Counterfactual Maps: What They Are and How to Find Them

Awa Khouna, Julien Ferry, Thibaut Vidal

CIRRELT & SCALE-AI Chair in Data-Driven Supply Chains,
Department of Mathematics and Industrial Engineering, Polytechnique Montréal, Canada

Introduction

What is the smallest change that would reverse a model's decision? A counterfactual explanation answers this question by showing, for example, how a rejected loan application could be modified to obtain approval from the model.

The gap. Exact methods solve a new optimization problem for each query. Heuristics may return suboptimal changes or fail to find a valid answer.

Our idea: turn counterfactual search into nearest-region search over a reusable geometric representation of the model.

Counterfactual Maps

A tree ensemble $\mathcal{T}$ partitions the input space into labeled, axis-aligned boxes $\mathbb{H}$. Its prediction is constant within each box. For a desired class y', let

$$\mathbb{H}_{y'} = \{\mathcal{H} \in \mathbb{H} : \mathcal{T}(\mathcal{H}) = y'\}.$$

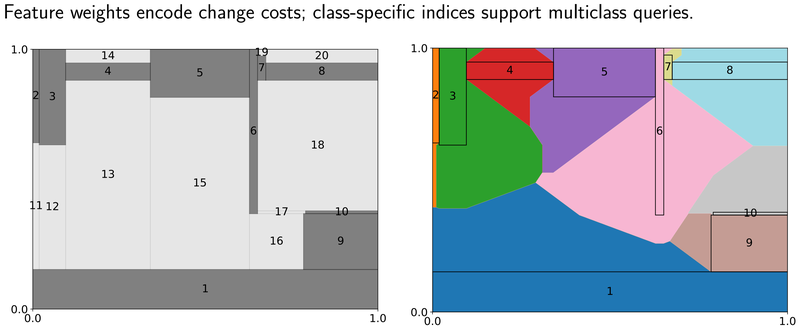
Nearest region, then projection. The closest counterfactual is the closest point in a nearest target-class box:

$$\mathcal{H}^\star \in \arg\min_{\mathcal{H}\in\mathbb{H}_{y'}} d_p(x,\mathcal{H}), \qquad x_{\text{cf}} = \hat{f}_{y'}(x) := \Pi_{\mathcal{H}^\star}(x).$$

Here $d_p(x,\mathcal{H})$ is the distance to a box. For $\mathcal{H} = \prod_j [a_j, b_j]$, projection clips each feature:

$$(x_{\text{cf}})_j = \min\{b_j, \max\{a_j, x_j\}\}.$$

Feature weights encode change costs; class-specific indices support multiclass queries.



(a) Model's decision regions **(b) Counterfactual map (L_1)**

Two depth-5 trees. Dark gray in (a) is the target class. Each color in (b) identifies its nearest target-class box: e.g., green inputs have a closest target-class counterfactual in box 3. Axes are input features.

A counterfactual map is a generalized Voronoi diagram: each input is assigned to its nearest target-class region.

Metric-dependent boundaries: piecewise polyhedral for L_1 and L_∞; potentially curved for L_2. The search index represents this map *implicitly*.

Contributions

1. We introduce **counterfactual maps**, a geometric representation of globally minimal recourse for tree ensembles.
2. We develop an **exact nearest-region algorithm** with reusable preprocessing, achieving substantially faster queries than existing exact methods.
3. We propose **budgeted map construction** with valid counterfactuals and explicit bounds on suboptimality.

References.
[1] Vidal, Pacheco & Schiffer. *Born-Again Tree Ensembles.* ICML 2020.
[2] Parmentier & Vidal. *Optimal Counterfactual Explanations in Tree Ensembles.* ICML 2021.
[3] Zhang & Zhong. *Efficient and Effective Counterfactual Explanations for Random Forests.* Expert Systems with Applications, 2025.
[4] Carreira-Perpiñán & Hada. *Very Fast, Approximate Counterfactual Explanations for Decision Forests.* AAAI 2023.
[5] Mothilal, Sharma & Tan. *Explaining Machine Learning Classifiers through Diverse Counterfactual Explanations.* FAT* 2020.
[6] Tolomei et al. *Interpretable Predictions of Tree-Based Ensembles via Actionable Feature Tweaking.* KDD 2017.

Methodology

Offline: build a faithful map.

1. Extract a complete partition $\mathbb{H}$ with **BA-Trees** [1], preserving every ensemble prediction.
2. Build a **volumetric KD-tree** for each target class. Each internal node stores a bounding box $B(v)$ containing all its descendant regions.

Online: search only promising regions.

1. Start with $U = \infty$. Visit queued nodes in increasing lower bound $\ell(v) = d_p(x, B(v))$.
2. At a leaf, evaluate its target region and update U. Otherwise, queue children whose bounds are below U.
3. Stop when the queue is empty or its smallest bound is $\geq U$; project onto the best region.

Whole groups of boxes can be skipped without losing the optimum.

Theoretical Guarantees

Safe pruning (KD-tree exploration at query time). For each KD-tree node v, its bounding box $B(v)$ contains all descendant regions. Thus,

$$\ell(v) := d_p(x, B(v)) \leq d_p(x, \mathcal{H}), \quad \forall\, \mathcal{H} \text{ below } v.$$

If $\ell(v) \geq U$, where U is the best distance found so far, no descendant can improve the solution. The entire subtree can therefore be safely pruned.

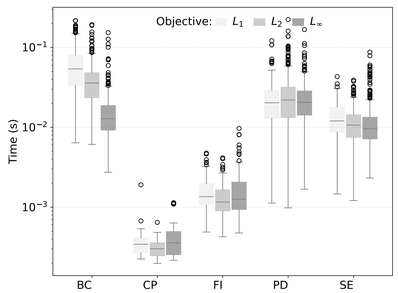
Global optimality. Given a complete, *faithful* partition of the feasible domain, the search finds a nearest target-class region. Projection onto that region yields a globally minimal counterfactual under L_p distance, $1 \leq p \leq \infty$.

Cost. Practical search visits a small fraction of regions, but worst-case search is linear in their number. Partition size and preprocessing can grow exponentially.

Experimental Analyses

Setup. Five datasets, five seeds, 1,000 counterfactual queries per tree ensemble. Ensemble sizes: 5–100 trees; depths: 2–7. Results below use **100 depth-5 trees**.

Datasets: BC: Breast Cancer; CP: COMPAS; FI: FICO; PD: Pima Diabetes; SE: Seeds (multiclass).
Baselines: exact : OCEAN [2]; heuristics : EECE [3], LIRE [4], DiCE [5] and FT [6].



Fast across norms. Millisecond-level query time for L_1, L_2 and L_∞.

Trend: L_∞ queries are generally fastest, followed by L_2 and L_1, reflecting metric-dependent search behavior.

1.4–45.2 ms mean L_1 query times across datasets.

Query time excludes preprocessing; log scale. Dataset differences reflect dimensionality and split thresholds, which affect partition size and search depth.

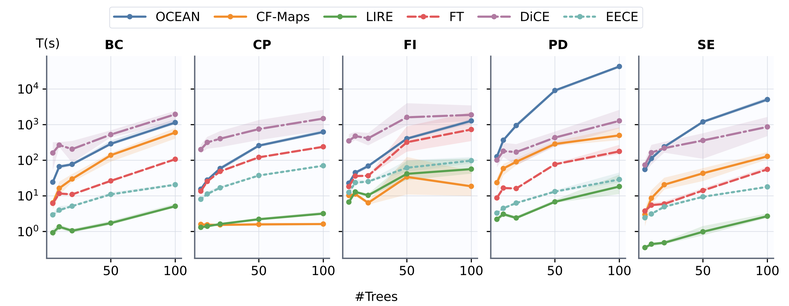
Exact vs. heuristic methods. 1,000 L_1 queries; times in seconds.

	OCEAN		**CF-Maps**		EECE		LIRE		DiCE			FT		
	T_P	T_{1000}	T_P	T_{1000}	T_{1000}	D	T_{1000}	D	T_{1000}	D	%F	T_{1000}	D	%F
BC	3.0	1,143	557	**45.2**	19.8	1.2	1.3	2.3	1,951	2.8	2.2	107	1.6	93.2
CP	5.0	619	0.2	**1.4**	67.9	1.0	1.2	1.1	1,483	1.0	15.1	239	1.0	41.4
FI	5.9	1,273	0.9	**17.6**	64.1	1.3	23.6	1.3	1,885	1.3	16.6	732	1.0	63
PD	13.9	42,983	467	**32.1**	28.7	1.1	2.4	2.4	1,278	4.4	1.0	178	3.2	79.3
SE	2.9	5,069	120	**8.5**	17.9	2.3	0.5	3.4	866	4.8	1.0	55.6	1.9	78.5

T_P: preprocessing (offline); T_{1000}: time to compute 1,000 counterfactuals (online); D: ratio distance / optimum, on valid answers; %F: failures. For both OCEAN and CF-Maps, by design, $D = 1$, %F=0; For EECE and LIRE, by design, %F=0. Other preprocessing: EECE 2.2–33.6 s, LIRE 2.0–33.0 s. **Bold:** Best online time across exact methods.

Experimental Analyses (continued)

Figure 5: what happens as tree ensembles grow?



Total time *including preprocessing* for 1,000 counterfactuals versus number of depth-5 trees. Shading: fastest–slowest of five seeds; log time axis.

Trend and takeaway. More trees generally create more decision regions, making CF-Maps preprocessing more expensive while queries remain fast. OCEAN's per-query optimization grows costly; LIRE and EECE stay fast but sacrifice distance quality.

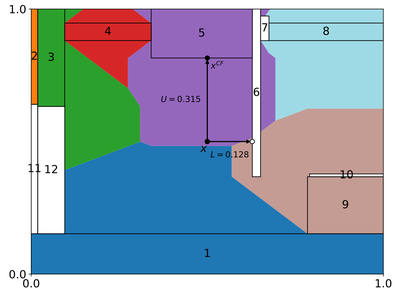
Budgeted CF-Maps: scale with certificates

The bottleneck is building the complete map. CF-Maps(B) stops after certifying N regions; it indexes known target boxes and retains unresolved space $\mathcal{U}$. This results in one KD-tree per class, plus an additional KD-tree for unresolved regions.

Upper bound U: Distance to the nearest certified target-class region (if any).
Unresolved-space bound $L_{\mathcal{U}}$: Minimum distance to any region whose class is not yet certified.

$$L_{\mathcal{U}} = \min_{R\in\mathcal{U}} d_p(x, R),$$
$$L = \min\{U, L_{\mathcal{U}}\} \leq d^*(x) \leq U.$$

Certificate: If $L = U$, the counterfactual is globally optimal; otherwise, $U - L$ bounds its suboptimality.



Illustrative example: L is tight, as unresolved region #6 belongs to the target class (not guaranteed in general).

Scaling result: 1,000 depth-10 trees, budget $N = 10^6$.

	CF-Maps			CF-Maps(B)			EECE		
	T_P [s]	T_q [ms]	D	T_P [s]	T_q [ms]	D	T_P [s]	T_q [ms]	D
BC	6633.01	34.4	**1.0**	180.09	**23.1**	**1.0**	39.64	384.1	1.11
PD	2198.57	13.2	**1.0**	302.58	**7.1**	**1.0**	130.28	1494.2	1.05
SE	322.11	9.2	**1.0**	21.70	**3.6**	**1.0**	24.22	393.8	1.21

T_P: preprocessing time; T_q: mean query time; D: the distance ratio relative to the optimal distance. Five-seed averages. All CF-Maps(B) queries in these rows are certified optimal ($L = U$); smaller budgets can leave gaps.

Preprocessing time reduced by up to 97.3%, with every tested query certified optimal.

Conclusion and Takeaway

Counterfactual optimization can be replaced by reusable geometric search. CF-Maps provides globally optimal counterfactuals for tree ensembles with millisecond-level queries.

CF-Maps(B) goes further: an incomplete map can still produce valid counterfactuals with certified optimality gaps, substantially reducing preprocessing.