York), 8422–8431.

Pawelczyk M, Datta T, van-den Heuvel J, Kasneci G, Lakkaraju H (2022) Probabilistically robust recourse: Navigating the trade-offs between costs and robustness in algorithmic recourse. Preprint, submitted March 13, https://arxiv.org/abs/2203.06768.

Rawal K, Kamar E, Lakkaraju H (2020) Algorithmic recourse in the wild: Understanding the impact of data and model shifts. Preprint, submitted December 22, https://arxiv.org/abs/2012.11788.

Russell C (2019) Efficient search for diverse coherent explanations. *Proc. Conf. Fairness Accountability Transparency* (Association for Computing Machinery, New York), 20–28.

Slack D, Hilgard A, Lakkaraju H, Singh S (2021) Counterfactual explanations can be manipulated. Ranzato M, Beygelzimer A, Dauphin Y, Liang P, Vaughan JW, eds. *Advances in Neural Information Processing Systems*, vol. 34 (Curran Associates, Inc., Red Hook, NY), 62–75.

Upadhyay S, Joshi S, Lakkaraju H (2021) Toward robust and reliable algorithmic recourse. Ranzato M, Beygelzimer A, Dauphin Y, Liang P, Vaughan JW, eds. *Advances in Neural Information Processing Systems*, vol. 34 (Curran Associates, Inc., Red Hook, NY), 16926–16937.

Ustun B, Spangher A, Liu Y (2019) Actionable recourse in linear classification. *Proc. Conf. Fairness Accountability Transparency* (Association for Computing Machinery, New York), 10–19.

Virgolin M, Fracaros S (2023) On the robustness of sparse counterfactual explanations to adverse perturbations. *Artificial Intelligence* 316:103840.

Wachter S, Mittelstadt B, Russell C (2018) Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harvard J. Law Tech.* 31(2):841–887.