

Counterfactual Explanations for Oblique Decision Trees: Exact, Efficient Algorithms*

Miguel Á. Carreira-Perpiñán Suryabhan Singh Hada
 Dept. of Computer Science & Engineering, University of California, Merced
<http://eecs.ucmerced.edu>

March 1, 2021

Abstract

We consider counterfactual explanations, the problem of minimally adjusting features in a source input instance so that it is classified as a target class under a given classifier. This has become a topic of recent interest as a way to query a trained model and suggest possible actions to overturn its decision. Mathematically, the problem is formally equivalent to that of finding adversarial examples, which also has attracted significant attention recently. Most work on either counterfactual explanations or adversarial examples has focused on differentiable classifiers, such as neural nets. We focus on classification trees, both axis-aligned and oblique (having hyperplane splits). Although here the counterfactual optimization problem is nonconvex and nondifferentiable, we show that an exact solution can be computed very efficiently, even with high-dimensional feature vectors and with both continuous and categorical features, and demonstrate it in different datasets and settings. The results are particularly relevant for finance, medicine or legal applications, where interpretability and counterfactual explanations are particularly important.

1 Introduction

Practical deployment of deep learning and machine learning models has become widespread in the last decade, and there is enormous societal interest in AI as a technology that can provide intelligent, automated processing of tasks that up to now were hard for machines. At the same time, some concerns about AI systems (ethical, safety and others) have arisen as well. One is the problem of interpretability, i.e., explaining the functionality of an automated system. This is an old problem, which has been studied (possibly under different names, such as explainability or transparency) since decades ago in statistics and machine learning (e.g. [4, 13, 16, 23]). A second problem is explaining why the system made a decision, and how to contest it or—of particular interest here—how to change it. This problem is more recent and has become more pressing and widely known with the advent of legislation, such as the European Union’s General Data Protection Regulation (GDPR) [15, 34], which requires AI systems to explain in some way its decisions to humans. That said, related problems have been studied in data mining or knowledge discovery from databases [2, 10, 25, 37], in particular in applications such as customer relationship management (CRM).

In this paper we focus on a specific version of the second problem that, following Wachter et al. [34], we will call a *counterfactual explanation*, which refers to the fact that an event did not actually happen. For example, “You were denied a loan because your annual income was \$30,000. If your income had been \$45,000, you would have been offered a loan.” The second statement, or counterfactual, offers an alternative event that would result in the desired outcome (loan approval). Formally, a counterfactual explanation seeks the minimal change to a given feature vector that will change a classifier’s decision in a prescribed way, and we will make this more precise as an optimization problem later. Compared to the problem of explaining the functionality of an automated system, counterfactual explanations are easier in that they do not require interpretability, as long as the explanations help a subject act rather than understand [34].

*A shorter version of this paper appears as [6]. Code implementing the algorithms is available from the authors’ web page.

Formally, the problem of finding a counterfactual explanation is the same as that of finding an adversarial example [14, 29, 30, 38]. The difference is in the underlying motivation. In a counterfactual explanation, one typically seeks a change of the source feature vector that is as small as possible (because changing features is seen as costly) and changes the classifier outcome (to a prescribed and more desirable one). In an adversarial example, one typically seeks a change of the source feature vector that is as small as possible (so that it is hard to detect) and changes the classifier outcome (to trick it into predicting the wrong outcome). Our focus will be in the solution of such optimization problems, although we will use as running example the case of counterfactual explanations.

Recent work has explored various ways of defining counterfactual explanation problems and solving them for different classifiers, in particular neural nets. Here we consider classification trees¹, which are of particular interest compared to other models for several reasons. Trees are widely used in practice, can handle continuous and discrete features, are extremely fast for inference, can model nonlinear boundaries, and provide multiclass models naturally (without the need of constructions such as one-vs-all). Perhaps most importantly, trees are generally considered among the most interpretable models (certainly much more so than neural nets or forests). And, while (axis-aligned) trees have traditionally not been competitive in terms of predictive accuracy with other models, a recent algorithm (tree alternating optimization, TAO; [5, 8, 42]) is able to learn oblique trees whose accuracy is much higher, which makes such trees competitive with other models.

Mathematically, trees provide a nonlinear classifier based on hierarchical, discontinuous splits, so the corresponding counterfactual problem is nonconvex and nondifferentiable. Yet, we show it can be solved exactly and efficiently, even if categorical variables exist. Our algorithm is very fast and suitable for interactive exploration of counterfactual explanations under different objectives or constraints. It can also provide, in a natural way, not just one but a diverse set of counterfactual explanations, which provide a range of ways in which the desired classifier outcome may be achieved.

2 Related work

Counterfactual explanations can be seen as a form of knowledge extraction from a trained machine learning model. This is the traditional realm of data mining, particularly in business and marketing [1, 19, 20, 36]. However, the precise formulation of counterfactual explanations as optimization problems given a classifier, source instance and target class (such as the one we follow in section 3.3) and the various works exploring this research topic are quite recent. The formulation of counterfactual explanations can take different forms but always involve a distance function to measure how costly it is to change features (attributes) in a source instance, as well as a constraint or penalty term that ensures a target class is predicted. Optimizing a tradeoff of both of these yields the counterfactual instance. 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. This has been particularly exploited with deep nets for adversarial examples and model inversion [12, 14, 18, 24, 29, 30, 38], two problems that are very similar to counterfactual explanations. Other methods are specific for linear models [7, 27, 32]. However, none of these algorithms apply to decision trees, which define nondifferentiable classifiers.

Some agnostic algorithms have been proposed which assume nothing about the classifier other than it can be evaluated on arbitrary instances, using some kind of random or approximate search [22, 28]. While these approaches are very general, they are computationally slow, particularly with high-dimensional instances, and give poor approximations to the optimal solution. One agnostic approach is to restrict the instance search space to a finite set of instances (such as the training set of the classifier), so the optimization involves a simple brute-force search, as in a database search. While this may be useful in some applications, it has a limited ability to explore the instance space, particularly if the problem constraints are satisfied by few instances, and is slow if the set has many, high-dimensional instances. A recent implementation of this approach is in the What-If Tool [35] for interaction and visualization of machine learning systems.

Our paper is specifically about decision trees, both axis-aligned and oblique, for multiclass classification

¹Throughout, we consider hard decision trees, where an instance goes either left or right at each decision node, and hence it reaches a single leaf. We do not consider soft decision trees (such as hierarchical mixtures of experts; [21]), where the instance traverses all root-leaf paths, each weighted by a certain probability.

and using continuous and categorical features. What little work exists in counterfactual explanations research about decision trees has focused on axis-aligned trees (or forests) for binary classification only, as far as we know. Yang et al. [37], motivated by customer relationship management, seek to infer actions from a binary classification tree (attrition vs no attrition), specifically to move a group of instances (customers) from some source leaves to some target leaves of the tree. The problem is different from a standard counterfactual explanation and is restricted to categorical features only. It takes the form of a maximum coverage problem, which is NP-complete and is approximated with a greedy algorithm. Bella et al. [2] consider a restricted form of counterfactual explanation over a single, “negotiable” feature, which must be continuous and satisfy certain sensitivity and monotonicity conditions. Cui et al. [10] formulate a type of counterfactual problem for binary classification forests, show it is NP-hard, and encode it as an integer linear program, which can be (approximately) solved by existing solvers. It is practical only for low-dimensional feature vectors, and even then it takes seconds or minutes for one instance. Tolomei et al. [31] also consider a restricted form of counterfactual problem for a binary classification forest and propose an approximate algorithm, based on propagating the source instance down each tree towards a leaf. This is claimed to be optimal if the forest contains a single tree. However, as our experiments show, this is not true.

3 Counterfactual explanation for oblique trees: an exact algorithm

3.1 Definitions

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 $\mathcal{N} = \{1, \dots, 17\}$, $\mathcal{L} = \{8, 10, \dots, 17\}$ and $\mathcal{D} = \mathcal{N} \setminus \mathcal{L}$. Each decision node $i \in \mathcal{D}$ has a real-valued decision function $f_i(\mathbf{x})$ such that input instance $\mathbf{x} \in \mathbb{R}^D$ is sent down i 's right child if $f_i(\mathbf{x}) \geq 0$ and down i 's left child otherwise. For oblique trees, the decision function is 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). Each leaf $i \in \mathcal{L}$ is labeled with one class label $y_i \in \{1, \dots, K\}$. The class $T(\mathbf{x}) \in \{1, \dots, K\}$ predicted by the tree for an input instance \mathbf{x} is found by sending \mathbf{x} down, 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 [5, 8] (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 figures 1 and 2 (right panels). Each region is an axis-aligned box (for axis-aligned trees) or polytope (for oblique trees) given 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 $\mathcal{F}_i = \{\mathbf{x} \in \mathbb{R}^D: \mathbf{h}_i(\mathbf{x}) \geq \mathbf{0}\}$ the corresponding feasible set, i.e., the region in input space of leaf i . For example, in fig. 2 (left) the path from the root to leaf 15 is $\mathcal{P}_{15} = \{1, 3, 6, 10, 15\}$ and its region is given by:

$$\mathbf{h}_{15}(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) \\ -f_3(\mathbf{x}) \\ -f_6(\mathbf{x}) \\ f_{10}(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} \mathbf{w}_1^T \mathbf{x} + b_1 \\ -\mathbf{w}_3^T \mathbf{x} - b_3 \\ -\mathbf{w}_6^T \mathbf{x} - b_6 \\ \mathbf{w}_{10}^T \mathbf{x} + b_{10} \end{pmatrix} \geq \mathbf{0}.$$

3.2 Learning axis-aligned and oblique trees: Tree Alternating Optimization (TAO)

Traditionally, classification trees have been learned from a labeled training set using greedy top-down algorithms that split an initial, root node (using a class purity criterion) and proceed recursively with its children until a stopping criterion is achieved. The classical examples are CART [4] and C4.5 [26]. However,

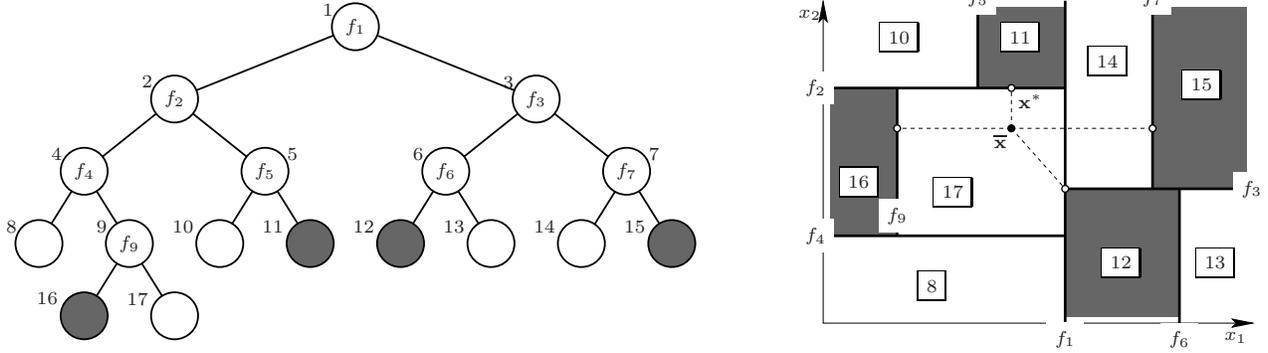


Figure 1: *Left*: an axis-aligned classification tree with $K = 2$ classes (colored white and gray). A decision node i sends an input instance \mathbf{x} to its right child if $f_i(\mathbf{x}) \geq 0$ and to its left child otherwise. For example, for node 5 we have $f_5(\mathbf{x}) = x_1 + b_5$, i.e., it thresholds x_1 and hence creates a vertical split. Likewise, for node 4 we have $f_4(\mathbf{x}) = x_2 + b_4$, i.e., it thresholds x_2 and hence creates a horizontal split. Each leaf i is colored according to its class label $y_i \in \{1, \dots, K\}$. *Right*: the space of the input instances $\mathbf{x} \in \mathbb{R}^2$, assumed two-dimensional, partitioned according to each leaf’s region, which is an axis-aligned box (the region boundaries are labeled with the corresponding decision node function). The source instance is $\bar{\mathbf{x}}$, of the white class. The counterfactual instance (using the squared ℓ_2 distance, $\|\mathbf{x} - \bar{\mathbf{x}}\|^2$) subject to changing to the gray class is \mathbf{x}^* , which is closest to $\bar{\mathbf{x}}$. We also show the closest instances to $\bar{\mathbf{x}}$ within each leaf of the gray class.

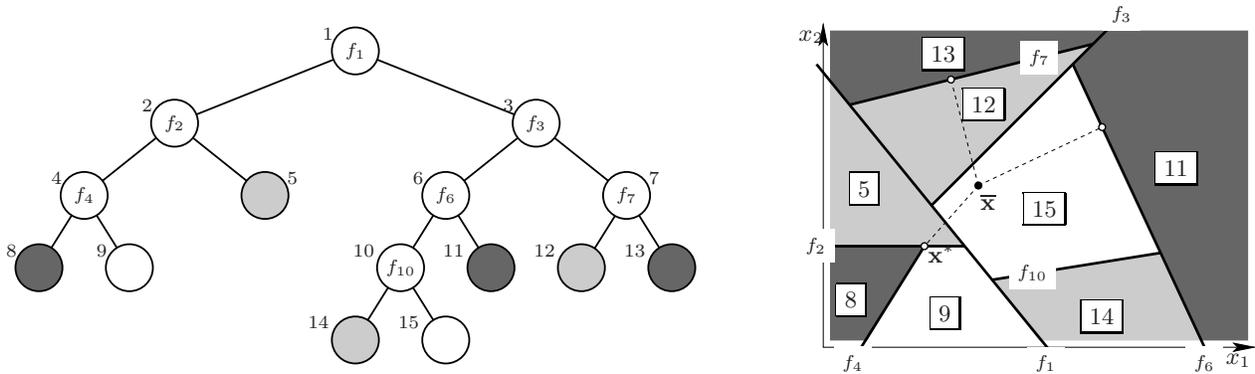


Figure 2: *Left*: like fig. 1 but for an oblique classification tree with $K = 3$ classes (colored white, light gray and gray). Unlike in an axis-aligned tree, where each decision function uses a single feature, in the oblique tree it uses a linear combination of them: $f_i(\mathbf{x}) = \mathbf{w}_i^T \mathbf{x} + b_i$. The source instance $\bar{\mathbf{x}}$ is in the white class and the counterfactual one (using the ℓ_2 distance) subject to being in the gray class is \mathbf{x}^* .

these algorithms achieve quite suboptimal trees, particularly if they are applied to oblique trees (having hyperplane decision nodes). Still, they are widely used in practice to learn axis-aligned (univariate) trees.

Recently, a new algorithm has been proposed that is able to train trees more accurately, both axis-aligned and oblique: Tree Alternating Optimization (TAO) [5, 8]. TAO does not grow a tree greedily. Instead, it takes a given tree structure with initial parameter values at the nodes and optimizes a loss function over these parameters—much as one would train, say, a neural net, except the tree is not differentiable. TAO essentially works by optimizing the parameters of each node (decision node or leaf) at a time. Each iteration of TAO is guaranteed to reduce or leave unchanged the classification error, which results in trees that are smaller yet much more accurate than those trained with CART, C4.5 or other algorithms, as shown in a range of datasets [42]. Furthermore, the predictive accuracy of oblique trees trained with TAO becomes comparable to that of state-of-the-art classifiers. Ensembles of TAO trees also improve considerably over traditional forests [9, 40]. Other types of trees can also be learned [41]. More details about TAO can be found in [5, 42].

In this paper, we illustrate our counterfactual explanation algorithm with axis-aligned trees trained with

CART and oblique trees trained with TAO.

3.3 Basic counterfactual optimization problem

We start by giving the simplest, but also most important, formulation of finding an optimal counterfactual explanation. 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*)². 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 features of $\bar{\mathbf{x}}$, and $\mathbf{c}(\mathbf{x})$ and $\mathbf{d}(\mathbf{x})$ are equality and inequality constraints (in vector form), all of which will be defined more precisely in sections 3.6 and 3.7. 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, the following simple observation, whose proof is obvious, shows that we can solve the problem exactly and efficiently.

Theorem 3.1. *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 and then picking the leaf with the best solution. This is shown in figures 1 and 2 (right panels). That is, the problem has the form of a mixed-integer optimization where the integer part is done by enumeration (over the leaves) and the continuous part (within each leaf) by other means to be described later. This is true for any tree as long as it has hard decisions at the decision nodes, even if the decision functions and leaf predictors are more complex than the hyperplanes and constant labels, respectively, that we consider here. Since the number of leaves in a tree is relatively small, this is computationally possible (in particular, oblique trees have very few leaves compared to axis-aligned ones).

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:

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

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 linear or quadratic then the problem is a linear program (LP) or a convex quadratic program (QP), both of which can be solved very efficiently with existing solvers, more so because the number of variables D is usually not very large in practice (thousands at most, and in some important applications it can be very small).

3.4 Separable problems: axis-aligned trees

The following result, whose proof is immediate, vastly simplifies the problem for axis-aligned trees.

Theorem 3.2. *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.

²We can also consider a formulation of the problem where the target class y is the same as the source class \bar{y} , but we seek an instance \mathbf{x} having a lower cost than the source instance $\bar{\mathbf{x}}$. For example, even if a subject is classified as approved for a mortgage, it may be possible to reduce the initial payment and still be approved. This requires redefining the cost $E(\mathbf{x})$ accordingly.

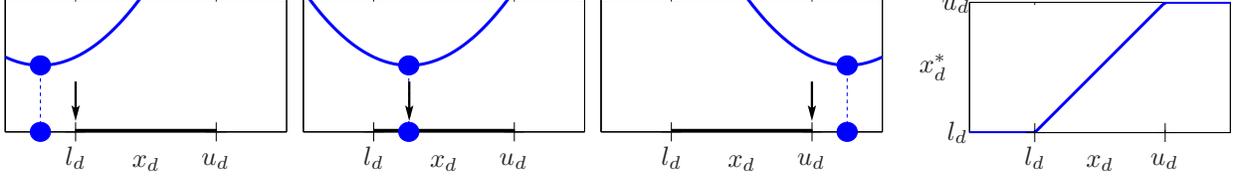


Figure 3: *Left*: three possible cases for the location of the solution for the scalar box-constrained problem (4). *Right*: the solution $x_d^* = \text{median}(\bar{x}_d, l_d, u_d)$ is the median of \bar{x}_d , l and u .

Theorem 3.3. 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:

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

Proof. From the assumption over E_d we have that \bar{x}_d is a global minimizer of E_d . The result follows by comparing \bar{x}_d with l_d and u_d ; see fig. 3. \square

The previous theorem does not consider equality constraints because, in a scalar problem, they trivially provide the solution (an equality constraint “ $x_d = \text{value}$ ” implies $x_d^* = \text{value}$). The inequalities “ $l_d \leq x_d \leq u_d$ ” in the theorem are obtained by collecting all the inequalities in the problem (1) that involve x_d .

Importantly, these theorems apply 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} . This makes solving the counterfactual explanation problem exceedingly fast for axis-aligned trees. We can represent each leaf $i \in \mathcal{L}$ by a bounding box $\mathbf{h}_i \leq \mathbf{x} \leq \mathbf{u}_i$ (which collects the constraints along the path from the root to i), solve elementwise by applying the median formula above within each leaf, and finally return the result of the best leaf.

Finally, the following result shows that, with axis-aligned trees, we obtain the same solution whether we use the ℓ_1 norm or the ℓ_2 norm or a linear combination of both.

Corollary 3.4. 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 $E(\mathbf{x}; \bar{\mathbf{x}}) = \lambda_1 \|\mathbf{x} - \bar{\mathbf{x}}\|_1 + \lambda_2 \|\mathbf{x} - \bar{\mathbf{x}}\|_2^2$ with coefficients $\lambda_1, \lambda_2 \geq 0$. Then the solution of the problem is the same, regardless of the values of the coefficients, and it is given by applying the median formula of theorem 3.3 elementwise within each leaf and then picking the best leaf.

However, note that if we use a different weight *per feature*, e.g. $E(\mathbf{x}; \bar{\mathbf{x}}) = \sum_{d=1}^D \lambda_d (x_d - \bar{x}_d)^2$, then the optimal solution does depend on those weights: while it can still be computed elementwise within each leaf, which leaf is the best depends on the weights.

3.5 Non-separable problems: oblique trees

With oblique trees, the root-leaf path constraints $\mathbf{h}_i(\mathbf{x}) \geq \mathbf{0}$ involve each a linear combination of multiple features, as shown in fig. 2 (right). Hence the problem (3) over a leaf does not separate and cannot be solved elementwise over each feature. However, it can still be solved exactly and efficiently using LP or QP solvers.

Computationally, it is convenient to store in each leaf its root-leaf path constraints and possibly to preprocess them in order to make the subsequent optimization more efficient (as is customary with LP or QP solvers). For example, one can remove redundant constraints in the root-leaf path (e.g. f_1 is redundant for leaf 16 in fig. 1). Also, a good initialization for each QP is given by the source instance $\bar{\mathbf{x}}$.

3.6 Useful cost or distance functions

The function $E(\mathbf{x}; \bar{\mathbf{x}})$ measures the cost of changing features in the source instance $\bar{\mathbf{x}}$, so it should satisfy $E(\bar{\mathbf{x}}; \bar{\mathbf{x}}) = 0$ and $E(\mathbf{x}; \bar{\mathbf{x}}) > 0$ if $\mathbf{x} \neq \bar{\mathbf{x}}$ (or perhaps $E(\mathbf{x}; \bar{\mathbf{x}}) \geq 0$). An appropriate definition of E is critical to find good counterfactual explanations, but such a definition depends on the application. For example, changing the amount of a loan is easier than changing the education level of a person. That said, a useful cost function can generally be written using a distance or a combination of distances, possibly weighted. Next, we give several generic distances that are convex and can be easily handled with decision trees. All of them have been used in earlier works, with the exception of the general quadratic distance, as far as we know.

- ℓ_2 distance: $E(\mathbf{x}; \bar{\mathbf{x}}) = \|\mathbf{x} - \bar{\mathbf{x}}\|_2^2$.
- ℓ_1 distance: $E(\mathbf{x}; \bar{\mathbf{x}}) = \|\mathbf{x} - \bar{\mathbf{x}}\|_1$. This encourages few features to be changed, while the ℓ_2 distance typically changes all features.

To optimize a problem of the form “ $\min_{\delta} \|\delta\|_1$ s.t. $\delta \in \mathcal{F}$ ” (where $\delta = \mathbf{x} - \bar{\mathbf{x}}$ and \mathcal{F} is a polytope), we use the standard reformulation as a LP “ $\min_{\delta, \mathbf{t}} \mathbf{1}^T \mathbf{t}$ s.t. $\delta \leq \mathbf{t}, \delta \geq -\mathbf{t}, \delta \in \mathcal{F}$ ”.

- General quadratic distance.

$$E(\mathbf{x}; \bar{\mathbf{x}}) = \delta^T \mathbf{Q} \delta = \sum_{d,e=1}^D q_{de} \delta_d \delta_e \quad (6)$$

where we call $\delta = \mathbf{x} - \bar{\mathbf{x}}$ and \mathbf{Q} is a symmetric positive definite (or perhaps positive semidefinite) matrix, so E is lower bounded. This means the distance is a sum of two types of costs:

- Singleton feature cost: $q_{dd} \delta_d^2 \geq 0$. This is the cost of changing feature d by δ_d .
- Pair of features’ cost: $q_{de} \delta_d \delta_e$. This is the cost of changing feature d by δ_d and feature e by δ_e . We can have $q_{de} < 0$ or > 0 as long as \mathbf{Q} is positive (semi)definite.

The singleton cost is useful to represent differential costs depending on the feature (as in a weighted ℓ_2 norm) while the pairwise cost is useful to represent costs that affect groups of features. For example, if \mathbf{x} are pixel values of a grayscale image, defining q_{de} according to whether pixels d and e are neighboring can encourage changing local groups of pixels rather than arbitrary, isolated pixels.

Making \mathbf{Q} positive semidefinite allows us to have zero cost for feature changes δ satisfying $\mathbf{Q}\delta = \mathbf{0}$, which can be useful to represent invariances. For example, if \mathbf{x} are pixel grayscale values, having $\mathbf{Q}\mathbf{1} = \mathbf{0}$ (where $\mathbf{1} = (1, \dots, 1)^T$) means that the cost of shifting all pixels by 1, i.e., globally changing the image intensity, is zero.

- Finally, we can have combinations of all the above, such as $E(\mathbf{x}; \bar{\mathbf{x}}) = \|\mathbf{x} - \bar{\mathbf{x}}\|_1 + \lambda \|\mathbf{x} - \bar{\mathbf{x}}\|_2^2$.

We emphasize that in practice the ℓ_1 or ℓ_2 distances should be weighted (or equivalently each feature should be normalized). Such weights should be chosen by the user according to the range of variation of each feature and to its perceived cost.

3.7 Useful constraints

The constraints $\mathbf{c}(\mathbf{x}) = \mathbf{0}$ and $\mathbf{d}(\mathbf{x}) \geq \mathbf{0}$ in problem (1) can be used to represent restrictions that must be obeyed for a counterfactual explanation to be reasonable. We consider the following:

- Constraints intrinsic to the problem: these typically represent natural equality constraints or lower and upper limits of each variable. We give some examples. Many variables are nonnegative, such as salary or cholesterol level. For a grayscale image each pixel should be in the interval $[0,1]$ (black to white). For a color image, suitable intervals must be obeyed depending on the color space (RGB, LUV, etc.). The race of an individual cannot change from what it is. The age of an individual must be nonnegative and smaller than, say, 120 years. Further, if feature d is the age of an individual, then we should constrain $x_d \geq \bar{x}_d$ to indicate that the given individual can get older but not younger.

- Constraints desirable for a particular explanation: these are given by the user on a case-by-case basis. For example, a loan applicant may not want to change her marital status, and cannot increase her age by more than say a few months, even if either of those were possible and resulted in the loan being approved by the tree.
- Categorical variables: because we handle them as continuous with a one-hot encoding, this introduces some constraints (see section 3.8).
- We can constrain \mathbf{x} to be in a discrete set of instances, such as the training set x_1, \dots, \mathbf{x}_N of the tree. The optimization reduces to a simple search in the set: we evaluate the objective $E(\mathbf{x}; \overline{\mathbf{x}})$ on every instance \mathbf{x}_n whose ground-truth class is the target class and satisfies the additional constraints $\mathbf{c}(\mathbf{x}_n) = \mathbf{0}$, $\mathbf{d}(\mathbf{x}_n) \geq \mathbf{0}$, and return the one with lowest objective.

3.8 Categorical variables

Although many popular benchmarks and models in machine learning use only continuous variables, categorical variables are very important in practice, particularly in legal, financial or medical applications. And it is precisely in these human-related applications where counterfactual explanations might be most useful.

We handle categorical variables by encoding them as one-hot. That is, 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. $\mathbf{1}^T \mathbf{x} = 1$.

During training with the TAO algorithm, we treat the dummy variables as if they were continuous and without the above constraints. This causes no problem because we only need to read the values of those variables; we do not need to update them.

When solving the counterfactual problem, we do modify those variables and 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 for two reasons: 1) categorical variables typically arise in low-dimensional problems and do not have many categories, so the total number of binary dummy variables is relatively small. And 2) modern mixed-integer optimization solvers, such as CPLEX or Gurobi [17], can solve relatively large problems exactly, and even larger ones approximately (providing a feasible result and a lower bound to the optimal objective) [3].

Note that the simple approach of relaxing each integer constraint $x_c \in \{0, 1\}$ to $x_c \geq 0$, solving a continuous optimization and rounding its result (i.e., picking the category c with the largest x_c value) has two drawbacks, which we have observed in our experiments: the result can be a poor approximation of the optimum; and, worse, applying the tree to it may not predict the target class (i.e., the rounded instance is infeasible).

Finally, with separable problems (section 3.4), all C dummy variables associated with an original categorical variable taking C categories can be separated from all the other variables in the problem, but the C dummy variables are optimized jointly. This is done simply by enumeration, i.e., trying each of the C categories and picking the best.

3.9 Extensions of the basic problem

The basic counterfactual problem (1) can be extended in several ways while remaining computationally easy:

- By their very nature, decision trees provide a simple way to offer the user not just a single solution for \mathbf{x} but a *diverse set of solutions*: we simply return the solutions of all the leaves having as label the target class label (assuming there are multiple leaves in that class), sorted in increasing cost E . Because the leaves' regions will usually lie far apart in instance space, their solutions will markedly differ in character.
- We can define as target not a single class $y \in \{1, \dots, K\}$ but a (nonempty) subset of classes $\mathcal{Y} \subset \mathcal{C}$

$\{1, \dots, K\} \setminus \bar{y}$:

$$\begin{aligned} & \min_{\mathbf{x} \in \mathbb{R}^D} E(\mathbf{x}; \bar{\mathbf{x}}) \quad \text{s.t.} \quad T(\mathbf{x}) \in \mathcal{Y}, \mathbf{c}(\mathbf{x}) = \mathbf{0}, \mathbf{d}(\mathbf{x}) \geq \mathbf{0} \implies \\ & \min_{i \in \mathcal{L}} \min_{\mathbf{x} \in \mathbb{R}^D} E(\mathbf{x}; \bar{\mathbf{x}}) \quad \text{s.t.} \quad y_i \in \mathcal{Y}, \mathbf{h}_i(\mathbf{x}) \geq \mathbf{0}, \mathbf{c}(\mathbf{x}) = \mathbf{0}, \mathbf{d}(\mathbf{x}) \geq \mathbf{0}. \end{aligned}$$

This is solved by enumeration over all leaves whose class is in \mathcal{Y} .

- More generally, rather than a preferred class or class subset, we can consider all classes as feasible but with different costs. We can move the tree constraint to the objective, which becomes “ $\min_{\mathbf{x}} E(\mathbf{x}; \bar{\mathbf{x}}) + L(T(\mathbf{x}))$ ”, where $L(y) \geq 0$ is the cost for class $y \in \{1, \dots, K\}$. We can still solve over y by enumeration over the leaves.
- We may want to keep the counterfactual explanation away from class boundaries. The solution of problem (1) is always on the boundary of a leaf region of the target class, hence an infinitesimal perturbation will make it change class. This may not be a problem in some applications, for example “barely passing” could be enough to get a loan approved. However, sometimes we may want to find a counterfactual explanation that is safely classified as the target class. One simple way to avoid this is to shrink the leaf region away from its boundaries by shifting each leaf’s constraints (box or polytope faces) by $\epsilon \geq 0$: $\mathbf{h}_i(\mathbf{x}) \geq \epsilon$. This provides a way to offer a range of counterfactual explanations for varying values of ϵ . The problem remains convex and can be efficiently solved by solving first for $\epsilon = 0$ and then warm-starting the problem for a larger ϵ from the solution for the previous ϵ value (for each leaf).

4 Experiments

Our algorithm is exact, as we have shown. It is also very efficient for both axis-aligned and oblique trees, even if there are categorical variables; solving a counterfactual explanation in our experiments takes usually milliseconds. We show 3 types of results: an example illustrating how the algorithm can be used interactively; summary results in several datasets, comparing with other algorithms; and results on MNIST digit images, which can be visualized.

4.1 Illustrative example

We consider the UCI Adult dataset for binary classification, where each instance corresponds to a person (age, education, sex, etc., involving both continuous and categorical features), and the classes are whether or not the person makes over \$50k a year. We are given an oblique decision tree trained by TAO. We take the source instance in table 1, which is classified by the tree as “below \$50k”, and seek counterfactual explanations \mathbf{x}^* (in ℓ_2 distance) classified as “above \$50k”. Without any constraints, 3 features change: race, capital-loss and native-country (\mathbf{x}_1^*). Changing a person’s race and native-country is not possible, so we constrain race, native-country as well as sex not to change. This results in 3 other features changing: education, marital-status and hours-per-week (\mathbf{x}_2^*). Based on the user’s preferences we then constrain education, marital-status and relationship not to change. Finally, this results in changing workclass, occupation, capital-gain and capital-loss (\mathbf{x}_3^*).

4.2 Algorithm comparison over different datasets

We now run our algorithm on several source instances from several datasets (of different numbers of features D and classes K , and with continuous and/or categorical features), on both an oblique tree trained by TAO (table 2) and an axis-aligned tree trained by CART (table 3). For each dataset we randomly select 20 source instances of each class from the test instances and generate a counterfactual explanation for them (to target a different class), using the ℓ_2 or ℓ_1 distance. As in the illustrative example, we try 3 levels of constraints: no constraints, some constraints and even more constraints (picked at random or manually); the tables show the percentage of features constrained in each case.

Feature	\bar{x} , source instance	\mathbf{x}_1^* , no constraints	\mathbf{x}_2^* , some constraints	\mathbf{x}_3^* , more constraints
age	25	=	=	=
workclass	Private	=	=	Federal-gov
education	11th	=	Assoc-voc	=
marital-status	Never-married	=	Married-AF-spouse	=
occupation	Machine-op-inspect	=	=	Armed-Forces
relationship	Own-child	=	=	=
race	Black	Asian-Pac-Islander	=	=
sex	Male	=	=	=
capital-gain	0	=	=	1
capital-loss	0	1	=	3
hours-per-week	40	=	39	=
native-country	United-States	Peru	=	=
income	<\$50k	≥\$50K	≥\$50K	≥\$50K

Table 1: Example illustrating the construction of counterfactual instances with our exact algorithm for an oblique decision tree on the Adult dataset. We show the dataset features, source instance \bar{x} (of class “<\$50k”), and 3 counterfactual instances (of class “≥\$50k”) with progressively more user constraints (\mathbf{x}_1^* , \mathbf{x}_2^* , \mathbf{x}_3^*). “=” means the feature value is the same as in the source instance.

	% c	Our exact algorithm				training & test set		
		ms	ℓ_2	ms	ℓ_1	ms	ℓ_2	% feasible
MNIST	0	40	0.63±0.48	580	2.85±1.40	40	53.50±17.24	100
	9	40	0.63±0.48	580	2.85±1.40	40	53.50±17.24	100
	47	40	9.28±6.70	550	12.80±7.89	—	—	0
Adult	0	710	2.40±0.83	600	2.40±0.83	70	3.4e4±1.2e5	100
	7	810	2.45±0.86	590	2.40±0.83	4300	1.6e7±5.2e7	100
	14	780	2.49±0.97	570	2.50±0.97	2700	1.9e7±5.8e7	100
Breast	0	4	0.35±0.33	2	1.14±0.48	0	0.86±0.49	100
	11	3	0.48±0.44	2	1.14±0.48	0	1.25±0.54	77
	22	3	0.52±0.48	2	1.16±0.48	0	1.61±0.89	17
\$pambase	0	7	0.002±0.01	12	0.060±0.06	0	0.060±0.06	100
	17	6	0.002±0.01	12	0.060±0.06	10	0.070±0.09	32
	53	7	0.003±0.01	12	0.070±0.06	20	0.020±0.01	17
Letter	0	18	0.02±0.01	20	0.31±0.12	0	0.19±0.09	100
	25	17	0.03±0.02	20	0.31±0.12	0	0.36±0.19	19
	62	16	0.20±0.84	190	0.55±0.47	—	—	0
Credit	0	50	3.97±3.50	80	2.70±1.20	0	3.0e3±7.3e3	100
	7	50	4.00±3.50	80	2.70±1.20	—	—	0
	16	50	4.25±5.80	80	2.80±1.30	—	—	0

Table 2: Comparison of counterfactuals generated by our exact algorithm and by searching over the training & test set, over a variety of datasets, using an oblique classification tree. We show: the percentage of features constrained in the source instance (% c), average runtime per instance in milliseconds (ms), ℓ_2 or ℓ_1 distance (mean and standard deviation over 20 instances per class), and the percentage of feasible counterfactuals (for the search in the training & test set only, since for our algorithm it is always 100%).

	% c	Our exact algorithm				Feature tweak [31]		
		ms	ℓ_2	ms	ℓ_1	ms	ℓ_2	% feasible
MNIST	0	110	0.04±0.11	110	0.16±0.26	190	1.48±1.09	100
	9	110	0.04±0.11	110	0.16±0.26	—	—	—
	47	120	5.83±3.27	120	13.16±3.65	—	—	—
Adult	0	130	2.05 ±0.31	130	2.05±0.31	50	1.00±0.00	28
	7	130	2.07±0.34	130	2.07±0.34	—	—	—
	14	140	26.10±14.97	140	2.85±4.54	—	—	—
Breast	0	0	0.13±0.17	0	0.42±0.28	0	0.36±0.19	100
	11	0	0.19 ±0.25	0	0.47±0.33	—	—	—
	22	0	0.22±0.29	0	0.48 ±0.35	—	—	—
Spambase	0	03	1.7e−5±0.0	30	0.003±0.002	0	0.005±0.009	100
	17	30	1.7e−5±0.0	30	0.004±0.002	—	—	—
	53	40	4.5e−5±0.0	40	0.005±0.004	—	—	—
Letter	0	50	0.014±0.01	50	0.16±0.08	30	0.22±0.13	100
	25	40	0.016±0.02	50	0.17±0.09	—	—	—
	62	40	0.058±0.05	60	0.28±0.02	—	—	—
Credit	0	40	2.60±1.01	40	2.60±1.01	0	87.88±13.20	22
	7	40	2.60±1.01	40	2.60±1.01	—	—	—
	16	40	26.50±50.2	40	4.87±4.64	—	—	—

Table 3: Like table 2 but for an axis-aligned classification tree. We also show the results for the Feature Tweaking method of [31] (which does not apply to problems with constraints, marked “—”).

In the oblique tree, we compare with searching only over those training & test set instances (as in the What-If tool [35]) which are classified as the target class by the tree, using the ℓ_2 distance. We label this as “training & test set” in the tables and figures. (Note that the tree is not perfect and may misclassify an instance, as happens in the first and last rows of fig. 4.) Clearly, this produces counterfactual instances with far larger distances and fails to find a feasible counterfactual instance (i.e., of the target class) if too many constraints are applied.

In the axis-aligned tree, we compare with the Feature Tweaking algorithm [31] in the ℓ_2 distance, by running the authors’ implementation (note this algorithm does not apply to oblique trees). Since it handles only continuous features, to use categorical ones we encode them as one-hot, solve as if they were continuous and round them at the end. We clearly see that Feature Tweaking is not exact for binary axis-aligned trees, contradicting the claim in [31]. This is shown by the larger distances and by the failure to find a feasible counterfactual instance if too many constraints are applied. Our algorithm is indeed exact for both oblique and axis-aligned trees and returns a feasible, minimal-distance counterfactual instance every time.

The runtime of our algorithm is a few milliseconds per instance, even in relatively high-dimensional cases such as the MNIST dataset ($D = 784$), or involving around 10 categorical features translating into almost 100 binary dummy variables as in the Adult dataset.

4.3 MNIST dataset with oblique trees

Figure 4 shows results using an oblique tree trained with TAO for MNIST digit images, so that we can visualize the result of different distances. We constrain each pixel to be in $[0,1]$ so that the counterfactual instance is a valid grayscale image. For both the ℓ_1 and ℓ_2 distances, the counterfactual instance is visually barely distinguishable from the source instance. As is well known, the ℓ_1 distance results in few pixels changing but by a large amount, while the ℓ_2 distance results in most pixels changing but by a small amount, and in both cases the pixel locations are arbitrary.

We then tried using a general quadratic distance. We constructed a positive definite matrix \mathbf{Q} of 784×784 having 1s in the diagonal and a value of $-\frac{1}{4}$ corresponding to neighboring pixels (up, down, left, right).

We also constrained each pixel to obey $\mathbf{x} \geq \bar{\mathbf{x}}$ (add ink only) or $\mathbf{x} \leq \bar{\mathbf{x}}$ (erase ink only). The resulting counterfactual images clearly show the changes occur on local groups of pixels.

In all those cases, the resulting counterfactual instances can be regarded as adversarial examples, in that they are visually hard to tell apart from the source instance, and not representative of the target class. The realism of the counterfactual instance can be improved by searching only over the training & test set instances (ℓ_2 distance). In this case, the counterfactual image is clearly recognizable as being from the target class, but the distance is far larger. Finding realistic counterfactual instances for images is a difficult, open problem [11, 33].

5 Conclusion

Classification trees are very important in applications such as business, law and medicine, where counterfactual explanations are of particular relevance. We have given an exact, efficient algorithm to compute counterfactual explanations for axis-aligned and oblique trees in multiclass problems, with different distances and constraints, and applicable to both continuous and categorical features. The algorithm is fast enough to allow interactive use. It should be possible to extend it to other cases, such as softmax classifier leaves (rather than constant-label leaves) and regression trees.

6 Acknowledgments

Work partially supported by NSF award IIS-2007147.

A Details about the experiments

A.1 Dataset information

In this section we describe the datasets used, in detail. All datasets are from UCI [39] except MNIST.

Adult It is a dataset with mixed type attributes. The prediction task is to determine whether a person makes over 50K a year. There are 12 attributes, out of which 4 are continuous, and the rest are categorical. In all our experiments we convert each categorical attribute to one-hot encoding attribute. Thus each instance has 102 attributes. There are 30 162 training instances, and separate 15 062 test instances. In table 5, we explain the attributes in detail.

Breast-Cancer The task is to classify whether the cancer is malignant or benign. There are 699 instances and each instance has 9 real-valued attributes. Since there is no separate test dataset, we randomly divide the entire data into training (80%) and test (20%).

Spambase This dataset consists of a collection of emails, and the task is to create a spam-filter that can tell whether an email is a spam or not. There are 4 601 instances and each instance has 56 real-valued attributes. Since there is no separate test dataset, we randomly divide the entire data into training (80%) and test (20%).

Letter The objective of this dataset is to classify 26 capital letters in the English alphabet. It has separate 5 000 test instances along with 15 000 training instances. Each instance has 16 real-valued attributes. The character images were based on 20 different fonts and each letter within these 20 fonts was randomly distorted to produce a file of 20 000 unique stimuli. Each stimulus was converted into 16 primitive numerical attributes (statistical moments and edge counts) which were then scaled to fit into a range of integer values from 0 through 15.

German Credit Similar to the Adult dataset, this dataset also has mixed type attributes. 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 continuous and rest are categorical. Similarly to the Adult dataset, we convert each categorical attribute to one-hot encoding attribute. Thus each

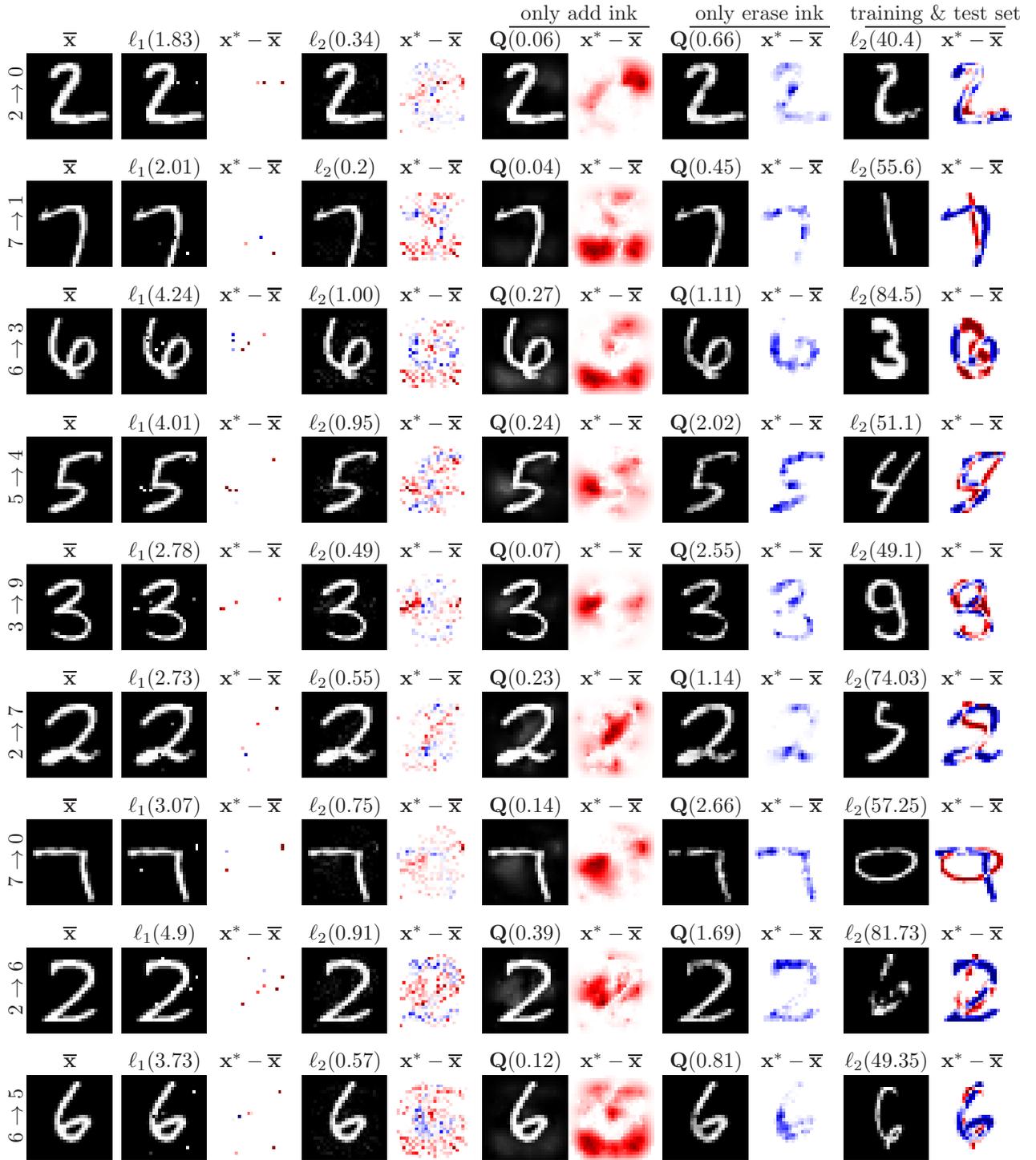


Figure 4: MNIST handwritten digit image counterfactuals using an oblique classification tree. Each row is a different source instance and target class (indicated at left, e.g. “2 → 0” means from class 2 to class 0). For each distance, we show the resulting counterfactual instance x^* (as a grayscale image) and its difference with the source instance $x^* - \bar{x}$ (red: positive, blue: negative, white: zero; rescaled within each image to $[-1, +1]$). We report the value of the distance in parenthesis. Zoom in to see details.

Dataset	D	Feature type	K
MNIST	784	Continuous	10
Adult	102	Continuous and categorical	2
Breast-Cancer	9	Continuous	2
Spambase	57	Continuous	2
Letter	16	Continuous	26
German Credit	61	Continuous and categorical	2

Table 4: Short description of the datasets used in the paper.

instance of the dataset has 61 attributes. Unlike in the Adult dataset, no separate test set is available, so we randomly divide the 1000 instances into training (80%) and test (20%).

MNIST The dataset consists of grayscale images of handwritten digits and the task is to classify them as 0 to 9. There are 60000 training images and 10000 test images. Each image is of size 28×28 with gray scales in $[0,1]$.

A.2 Details about the constraints

Below we describe the constraints used in tables 2-3.

Adult For the second row, we constrain “race” and “sex” to not change. Although they are two attributes, as a one-hot encoding these two attributes form 7 (7%) attributes. For the third row along with “race” and “sex” we also fix the “marital-status”. Thus in total 14 (14%) attributes are fixed.

Breast Cancer For the second row, we randomly select one (11%) attribute index and for a given input we fix the attribute value at that index. For the third row we again select one more attribute index at random and fix the attribute value at those indexes (including one from the second row too), so in total 2 (22%) attributes are fixed.

Spambase For the second row, we randomly select 10 (17%) attribute indexes and for a given input we fix the attribute values at those indexes. For the third row we select 20 more attribute indexes at random and fix the attribute value at those indexes (including one from the second row too), so in total 30 (53%) attributes are fixed.

Letter For the second row, we randomly select 4 (25%) attribute indexes and for a given input we fix the attribute values at those indexes. For the third row we select 6 more attribute indexes at random and fix the attribute value at those indexes (including one from the second row too), so in total 10 (62%) attributes are fixed.

German Credit For the second row, we constrain “sex” and “status” to not change. Although they are two attributes, as a one-hot encoding these two attribute form 4 (7%) attributes. For the third row along with “sex” and “status” we also fix “Credit history”. Thus in total 9 (14%) attributes are fixed.

MNIST For the second row, we fix those pixels which are always zero in the entire training dataset. There are 69 (9%) such pixels, we constrain them to be 0. For the third row along with 69 pixels we randomly select 200 more pixels and fix them too, so in total 269 (47%) attributes are constrained.

A.3 Details about the algorithms

All our algorithms and experiments are implemented in Python. We train CART and Random Forest using Scikit-learn (version 0.22). For CART we train by first letting the tree grow full, and then prune it back with optimal pruning parameter. We train Random Forest using the default parameters in Scikit-learn. We train oblique trees using TAO (ran for 60 iterations at most), also implemented in Python. For oblique trees, we initialize the tree with random parameters (weights and biases) and a user-set depth (mentioned in table 6).

Feature name	Feature type	Explanation
age	Continuous	range: 17 to 90
workclass	Categorical	7 categories Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked
education	Categorical	17 categories Bachelors, Some-college, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 11th, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
marital-status	Categorical	7 categories Married-civ-spouse, Divorced, Widowed, Separated, Never-married, Married-spouse-absent Married-AF-spouse.
occupation	Categorical	14 categories Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.
relationship	Categorical	6 categories Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
race	Categorical	5 categories White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
sex	Categorical	2 categories Female and Male.
capital-gain	Continuous	range: 0 to 99999
capital-loss	Continuous	range: 0 to 4356
hours-per-week	Continuous	range: 1 to 99
native-country	Categorical	41 countries
income (classes)	Categorical	1) less than 50K 2) greater than equal to 50k

Table 5: Feature explanation for the Adult dataset.

Dataset	CART			Oblique (TAO)			Random Forest	
	Train error	Test error	Depth	Train error	Test error	Depth	Train error	Test error
MNIST	4.2	11.9	19	1.4	5.2	9	0	3.1
Adult	12.6	14.7	12	13.9	14.6	12	13.2	14.1
Breast-Cancer	1.4	2.6	4	1.2	2.1	3	0.01	2.1
Spambase	3.1	7.6	10	4.5	4.8	5	0	4.2
Letter	1.2	12.1	25	3.8	7.7	12	0.0	3.7
German Credit	21.9	23.5	7	17.5	18.5	8	0.0	21.9

Table 6: Test/training error of oblique trees (TAO), axis-aligned trees (CART) and Random Forest over various datasets.

For each tree we use same ℓ_1 parameter which is equal to 10. For solving the quadratic and linear programs, we use Gurobi Python interface (version 9.0). We use Gurobi’s `Mvar` type variables to implement all our problems.

References

- [1] C. C. Aggarwal. *Data Mining. The Textbook*. Springer-Verlag, 2015.
- [2] A. Bella, C. Ferri, J. Hernández-Orallo, and M. J. Ramírez-Quintana. Using negotiable features for prescription problems. *Computing*, 91:135–168, Feb. 2011.
- [3] R. E. Bixby. A brief history of linear and mixed-integer programming computation. *Documenta Mathematica*, Extra Volume: Optimization Stories:107–121, 2012.
- [4] L. J. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, Calif., 1984.
- [5] M. Á. Carreira-Perpiñán. The Tree Alternating Optimization (TAO) algorithm: A new way to learn decision trees and tree-based models. arXiv, 2021.
- [6] M. Á. Carreira-Perpiñán and S. S. Hada. Counterfactual explanations for oblique decision trees: Exact, efficient algorithms. In *Proc. of the 35th AAAI Conference on Artificial Intelligence (AAAI 2021)*, Online, Feb. 2–9 2021.
- [7] M. Á. Carreira-Perpiñán and S. S. Hada. Inverse classification with logistic and softmax classifiers: Efficient optimization. arXiv, 2021.
- [8] M. Á. Carreira-Perpiñán and P. Tavallali. Alternating optimization of decision trees, with application to learning sparse oblique trees. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NEURIPS)*, volume 31, pages 1211–1221. MIT Press, Cambridge, MA, 2018.
- [9] M. Á. Carreira-Perpiñán and A. Zharmagambetov. Ensembles of bagged TAO trees consistently improve over random forests, AdaBoost and gradient boosting. In *Proc. of the 2020 ACM-IMS Foundations of Data Science Conference (FODS 2020)*, pages 35–46, Seattle, WA, Oct. 19–20 2020.
- [10] Z. Cui, W. Chen, Y. He, and Y. Chen. Optimal action extraction for random forests and boosted trees. In *Proc. of the 21st ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (SIGKDD 2015)*, pages 179–188, Sydney, Australia, Aug. 10–13 2015.
- [11] A. Dhurandhar, P.-Y. Chen, R. Luss, C.-C. Tu, P. Ting, K. Shanmugam, and P. Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NEURIPS)*, volume 31, pages 592–603. MIT Press, Cambridge, MA, 2018.

- [12] A. Dosovitskiy and T. Brox. Inverting visual representations with convolutional networks. In *Proc. of the 2016 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'16)*, Las Vegas, NV, June 26 – July 1 2016.
- [13] A. A. Freitas. Comprehensible classification models: A position paper. *SIGKDD Explorations*, 15(1): 1–10, June 2014.
- [14] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *Proc. of the 3rd Int. Conf. Learning Representations (ICLR 2015)*, San Diego, CA, May 7–9 2015.
- [15] B. Goodman and S. Flaxman. European Union regulations on algorithmic decision-making and a “right to explanation”. *AI Magazine*, 38(3):50–57, Fall 2017.
- [16] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi. A survey of methods for explaining black box models. *ACM Computing Surveys*, 51(5):93, May 2018.
- [17] Gurobi Optimization, LLC. Gurobi optimizer reference manual, 2019.
- [18] S. S. Hada and M. Á. Carreira-Perpiñán. Sampling the “inverse set” of a neuron: An approach to understanding neural nets. arXiv:1910.04857, Sept. 27 2019.
- [19] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, third edition, 2011.
- [20] D. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining*. Adaptive Computation and Machine Learning Series. MIT Press, Cambridge, MA, 2001.
- [21] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(2):181–214, Mar. 1994.
- [22] A.-H. Karimi, G. Barthe, B. Balle, and I. Valera. Model-agnostic counterfactual explanations for consequential decisions. arXiv:1905.11190, Oct. 8 2019.
- [23] Z. C. Lipton. The mythos of model interpretability. *Comm. ACM*, 81(10):36–43, Oct. 2018.
- [24] A. Mahendran and A. Vedaldi. Visualizing deep convolutional neural networks using natural pre-images. *Int. J. Computer Vision*, 120(3):233–255, Dec. 2016.
- [25] D. Martens and F. Provost. Explaining data-driven document classifications. *MIS Quarterly*, 38(1): 73–99, Mar. 2014.
- [26] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
- [27] C. Russell. Efficient search for diverse coherent explanations. In *Proc. ACM Conf. Fairness, Accountability, and Transparency (FAT 2019)*, pages 20–28, Atlanta, GA, Jan. 29–31 2019.
- [28] S. Sharma, J. Henderson, and J. Ghosh. CERTIFAI: Counterfactual explanations for robustness, transparency, interpretability, and fairness of artificial intelligence models. arXiv:1905.07857, May 20 2019.
- [29] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Proc. of the 2nd Int. Conf. Learning Representations (ICLR 2014)*, Banff, Canada, Apr. 14–16 2014.
- [30] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *Proc. of the 2nd Int. Conf. Learning Representations (ICLR 2014)*, Banff, Canada, Apr. 14–16 2014.
- [31] G. Tolomei, F. Silvestri, A. Haines, and M. Lalmas. Interpretable predictions of tree-based ensembles via actionable feature tweaking. In *Proc. of the 23rd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (SIGKDD 2017)*, pages 465–474, Halifax, Nova Scotia, Aug. 13–17 2017.

- [32] B. Ustun, A. Spangher, and Y. Liu. Actionable recourse in linear classification. In *Proc. ACM Conf. Fairness, Accountability, and Transparency (FAT 2019)*, pages 10–19, Atlanta, GA, Jan. 29–31 2019.
- [33] A. Van Looveren and J. Klaise. Interpretable counterfactual explanations guided by prototypes. arXiv:1907.02584, July 3 2019.
- [34] S. Wachter, B. Mittelstadt, and C. Russell. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harvard J. Law & Technology*, 31(2):841–887, Spring 2018.
- [35] J. Wexler, M. Pushkarna, T. Bolukbasi, M. Wattenberg, F. Viégas, and J. Wilson. The What-If Tool: Interactive probing of machine learning models. *IEEE Trans. Visualization and Computer Graphics*, 26(1):56–65, Jan. 2020.
- [36] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, fourth edition, 2016.
- [37] Q. Yang, J. Yin, C. X. Ling, and R. Pan. Extracting actionable knowledge from decision trees. *IEEE Trans. Knowledge and Data Engineering*, 18(1):43–56, Jan. 2006.
- [38] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Proc. 13th European Conf. Computer Vision (ECCV'14)*, pages 818–833, Zürich, Switzerland, Sept. 6–12 2014.
- [39] C. Zhang, C. Liu, X. Zhang, and G. Almpanidis. An up-to-date comparison of state-of-the-art classification algorithms. *Expert Systems with Applications*, 82:128–150, Oct. 1 2017.
- [40] A. Zharmagambetov and M. Á. Carreira-Perpiñán. Smaller, more accurate regression forests using tree alternating optimization. In H. Daumé III and A. Singh, editors, *Proc. of the 37th Int. Conf. Machine Learning (ICML 2020)*, pages 11398–11408, Online, July 13–18 2020.
- [41] A. Zharmagambetov and M. Á. Carreira-Perpiñán. Learning a tree of neural nets. In *Proc. of the IEEE Int. Conf. Acoustics, Speech and Sig. Proc. (ICASSP'21)*, Toronto, Canada, Mar. 21–25 2021.
- [42] A. Zharmagambetov, S. S. Hada, M. Á. Carreira-Perpiñán, and M. Gabidolla. An experimental comparison of old and new decision tree algorithms. arXiv:1911.03054, Mar. 20 2020.