





# Exploring Counterfactual Explanations for Classification and Regression Trees

Suryabhan Singh Hada<sup>(✉)</sup>  and Miguel Á. Carreira-Perpiñán 

Department of Computer Science and Engineering,  
University of California, Merced, CA 95343, USA  
{shada,mcarreira-perpnan}@ucmerced.edu

**Abstract.** The problem of counterfactual explanations is that of minimally adjusting attributes in a source input instance so that it is classified as a target class under a given classifier. They answer practical questions of the type “what should my annual income be for my loan to be approved?”, for example. We focus on classification and regression trees, both axis-aligned and oblique (having hyperplane splits), and formulate the counterfactual explanation as an optimization problem. Although this problem is nonconvex and nondifferentiable, an exact solution can be computed very efficiently, even with high-dimensional feature vectors and with both continuous and categorical features. We also show how the counterfactual explanation formulation can answer a range of important practical questions, providing a way to query a trained tree and suggest possible actions to overturn its decision, and demonstrate it in several case studies. The results are particularly relevant for finance, medicine or legal applications, where interpretability and counterfactual explanations are particularly important.

**Keywords:** Interpretability · Counterfactual explanations · Decision trees

## 1 Introduction

In the last decade, deep learning and machine learning models have become widespread in many practical applications. This has been beneficial as these models provide intelligent, automated processing of tasks that up to now were hard for machines. However, at the same time, concerns related to the ethics and safety of these models have arisen as well. One is the problem of interpretability, i.e., explaining how these models function. This is an old problem, which has been studied (possibly under different names, such as explainable AI) since decades ago in statistics and machine learning (e.g. [2, 11, 14, 21]). A second problem is explaining why the model made a decision or how to change it [28]. This problem is more recent and has become more pressing due to concerns over the opaqueness of current AI systems. That said, related problems have been studied in data mining or knowledge discovery from databases [1, 9, 23, 30], in particular in applications such as customer relationship management (CRM).

Much of the recent works focus on a specific version of the second problem that focuses on changing a classifier’s decision in a prescribed way [28]. This problem is called *counterfactual explanation* [28]. Formally, a counterfactual explanation seeks a minimal change to a given input’s features that will change the classifier’s prediction in the desired way. These explanations are very helpful in understanding the behavior of a model for a given instance, as they can answer questions like what input feature the model focuses on to make a certain prediction. Mathematically, the problem of counterfactual explanations can be formulated as an optimization problem. The objective is to minimize the change in input features subject to classified as user-defined class. This similar formulation has been used in deep nets for adversarial examples and model inversion [10, 13, 16, 17, 22, 25, 26, 31]. There are also formulations that are specific for linear models [6, 24, 27]. Most algorithms to solve the optimization assume differentiability of the classifier with respect to its input instance so that gradient-based optimization can be applied. However, none of these algorithms apply to decision trees, which define nondifferentiable classifiers.

Decision trees have long been widely used in applications and are regularly highly ranked in user surveys such as *KDnuggets.com* or data mining and machine learning reviews such as [29]. This is particularly owing to their ease of interpretability compared to more accurate classifiers such as neural nets or decision forests. However, the prediction accuracy of decision trees learned using the recently introduced *Tree Alternating Optimization (TAO)* algorithm [3, 7, 38] is much higher than using traditional algorithms such as CART or C5.0. This applies both to the traditional, axis-aligned trees, but also to the far more accurate oblique trees (having hyperplane splits) and sparse oblique trees (having hyperplane splits with few nonzero weights), as well as to trees of more complex forms [34, 37], and to forests using bagging, boosting or other ensembling mechanisms, but where each tree is trained with TAO [8, 12, 33, 35, 36]. Finally, the stronger predictive power of sparse oblique decision trees together with their interpretability and fast inference makes them useful for other uses, such as in understanding deep neural networks [18, 19] or compressing deep neural networks [20].

For these reasons, solving counterfactual explanations for decision trees is important in practice. Our recent work from [4, 5] shows that the counterfactual problem can be solved exactly and efficiently in a decision tree. We showed that the counterfactual problem in decision trees is equivalent to solving the counterfactual problem in each leaf region and picking the best among them (see Sect. 2). This exact formulation for solving the counterfactual problem in decision trees allows us to extend the problem, which can answer some very interesting questions. For instance, what is the closest class to change from the original class, what is the closest boundary if changing the only feature, which feature has the lowest cost to change the class to a target class, and more. We describe these extended problems in detail in Sect. 3. These problems are very difficult and probably very expensive to answer in other models like a neural

network. However, we show that in decision trees using the [4, 5] framework, we can solve these extended problems exactly and efficiently.

Next, we first briefly describe our formulation of the counterfactual problem and how to solve it exactly (Sect. 2); this follows closely the main results of [4, 5]. Then we describe each extended problem in detail (Sect. 3). Finally, in Sect. 4 we discuss these extended problems along with real-life use cases.

## 2 Solving Counterfactual Exactly in Decision Trees

### 2.1 Leaf Region: Definition

Assume we are given a classification tree that can map an input instance  $\mathbf{x} \in \mathbb{R}^D$ , with  $D$  real features (attributes), to a class in  $\{1, \dots, K\}$ . Assume the tree is rooted, directed and binary (where each decision node has two children) with decision nodes and leaves indexed in the sets  $\mathcal{D}$  and  $\mathcal{L}$ , respectively, and  $\mathcal{N} = \mathcal{D} \cup \mathcal{L}$ . We index the root as  $1 \in \mathcal{D}$ . For example, in Fig. 1 we have  $\mathcal{N} = \{1, \dots, 15\}$ ,  $\mathcal{L} = \{5, 8, 9, 11, \dots, 15\}$  and  $\mathcal{D} = \mathcal{N} \setminus \mathcal{L}$ . In oblique decision trees each decision node  $i \in \mathcal{D}$  has a real-valued decision function  $f_i(\mathbf{x})$  defined by a hyperplane (linear combination of all the features)  $f_i(\mathbf{x}) = \mathbf{w}_i^T \mathbf{x} + b_i$ , with fixed weight vector  $\mathbf{w}_i \in \mathbb{R}^D$  and bias  $b_i \in \mathbb{R}$ . For axis-aligned trees,  $\mathbf{w}_i$  is an indicator vector (having one element equal to 1 and the rest equal to 0). The decision function  $f_i(\mathbf{x})$  send down an input instance  $\mathbf{x} \in \mathbb{R}^D$  to  $i$ 's right child if  $f_i(\mathbf{x}) \geq 0$  and down  $i$ 's left child otherwise. Each leaf  $i \in \mathcal{L}$  is labeled with one class label  $y_i \in \{1, \dots, K\}$ . For an input instance  $\mathbf{x}$  the tree predicts its label by sending down  $\mathbf{x}$  via the decision nodes, to exactly one leaf and outputting its label. The parameters  $\{\mathbf{w}_i, b_i\}_{i \in \mathcal{D}}$  and  $\{y_i\}_{i \in \mathcal{L}}$  are estimated by TAO [3, 7] (or another algorithm) when learning the tree from a labeled training set.

The tree partitions the input space into  $|\mathcal{L}|$  regions, one per leaf, as shown in Fig. 1 (bottom panel). Each region is an axis-aligned box in case of axis-aligned trees and a polytope for oblique trees. This region is defined by the intersection of the hyperplanes found in the path from the root to the leaf. Specifically, define a linear constraint  $z_i(\mathbf{w}_i^T \mathbf{x} + b_i) \geq 0$  for decision node  $i$  where  $z_i = +1$  if going down its right child and  $z_i = -1$  if going down its left child. Then we define the constraint vector for leaf  $i \in \mathcal{L}$  as  $\mathbf{h}_i(\mathbf{x}) = (z_j(\mathbf{w}_j^T \mathbf{x} + b_j))_{j \in \mathcal{P}_i \setminus \{i\}}$ , where  $\mathcal{P}_i = \{1, \dots, i\}$  is the path of nodes from the root (node 1) to leaf  $i$ . We call  $\mathcal{F}_i = \{\mathbf{x} \in \mathbb{R}^D: \mathbf{h}_i(\mathbf{x}) \geq 0\}$  the corresponding feasible set, i.e., the region in input space of leaf  $i$ . For example, in Fig. 1 (top) the path from the root to leaf 14 is  $\mathcal{P}_{14} = \{1, 3, 6, 10, 14\}$  and its region is given by:

$$\mathbf{h}_{14}(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) \\ -f_3(\mathbf{x}) \\ -f_6(\mathbf{x}) \\ -f_{10}(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} \mathbf{w}_1^T \mathbf{x} + b_1 \\ -\mathbf{w}_3^T \mathbf{x} - b_3 \\ -\mathbf{w}_6^T \mathbf{x} - b_6 \\ -\mathbf{w}_{10}^T \mathbf{x} - b_{10} \end{pmatrix} \geq \mathbf{0}.$$

## 2.2 Counterfactual Problem in Decision Trees

In this section we briefly describe the counterfactual problem in a decision tree. Assume we are given a *source input instance*  $\bar{\mathbf{x}} \in \mathbb{R}^D$  which is classified by the tree as class  $\bar{y}$ , i.e.,  $T(\bar{\mathbf{x}}) = \bar{y}$ , and we want to find the closest instance  $\mathbf{x}^*$  that would be classified as another class  $y \neq \bar{y}$  (the *target class*). We define the *counterfactual explanation* for  $\bar{\mathbf{x}}$  as the (or a) minimizer  $\mathbf{x}^*$  of the following problem:

$$\min_{\mathbf{x} \in \mathbb{R}^D} E(\mathbf{x}; \bar{\mathbf{x}}) \quad \text{s.t.} \quad T(\mathbf{x}) = y, \quad \mathbf{c}(\mathbf{x}) = \mathbf{0}, \quad \mathbf{d}(\mathbf{x}) \geq \mathbf{0} \quad (1)$$

where  $E(\mathbf{x}; \bar{\mathbf{x}})$  is a cost of changing attributes of  $\bar{\mathbf{x}}$ , and  $\mathbf{c}(\mathbf{x})$  and  $\mathbf{d}(\mathbf{x})$  are equality and inequality constraints (in vector form). The fundamental idea is that problem (1) seeks an instance  $\mathbf{x}$  that is as close as possible to  $\bar{\mathbf{x}}$  while being classified as class  $y$  by the tree and satisfying the constraints  $\mathbf{c}(\mathbf{x})$  and  $\mathbf{d}(\mathbf{x})$ .

The constraint  $T(\mathbf{x}) = y$  makes the problem severely nonconvex, nonlinear and nondifferentiable because of the tree function  $T(\mathbf{x})$ . However as described in [4, 5] this problem can be solved exactly and efficiently. In [4] we show that problem (1) is equivalent to:

$$\min_{i \in \mathcal{L}} \min_{\mathbf{x} \in \mathbb{R}^D} E(\mathbf{x}; \bar{\mathbf{x}}) \quad \text{s.t.} \quad y_i = y, \quad \mathbf{h}_i(\mathbf{x}) \geq \mathbf{0}, \quad \mathbf{c}(\mathbf{x}) = \mathbf{0}, \quad \mathbf{d}(\mathbf{x}) \geq \mathbf{0}. \quad (2)$$

*In English, what this means is that solving problem (1) over the entire space can be done by solving it within each leaf's region (defined by  $\mathbf{h}_i$ , Sect. 2.1) and then picking the leaf with the best solution.* This is shown in Fig. 1 (bottom panel). That is, the problem has the form of a mixed-integer optimization where the integer part is done by enumeration (over the leaves ( $\mathcal{L}$ )) and the continuous part (within each leaf) by other means to be described later.

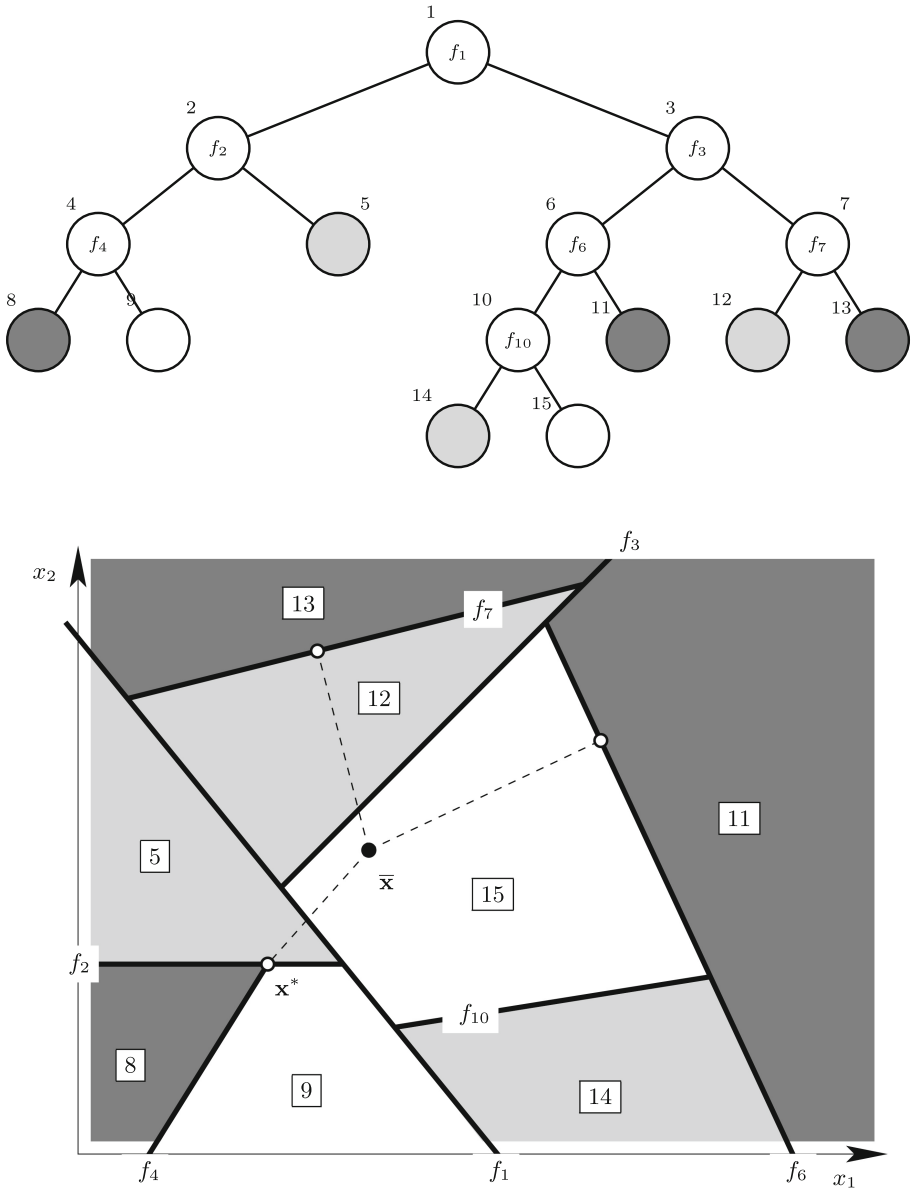
Hence, the problem we still need to solve is the problem over a single leaf  $i \in \mathcal{L}$  (having the desired label  $y_i = y$ ), and henceforth we focus on this. We write it as:

$$\min_{\mathbf{x} \in \mathbb{R}^D} E(\mathbf{x}; \bar{\mathbf{x}}) \quad \text{s.t.} \quad \mathbf{h}_i(\mathbf{x}) \geq \mathbf{0}, \quad \mathbf{c}(\mathbf{x}) = \mathbf{0}, \quad \mathbf{d}(\mathbf{x}) \geq \mathbf{0}. \quad (3)$$

here,  $\mathbf{h}_i(\mathbf{x})$  is the set of hyperplanes that represents decision rule of the nodes in the path from root to leaf  $i$  (Sect. 2.1). If the function  $E(\mathbf{x}; \cdot)$  is convex over  $\mathbf{x}$  and the constraints  $\mathbf{c}(\mathbf{x})$  and  $\mathbf{d}(\mathbf{x})$  are linear, then this problem is convex (since for oblique trees  $\mathbf{h}_i(\mathbf{x})$  is linear). In particular, if  $E$  is quadratic then the problem is convex quadratic program (QP), which can be solved very efficiently with existing solvers. See [4, 5] for more details.

## 2.3 Separable Problems: A Special Case for Axis-Aligned Trees

Our earlier work [4, 5] provides following result, which vastly simplifies the problem for axis-aligned trees.



**Fig. 1.** *Top:* an oblique classification tree with  $K = 3$  classes (colored white, light gray and gray) from [4]. A decision node  $i$  sends an input instance  $\mathbf{x}$  to its right child if  $f_i(\mathbf{x}) \geq 0$  and to its left child otherwise. The decision function  $f_i(\mathbf{x}) = \mathbf{w}_i^T \mathbf{x} + b_i$ , where  $\mathbf{w}$  and  $b$  are weights and the bias respectively. *Bottom* the space of the input instances  $\mathbf{x} \in \mathbb{R}^2$ , assumed two-dimensional, partitioned according to each leaf's region in polytopes (the region boundaries are labeled with the corresponding decision node function). The source instance  $\bar{\mathbf{x}}$  is in the white class and the counterfactual one (using the  $\ell_2$  distance) subject to being in the gray class is  $\mathbf{x}^*$ .

**Theorem 1.** *In problem (1), assume that each constraint depends on a single element of  $\mathbf{x}$  (not necessarily the same) and that the objective function is separable, i.e.,  $E(\mathbf{x}; \bar{\mathbf{x}}) = \sum_{d=1}^D E_d(x_d; \bar{x}_d)$ . Then the problem separates over the variables  $x_1, \dots, x_D$ .*

This means that, within each leaf, we can solve for each  $x_d$  independently, by minimizing  $E_d(x_d; \bar{x}_d)$  subject to the constraints on  $x_d$ . Further, the solution is given by the following result.

**Theorem 2.** *Consider the scalar constrained optimization problem, where the bounds can take the values  $l_d = -\infty$  and  $u_d = \infty$ :*

$$\min_{x_d \in \mathbb{R}} E_d(x_d; \bar{x}_d) \quad \text{s.t.} \quad l_d \leq x_d \leq u_d. \quad (4)$$

Assume  $E_d$  is convex on  $x_d$  and satisfies  $E_d(\bar{x}_d; \bar{x}_d) = 0$  and  $E_d(x_d; \bar{x}_d) \geq 0 \forall x_d \in \mathbb{R}$ . Then  $x_d^*$ , defined as the median of  $\bar{x}_d, l_d$  and  $u_d$ , is a global minimizer of the problem:

$$x_d^* = \text{median}(\bar{x}_d, l_d, u_d) = \begin{cases} l_d, & \bar{x}_d < l_d \\ u_d, & \bar{x}_d > u_d \\ \bar{x}_d, & \text{otherwise} \end{cases}. \quad (5)$$

We can apply these theorems to axis-aligned trees (assuming each of the extra constraints  $\mathbf{c}(\mathbf{x})$  and  $\mathbf{d}(\mathbf{x})$  depends individually on a single feature), because each of the constraints  $\mathbf{h}_i(\mathbf{x}) \geq \mathbf{0}$  in the path from the root to leaf  $i$  involve a single feature of  $\mathbf{x}$ . Within each leaf  $i \in \mathcal{L}$  we can represent the constraints (which represents the path from the root to  $i$ ) as bounding box  $\mathbf{l}_i \leq \mathbf{x} \leq \mathbf{u}_i$ , and solve elementwise by applying the median formula described above. After solving the counterfactual problem in each leaf, we return the result of the best leaf. This makes solving the counterfactual explanation problem exceedingly fast for axis-aligned trees.

## 2.4 Categorical Variables

Although many popular benchmarks in machine learning use only continuous variables, in practice, most of the datasets contain categorical variables. This is true especially in legal, financial, or medical applications, for instance, use cases in Sect. 4.

In this work we handle categorical variables as described in [4, 5]. That is we encode the categorical variables as one-hot. This means, if an original categorical variable can take  $C$  different categories, we encode it using  $C$  dummy binary variables jointly constrained so that exactly one of them is 1 (for the corresponding category):  $x_1, \dots, x_C \in \{0, 1\}$  s.t.  $\mathbf{1}^T \mathbf{x} = 1$ .

Since we only need to read the values of dummy variables during training, we treat them as if they were continuous and without the above constraints. However, when solving the counterfactual problem, we modify those variables,

so we need to respect the above constraints. This makes the problem a mixed-integer optimization, where some variables are continuous and others binary (the dummy variables). While these problems are NP-hard in general, in many practical cases, we can expect to solve them exactly and quickly using modern mixed-integer optimization solvers, such as CPLEX or Gurobi [15].

### 3 Exploring Different Types of Counterfactual Explanation Questions

The counterfactual problem (2) accommodates a variety of useful, practical questions about the source instance ( $\bar{\mathbf{x}}$ ). We list each below and explain how that can be solved exactly.

1. *Finding the closest boundary.* The minimum-distance change to  $\bar{\mathbf{x}}$  that changes its original class  $k$ . Solve the problem (3) in every leaf except the ones with label  $k$ , and pick the solution with the lowest cost.
2. *Critical attribute for change to the target class  $y$ .* Which attribute has the lowest cost to change the class of  $\bar{\mathbf{x}}$  to a target class  $y$ , if changing only one attribute? For given a attribute  $d$ , we add all other attributes to the equality constraint ( $\mathbf{c}(\mathbf{x}) = \mathbf{0}$ ) and solve the counterfactual problem (2). We repeat this process for each attribute in  $\bar{\mathbf{x}}$  and pick the attribute for which the counterfactual ( $\mathbf{x}^*$ ) has the lowest cost.
3. *Critical attribute for changing the class.* Which attribute has the lowest cost to change the class of  $\bar{\mathbf{x}}$  to any other class if changing only one attribute? For each attribute, we solve the *finding the closest boundary* problem, where other attributes are in the equality constraint; and pick the attribute for which the counterfactual ( $\mathbf{x}^*$ ) have the lowest cost.
4. *Robust counterfactuals.* Here, we want to find the counterfactuals that are well inside a leaf region rather than on the boundary, so they are more robust to flipping their class due to small changes. This problem can easily be solved by shrinking the leaf region size in problem 3. That is in problem 3, the constraint " $\mathbf{h}_i(\mathbf{x}) \geq \mathbf{0}$ " becomes " $\mathbf{h}_i(\mathbf{x}) \geq \epsilon$ ", where  $\epsilon > 0$ .

We can also use the counterfactual problem (2) to explore more practical problems that are related to the regression trees. Consider a regression tree  $T$ , where  $T(\bar{\mathbf{x}})$  and  $T(\mathbf{x}^*)$  represent the predicted value of the source instance ( $\bar{\mathbf{x}}$ ) and the counterfactual ( $\mathbf{x}^*$ ). Similar, to above we list each problem below and explain how that problem can be solved exactly.

1.  $T(\mathbf{x}^*) > T(\bar{\mathbf{x}})$ : find the minimum change in  $\bar{\mathbf{x}}$  that increase its predicted value. For this we only consider the leaves whose label is larger than the  $T(\bar{\mathbf{x}})$ , and solve problem (3) in each of them and pick the  $\mathbf{x}^*$  with the lowest cost.
2.  $T(\mathbf{x}^*) \geq T(\bar{\mathbf{x}}) + \beta$  find the minimum change in  $\bar{\mathbf{x}}$  that increase its predicted value atleast by  $\beta$ . Same as above, the only difference is the leaves we consider have the label greater than  $T(\bar{\mathbf{x}}) + \beta$ .

3.  $\alpha \geq T(\mathbf{x}^*) \geq \beta$  find the minimum change in  $\bar{\mathbf{x}}$  that change its predicted value between  $\alpha$  and  $\beta$ . It is again same as the previous two, but this time we only consider the leaves with label between  $\alpha$  and  $\beta$ .

All these extended problems can be applied to any type of decision tree with hard thresholds, but here we only focus on oblique trees.

## 4 Experiments

Since we show all extension problems (Sect. 3) can be solved exactly by converting them into the problem (2), we do not need to assess the optimality and validity of the generated counterfactual examples experimentally. Instead, we apply each problem to a real-life dataset [32] and explain their usage with three use case studies.

- Our first use case deals with the students’ grades in secondary education of two Portuguese schools. For each student, we have social, demographic, and school-related attributes and their final grades. In this study, we focus on the students who have failing grades, and we suggest the changes that can improve students’ grades by applying our extension problems.
- Next, we focus on the loan applicants as credit risk from the German credit dataset. We focus on the applicants that are a bad credit risk. Our goal here is not only to generate counterfactuals that can classify these candidates as good credit risk but also to suggest more changes to have a stronger credit profile. We do this by generating counterfactuals that are more robust to small changes.
- Our last use case study focuses on the median value of owner-occupied homes in multiple suburbs of Boston. We treat this as a regression problem whose goal is to predict the value of the home. Then we apply our formulations to explore what factors affect the value of a home at different price points.

In all the case studies, our model is an oblique decision tree that we train using a recently proposed algorithm called tree alternating optimization (TAO) [3, 7]. To generate counterfactuals we use  $\ell_2$  distance as our cost function ( $E$ ). Also, these datasets contain categorical variables, so to generate counterfactuals, we use Gurobi [15] to solve the mixed-integer optimization problem.

Next, we describe each dataset in brief and then discuss each use case study in detail.

### 4.1 Dataset Information

All our datasets are from UCI [32], and they are described in the same order as the user case study. Since none of the datasets contains a separate test set, we randomly divide the instances into training (80%) and test (20%) sets.



**Student Portuguese Grades.** Each instance have three grades, but we use only the *final grades*, and remove the other two from the attributes. The *final grades* are in the range of 1 to 20. So, we divide these grades into five categories. These categories are as follows:

- If the grades are greater or equal to 16, then *excellent*.
- If grades are either 15 or 14, then *good*.
- If grades are either 14 or 13, then *satisfactory*.
- If grades are in range between 12 and 10, then *sufficient*.
- If grades are 9 or lower, then *fail*.

The prediction task is to determine the *final grades* of a student. There are 649 instances, and each has 30 attributes (after removing the two grades). Out of 30 attributes, 11 are integer, and the rest are categorical. We convert each categorical attribute into a one-hot encoding attribute. Thus each instance of the dataset has 64 attributes.

**German Credit.** The prediction task is to determine whether an applicant is considered a *good* or a *bad* credit risk for 1 000 loan applicants. There are 20 attributes, out of which 7 are integer, and the rest are categorical. Similarly to the above dataset, we convert each categorical attribute to a one-hot encoding attribute. Thus each instance of the dataset has 61 attributes.

**Boston Housing Dataset.** We use this dataset for regression. The task is to predict the *median price* of owner-occupied homes. There are 506 instances, and each instance describes a Boston suburb or town. For each instance, there is 13 attribute all continuous except one which is binary.

## 4.2 Use Case Study 1

Our oblique tree for this classification task achieves 56.92% test error and 33.14% train error. The tree has a depth of 9 and 34 leaves.

First consider a student whose attributes are described in the second column of Table 1. The current *final grades* of the student is *fail*. If create a counterfactual with target class as *excellent*, the following attributes change (third column in Table 1):

- reduce previous class failures from 2 to 1.
- mother’s job should be changed from services to teacher.
- father’s job should be changed from services to teacher.
- change higher education plan from no to yes.

However, it is hard to jump from failing grades to excellent grades in real life. Also, it requires changing many attributes, which might not be feasible like mother’s job and father’s job. Instead, the student can try making small changes, which are enough for passing the class. For this, we try to find the closest other class boundary problem here since any other class will lead to passing the class. The closest class we find is *sufficient* and requires the following change (fourth column in Table 1).

- reduce previous class failures from 2 to 1.

This is also expected the closest class to change is *sufficient* as this lowest thing to be considered as passing the class. Moreover, it does not suggest big changes like changing parent’s jobs, but only change **previous class failures** which easier to achieve.

Next, we consider another student (fifth column in Table 1). This student also failed the class. If we upgrade the grades directly to *satisfactory* allowing only attribute to change, then the attributes that will change the class with the lowest cost will be (sixth column in Table 1):

- reduce **free time** from high(4) to very low(1).

This also makes sense as reducing **free time** after school can be devoted to studies. That said, it is still a large change. So, we search for the closest different class allowing only one attribute to change (last column in Table 1). The closet class would be *sufficient*, and the attribute with the lowest cost is:

- reduce **going out frequency** from medium(3) to low(2).

This makes sense as, again, time spent on going out can be used for studies. Also, the numeric cost to change to lower grade (*sufficient*) is 3 times lower than the upper grade (*satisfactory*).

### 4.3 Use Case Study 2

The predictive task in the study is to predict how much a loan applicant is of credit risk. Our oblique tree achieves a test error of 18.5% and a train error of 17.5% for this dataset. The depth of the tree is 8.

The goal here is to generate diverse counterfactuals, where each subsequent counterfactual is more robust than the previous one. When generating a counterfactual, the algorithm generates a counterfactual in each target leaf that is closest to the input instance. If all the attributes are continuous, then this generated counterfactual will always exist on the boundary of the leaf, as shown in the Fig. 1. However, if we shrunk these leaves’ regions, we can force the algorithm to generate counterfactuals inside the leaf region (generating robust counterfactuals). As described in Sect. 3, for a leaf  $i$  this can be done by adding a small positive constant ( $\epsilon > 0$ ) in Eq. (3).

$$\min_{\mathbf{x} \in \mathbb{R}^D} E(\mathbf{x}; \bar{\mathbf{x}}) \quad \text{s.t.} \quad \mathbf{h}_i(\mathbf{x}) \geq \epsilon, \quad \mathbf{c}(\mathbf{x}) = \mathbf{0}, \quad \mathbf{d}(\mathbf{x}) \geq \mathbf{0}. \quad (6)$$

Consider an example described in the Table 2. This instance is classified as *bad* creditor. Next, we generate a counterfactual for it. The generated counterfactual will able to change its label from *bad* to *good* creditor. However, it is either on the boundary of the target leaf region or very close to it (due to categorical variables). This means the changes suggested by the counterfactual ( $\mathbf{x}_1^*$ ) will only suggest changes that are the bare minimum for a good creditor. However, if we force the algorithm to generate counterfactuals deep in the leaf region, it will suggest the changes that make the counterfactual much better creditor.

**Table 1.** Example illustrating the construction of counterfactual instances with our extended formulations for an oblique decision tree on the Student Portuguese grades dataset. *Column 2-4:* For source  $\bar{\mathbf{x}}_1$ ,  $\mathbf{x}_{11}^*$  and  $\mathbf{x}_{12}^*$  represents counterfactual with the given target class and the closest class respectively. *Column 5-7:* For source  $\bar{\mathbf{x}}_2$ , both counterfactuals are generated by changing only one attribute that has the lowest cost.  $\mathbf{x}_{21}^*$  and  $\mathbf{x}_{22}^*$  represent counterfactual with the given target class and the closest class, respectively. “=” means the attribute value is the same as in the source instance.

Attribute	$\bar{\mathbf{x}}_1$ , source instance	$\mathbf{x}_{11}^*$ , target class <i>excellent</i>	$\mathbf{x}_{12}^*$ , closest class	$\bar{\mathbf{x}}_2$ , source instance	$\mathbf{x}_{21}^*$ , target class <i>satisfactory</i>	$\mathbf{x}_{22}^*$ , closest class
	Only one attribute can change					
School	GP	=	=	MS	=	=
Sex	Male	=	=	Male	=	=
Age	18	=	=	17	=	=
Address	Urban	=	=	Urban	=	=
Family size	>3	=	=	>3	=	=
Parent’s status	Together	=	=	Together	=	=
Mother’s education	2	=	=	1	=	=
Father’s education	1	=	=	1	=	=
Mother’s job	Services	Teacher	=	Other	=	=
Father’s job	Services	Teacher	=	Other	=	=
Reason	Other	=	=	Home	=	=
Guardian	Mother	=	=	Mother	=	=
Traveltime <sup>†</sup>	1	=	=	1	=	=
Study time <sup>◊</sup>	1	=	=	2	=	=
Failures	2	1	1	0	=	=
School support	No	=	=	No	=	=
Family support	No	=	=	No	=	=
Paid	No	=	=	Yes	=	=
Activities	No	=	=	No	=	=
Nursery	No	=	=	No	=	=
Plan for higher education	No	Yes	=	Yes	=	=
Internet access	Yes	=	=	Yes	=	=
In romantic relationship	No	=	=	No	=	=
Family relationship <sup>‡</sup>	3	=	=	4	=	=
Free time <sup>‡</sup>	2	=	=	4	1	=
Going out frequency <sup>‡</sup>	5	=	=	3	=	2
Workday alcohol consumption <sup>‡</sup>	2	=	=	2	=	=
Weekend alcohol consumption <sup>‡</sup>	5	=	=	4	=	=
Health <sup>‡</sup>	5	=	=	5	=	=
Absences	4	=	=	4	=	=
Grades	Fail	Excellent	Sufficient	Fail	Satisfactory	Sufficient

<sup>†</sup> 1-<15 min, 2-15 to 30 min, 3-30 min to 1h or 4 -> 1h).

<sup>◊</sup> 1-<2h, 2-2 to 5 h, 3-5 h to 10h or 4 -> 10h).

<sup>‡</sup> from 1 - very low to 5 - very high.

**Table 2.** Counterfactual solution trajectory as a function of  $\epsilon$  for Credit dataset.  $\ell_2$  distance is also mentioned between source instance ( $\bar{\mathbf{x}}$ ) and the counterfactual  $\mathbf{x}^*$ .

Attribute	$\bar{\mathbf{x}}$ , source instance	$\mathbf{x}_1^*$ , $\epsilon = 0.00$ $\ell_2 = 1.73$	$\mathbf{x}_2^*$ , $\epsilon = 0.10$ $\ell_2 = 1.73$	$\mathbf{x}_3^*$ , $\epsilon = 0.20$ $\ell_2 = 2.00$	$\mathbf{x}_4^*$ , $\epsilon = 0.25$ $\ell_2 = 2.23$
Existing checking	<0 DM	=	=	=	=
Duration	15 months	16 months	=	=	16 months
Credithistory	Critical account	=	=	Delay in paying off in the past	Delay in paying off in the past
Purpose	Furniture/equipment	=	=	=	=
Credit amount	1478	=	=	=	=
Savings	<100 DM	=	=	=	=
Employment since	$\geq 7$ years	=	=	=	=
Installment rate	4	=	=	=	=
Status and sex	Male: single	=	=	=	=
Other debtors	None	=	=	=	=
Residence since	4	=	3	3	3
Property	Car or other	=	=	=	=
Age	44	=	=	=	=
Other installment plans	None	Stores	stores	=	=
Housing	Own	=	=	=	=
Existing credits	2	=	=	1	1
Job	Skilled employee	=	=	=	=
People liable	2	=	=	=	=
Telephone	Yes	=	=	=	=
Foreignworker	Yes	=	=	=	=
Credit	Bad	Good	Good	Good	Good

This is also evident from the Table 2. As we increase the value of  $\epsilon$ , the number of features changed also increases. Thus, creating a user profile ( $\mathbf{x}_4^*$ ) that is a better creditor than the first counterfactual ( $\mathbf{x}_1^*$ ). This can also be seen as the distance between the original instances and the counterfactual increases as the leaf regions get smaller (increasing value of  $\epsilon$ ).

These kinds of problems can be very useful, as shown in the above case. Here, we give a solution for now and give a strategy to become much better creditor.

In theory, the distance between the source instance and counter should increase with the value of  $\epsilon$  continuously, but in the presented case, it happens in intervals. The reason is due to the presence of integer and categorical attributes because the solution can only change when an attribute changes by an integer or category.

**Table 3.** Example illustrating the construction of counterfactual instances for regression with our extended formulations for an oblique decision tree on the Boston housing dataset. We show the dataset attributes, source instance  $\bar{\mathbf{x}}$ , and 3 counterfactual instances (with different Median home value) with various conditions. We have rounded each attribute value to 2 decimal places. “=” means the attribute value is the same as the source after rounding.

Attribute	$\bar{\mathbf{x}}$ , source instance	$\mathbf{x}^*$		
		$T(\mathbf{x}^*) > T(\bar{\mathbf{x}})$	$T(\mathbf{x}^*) \geq T(\bar{\mathbf{x}}) + 5$	$30 \geq T(\mathbf{x}^*) \geq 25$
Crime rate	2.37	2.15	1.93	1.81
Residential land zoned proportion	0.0	0.02	=	0.03
Proportion of non-retail business	19.58	19.48	=	19.48
Tract bounds river	0	=	1	=
Nitric oxides concentration	0.87	0.39	0.39	0.385
Avg. rooms per dwelling	4.92	5.13	5.73	8.09
Proportion of units before 1940	95.70	95.67	95.71	95.66
Distances to Boston employment centres	1.46	1.17	=	1.16
Accessibility to highways	5.00	5.04	5.1	5.41
Property-tax rate	403.00	=	=	402.99
Pupil-teacher ratio	14.70	14.47	14.58	14.60
Proportion of african american by town	391.71	=	=	391.67
% lower status of the population	29.53	29.39	29.54	29.41
Median home value in \$1000's	14.74	15.96	20.52	29.14

#### 4.4 Use Case Study 3

This user study deals with predicting the median home value of different suburbs and towns in Boston. We train a regression tree for this task. Our tree achieves a test mean squared error of 14.57 and train mean squared error of 13.19. The depth of the tree is 6, and the number of leaves is 21.

We consider an instance that has attributes described in the second column of Table 3. We apply our extension problems here to determine how for the given instance ( $\bar{\mathbf{x}}$ ), the attributes need to change to accommodate the new median home value. We investigate three scenarios:

$T(\mathbf{x}^*) > T(\bar{\mathbf{x}})$  meaning have higher median home value. Almost all of the attributes change (third column of the Table 3), the ones that change the most are as follows. The crime rate, nitric oxides concentration, distances to Boston employment centres, and pupil-teacher ratio decreases which make sense. Because having low crime rate means safer neighborhood and low nitric oxides concentration means the air quality is better. Also, having low pupil-teacher ratio means more teacher per student, which is beneficial for the families with children in school, and then distances to Boston employment centers is important for the people who need employment maybe for other family members. On the other side, the average rooms per dwelling increases, which is reasonable as it means bigger homes and thus higher prices.

$T(\mathbf{x}^*) > T(\bar{\mathbf{x}}) + 5$  meaning the median home value should be greater than 19.72. As shown in the fourth column of the Table 3 the crime rate and nitric oxides concentration further decrease. The pupil-teacher ratio also decreases but com-

pare to the previous scenario it is higher by a small reason. It may be the increased value of average rooms per dwelling and tract bounds river compensate for it.

$30 \geq T(\mathbf{x}^*) \geq 25$  meaning the median home value should be between 25 and 30. As shown in the fifth column of the Table 3 again the crime rate and nitric oxides concentration plays an important role in the value. The biggest change is the average rooms per dwelling which almost doubled. This is expected as described earlier.

This case study shows for this particular instances the crime rate, nitric oxides concentration and average rooms per dwelling plays an important role in deciding the median home value of the home. Also, average rooms per dwelling compensates for some attributes like pupil-teacher ratio.

## 5 Conclusion

Classification and regression trees are very important in applications such as education, business, law and medicine, where counterfactual explanations are of particular relevance. These can be formulated as a constrained optimization problem and solved exactly and efficiently for both continuous and categorical features, possibly in an interactive way. The formulation can be applied to answer a variety of practical questions and we have illustrated this in several case studies. Python code implementing the algorithm is available at the authors' web page.

**Acknowledgments.** Work partially supported by NSF award IIS-2007147.

## References

1. Bella, A., Ferri, C., Hernández-Orallo, J., Ramírez-Quintana, M.J.: Using negotiable features for prescription problems. *Computing* **91**, 135–168 (2011)
2. Breiman, L.J., Friedman, J.H., Olshen, R.A., Stone, C.J.: *Classification and Regression Trees*. Wadsworth, Belmont (1984)
3. Carreira-Perpiñán, M.Á.: The Tree Alternating Optimization (TAO) algorithm: a new way to learn decision trees and tree-based models (2021). arXIV
4. Carreira-Perpiñán, M.Á., Hada, S.S.: Counterfactual explanations for oblique decision trees: exact, efficient algorithms. In: *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI 2021)*, 2–9 February 2021, pp. 6903–6911 (2021)
5. Carreira-Perpiñán, M.Á., Hada, S.S.: Counterfactual explanations for oblique decision trees: exact, efficient algorithms. [arXiv:2103.01096](https://arxiv.org/abs/2103.01096) (2021)
6. Carreira-Perpiñán, M.Á., Hada, S.S.: Inverse classification with logistic and softmax classifiers: efficient optimization (2021). arXIV
7. Carreira-Perpiñán, M.Á., Tavallali, P.: Alternating optimization of decision trees, with application to learning sparse oblique trees. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) *Advances in Neural Information Processing Systems (NEURIPS)*, vol. 31, pp. 1211–1221. MIT Press, Cambridge (2018)

8. Carreira-Perpiñán, M.Á., Zharmagambetov, A.: Ensembles of bagged TAO trees consistently improve over random forests, AdaBoost and gradient boosting. In: Proceedings of the 2020 ACM-IMS Foundations of Data Science Conference (FODS 2020), Seattle, WA, 19–20 October 2020, pp. 35–46(2020)
9. Cui, Z., Chen, W., He, Y., Chen, Y.: Optimal action extraction for random forests and boosted trees. In: Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD 2015), Sydney, Australia, 10–13 August 2015, pp. 179–188 (2015)
10. Dosovitskiy, A., Brox, T.: Inverting visual representations with convolutional networks. In: Proceedings of the 2016 IEEE Computer Society Conference Computer Vision and Pattern Recognition (CVPR’16), Las Vegas, NV, 26 June–1 July 2016 (2016)
11. Freitas, A.A.: Comprehensible classification models: a position paper. *SIGKDD Explor.* **15**(1), 1–10 (2014)
12. Gabidolla, M., Zharmagambetov, A., Carreira-Perpiñán, M.Á.: Improved multi-class adaboost using sparse oblique decision trees (2021), submitted
13. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), San Diego, CA, 7–9 May 2015 (2015)
14. Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A survey of methods for explaining black box models. *ACM Comput. Surv.* **51**(5), 93 (2018)
15. Gurobi Optimization, LLC: Gurobi optimizer reference manual (2019)
16. Hada, S.S., Carreira-Perpiñán, M.Á.: Sampling the “inverse set” of a neuron: an approach to understanding neural nets. [arXiv:1910.04857](https://arxiv.org/abs/1910.04857) (2019)
17. Hada, S.S., Carreira-Perpiñán, M.Á.: Sampling the “inverse set” of a neuron. In: IEEE International Conference on Image Processing (ICIP 2021), Anchorage, AK, 9–12 September 2021 (2021)
18. Hada, S.S., Carreira-Perpiñán, M.Á., Zharmagambetov, A.: Sparse oblique decision trees: a tool to understand and manipulate neural net features. [arXiv:2104.02922](https://arxiv.org/abs/2104.02922) (2021)
19. Hada, S.S., Carreira-Perpiñán, M.Á., Zharmagambetov, A.: Understanding and manipulating neural net features using sparse oblique classification trees. In: IEEE International Conference on Image Processing (ICIP 2021), Anchorage, AK, 19–12 September 2021 (2021)
20. Idelbayev, Y., Zharmagambetov, A., Gabidolla, M., Carreira-Perpiñán, M.Á.: Faster neural net inference via forests of sparse oblique decision trees (2021). [arXIV](https://arxiv.org/abs/2104.02922)
21. Lipton, Z.C.: The mythos of model interpretability. *Comm. ACM* **81**(10), 36–43 (2018)
22. Mahendran, A., Vedaldi, A.: Visualizing deep convolutional neural networks using natural pre-images. *Int. J. Comput. Vision* **120**(3), 233–255 (2016)
23. Martens, D., Provost, F.: Explaining data-driven document classifications. *MIS Q.* **38**(1), 73–99 (2014)
24. Russell, C.: Efficient search for diverse coherent explanations. In: Proceedings of ACM Conference Fairness, Accountability, and Transparency (FAT 2019), Atlanta, GA, 29–31 January 2019, pp. 20–28 (2019)
25. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: visualising image classification models and saliency maps. In: Proceedings of the 2nd International Conference Learning Representations (ICLR 2014), Banff, Canada, 14–16 April 2014 (2014)

26. Szegedy, C., et al.: Intriguing properties of neural networks. In: Proceedings of the 2nd International Conference on Learning Representations (ICLR 2014), Banff, Canada, 14–16 April 2014 (2014)
27. Ustun, B., Spangher, A., Liu, Y.: Actionable recourse in linear classification. In: Proceedings of ACM Conference Fairness, Accountability, and Transparency (FAT 2019), Atlanta, GA, 29–31 January 2019, pp. 10–19 (2019)
28. Wachter, S., Mittelstadt, B., Russell, C.: Counterfactual explanations without opening the black box: automated decisions and the GDPR. *Harvard J. Law Technol.* **31**(2), 841–887 (2018)
29. Wu, X., Kumar, V. (eds.): *The Top Ten Algorithms in Data Mining*. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series, CRC Publishers, Boca Raton (2009)
30. Yang, Q., Yin, J., Ling, C.X., Pan, R.: Extracting actionable knowledge from decision trees. *IEEE Trans. Knowl. Data Eng.* **18**(1), 43–56 (2006)
31. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: Proceedings of 13th European Conference Computer Vision (ECCV’14), Zürich, Switzerland, 6–12 September 2014, pp. 818–833 (2014)
32. Zhang, C., Liu, C., Zhang, X., Almpandis, G.: An up-to-date comparison of state-of-the-art classification algorithms. *Expert Syst. Appl.* **82**, 128–150 (2017)
33. Zharmagambetov, A., Carreira-Perpiñán, M.Á.: Smaller, more accurate regression forests using tree alternating optimization. In: Daumé III, H., Singh, A. (eds.) Proceedings of the 37th International Conference on Machine Learning (ICML 2020), 13–18 July 2020, pp. 11398–11408 (2020)
34. Zharmagambetov, A., Carreira-Perpiñán, M.Á.: Learning a tree of neural nets. In: Proceedings of the IEEE International Conference Acoustics, Speech and Signal Processing (ICASSP’21), , Toronto, Canada, 6–11 June 2021, pp. 3140–3144 (2021)
35. Zharmagambetov, A., Gabidolla, M., Carreira-Perpiñán, M.Á.: Improved boosted regression forests through non-greedy tree optimization. In: International Joint Conference Neural Networks (IJCNN’21), Virtual Event, 18–22 July 2021 (2021)
36. Zharmagambetov, A., Gabidolla, M., Carreira-Perpiñán, M.Á.: Improved multi-class AdaBoost for image classification: the role of tree optimization. In: IEEE International Conference on Image Processing (ICIP 2021), Anchorage, AK, 19–22 September 2021 (2021)
37. Zharmagambetov, A., Gabidolla, M., Carreira-Perpiñán, M.Á.: Softmax tree: an accurate, fast classifier when the number of classes is large (2021), submitted
38. Zharmagambetov, A., Hada, S.S., Gabidolla, M., Carreira-Perpiñán, M.Á.: Non-greedy algorithms for decision tree optimization: an experimental comparison. In: International Joint Conference on Neural Networks (IJCNN’21), Virtual event, 18–22 July 2021 (2021)