S-CFE: Simple Counterfactual Explanations

Shpresim Sadiku[1,2] **Moritz Wagner**[1,2] **Sai Ganesh Nagarajan**[1] **Sebastian Pokutta**[1,2]

[1]Zuse Institute Berlin
[2]Technische Universität Berlin

Abstract

We study the problem of finding optimal sparse, manifold-aligned counterfactual explanations for classifiers. Canonically, this can be formulated as an optimization problem with multiple non-convex components, including classifier loss functions and manifold alignment (or *plausibility*) metrics. The added complexity of enforcing *sparsity*, or shorter explanations, complicates the problem further. Existing methods often focus on specific models and plausibility measures, relying on convex ℓ_1 regularizers to enforce sparsity. In this paper, we tackle the canonical formulation using the accelerated proximal gradient (APG) method, a simple yet efficient first-order procedure capable of handling smooth non-convex objectives and non-smooth ℓ_p (where $0 \leq p < 1$) regularizers. This enables our approach to seamlessly incorporate various classifiers and plausibility measures while producing sparser solutions. Our algorithm only requires differentiable data-manifold regularizers and supports box constraints for bounded feature ranges, ensuring the generated counterfactuals remain *actionable*. Finally, experiments on real-world datasets demonstrate that our approach effectively produces sparse, manifold-aligned counterfactual explanations while maintaining proximity to the factual data and computational efficiency.

Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258. Copyright 2025 by the author(s).

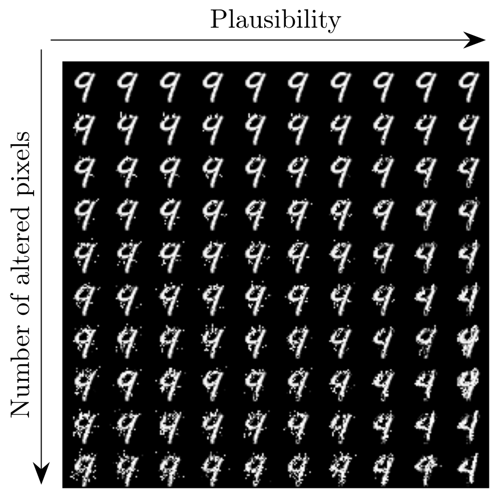

Figure 1: Examples of possible CFEs for an input image of the digit 9 when changing the classification to 4: Sparsity constraints alone produce adversarial examples, while plausibility constraints lead to unrealistic CFEs. Combining both yields CFEs that are sparse yet aligned with the target class 4's data manifold.

1 Introduction

Machine learning models are increasingly deployed in critical decision-making scenarios, from finance and healthcare to criminal justice and hiring. While these classifiers can be highly accurate, their decision-making processes are often opaque, raising concerns about transparency, fairness, and accountability. Counterfactual explanations (CFEs) have emerged as powerful tools to provide insights into a classifier's decision-making process by offering hypothetical "what-if" scenarios. Unlike explanation methods like LRP [Bach et al., 2015] and LIME [Ribeiro et al., 2016], which identify the minimal set of features contributing to the current classification, CFEs focus on detecting the minimal set of absent features whose presence would change the classification [Wachter et al., 2017].

Basic Principles. In essence, a CFE suggests small changes to input features *(Proximity)* that could lead to

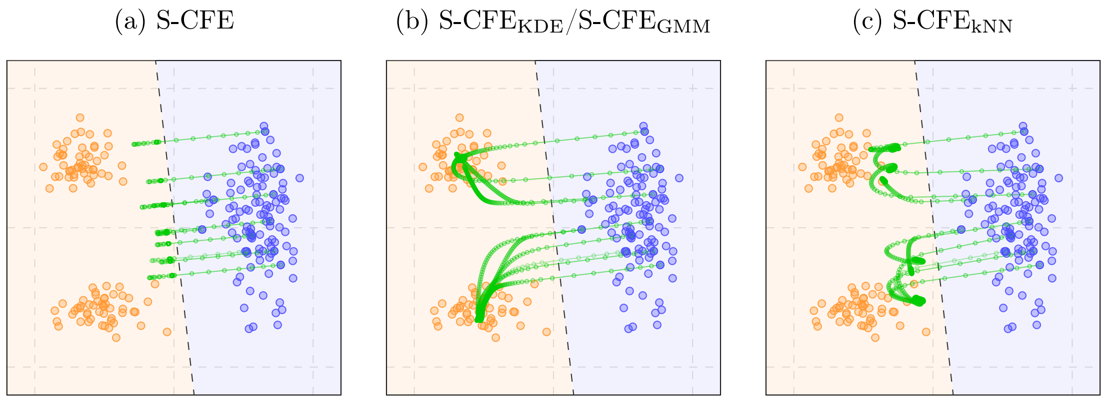

Figure 2: A simple dataset illustrates the need for a plausibility term in CFE algorithms. (a) Our S-CFE method without a plausibility term generates CFEs near the factual blue data points, but they remain distant from the distribution of correctly classified orange data points. (b) Our S-CFE$_{\text{KDE}}$ method, which combines S-CFE with a KDE-based plausibility term, produces CFEs within high-density regions. (c) Similarly, S-CFE$_{\text{kNN}}$, combining S-CFE with a $k-$NN-based plausibility term generates CFEs near the boundary of high-density regions. The green trajectory connecting the green data points represents the iterates of our S-CFE algorithm. The dashed black line represents the decision boundary of a linear classifier.

a different, more favorable outcome *(Validity)*. These changes must be *Actionable*, meaning they should apply only to valid feature ranges and avoid unrealistic suggestions. For example, a CFE in a loan application scenario should not propose that Alice reduce her age by ten years.

Plausibility. The basic principles of CFEs —*Proximity*, *Validity*, and *Actionability* —highlight their similarity to adversarial examples. However, a critical distinguishing conceptual feature lies in their *Plausibility*. While adversarial attacks introduce small changes, known as perturbations, to mislead the classifier into incorrect predictions—often pushing the data point out of its original class distribution—CFEs aim to nudge the data point toward the target class's distribution. This ensures that the explanations are not only effective but also grounded in plausible real-world scenarios. Fig. 2 illustrates this property using a synthetic 2D Gaussian dataset, comparing CFEs generated without a data distribution penalty to those incorporating a plausibility term, ensuring the CFEs align with the target class distribution.

Sparsity. Additionally, CFEs should modify as few features as possible to promote simplicity, a concept known as *Sparsity*. Studies show that shorter explanations are easier for people to understand, making sparsity critical [Mothilal et al., 2020, Naumann and Ntoutsi, 2021]. Combined with the proximity requirement, sparsity implies that feature changes should be minimal and low in magnitude. For instance, rather than suggesting multiple changes, a CFE might recommend that Alice raise her income by just \$10K to shift the loan decision from rejection to approval, as opposed to requiring a \$50K increase or multiple simultaneous feature adjustments.

However, existing methods for generating CFEs often struggle to produce results that are both sparse and adhere to the data manifold, leading to unrealistic or impractical suggestions. To better understand the tradeoff between sparsity and plausibility, we provide an illustration in Fig. 1 using an MNIST image [LeCun et al., 1998]. The goal is to minimally alter a 9 to resemble a 4. Sparsity alone produces perturbations similar to adversarial attacks (leftmost column), while plausibility alone leads to unrealistic CFEs (rightmost column). By balancing both, we achieve simple, realistic changes, as shown in the bottom-right examples.

To capture these requirements, one would ideally solve the following optimization problem to find a CFE for a given factual data point $\boldsymbol{x}_f$

$$\begin{aligned} \min_{\boldsymbol{x} \in \text{actionable set}} \quad & \text{counterfactual loss of } \boldsymbol{x} \\ & + \text{dist to } \boldsymbol{x}_f \\ & + \text{dist to data manifold} \\ & + \text{No. of feature changes,} \end{aligned} \tag{1}$$

which we refer to as the *canonical form* of a CFE. The key technical challenge lies in the fact that the objective terms can be *non-convex* and *non-smooth*, due to the use of complex classifiers, intricate distance measures to the data manifold, and sparsity terms, such as those introduced by ℓ_0 regularizers. Additionally, the actionable set imposes constraints on how much the CFE can change. Numerous methods have tackled partial aspects of this problem, often tailoring their approaches to specific classifiers, plausibility, or sparsity constraints. For example, Artelt and Hammer [2020] generate CFEs for simple linear classifiers and decision trees, while Tsiourvas et al. [2024] focus on ReLU neural networks (NNs) using the Local Outlier Factor (LOF) metric as a plausibility constraint. For a comprehensive survey, see [Verma et al., 2024].

Our Contributions:

1. We introduce S-CFE (**S**imple **C**ounter**f**actual **E**xplanations), a *simple* method to solve the canonical formulation in Eq. (1), using the Accelerated Proximal Gradient (APG) method [Beck and Teboulle, 2009].
2. Our method is capable of handling smooth nonconvex objectives and non-smooth ℓ_p (where $0 \leq p < 1$) regularizers. This enables our approach to seamlessly incorporate various classifiers and plausibility measures while producing *sparser* solutions, that are *actionable*.
3. Extensive evaluations on real-world datasets show significant improvements over existing methods, particularly in sparsity and adherence to the data manifold, which is crucial for generating meaningful CFEs.

In summary, our proposed method addresses the key challenges in generating CFEs, offering a practical and effective tool for enhancing the interpretability of classifiers. This advancement is particularly valuable in safety-critical domains, where understanding and trusting classifier decisions is essential.

2 Related Work

CFEs have seen growing interest in recent years [Wachter et al., 2017, Verma et al., 2024, Karimi et al., 2020]. Most methods enforce sparsity by optimizing weighted Manhattan or Mahalanobis distances between the factual data and the generated CFE. Only recently has the importance of plausibility in CFEs been recognized, prompting a focus on methods addressing both sparsity and plausibility. For DNN-based classifiers, Dhurandhar et al. [2018] were among the first to frame this as an unconstrained problem using the ℓ_1 norm for sparsity and VAEs trained on the data distribution for plausibility. Building on Van Looveren and Klaise [2021], Zhang et al. [2023] replace autoencoders with a density-based distance term for plausibility, while adopting the elastic-net regularizer [Zou and Hastie, 2005] for sparsity. They also highlight that autoencoder-based approaches often struggle with data quality issues, undermining the robustness and credibility of CFEs. Artelt and Hammer [2020] formulate a constrained optimization problem, using Gaussian Mixture Models (GMMs) to ensure high target-class density, with ℓ_1-distance enforcing sparsity. The nonconvex GMM problem is approximated by solving convex quadratic subproblems for each GMM component, leveraging simple classifiers like generalized linear classifiers and linear SVMs. The recent paper of Tsiourvas et al. [2024] takes a different approach, formulating a mixed-integer programming (MIP) problem for ReLU networks, using the Local Outlier Factor (LOF) for plausibility and adding a sparsity constraint. The MIP is solved efficiently by restricting the search to polytopes containing the correct class, but this framework is limited to ReLU architectures.

3 Preliminaries

Assume a classification setting where $\mathcal{X} \subseteq \mathbb{R}^d$ denotes the input space, the discrete finite set $\mathcal{Y}$ denotes the set of possible class labels, and $\mathcal{D} = \{(\boldsymbol{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}_{i=1}^n$ is a dataset consisting of n independent and identically distributed data points generated from a joint density $\psi : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}_+$. Furthermore, we define $q(\boldsymbol{x}, y) := \psi(\boldsymbol{x}|y)$, which is the corresponding density of the inputs conditioned on the given label y.

We let $f_l : \mathcal{X} \to \mathbb{R}^{|\mathcal{Y}|}$ denote a classifier that takes a d-dimensional sample as input and outputs logits of $|\mathcal{Y}|$ classes. The final decision is denoted by $f(\boldsymbol{x}) := \arg\max_i [f_l(\boldsymbol{x})]_i$. Furthermore, let $\theta_p : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}_+$ be a distance function on $\mathcal{X}$, such as the one given by ℓ_p-(quasi) norm $\theta_p(\boldsymbol{x}, \boldsymbol{x}') := \|\boldsymbol{x} - \boldsymbol{x}'\|_p^p = \sum_{i=1}^d |x_i - x_i'|^p$, for $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d$ and $0 < p < \infty$. For $p = 0$, we define $\theta_0(\boldsymbol{x} - \boldsymbol{x}') := \|\boldsymbol{x} - \boldsymbol{x}'\|_0$ as the cardinality of the support of $\boldsymbol{x} - \boldsymbol{x}'$. For $d \in \mathbb{N}$ let $[d] = \{1, ..., d\}$.

Definition 3.1 ([Parikh et al., 2014])**.** *The proximal operator with respect to a (possibly non-smooth) function $g : \mathbb{R}^d \to \mathbb{R}$ is defined for any $\boldsymbol{x}' \in \mathbb{R}^d$*

$$\operatorname{prox}_{\lambda g}(\boldsymbol{x}') := \underset{\boldsymbol{x} \in \mathbb{R}^d}{\arg\min} \frac{1}{2\lambda} \theta_2(\boldsymbol{x}, \boldsymbol{x}') + g(\boldsymbol{x}),$$

where $\lambda > 0$ is a given parameter.

The proximal operator is particularly useful for analyzing non-smooth functions g and can often be computed analytically for many such functions.

3.1 Background on CFEs

Definition 3.2. (Closest Data-Manifold CFE). *Given a factual data sample $\boldsymbol{x}_f \in \mathbb{R}^d$ such that $f(\boldsymbol{x}_f) = y_f$, its closest data-manifold CFE (in terms of θ_2) with respect to $f(\cdot)$ and the data manifold of the target class y_{cf} is defined as a point $\boldsymbol{x}_{cf} \in \mathcal{X}$ that is the solution of the following optimization problem*

$$\begin{aligned} \boldsymbol{x}_{cf} := \quad & \underset{\boldsymbol{x}\in\mathcal{X}}{\arg\min}\, \theta_2(\boldsymbol{x}, \boldsymbol{x}_f) \\ s.t. \quad & \boldsymbol{x} \in \mathcal{A} \\ & f(\boldsymbol{x}) = y_{cf} \\ & q(\boldsymbol{x}, y_{cf}) \geq \tau, \end{aligned} \tag{2}$$

where $\mathcal{A} := \bigtimes_{i=1}^{d}[-\mathcal{A}_i, \mathcal{A}_i]$, for $\mathcal{A}_i \in \mathbb{R}$, denotes the value range for features, extracted from the observed dataset or given by the user, and $\tau > 0$ denotes a minimum density at which we consider a sample to lie in the data manifold of the target class y_{cf}.

The objective function in Eq. (2) ensures the *proximity* of the generated CFE, while the three constraints account for *actionability*, *validity*, and *plausibility*. Without the plausibility constraint, the closest CFEs often deviate significantly from the data manifold (cf. Fig. 2), leading to unrealistic and anomalous instances. This underscores the importance of the plausibility term, which ensures that the generated CFEs remain on the data manifold.

To find the closest data-manifold CFEs while altering as few features as possible, ensuring *sparsity* of the generated CFEs, the non-smooth constrain $\theta_0(\boldsymbol{x}, \boldsymbol{x}_f) \leq m$ can be added to Eq. (2) as follows.

Definition 3.3. (Closest Sparse Data-Manifold CFE). *Given a factual data sample $\boldsymbol{x}_f \in \mathbb{R}^d$ such that $f(\boldsymbol{x}_f) = y_f$, its closest sparse data-manifold CFE with respect to $f(\cdot)$ and the data manifold of the target class y_{cf} is defined as a point $\boldsymbol{x}_{cf} \in \mathcal{X}$ that is the solution of the following optimization problem*

$$\begin{aligned} \boldsymbol{x}_{cf} := \quad & \underset{\boldsymbol{x}\in\mathcal{X}}{\arg\min}\, \theta_2(\boldsymbol{x}, \boldsymbol{x}_f) \\ s.t. \quad & \boldsymbol{x} \in \mathcal{A} \\ & f(\boldsymbol{x}) = y_{cf} \\ & q(\boldsymbol{x}, y_{cf}) \geq \tau \\ & \theta_0(\boldsymbol{x}, \boldsymbol{x}_f) \leq m, \end{aligned} \tag{3}$$

where $m \in \mathbb{N}$ is a parameter to explicitly control the sparsity of the generated CFE.

While for the validity constraint in Eq. (3), common measures such as checking if the CFE belongs to the target class can be used, plausibility can be assessed with a wider range of metrics. Techniques like k-nearest neighbors, kernel density, and Mahalanobis distance estimate whether a sample aligns with the data distribution. However, a widely used metric in the CFE literature (e.g., [Zhang et al., 2023, Hamman et al., 2023, Tsiourvas et al., 2024]) for assessing the similarity or anomalous nature of a generated CFE relative to the dataset $\mathcal{D} \subseteq \mathcal{X}$ is the Local Outlier Factor [Breunig et al., 2000].

Definition 3.4 (Local Outlier Factor (LOF))**.** *For $\boldsymbol{x} \in \mathcal{D}$, let $N_k(\boldsymbol{x}) = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_k\}$ be the set of $k-$Nearest Neighbors ($k-$NN) in $\mathcal{D}$. The $k-$reachability distance rd_k of $\boldsymbol{x}$ with respect to $\boldsymbol{x}'$ is defined by $\mathrm{rd}_k(\boldsymbol{x}, \boldsymbol{x}') = \max\{\|\boldsymbol{x} - \boldsymbol{x}'\|_p, d_k(\boldsymbol{x}')\}$, where $d_k(\boldsymbol{x}')$ is the ℓ_p-norm distance between $\boldsymbol{x}'$ and its $k-$th nearest instance in $\mathcal{D}$. The $k-$local reachability of $\boldsymbol{x}$ is defined by $\mathrm{lrd}_k(\boldsymbol{x}) = |N_k(\boldsymbol{x})|(\sum_{\boldsymbol{x}_i \in N_k(\boldsymbol{x})} \mathrm{rd}_k(\boldsymbol{x}, \boldsymbol{x}_i))^{-1}$. Then, the $k-$LOF of $\boldsymbol{x}$ on $\mathcal{D}$ is defined as*

$$LOF_{k,\mathcal{D}}(\boldsymbol{x}) = \frac{1}{|N_k(\boldsymbol{x})|} \sum_{\boldsymbol{x}_i \in N_k(\boldsymbol{x})} \frac{\mathrm{lrd}_k(\boldsymbol{x}_i)}{\mathrm{lrd}_k(\boldsymbol{x})}.$$

We consider $p \in \{1, 2, \infty\}$ and use LOF as a *post-process* evaluation metric to measure whether the learned closest sparse CFE follows the data manifold. By convention, a value of $LOF_{k,\mathcal{D}}(\boldsymbol{x})$ close to 1 indicates that $\boldsymbol{x}$ is an inlier that is aligned with the data manifold, while larger values (especially $LOF_{k,\mathcal{D}}(\boldsymbol{x}) > 1.5$) indicate that $\boldsymbol{x}$ is an outlier.

4 A Simple Algorithm for Generating CFEs

There are two main issues with solving Eq. (3) in practice.

1. Setting the sparsity constraint aside for the moment, the first problem arises from the fact that the conditional distribution $q(\cdot, y)$ underlying the data $\mathcal{D}$ is often unknown. One possible solution is to incorporate plausibility constraints based on density estimates or the LOF metric. Density estimators like Gaussian mixture models (GMM) or kernel density estimation (KDE) are a common approach to estimate the density for each class based on training samples (see Appendix A). Then, the plausibility constraint requires the resulting CFE to lie near the data manifold by enforcing $\hat{q}_{KDE}(\boldsymbol{x}, y_{cf}) \geq \tau$, respectively $\hat{q}_{GMM}(\boldsymbol{x}, y_{cf}) \geq \tau$. Similarly, the LOF metric can be used, by requiring $LOF_{k,\mathcal{D}}(\boldsymbol{x}) \leq \nu$, for a user-defined threshold ν. However, this highly non-linear constraint (either using density-based estimates or the LOF metric) results in the density estimator being highly non-convex, thus further exacerbating the known

computational challenges in optimizing non-linear classifiers (e.g., NNs). This is why Artelt and Hammer [2020] solve the problem only for simple linear classifiers and decision trees, leveraging linearized GMMs for plausibility, resulting in convex quadratic subproblems. Tsiourvas et al. [2024] on the other hand, use LOF as a plausibility constraint and approximate the complex mixed-integer optimization (MIP) problem by considering it only for ReLU NNs, known for their piece-wise linear structure [Lee et al., 2019], and solve the MIP only over polytopes consisting of points of the correct class.

2. Enforcing the sparsity constraint in Eq. (3) through the non-smooth θ_0-distance is well known to be an NP-hard problem. Consequently, prior works of Dhurandhar et al. [2018], Artelt and Hammer [2020] and Zhang et al. [2023], mostly relax the sparsity constraint to θ_1 and consider the regularized version of Eq. (3).

4.1 Problem Relaxation and Solution Heuristic

For the actionability constraint in Eq. (3), we can utilize the indicator function such that

$$I_{\mathcal{A}}(\boldsymbol{x}) := \begin{cases} 0, & \text{if } \boldsymbol{x} \in \mathcal{A}, \\ +\infty, & \text{otherwise.} \end{cases} \tag{4}$$

To address the combinatorial optimization problem in Eq. (3) we make use of the following relaxations. First, we formulate the closest sparse data-manifold CFE problem by replacing the validity, plausibility, and sparsity constraints with penalty terms, where for sparsity instead of the θ_1-distance regularization we consider any sparsity-inducing θ_p-distance as a penalty. We incorporate the box constraints as indicator functions

$$\begin{aligned} \boldsymbol{x}_{cf} := \; & \arg\min_{\boldsymbol{x}\in\mathbb{R}^d} \theta_2(\boldsymbol{x}, \boldsymbol{x}_f) + I_{\mathcal{A}}(\boldsymbol{x}) + \gamma\mathcal{L}_f(\boldsymbol{x}, y_{cf}) \\ & \quad - \tau\hat{q}(\boldsymbol{x}, y_{cf}) + \beta\theta_p(\boldsymbol{x}, \boldsymbol{x}_f), \end{aligned} \tag{5}$$

where we consider an estimate $\hat{q}(\cdot, y_{cf})$ for the density of target class y_{cf} in $\mathcal{X}$, and $\mathcal{L}_f$ is a suitable (differentiable) classification loss. $\gamma, \tau > 0$ denote tradeoff parameters for validity and plausibility. The only requirement for the plausibility term $\hat{q}(\cdot, y_{cf})$ is to be differentiable so that we are able to learn from its gradient information.

We solve the problem in Eq. (5) by making use of a commonly used algorithm for non-convex and non-smooth programs, based on accelerated proximal gradient (APG) method [Beck and Teboulle, 2009].

We start by denoting $h(\boldsymbol{x}, y_{cf}) := \theta_2(\boldsymbol{x}, \boldsymbol{x}_f) + \gamma\mathcal{L}_f(\boldsymbol{x}, y_{cf}) - \tau\hat{q}(\boldsymbol{x}, y_{cf})$ and $g_p(\boldsymbol{x}) := I_{\mathcal{A}}(\boldsymbol{x}) + \beta\theta_p(\boldsymbol{x}, \boldsymbol{x}_f)$. Assuming $h(\cdot, y_{cf})$ is a smooth, possibly non-convex function, whose gradient has Lipschitz constant L, we make a quadratic approximation $\tilde{h}_L(\boldsymbol{x}, y_{cf})$ to $h(\boldsymbol{x}, y_{cf})$ and replace $\nabla^2 h(\boldsymbol{x}, y_{cf})$ by $\frac{L}{2}I$. Given iterate $\boldsymbol{x}^t$ of the algorithm, it holds that

$$\begin{aligned} \boldsymbol{x}^{t+1} := \; & \arg\min_{\boldsymbol{x}\in\mathbb{R}^d} \tilde{h}_L(\boldsymbol{x}^t, y_{cf}) + g_p(\boldsymbol{x}) \\ = \; & \arg\min_{\boldsymbol{x}\in\mathbb{R}^d} \nabla_{\boldsymbol{x}^t} h(\boldsymbol{x}^t, y_{cf})^\top(\boldsymbol{x} - \boldsymbol{x}^t) + \frac{L}{2}\theta_2(\boldsymbol{x}, \boldsymbol{x}^t) \\ & \quad + g_p(\boldsymbol{x}) \\ = \; & \arg\min_{\boldsymbol{x}\in\mathbb{R}^d} \frac{L}{2}\theta_2\left(\boldsymbol{x}, \boldsymbol{x}^t - \frac{1}{L}\nabla_{\boldsymbol{x}^t} h(\boldsymbol{x}^t, y_{cf})\right) \\ & \quad + g_p(\boldsymbol{x}) \\ = \; & \operatorname{prox}_{\frac{1}{L}g_p}\left(\boldsymbol{x}^t - \frac{1}{L}\nabla_{\boldsymbol{x}^t} h(\boldsymbol{x}^t, y_{cf})\right). \end{aligned} \tag{6}$$

In practice, the inverse Lipschitz constant is further replaced by a step size sequence $(\sigma_t)_{t\in\mathbb{N}}$ [Karimi et al., 2016]. In order to solve the proximal operator in Eq. (6) for the function $g_p(\cdot)$, we first need to compute $\nabla_{\boldsymbol{x}^t} h(\boldsymbol{x}^t, y_{cf})$, which we will denote as $\nabla_{\boldsymbol{x}} h(\boldsymbol{x}, y_{cf})$ for simplicity in the following sections.

4.1.1 Computing $\nabla_{\boldsymbol{x}} h(\boldsymbol{x}, y_{cf})$

As long as we use differentiable terms for proximity, classifier loss function, and the density term, any suitable differentiation technique can be applied to compute $\nabla_{\boldsymbol{x}} h(\boldsymbol{x}, y_{cf})$, and we use backpropagation in our experiments.

As for $\hat{q}(\boldsymbol{x}, y_{cf})$, we adopt traditional differentiable estimates, such as KDE ($\hat{q}_{KDE}(\boldsymbol{x}, y_{cf})$) and GMM ($\hat{q}_{GMM}(\boldsymbol{x}, y_{cf})$). For completeness, we also consider a recently proposed density term based on the LOF metric. Zhang et al. [2023] consider an approximation to LOF given by density gravity on an instance - $G(\boldsymbol{x})$ (see Definition A.3). A suitable plausibility term then minimizes the ℓ_2-distance between the CFE $\boldsymbol{x}$ and the density gravity point $G_{y_{cf}}(\boldsymbol{x}_f)$, generated by a convex combination of $k-$nearest neighbors of the factual data point belonging to the target class y_{cf}, weighted by their local density

$$\hat{q}_{kNN}(\boldsymbol{x}, y_{cf}) := -\left\|\boldsymbol{x} - G_{y_{cf}}(\boldsymbol{x}_f)\right\|_2.$$

The choice of plausibility term ($\hat{q}_{KDE}, \hat{q}_{GMM}$, or $\hat{q}_{kNN}$) leads to different algorithm variants: S-CFE$_{\text{KDE}}$, S-CFE$_{\text{GMM}}$, and S-CFE$_{\text{kNN}}$.

4.2 Computing the Proximal Operator

After computing $\nabla_{\boldsymbol{x}} h(\boldsymbol{x}, y_{cf})$ as outlined in Sec. 4.1.1, we define $S_\sigma(\boldsymbol{x}, y_{cf}) = \boldsymbol{x} - \sigma\nabla_{\boldsymbol{x}} h(\boldsymbol{x}, y_{cf})$. The solution to the proximal operator in Eq. (6) for $p = 0$ is detailed in Zhu et al. [2021]. Notably, our approach is quite general; similar results can be derived for any θ_p, with a straightforward proximal operator (e.g., $p \in \{1/2, 2/3\}$ [Cao et al., 2013]). The method proposed by Zhang et al. [2023] is a specific case using θ_1 and the density gravity term for plausibility.

We summarize our procedure in Algorithm 1. This method generates sparse, manifold-aligned CFEs, but we cannot precisely control the number of features that change.

4.2.1 Constraining the Sparsity

Rather than penalizing sparsity in Eq. (5), we can regularize using the indicator function of the sparsity constraint for improved control. This approach begins with the observation that

$$I_{\theta_p(\boldsymbol{x},\boldsymbol{x}_f)\le m}(\boldsymbol{x}) := \begin{cases} 0, & \text{if } \theta_p(\boldsymbol{x},\boldsymbol{x}_f) \le m, \\ +\infty, & \text{otherwise.} \end{cases}$$

Thus, problem in Eq. (5) can be reframed as

$$\begin{aligned} \boldsymbol{x}_{cf} := \; & \arg\min_{\boldsymbol{x}\in\mathbb{R}^d} \theta_2(\boldsymbol{x},\boldsymbol{x}_f) + I_{\mathcal{A}}(\boldsymbol{x}) + \gamma\mathcal{L}_f(\boldsymbol{x}, y_{cf}) \\ & - \tau\hat{q}(\boldsymbol{x}, y_{cf}) + \beta I_{\theta_p(\boldsymbol{x},\boldsymbol{x}_f)\le m}(\boldsymbol{x}). \end{aligned} \tag{7}$$

Since the new function $g_p(\boldsymbol{x}) := I_{\mathcal{A}}(\boldsymbol{x}) + \beta I_{\theta_p(\boldsymbol{x},\boldsymbol{x}_f)\le m}(\boldsymbol{x})$ is an indicator function, its proximal operator for $p = 0$ coincides with the projection onto the intersection $\{\theta_0(\boldsymbol{x},\boldsymbol{x}_f) \le m\} \cap \mathcal{A}$ of a 0−norm ball and our box constraints. A closed-form solution for this projection was derived by Croce and Hein [2019] and is given by

$$\left[P_{\{\theta_0(\boldsymbol{x},\boldsymbol{x}_f)\le m\}\cap\mathcal{A}}(S_\sigma(\boldsymbol{x}, y_{cf}))\right]_i = \begin{cases} z_i, & \text{if } i \in Q, \\ 0, & \text{otherwise,} \end{cases}$$

$$\boldsymbol{z} = \Pi_{\mathcal{A}}(S_\sigma(\boldsymbol{x}, y_{cf})),$$

$$Q = \operatorname{argtopk}(\boldsymbol{v}, m),$$

where $\boldsymbol{v} = \boldsymbol{w}\odot\boldsymbol{w} - (\boldsymbol{w}-\boldsymbol{z})\odot(\boldsymbol{w}-\boldsymbol{z})$ with $\boldsymbol{w} = \boldsymbol{x} - \boldsymbol{x}_f$ and $\odot$ the element-wise product, argtopk$(\boldsymbol{v}, m)$ represents the indices corresponding to the m largest absolute values of the entries of $\boldsymbol{v}$, and $\Pi_{\mathcal{A}}(\boldsymbol{x}) = \arg\min_{\boldsymbol{x}}\{\|\boldsymbol{x}' - \boldsymbol{x}\|_2 \mid \boldsymbol{x}' \in \mathcal{A}\}$.

Remark 4.1. *Since $g_0(\boldsymbol{x}) := I_{\mathcal{A}}(\boldsymbol{x}) + \beta I_{\theta_0(\boldsymbol{x},\boldsymbol{x}_f)\le m}(\boldsymbol{x})$ is a proper and lower semicontinuous function, the convergence of APG to a critical point of the minimization problem specified in Eq. (7) can be assured (even for non-convex and non-smooth $g_0(\cdot)$), under some mild conditions [Li and Lin, 2015].*

5 Experiments

We conduct experiments on four real-world datasets to validate our methods against various benchmarks.

Algorithm 1 S-CFE: Simple Counterfactual Explanations

```
Input: Classifier $f$, density penalty $\hat{q}$, sparsity penalty $\theta_p$, classification loss function $\mathcal{L}_f$, feature ranges $\mathcal{A}$, original point $\boldsymbol{x}_f \in \mathbb{R}^d$, target label $y_{cf}$, parameters of the objective function $\beta, \gamma, \tau$, extrapolation parameters $(\alpha_t)$, step sizes $(\sigma_t)$, number of iterations $T$.
Output: $\boldsymbol{x}_{cf} \in \mathbb{R}^d$ the CFE of $\boldsymbol{x}_f$.
1: Initialize $\boldsymbol{x}^0, \boldsymbol{z}^0 \leftarrow \mathbf{0} \in \mathbb{R}^d$.
2: $h(\boldsymbol{x}, y_{cf}) := \theta_2(\boldsymbol{x},\boldsymbol{x}_f) + \gamma\mathcal{L}_f(\boldsymbol{x}, y_{cf}) - \tau\hat{q}(\boldsymbol{x}, y_{cf})$
3: $g_p(\boldsymbol{x}) := I_{\mathcal{A}}(\boldsymbol{x}) + \beta\theta_p(\boldsymbol{x},\boldsymbol{x}_f)$
4: for $t = 0, ..., T-1$ do
5:     $\boldsymbol{r}^{t+1} \leftarrow \nabla_{\boldsymbol{z}^t} h(\boldsymbol{z}^t, y_{cf})$
6:     $S_{\sigma_{t+1}}(\boldsymbol{z}^{t+1}, y_{cf}) \leftarrow \boldsymbol{z}^t - \sigma_{t+1}\boldsymbol{r}^{t+1}$
7:     $\boldsymbol{x}^{t+1} \leftarrow \text{prox}_{\sigma_{t+1}g_p}(S_{\sigma_{t+1}}(\boldsymbol{z}^{t+1}, y_{cf}))$
8:     $\boldsymbol{z}^{t+1} \leftarrow \boldsymbol{x}^{t+1} + \alpha_{t+1}(\boldsymbol{x}^{t+1} - \boldsymbol{x}^t)$
9: end for
10: return $\boldsymbol{x}^T =: \boldsymbol{x}_{cf}$
```

All computations are performed using Python 3.12 [Van Rossum and Drake, 2009], PyTorch 2.4.1 [Paszke et al., 2019], and Scikit-learn 1.5.2 [Buitinck et al., 2013], utilizing eight cores of an Intel Xeon Gold 6342 CPU @ 2.80GHz with 16GB of memory. Our codes are available at https://github.com/wagnermoritz/SCFE.

5.1 Setup

5.1.1 Datasets

We use the Boston Housing dataset ($d = 12$) [Harrison and Rubinfeld, 1978] to predict whether the median house value exceeds a threshold, the Breast Cancer Wisconsin dataset ($d = 30$) [H. Wolberg and Street, 1995] to classify malignant versus benign tumors, Wine dataset [Aeberhard et al., 1992] for wine quality classification, and the MNIST dataset [Asuncion et al., 2007] for handwritten digit classification. All continuous features are scaled using a min-max scaler to the $[0, 1]$ range. MNIST consists of 60,000 training images and 10,000 test images, while the other datasets are split with 100 test points.

5.1.2 Method Implementation

For the experiments, we consider a 2−layer, densely connected ReLU network with 20 neurons per hidden layer for the smaller datasets, and a convolutional neural network (CNN) consisting of two convolutional layers followed by three densely connected layers for the MNIST dataset, as the underlying machine learning models that give predictions. We train each network with a learning rate of 10^{-3} using the Adam optimizer. The densely connected networks are trained using a batch size of 32 for 20 epochs and the CNN is trained using a batch size of 128 for 80 epochs. All classifiers reach an accuracy above 95% on the test data.

Following *Alibi Explain* [Klaise et al., 2021], we conduct a logarithmic grid search over $[10^{-3}, 10^{3}]^2$ for β and τ, and perform a section search with 10 steps for γ for each test instance. For binary classifiers $f : \mathcal{X} \to [0,1]$ we utilize the classification loss with $y_{cf} \in \{0,1\}$

$$\mathcal{L}_f(\boldsymbol{x}, y_{cf}) := \max\{(1 - 2y_{cf})f(\boldsymbol{x}), -c\},$$

and for multi-class classifiers $f : \mathcal{X} \to \mathbb{R}^{|\mathcal{Y}|}$ with $y_{cf} \in \{1, ..., |\mathcal{Y}|\}$ we use

$$\mathcal{L}_f(\boldsymbol{x}, y_{cf}) := \max\{\max\{f_i(\boldsymbol{x}) \mid i \neq y_{cf}\} - f_{y_{cf}}(\boldsymbol{x}), -c\},$$

where $\boldsymbol{x}$ a datapoint of class y_{cf} and c is the cut-off parameter for the hinge loss. Similar to most adversarial attacks utilizing the Hinge Loss (or Carlini-Wagner loss) [Carlini and Wagner, 2017], we set its threshold to $c = 0$. Following Zhang et al. [2023], in the S-CFE$_{\text{kNN}}$ approach, we use 3, 4, and 5 for the grid search of nearest neighbors k, as higher values tend to confuse the classifier and degrade performance (see Appendix B).

In line with Beck and Teboulle [2009], we use the sequence (α_t) of extrapolation parameters defined by

$$\beta_t = \frac{1}{2}\left(1 + \sqrt{1 + 4\beta_{t-1}^2}\right), \quad \alpha_t = \frac{\beta_t - 1}{\beta_{t+1}}, \qquad (8)$$

with $\beta_0 = 0$. The sequence of step sizes (σ_t) is given by an initial step size σ_0 and a square-root decay

$$\sigma_{t+1} = \sigma_t \sqrt{1 - \frac{t}{t_{\max}}},$$

where $t_{\max}$ is the maximum number of iterations. We run all the methods for a total of 200 iterations.

5.1.3 Evaluation Metrics

In our experiments, we report validity, defined as the ratio of CFEs with the desired class label multiplied by 100%. We measure proximity using the ℓ_2-norm, $\|\boldsymbol{x}_{cf} - \boldsymbol{x}_f\|_2$, and sparsity using the ℓ_0-(quasi) norm, $\|\boldsymbol{x}_{cf} - \boldsymbol{x}_f\|_0$, where $\boldsymbol{x}_{cf}$ is the generated CFE and $\boldsymbol{x}_f$ the original factual point. To assess plausibility, we use the LOF metric (Definition 3.4) to determine how much a generated CFE deviates from the data manifold, based on its k nearest neighbors in $\mathcal{D}$. The scikit-learn implementation of LOF is used with default parameters. Additionally, we report the average runtime per method.

5.2 Results

Proximity, sparsity, and plausibility are often conflicting objectives, as data near the decision boundary is typically sparse, and small shifts across it can yield implausible CFEs [Van Looveren and Klaise, 2021, Dandl et al., 2024]. By applying our S-CFE algorithm with various plausibility terms, and the constrained θ_0-metric via indicator functions to achieve sparsity, we are able to generate sparse, plausible CFEs that remain close to the factual data points and maintain the computation speed.

5.2.1 Quantitative Evaluation

As Artelt and Hammer [2020] consider only linear classifiers and decision trees, Tab. 1 reports results for logistic regression classifiers on the Housing and Wine datasets, including the CFE validity, average proximity, average sparsity, plausibility (measured via LOF), and generation time for both CFE algorithms. Tab. 2 presents the same metrics for the Housing, Wine, and MNIST datasets using a CNN classifier, comparing variants of our method against [Dhurandhar et al., 2018, Zhang et al., 2023].

In terms of validity, S-CFE consistently outperforms other methods (e.g., on MNIST), generating the desired CFE for nearly all original points. Thanks to our formulation in Eq. (7), we control the number of altered features, setting it to 1 in Tab. 1 and 2 in Tab. 2, producing the sparsest CFEs. The integration of differentiable plausibility terms guides the search towards high-density regions, resulting in the lowest LOF. This allows us to generate superior sparse, manifold-aligned CFEs compared to benchmark works of Dhurandhar et al. [2018], Artelt and Hammer [2020] and Zhang et al. [2023]. Additionally, we achieve comparable or slightly better proximity to factual data while maintaining low computation time, underscoring the simplicity of our method. See Tab. 3 and Tab. 4 in Appendix D for additional experiments.

Note that CEM [Dhurandhar et al., 2018] can only be applied to binary classification datasets, where the goal is to find CFEs for a certain target class. Their pertinent positives represent the minimal set of features required to maintain the current classification, while pertinent negatives are the minimal set needed to change the classification. In datasets like MNIST, the altered class could be any of the nine remaining classes (e.g., a "9" could change into any class other than "4"), making CEM unsuitable for targeted CFEs in multi-class scenarios.

Tsiourvas et al. [2024], on the other hand, reference a link to an empty repository, hindering result reproducibility. The MIP formulation of Tsiourvas et al. [2024] faces scalability issues with complex networks, addressed by limiting the search to a few live polytopes from the correct class. However, this partition-

Table 1: CFEs for linear classifiers on the Boston Housing and Wine datasets. Dimensionality was reduced to 8 using PCA, with 100 test points assessed. Compute time is reported in seconds per 100 CFEs. The best values for each dataset are highlighted. The method proposed by Artelt and Hammer [2020] is referred to as PCFE.

Dataset	Method	Validity (std)	ℓ_2 (std)	ℓ_0 (std)	LOF (std)	Time
Housing 8 features	S-CFE$_{KDE}$	**100** (0.00)	3.06 (1.42)	**1.00** (0.00)	1.22 (0.27)	11.8
	S-CFE$_{GMM}$	**100** (0.00)	2.62 (1.25)	**1.00** (0.00)	**1.20** (0.27)	12.1
	S-CFE$_{k-\text{NN}}$	**100** (0.00)	2.51 (1.19)	**1.00** (0.00)	1.34 (0.58)	**4.61**
	PCFE	**100** (0.00)	**1.61** (1.03)	1.13 (0.87)	1.31 (0.57)	26.7
Wine 8 features	S-CFE$_{KDE}$	**100** (0.00)	2.10 (1.10)	**1.00** (0.00)	**0.98** (0.01)	9.88
	S-CFE$_{GMM}$	**100** (0.00)	2.09 (1.10)	**1.00** (0.00)	**0.98** (0.02)	11.4
	S-CFE$_{k-\text{NN}}$	**100** (0.00)	2.15 (1.12)	**1.00** (0.00)	0.99 (0.02)	**4.50**
	PCFE	**100** (0.00)	**1.39** (1.02)	1.37 (0.96)	0.99 (0.02)	22.3

Table 2: CFEs for DNN classifiers on the Boston Housing and Wine datasets, and for a CNN classifier on the MNIST dataset. Evaluated on 1000 test points for MNIST and 100 test points for the other two datasets. The compute time is given in seconds per 100 CFEs. The best values for each dataset are highlighted. The methods proposed by Zhang et al. [2023] and Dhurandhar et al. [2018] are referred to as DCFE and CEM, respectively.

Dataset	Method	Validity (std)	ℓ_2 (std)	ℓ_0 (std)	LOF (std)	Time
Housing 12 features	S-CFE$_{KDE}$	**100** (0.00)	**2.59** (1.21)	**2.00** (0.00)	1.23 (0.29)	12.7
	S-CFE$_{GMM}$	**100** (0.00)	2.91 (1.38)	**2.00** (0.00)	**1.12** (0.26)	13.3
	S-CFE$_{k\text{NN}}$	**100** (0.00)	3.64 (1.73)	**2.00** (0.00)	1.17 (0.31)	5.85
	DCFE	**100** (0.00)	3.50 (1.68)	6.86 (1.42)	1.27 (0.38)	**5.33**
	CEM	94.0 (0.23)	2.93 (2.23)	2.99 (1.17)	1.36 (0.60)	7.51
Wine 13 features	S-CFE$_{KDE}$	**100** (0.00)	3.31 (1.16)	**2.00** (0.00)	0.99 (0.01)	12.4
	S-CFE$_{GMM}$	**100** (0.00)	3.44 (1.09)	**2.00** (0.00)	**0.98** (0.02)	13.1
	S-CFE$_{k-\text{NN}}$	**100** (0.00)	4.04 (1.59)	**2.00** (0.00)	1.01 (0.07)	5.80
	DCFE	**100** (0.00)	**3.21** (2.70)	7.13 (1.31)	1.03 (0.18)	**4.95**
	CEM	92.0 (0.29)	5.40 (3.25)	5.14 (2.68)	1.07 (0.14)	5.71
MNIST 784 features	S-CFE$_{GMM}$	99.1 (0.09)	**6.74** (2.92)	**25.0** (0.00)	**1.21** (0.18)	55.3
	S-CFE$_{k-\text{NN}}$	**99.8** (0.04)	7.04 (2.99)	**25.0** (0.00)	1.30 (0.22)	13.1
	DCFE	99.3 (0.08)	8.06 (3.48)	118 (6.30)	1.32 (2.24)	**11.8**

ing is restricted to ReLU NNs and often compromises plausibility. Fig. 5 conceptually shows our method outperforming MIP-Live in terms of plausibility on their simple example, and we conjecture this advantage extends to higher dimensions, as their approach overly constrains the search space, leading to a poor sparsity-plausibility tradeoff. Moreover, limiting the method to ReLU networks ignores the growing success of other architectures with other nonlinearities than ReLU (e.g., Swin transformers in Computer Vision using GELU activations [Liu et al., 2021]). In contrast, our approach supports any architecture that utilizes standard gradient computation techniques like backpropagation. Consequently, we exclude MIP-Live from our experiments.

5.2.2 Robustness of Plausible CFEs to Input Manipulations

CFEs without plausibility constraints have been shown to diverge significantly with even minor input perturbations, underscoring their lack of robustness [Slack et al., 2021]. This presents a challenge for CFEs, as two similar individuals may receive drastically different explanations. In contrast, incorporating plausibility constraints improves robustness against such input shifts, enhancing the individual fairness of CFEs [Artelt et al., 2021, Zhang et al., 2023]. Fig. 3 demonstrates that sparse, manifold-aligned CFEs generated with various plausibility regularizers further enhance robustness against input shifts. For additional results across various datasets and different sparsity modes, refer to Fig. 6 in Appendix D.

6 Conclusion and Discussion

We introduced “S-CFE”, a novel yet simple framework for generating sparse and plausible counterfactual explanations. Our method is based on proximal gradient techniques for non-convex and non-smooth optimization, offering enhanced control over feature changes and leveraging density estimates to ensure plausibility. Extensive experiments demonstrate that S-CFE outperforms existing methods in producing sparse, plausible CFEs while maintaining proximity to input data and computational efficiency.

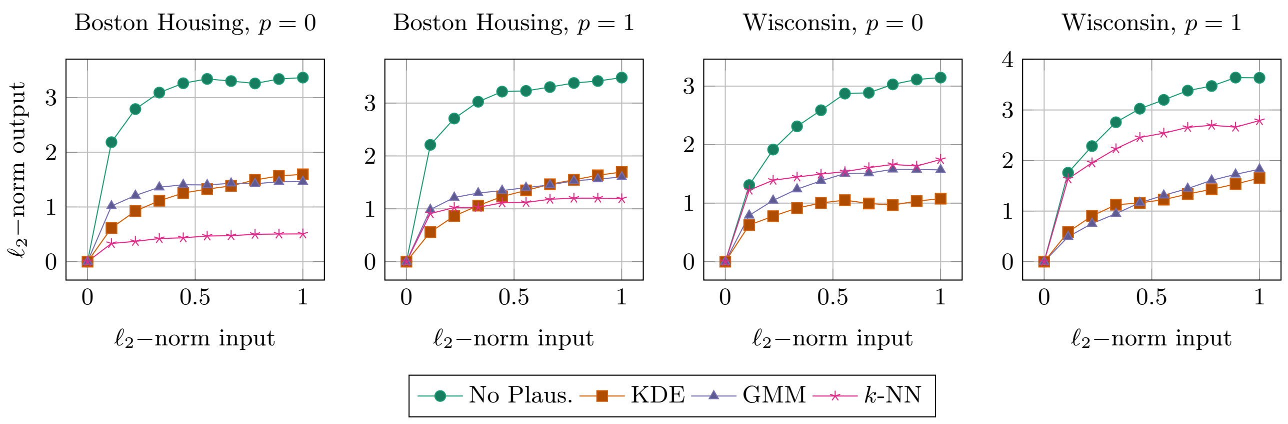

Figure 3: Robustness of the different methods. The distance of the input data points to the original data points on the x-axis and the distance of the generated CFEs to the CFE generated from the original data points on the y-axis. Tested on 100 data points from each data set.

Limitations and Broader Impact: Sparse plausible CFEs highlight which feature changes may lead to different predictions but offer no guidance on real-world interventions for achieving the desired outcome, which requires causal knowledge. *Improving* the underlying target is more important than merely gaining predictor acceptance [Tsirtsis and Gomez Rodriguez, 2020]. For example, altering symptoms may change a COVID-19 diagnosis, but not the actual infection status [König et al., 2023]. Our method acts as an adversary, guiding users toward simple changes for predictor acceptance without improving the real-world state. Future work will explore training S-CFE directly on data rather than relying on classifier predictions.

Acknowledgements

This research was partially supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy through the Berlin Mathematics Research Center MATH+ (EXC-2046/1, Project ID: 390685689). Additional support was provided by the Research Campus MODAL, funded by the German Federal Ministry of Education and Research (BMBF, Grant No. 05M14ZAM).

References

S. Aeberhard, D.C., and O. de Vel. Comparison of classifiers in high dimensional settings. *Tech. Rep. no. 92-02*, 1992.

André Artelt and Barbara Hammer. Convex density constraints for computing plausible counterfactual explanations. In *Artificial Neural Networks and Machine Learning–ICANN 2020: 29th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 15–18, 2020, Proceedings, Part I 29*, pages 353–365. Springer, 2020.

André Artelt, Valerie Vaquet, Riza Velioglu, Fabian Hinder, Johannes Brinkrolf, Malte Schilling, and Barbara Hammer. Evaluating robustness of counterfactual explanations. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 01–09. IEEE, 2021.

Arthur Asuncion, David Newman, et al. Uci machine learning repository, 2007.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1): 183–202, 2009.

Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.

Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, et al. Api design for machine learning software: experiences from the scikit-learn project. *arXiv preprint arXiv:1309.0238*, 2013.

Wenfei Cao, Jian Sun, and Zongben Xu. Fast image deconvolution using closed-form thresholding formulas of lq (q= 12, 23) regularization. *Journal of visual communication and image representation*, 24(1):31–41, 2013.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. Ieee, 2017.

Francesco Croce and Matthias Hein. Sparse and imperceivable adversarial attacks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4724–4732, 2019.

Susanne Dandl, Kristin Blesch, Timo Freiesleben, Gunnar König, Jan Kapar, Bernd Bischl, and Marvin Wright. Countarfactuals–generating plausible model-agnostic counterfactual explanations with adversarial random forests. *arXiv preprint arXiv:2404.03506*, 2024.

Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Paishun Ting, Karthikeyan Shanmugam, and Payel Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. *Advances in neural information processing systems*, 31, 2018.

William H. Wolberg and W. Nick Street. O.l.m.: Breast cancer wisconsin (diagnostic) data set., 1995. URL https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic).

Faisal Hamman, Erfaun Noorani, Saumitra Mishra, Daniele Magazzeni, and Sanghamitra Dutta. Robust counterfactual explanations for neural networks with probabilistic guarantees. In *International Conference on Machine Learning*, pages 12351–12367. PMLR, 2023.

D. Harrison and D.L. Rubinfeld. Boston housing data set., 1978. URL https://archive.ics.uci.edu/ml/datasets/Housing.

Amir-Hossein Karimi, Gilles Barthe, Bernhard Schölkopf, and Isabel Valera. A survey of algorithmic recourse: definitions, formulations, solutions, and prospects. *arXiv preprint arXiv:2010.04050*, 2020.

Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part I 16*, pages 795–811. Springer, 2016.

Janis Klaise, Arnaud Van Looveren, Giovanni Vacanti, and Alexandru Coca. Alibi explain: Algorithms for explaining machine learning models. *Journal of Machine Learning Research*, 22(181):1–7, 2021.

Gunnar König, Timo Freiesleben, and Moritz Grosse-Wentrup. Improvement-focused causal recourse (icr). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 11847–11855, 2023.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324, 1998.

Guang-He Lee, David Alvarez-Melis, and Tommi S. Jaakkola. Towards robust, locally linear deep networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SylCrnCcFX.

Huan Li and Zhouchen Lin. Accelerated proximal gradient methods for nonconvex programming. *Advances in neural information processing systems*, 28, 2015.

Rongrong Lin, Shihai Chen, Han Feng, and Yulan Liu. Computing the proximal operator of the $\ell_{1,q}$-norm for group sparsity. *arXiv preprint arXiv:2409.14156*, 2024.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

Ramaravind K Mothilal, Amit Sharma, and Chenhao Tan. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pages 607–617, 2020.

Philip Naumann and Eirini Ntoutsi. Consequence-aware sequential counterfactual generation. In *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part II 21*, pages 682–698. Springer, 2021.

Neal Parikh, Stephen Boyd, et al. Proximal algorithms. *Foundations and trends® in Optimization*, 1(3):127–239, 2014.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Jeffrey S Racine et al. Nonparametric econometrics: A primer. *Foundations and Trends® in Econometrics*, 3(1):1–88, 2008.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

Dylan Slack, Anna Hilgard, Himabindu Lakkaraju, and Sameer Singh. Counterfactual explanations can be manipulated. *Advances in neural information processing systems*, 34:62–75, 2021.

Asterios Tsiourvas, Wei Sun, and Georgia Perakis. Manifold-aligned counterfactual explanations for neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 3763–3771. PMLR, 2024.

Stratis Tsirtsis and Manuel Gomez Rodriguez. Decisions, counterfactual explanations and strategic behavior. *Advances in Neural Information Processing Systems*, 33:16749–16760, 2020.

Arnaud Van Looveren and Janis Klaise. Interpretable counterfactual explanations guided by prototypes. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 650–665. Springer, 2021.

Guido Van Rossum and Fred L Drake. *Introduction to python 3: python documentation manual part 1*. CreateSpace, 2009.

Sahil Verma, Varich Boonsanong, Minh Hoang, Keegan E Hines, John P Dickerson, and Chirag Shah. Counterfactual explanations and algorithmic recourses for machine learning: A review. *ACM Computing Surveys*, 56, 2024.

Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31:841, 2017.

Songming Zhang, Xiaofeng Chen, Shiping Wen, and Zhongshan Li. Density-based reliable and robust explainer for counterfactual explanation. *Expert Systems with Applications*, 226:120214, 2023.

Mingkang Zhu, Tianlong Chen, and Zhangyang Wang. Sparse and imperceptible adversarial attack via a homotopy algorithm. In *International Conference on Machine Learning*, pages 12868–12877. PMLR, 2021.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 67(2):301–320, 2005.

Checklist

1. For all classifiers and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or classifier. **Yes**
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. **Yes. Specifically, for each algorithm we report the average runtime and the sample size.**
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. **Yes**

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. **Yes**
   (b) Complete proofs of all theoretical results. **Yes**
   (c) Clear explanations of any assumptions. **Yes**

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). **Yes**
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). **Yes**
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). **Yes**
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). **Yes**

4. If you are using existing assets (e.g., code, data, classifiers) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. **Yes**
   (b) The license information of the assets, if applicable. **Yes**
   (c) New assets either in the supplemental material or as a URL, if applicable. **Yes**
   (d) Information about consent from data providers/curators. **Not Applicable**
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. **Not Applicable**

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. **Not Applicable**
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. **Not Applicable**

(c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. **Not Applicable**

Appendix

A Density Estimate Instances

Definition A.1. *A kernel density estimator (KDE) $\hat{q}_{\text{KDE}}$ is defined as*

$$\hat{q}_{\text{KDE}}(\boldsymbol{x}) = \sum_{i=1}^{m} \boldsymbol{w}_i k(\boldsymbol{x}, \boldsymbol{x}_i),$$

where $k(\cdot,\cdot)$ denotes a suitable kernel function, $\boldsymbol{x}_i$ denotes the $i-$th correctly classified sample in the training data set and $\boldsymbol{w}_i > 0$ denotes the weighting of the $i-$th sample.

For the KDE plausibility term, we utilize a Gaussian normal kernel of bandwidth parameter $\sigma > 0$ (standard choice from Racine et al. [2008]) $k(\boldsymbol{x}, \boldsymbol{x}_i) := e^{-\theta_2(\boldsymbol{x},\boldsymbol{x}_i)/2\sigma^2}$ for correctly classified points $\boldsymbol{x}_i$ and we set $\boldsymbol{w}_i = 1/m$, for $i \in [m]$.

Definition A.2. *A Gaussian mixture classifier (GMM) $\hat{q}_{\text{GMM}}$ with m components is defined as*

$$\hat{q}_{\text{GMM}}(\boldsymbol{x}) = \sum_{i=1}^{m} \pi_i \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i),$$

where π_i represents the prior probability of the $i-$th component, $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ denote the mean and covariance of the $i-$th component, a $d-$dimensional Gaussian density.

Definition A.3. (Density Gravity on an Instance ([Zhang et al., 2023]) *For $\boldsymbol{x} \in \mathcal{D}$, let $N_k(\boldsymbol{x}) = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_k\}$ be the set of $k-$NNs of $\boldsymbol{x}$ in $\mathcal{D}$. The $k-$local density of $\boldsymbol{x}$ is defined by $\rho(\boldsymbol{x}) = |N_k(\boldsymbol{x})|(\sum_{\boldsymbol{x}_i \in N_k(\boldsymbol{x})} \|\boldsymbol{x} - \boldsymbol{x}_i\|_p)^{-1}$. The set of relative local densities $\{\hat{\rho}_1, ..., \hat{\rho}_k\}$ of the points in $N_k(\boldsymbol{x})$ is the result of normalization as $\hat{\rho}_i = \frac{\rho_i}{\sum_{i=1}^{k} \rho_i}, \rho_i \geq 0, i \in [k]$. Then, the density gravity G of $\boldsymbol{x}$ on $\mathcal{D}$ is defined as*

$$G(\boldsymbol{x}) = \sum_{i=1}^{k} \hat{\rho}_i \boldsymbol{x}_i, \text{ where } \boldsymbol{x}_i \in N_k(\boldsymbol{x}), \; \sum_{i=1}^{k} \hat{\rho}_i = 1, \text{ and } \hat{\rho}_i \geq 0.$$

Note that the maximum operator $\text{rd}_k(\boldsymbol{x}, \cdot)$ of Definition 3.4 is linearized for simplicity. In simple words, the density gravity of $\boldsymbol{x}$, finds the closest point $G(\boldsymbol{x})$ that lies in a high-density data region by finding a convex combination of $k-$nearest neighbors of $\boldsymbol{x}$ weighted by their local density. Note that by definition, the local density, denoted by $\rho(\boldsymbol{x})$ in Definition A.3, is higher for a point $\boldsymbol{x}$ if it has more neighbors that are closer to it.

B Issues with Using Many Neighbors for the S-CFE$_{\text{kNN}}$ Approach

In Fig. 4 we experiment with choosing different neighbors for the plausibility term given by density gravity, which minimizes the distance between the CFE and a point in the convex combination of the $k-$nearest neighbors of given factual data point, weighted by their local density. Choosing higher values of k results in neighbors being chosen from different high-density areas of correctly classified data points, thus the density gravity point can result in low-density areas (as can be seen in the middle and the right figure in Fig. 4).

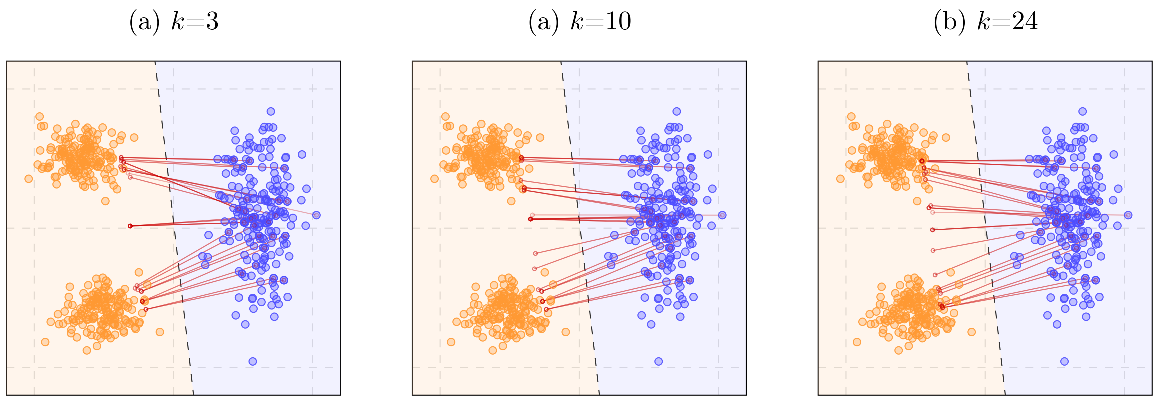

Figure 4: A toy example illustrating the positioning of convex combinations of $k-$NNs obtained via density gravity relative to the original points.

C Our S-CFE Approach vs MIP-LIVE-m=1

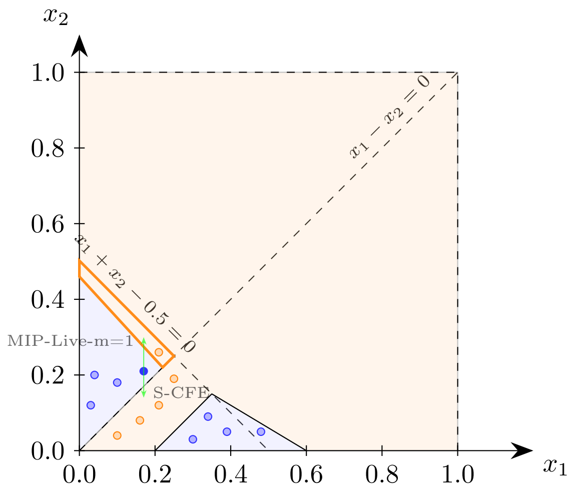

Figure 5: Example reproduced from Tsiourvas et al. [2024]. Compares geometrically MIP-Live-m=1 [Tsiourvas et al., 2024] vs. our S-CFE approach. The generated CFE of our method resides in a high-density region and is sparse. MIP-Live-m=1 considerably restricts the working space - the small bounded red region, and uses only 1 neighbor for the LOF manifold adhering constraint.

D Additional Experiments

We provide additional results in Tab. 3 and Tab. 4, demonstrating that our method generates sparse, manifold-aligned CFEs that closely follow the data manifold while maintaining efficient computation times. These findings align with Sec. 5.2.1.

We exclude CEM [Dhurandhar et al., 2018] from further experiments since Zhang et al. [2023] already benchmark against it. Instead, we present additional results using the closed-form solution for the proximal operator in Eq. (6) for $p = 1/2$ [Cao et al., 2013, Lin et al., 2024]. The solution, for $i \in [d]$ and the inverse Lipschitz constant

Table 3: CFEs for linear classifiers on the Boston Housing and Wine datasets. The dimensionality of both datasets has been reduced to 8 using PCA. Evaluated 100 test points. The S-CFE $p = 0$ variants use sparsity constraints, while the other variants use the corresponding norm as a regularizer. The compute time is given in seconds per 100 CFEs. The compute time is given in seconds per 100 CFEs. We denote by DCFE and PCFE the methods proposed by Zhang et al. [2023] and Artelt and Hammer [2020], respectively.

Dataset	Method	Validity	ℓ_2	ℓ_0	LOF	KDE	GMM	Time
Wine 8 features PCA	S-CFE$_{KDE}$ $p=0$	100	2.10	**1.00**	**0.98**	**-2.61**	-19.0	9.88
	S-CFE$_{KDE}$ $p=\frac{1}{2}$	100	1.93	1.52	**0.98**	-2.71	-19.1	12.6
	S-CFE$_{KDE}$ $p=1$	100	2.11	2.02	0.99	-2.68	-23.5	11.0
	S-CFE$_{GMM}$ $p=0$	100	2.09	**1.00**	**0.98**	-2.86	**-14.3**	11.4
	S-CFE$_{GMM}$ $p=\frac{1}{2}$	100	2.86	1.31	0.99	-2.89	-15.3	13.6
	S-CFE$_{GMM}$ $p=1$	100	2.40	2.00	0.99	-3.01	-20.1	11.6
	S-CFE$_{k-\text{NN}}$ $p=0$	100	2.15	**1.00**	0.99	-2.92	-19.8	4.50
	S-CFE$_{k-\text{NN}}$ $p=\frac{1}{2}$	100	2.49	1.55	1.01	2.93	-20.9	4.79
	DCFE	100	2.08	1.99	1.01	-2.96	-23.4	**3.49**
	PCFE	100	**1.39**	1.37	0.99	-2.93	-20.6	22.3
Housing 8 features PCA	S-CFE$_{KDE}$ $p=0$	100	3.06	**1.00**	1.22	**-2.55**	-14.1	11.8
	S-CFE$_{KDE}$ $p=\frac{1}{2}$	100	2.82	1.31	**1.20**	-2.69	-15.8	13.2
	S-CFE$_{KDE}$ $p=1$	100	2.14	2.44	1.23	-2.75	-14.8	10.3
	S-CFE$_{GMM}$ $p=0$	100	2.62	**1.00**	**1.20**	-2.89	**-10.8**	12.1
	S-CFE$_{GMM}$ $p=\frac{1}{2}$	100	3.01	1.92	1.23	-2.84	-11.3	12.5
	S-CFE$_{GMM}$ $p=1$	100	2.73	3.19	1.26	-2.92	-11.6	12.0
	S-CFE$_{k-\text{NN}}$ $p=0$	100	2.51	**1.00**	1.34	-3.01	-14.2	4.61
	S-CFE$_{k-\text{NN}}$ $p=\frac{1}{2}$	100	3.05	1.87	1.38	-3.01	-20.8	4.48
	DCFE	100	3.05	3.22	1.31	-2.99	-16.6	**4.17**
	PCFE	100	**1.61**	1.13	1.31	-3.06	-11.4	26.7

L replaced by a step size sequence $(\sigma_t)_{t\in\mathbb{N}}$, is given by

$$[\boldsymbol{x}^{t+1}_{cf,\frac{1}{2}}]_i = \begin{cases} \frac{2}{3}[S_{\sigma_t}(\boldsymbol{x}^t, y_{cf})]_i\left(1+\cos\left(\frac{2\pi}{3} - \frac{2\phi_{2\beta\sigma_t}([S_{\sigma_t}(\boldsymbol{x}^t,y_{cf})]_i)}{3}\right)\right), & \text{if } |[S_{\sigma_t}(\boldsymbol{x}^t, y_{cf})]_i| > g(2\beta\sigma_t), \\ 0, & \text{otherwise,} \end{cases}$$

where $\phi_{2\beta\sigma_t}([S_{\sigma_t}(\boldsymbol{x}^t, y_{cf})]_i) = \arccos\left(\frac{\beta\sigma_t}{4}\left(\frac{|[S_{\sigma_t}(\boldsymbol{x}^t,y_{cf})]_i|}{3}\right)^{-\frac{3}{2}}\right)$, and $g(2\beta\sigma_t) = \frac{\sqrt[3]{54}}{4}(2\beta\sigma_t)^{\frac{2}{3}}$.

Results in Fig. 6 of Appendix D further confirm findings of section Sec. 5.2.2 that our S-CFE generated CFEs are robust to input shifts.

Table 4: CFEs for DNN classifiers on the Boston Housing Wine datasets, and for a CNN classifier on the MNIST dataset. Evaluated on 1000 test points for MNIST and 100 test points for the other two datasets. The S-CFE $p = 0$ variants use sparsity constraints, while the other variants use the corresponding norm as a regularizer. The compute time is given in seconds per 100 CFEs. We denote by DCFE the method proposed by Zhang et al. [2023].

Dataset	Method	Validity	ℓ_2	ℓ_0	LOF	KDE	GMM	Time
Housing 12 features	S-CFE$_{KDE}$ $p = 0$	100	2.59	**2.00**	1.23	-2.91	-15.3	12.7
	S-CFE$_{KDE}$ $p = \frac{1}{2}$	100	**2.41**	2.78	1.18	**-2.87**	-16.8	14.5
	S-CFE$_{KDE}$ $p = 1$	100	2.52	6.30	1.27	-3.02	-17.7	12.3
	S-CFE$_{GMM}$ $p = 0$	100	2.91	**2.00**	**1.12**	-3.23	**-12.8**	13.3
	S-CFE$_{GMM}$ $p = \frac{1}{2}$	100	2.74	3.09	1.17	-3.55	-13.6	15.1
	S-CFE$_{GMM}$ $p = 1$	100	2.76	5.76	1.24	-3.47	-14.1	12.6
	S-CFE$_{k-\text{NN}}$ $p = 0$	100	3.64	**2.00**	1.17	-3.08	-17.6	5.85
	S-CFE$_{k-\text{NN}}$ $p = \frac{1}{2}$	100	3.61	4.04	1.20	-3.31	-19.1	6.04
	DCFE	100	3.50	6.86	1.27	-3.49	-20.7	**5.33**
Wine 13 features	S-CFE$_{KDE}$ $p = 0$	100	3.31	**2.00**	0.99	-2.87	-24.1	12.4
	S-CFE$_{KDE}$ $p = \frac{1}{2}$	100	3.30	5.40	**0.98**	**-2.76**	-21.8	11.8
	S-CFE$_{KDE}$ $p = 1$	100	**2.83**	6.73	1.00	-2.85	-25.4	12.3
	S-CFE$_{GMM}$ $p = 0$	100	3.44	**2.00**	**0.98**	-3.11	**-14.8**	13.1
	S-CFE$_{GMM}$ $p = \frac{1}{2}$	100	3.83	5.73	0.99	-2.98	-15.9	12.6
	S-CFE$_{GMM}$ $p = 1$	100	3.08	6.98	1.01	-3.05	-16.2	13.1
	S-CFE$_{k-\text{NN}}$ $p = 0$	100	4.04	**2.00**	1.01	-3.17	-37.9	5.80
	S-CFE$_{k-\text{NN}}$ $p = \frac{1}{2}$	100	3.71	6.82	1.02	-3.66	-40.2	5.72
	DCFE	100	3.21	7.13	1.03	-3.77	-42.1	**4.95**
MNIST 784 features	S-CFE$_{GMM}$ $p = 0$	99.1	**6.74**	**25.0**	**1.21**	-	**-1058**	55.3
	S-CFE$_{GMM}$ $p = \frac{1}{2}$	98.4	7.12	77.4	1.34	-	-1112	57.3
	S-CFE$_{GMM}$ $p = 1$	99.3	8.07	115	1.47	-	-1132	54.2
	S-CFE$_{k-\text{NN}}$ $p = 0$	99.8	7.04	**25.0**	1.30	-	-1075	13.1
	S-CFE$_{k-\text{NN}}$ $p = \frac{1}{2}$	**99.9**	8.13	80.9	1.20	-	-1069	12.7
	DCFE	99.3	8.06	118	1.32	-	-1122	**11.8**

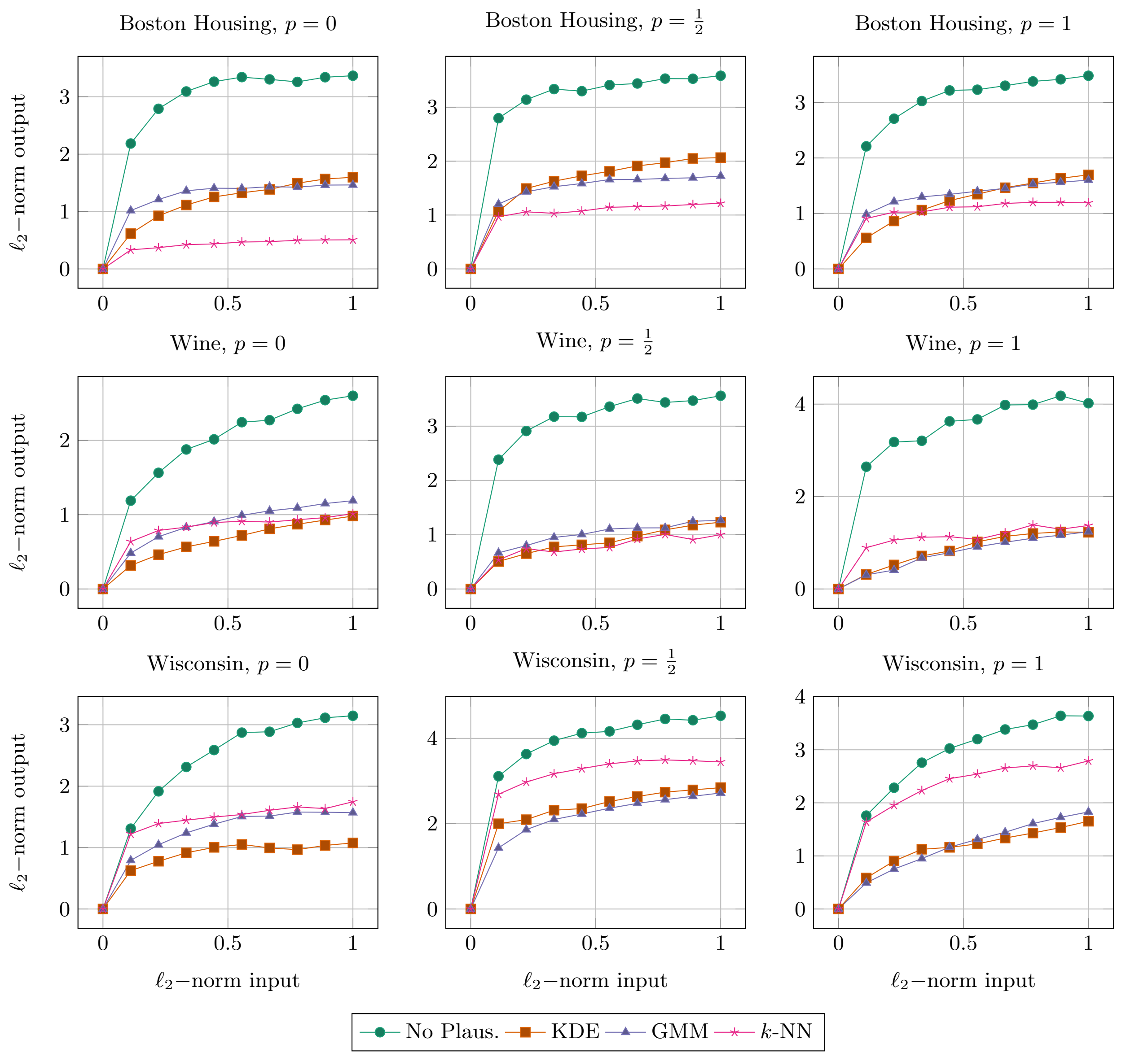

Figure 6: Robustness of the different methods. The distance of the input data points to the original data points on the x-axis and the distance of the generated CFEs to the CFE generated from the original data points on the y-axis. Tested on 100 data points from each data set.