GENERATING LIKELY COUNTERFACTUALS USING SUM-PRODUCT NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

The need to explain decisions made by AI systems is driven by both recent regulation and user demand. The decisions are often explainable only post hoc. In counterfactual explanations, one may ask what constitutes the best counterfactual explanation. Clearly, multiple criteria must be taken into account, although "distance from the sample" is a key criterion. Recent methods that consider the plausibility of a counterfactual seem to sacrifice this original objective. Here, we present a system that provides high-likelihood explanations that are, at the same time, close and sparse. We show that the search for the most likely explanations satisfying many common desiderata for counterfactual explanations can be modeled using Mixed-Integer Optimization (MIO). We use a Sum-Product Network (SPN) to estimate the likelihood of a counterfactual. To achieve that, we propose an MIO formulation of an SPN, which can be of independent interest.

1 INTRODUCTION

A better understanding of deployed AI models is needed, especially in high-risk scenarios (Dwivedi et al., 2023). Trustworthy and explainable AI (XAI), is concerned with techniques that help people understand, manage, and improve trust in AI models (Gunning, 2016; Burkart & Huber, 2021; Bodria et al., 2023). Explanations also serve an important role in debugging models to ensure that they do not rely on spurious correlations and traces of processing correlated with labels, such as timestamps. In a *post-hoc* explanation, a vendor of an AI system provides an individual user with a personalized explanation of an individual decision made by the AI system, improving the model's trustworthiness (Karimi et al., 2020; Li et al., 2023). In this context, personalized explanations are often called local explanations because they explain the model's decision locally, around a given sample, such as one person's input. Thus, local explanations provide information relevant to the user without revealing global information about the model, regardless of whether the model is interpretable *a priori*.

Consider, for example, credit decision-making in financial services. The models utilized need to be interpretable *a priori*, cf. Equal Credit Opportunity Act in the US (Equal Credit Opportunity Act , ECOA) and related regulation (European Commission, 2016a;b) in the European Union, but an individual who is denied credit may still be interested in a personalized, local explanation. A well-known example of local explanations is the counterfactual explanation (CE). CE answers the question "How should a sample be changed to obtain a different result?" (Wachter et al., 2017). In the example of credit decision-making, a denied client might ask what they should do to obtain the loan. The answer would take the form of a CE. For example, "Had you asked for half the loan amount, your application would have been accepted". As illustrated, CE can be easily understood (Byrne, 2005; Guidotti, 2022). However, their usefulness is influenced by many factors (Guidotti, 2022), including validity, similarity, sparsity, actionability, and plausibility.

This work focuses on the plausibility of counterfactual explanations. Unfortunately, plausibility does not have a clear definition. The definition of Guidotti (2022) suggests CE not being an outlier and measures it as the mean distance to data. A Local Outlier Factor is often used (e.g., Kanamori et al., 2020), but this method is not invariant of the data size. Alternatively, Jiang et al. (2024) define a "plausible region" as a convex hull of k nearest neighbors of the factual. This region can, however, still contain outliers.

Table 1: *Method comparison.* A check mark indicates that a given method claims to possess the given feature. The star symbol (*) means that the method is model-agnostic as long as the classifier can be expressed using MIO. Complex data means data with continuous, categorical, ordinal, and discrete contiguous values. Exogenous property holds if a method can generate unseen data samples as CEs. Regarding Actionability, C-CHVAE disregards the monotonicity of features, and DiCE claims to achieve actionability through diversity without any data-specific constraints in place. All methods require validity and optimize some notion of similarity.

Method		Plausibility	Sparsity	Actionability	Complex data	Model-agnostic	Exogenous
LiCE	(Proposed method)	✓	✓	✓	✓	✓*	✓
PROPLACE	(Jiang et al., 2024)	✓					✓
C-CHVAE	(Pawelczyk et al., 2020)	✓		only immut.		✓	✓
FACE	(Poyiadzi et al., 2020)	✓		✓	✓	✓	
DiCE	(Mothilal et al., 2020)		✓	✓			✓
PlaCE	(Artelt & Hammer, 2020)	✓	✓				✓
DACE	(Kanamori et al., 2020)	✓		✓	✓	✓*	✓

Many other methods consider estimating the likelihood of CEs as a proxy for plausibility. This approach aligns with the definition of CE not being an outlier, since outliers will have low likelihood. One such approach uses (Conditional) Variational Auto-Encoders (Jordan et al., 1998; Pawelczyk et al., 2020) in likelihood estimation. This approach does not provide a good way to handle categorical inputs and does not provide an efficient way to compute the exact likelihood of a CE. Plausible CE (PlaCE) proposed in (Artelt & Hammer, 2020) uses Gaussian Mixture models in the framework of convex optimization to maximize likelihood in CE generation. Its limitations are the inability to handle categorical features and non-linear classifiers. Another common way to estimate likelihood is Kernel Density Estimation (KDE), which shares the inability to handle categorical features well. KDE is utilized by, e.g., FACE (Poyiadzi et al., 2020), which can also return CEs only from the training set.

Our Contribution We propose *Likely Counterfactual Explanations* (LiCE) method which optimizes plausibility in combination with other desiderata (see Table 1). LiCE uses Sum-Product Networks (SPN) of Poon & Domingos (2011), which are state-of-the-art tractable models for estimating likelihood. They naturally handle categorical features. This work combines the tradition of tractable probabilistic models with mixed-integer formulations, by formulating the former in the latter.

In particular, we propose:

- A mixed-integer formulation of a trained Sum-Product Network estimating log-likelihood.
- Sum-Product Network as a measure of plausibility of CE, which allows the integration of plausibility directly into the MIO formulation.
- LiCE method for the generation of CEs. An MIO model that can be constrained by or optimized with respect to the most common desiderata regarding CE generation.

The advantage of our approach can be illustrated on the German Credit dataset (Hofmann, 1994). See Figure 1, where CEs produced by several methods considering the diversity or plausibility of CE are compared against the factual (white cross) in the plane, where the horizontal axis represents the amount of credit and where the vertical axis is the duration.

For example, C-CHVAE (Pawelczyk et al., 2020) and FACE (Poyiadzi et al., 2020) suggest approximately halving the credit amount. The most plausible explanation produced by DiCE (Mothilal et al., 2020) suggests decreasing the credit amount by almost a third while reducing the Duration of the loan to a single year. VAE and PROPLACE (Jiang et al., 2024) suggest decreasing the credit amount even further to below 3000. In contrast, MIO finds a counterfactual with the sought credit amount and suggests decreasing the Duration of the loan by only two months. Because the visualization is a 2-dimensional projection, some changes are not visualized. LiCE changes only one "hidden" feature (Installment rate as a percentage of disposable income). Additionally, all other methods change at least six features (except MIO, which changes two), showing poor sparsity.

This example illustrates the issue of considering solely plausibility. High plausibility should ensure that the counterfactual is not an outlier, i.e., it is "realizable" by the client. However, this can lead to non-sparse, distant CEs, which are nonetheless difficult to realize.

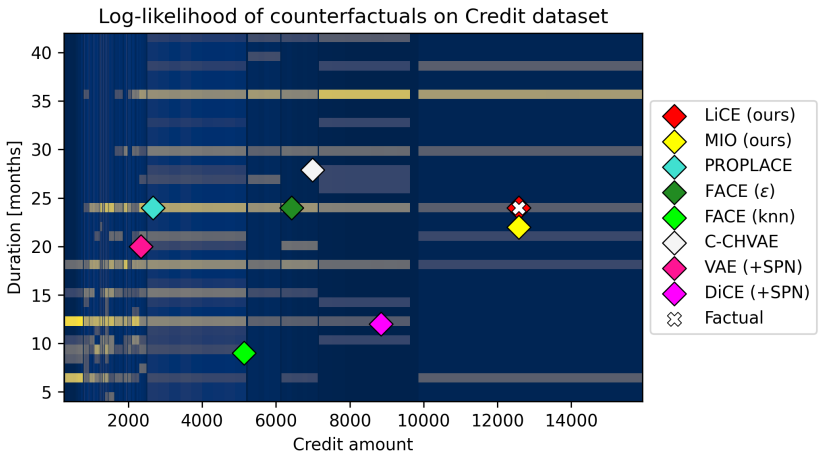

Figure 1: The heatmap shows the marginalized log-likelihood distribution of the German Credit dataset into a 2-dimensional space of Credit amount and Duration features, with extremely low values clipped to -35 for visual clarity. The factual (white cross) and CEs are also projected to the two dimensions. The factual is classified as being denied. Most CE methods choose distant points, sometimes with poor likelihood. The proposed method (LiCE) strikes a balance between likelihood and proximity.

Notation used Throughout the paper, we consider a classification problem where the dataset $\mathcal{D}$ is a set of 2-tuples $(\mathbf{x}, y) \in \mathcal{D}$. Each input vector $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^P$ consists of P features and is taken from the input space $\mathcal{X}$ that can be smaller than P-dimensional real space (e.g., can contain categorical values). x_j is the value of the j-th feature of the sample $\mathbf{x}$. We have C classes and describe the set of classes $[C] = \{1, \ldots, C\}$. $y \in [C]$ is the true class of the sample $\mathbf{x}$. Finally, we have a classifier $h(\mathbf{x}) = \hat{y} \in [C]$ that predicts the class $\hat{y}$ for the sample $\mathbf{x}$. More details on the notation are in Appendix A.1.

2 Prerequisites

2.1 Counterfactual Explanations

We define a counterfactual explanation in accordance with previous works as $\mathbf{x}' \in \mathcal{X}$ such that $h(\mathbf{x}) \neq h(\mathbf{x}')$ and the distance between $\mathbf{x}'$ and $\mathbf{x}$ is in some sense minimal (Guidotti, 2022; Wachter et al., 2017). We refer to $\mathbf{x}$ as factual and $\mathbf{x}'$ as counterfactual or CE. As mentioned above, there are many desiderata regarding the properties of CEs. Following Guidotti (2022), the common desiderata we are interested in are:

- *Validity.* $\mathbf{x}'$ should be classified differently than $\mathbf{x}$
- *Similarity.* $\mathbf{x}'$ should be similar (close) to $\mathbf{x}$
- *Sparsity.* $\mathbf{x}'$ should change only a few features compared to $\mathbf{x}$, i.e., minimize $\|\mathbf{x}' - \mathbf{x}\|_0$
- *Actionability.* A counterfactual should not change features that cannot be changed (immutability). This includes the monotonicity of some features, e.g., age can only increase.
- *Plausibility.* CEs should have a high likelihood (be plausible) with respect to distribution that has generated the dataset $\mathcal{D}$. This is sometimes interpreted as not being an outlier.

Guidotti (2022) describes also other desiderata, which we discuss in Appendix A.2

2.2 Mixed-Integer Optimization

Mixed-Integer Optimization (MIO, (Wolsey, 2020)) is a powerful framework for modeling and solving optimization problems, where some decision variables take values from a discrete set while others are continuously valued. Non-trivially, the problem is in NP (Papadimitriou, 1981) and

is NP-Hard, in general. There has been fascinating progress in the field in the past half-century (Bixby, 2012). State-of-the-art solvers based on the branch-and-bound-and-cut approach can often find global, certified optima for instances with millions of binary variables within hours, while there are pathological instances on under a thousand variables whose global optima are still unknown. Naturally, MIO is widely used in areas of machine learning, where both discrete and continuous decision variables need to be optimized jointly (e.g., Huchette et al., 2023). We use the more general abbreviation MIO, though we consider only mixed-integer *linear* formulations.

A crucial advance has been the mixed polytope formulation by Russell (2019), which neatly combines categorical and continuous values. A feature j takes a continuous value from the $[L_j, U_j]$ range or one of K_j distinct categorical values. This is useful for modeling data with missing values, especially when there is a description of why the value is missing (Russell, 2019). To model the mixed polytope (Russell, 2019) of a counterfactual for the feature j, we create a one-hot encoding for K_j discrete values into binary variables $d_{j,k}$ and a continuous variable c_j with a binary indicator variable d_j^{cont} equal to 1 when the feature takes a continuous value. In summary:

$$\sum_{k=1}^{K_j} d_{j,k} + d_j^{\text{cont}} = 1 \tag{1}$$

$$c_j = F_j d_j^{\text{cont}} - l_j + u_j \tag{2}$$

$$0 \leq l_j \leq (F_j - L_j) d_j^{\text{cont}} \tag{3}$$

$$0 \leq u_j \leq (U_j - F_j) d_j^{\text{cont}} \tag{4}$$

$$d_j^{\text{cont}}, d_{j,k} \in \{0, 1\} \qquad \forall k \in [K_j], \tag{5}$$

where F_j is either the original value x_j or the median value of continuous data of the mixed feature j if the factual x_j has one of the categorical values instead. Computing c_j in (2) uses two non-negative variables, l_j and u_j, representing the decrease and increase in the continuous value, respectively. This construction facilitates the computation of the absolute difference from the factual. Since we minimize their (weighted) sum, at least one of them will always equal 0 (Russell, 2019).

2.3 Sum-Product Networks

Probabilistic circuits (PCs) (Choi et al., 2020) are tractable probabilistic models (or rather, computational graphs) that support exact probabilistic inference and marginalization in time linear w.r.t. their representation size. Probabilistic circuits are defined by a tuple $(\mathcal{G}, \psi, \theta)$, where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a Directed Acyclic Graph (DAG) defining the computation model, a scope-function $\psi : \mathcal{V} \to 2^{[P]}$ defines a subset of features over which the node defines its distribution, and a set of parameters θ. The root node n^{root} (a node without parents) has the scope equal to all features, i.e., $\psi(n^{\text{root}}) = [P]$. To simplify the notation, we define a function $\text{pred}(n)$, giving a set of children (predecessors) of an inner node n and denote $x_{\psi(n)}$ the features of $\mathbf{x}$ within the scope of n.

An important subclass of PCs are Sum-Product Networks (SPNs), which restrict PCs such that the inner (non-leaf) nodes can be only sum nodes ($\mathcal{V}^{\Sigma}$) or product nodes ($\mathcal{V}^{\Pi}$).

Leaf node $n^{\text{L}} \in \mathcal{V}^{\text{L}} = \{n \,|\, \text{pred}(n) = \emptyset\}$ within SPNs takes a value $O_{n^{\text{L}}}$ from a (tractable) distribution over its scope $\psi(n^{\text{L}})$ parametrized by $\theta_{n^{\text{L}}}$.

Product node $n^{\Pi} \in \mathcal{V}^{\Pi}$ performs a product of probability distributions defined by its children

$$O_{n^{\Pi}}(x_{\psi(n)}) = \prod_{a \in \text{pred}(n^{\Pi})} O_a(x_{\psi(a)}). \tag{6}$$

The scope of product nodes must satisfy decomposability, meaning that the scopes of its children are disjoint, i.e., $\bigcap_{a \in \text{pred}(n^{\Pi})} \psi(a) = \emptyset$, but complete $\bigcup_{a \in \text{pred}(n^{\Pi})} \psi(a) = \psi(n^{\Pi})$.

Sum node $n^{\Sigma} \in \mathcal{V}^{\Sigma}$ has its value defined as

$$O_{n^{\Sigma}}(x_{\psi(n)}) = \sum_{a \in \text{pred}(n^{\Sigma})} w_{a,n^{\Sigma}} \cdot O_a(x_{\psi(a)}), \tag{7}$$

where weights $w_{a,n^{\Sigma}} \geq 0$ and $\sum_{a \in \text{pred}(n^{\Sigma})} w_{a,n^{\Sigma}} = 1$. The value of a sum node is thus a mixture of distributions defined by its children. The scope of each sum node must satisfy completeness (smoothness), i.e., it must hold that $\psi(a_1) = \psi(a_2)\ \forall a_1, a_2 \in \text{pred}(n^{\Sigma})$.

3 Related Work

Pioneering work on counterfactual explanations (under the name "optimal action extraction") by Cui et al. (2015) considered classifiers based on additive tree models and extracted an optimal plan to change a given input to a desired class at a minimum cost using MIO. In parallel, similar approaches have been developed under the banner of "actionable recourse" (Ustun et al., 2019) or "algorithmic recourse" (Karimi et al., 2020; 2021). Developing upon this, Karimi et al. (2021) distinguish between contrasting explanations and consequential explanations, where actions are modeled explicitly in a causal model. We use the term counterfactual explanations (CEs), popularized by, e.g., Wachter et al. (2017).

There is a plethora of work on the search for CEs, as recently surveyed (e.g., Karimi et al., 2020; Burkart & Huber, 2021; Guidotti, 2022; Bodria et al., 2023). Below, we focus on methods with objectives related to the plausibility of the CE.

DACE (Kanamori et al., 2020) utilizes an MIO formulation, minimizing a combination of ℓ_1-norm based Mahalanobis' distance and 1-Local Outlier Factor for plausibility. The use of 1-LOF requires the use of $\mathcal{O}(|\mathcal{D}|)$ variables and $\mathcal{O}(|\mathcal{D}|^2)$ constraints. We improve on DACE by formulating the SPN as MIO to compute the likelihood. The number of variables and constraints does not depend on the dataset size, but rather on the size of the SPN. Moreover, our flexible formulation allows us to maximize plausibility or just require the CE not to be an outlier, as does DACE. **PROPLACE** (Jiang et al., 2024) is also an MIO-based method for finding robust CEs within a "plausible region". The region is constructed as a convex hull of the factual and its (robust) nearest neighbors. The neighbors can, however, be outliers if a factual is not in a dense region of the data. This approach is, therefore, not faithful to the definition of plausibility we use (Section 2.1). **FACE** (Poyiadzi et al., 2020) selects a CE from the training set $\mathcal{D}$, rather than generating it from $\mathcal{X}$. It works by navigating a graph of samples $\mathbf{x} \in \mathcal{D}$, where an edge exists between 2 samples if they are close or by connecting k-nearest neighbors. It further requires that a sample has density (evaluated by KDE) above a certain threshold. This approach is limited by the inability to generate exogenous CEs, which is not the case for our method.

Similar to LiCE, some works estimate the data distribution via a probabilistic model, such as Variational Auto-Encoder (VAE, Mahajan et al. (2020)). **C-CHVAE** (Pawelczyk et al., 2020) uses a Conditional VAE to search for plausible (they use the term *faithful*) CEs without the need of a metric in the original space. However, VAE provide only a lower bound on likelihood and it is non-trivial to formulate within MIO, therefore the solution lacks any guarantees on optimality. **PlaCE** (Artelt & Hammer, 2020) uses Gaussian Mixture Model (GMM) to represent the data distribution. Their formulation approximates a GMM by a quadratic term and uses a general convex optimization solver. However, GMMs cannot handle categorical features, which are frequent in datasets of interest. LiCE uses SPNs, probabilistic models that are tractable, naturally handle categorical features, and can have linear MIO formulation. SPNs are a strict generalization of GMMs (Aden-Ali & Ashtiani, 2020). We refer to Appendix A.8.3 for further discussion on using SPNs.

4 Mixed-Integer Formulation of SPN

Our contribution is built on a novel formulation of the likelihood estimates provided by a Sum-Product Network (SPN) in Mixed-Integer Optimization (MIO). Eventually, this makes it possible to utilize the estimate to ensure plausible counterfactuals generated using MIO. Specifically, we propose an MIO formulation for a log-space variant of a *fitted* SPN (Poon & Domingos, 2011) with fixed parameters. We perform all computations in log-space because it enables the approximation of both sum and product nodes by linear constraints. Additionally, it makes the optimization less prone to numerical instabilities.

Let us introduce the MIO formulation following the definition of SPN in Section 2.3:

Leaf nodes In any SPN, leaves are represented by probability distributions over a single feature. In the case of discrete random variables, we can utilize the indicator $d_{j,k}$ that feature j has value k in the one-hot encoding. In the case of continuous random variables, we can utilize histogram approximations, i.e., piece-wise linear functions, whose mixed-integer formulations have been studied in

considerable detail (cf. Huchette & Vielma, 2023). We suggest and utilize an alternative formulation of a histogram described in Appendix A.4.2.

Product nodes In any SPN, each node n combines the outputs $o_a,\ a \in \text{pred}(n)$ of its predecessors. Consider now a product node $n \in \mathcal{V}^{\Pi}$, with output defined as a product of predecessor outputs. Since we consider all computations in log space, this translates to

$$o_n = \sum_{a \in \text{pred}(n)} o_a \quad \forall n \in \mathcal{V}^{\Pi}. \tag{8}$$

Sum nodes Sum node $n \in \mathcal{V}^{\Sigma}$ is defined as a weighted sum of predecessor a outputs. In log space, the sum would translate to $o_n^* = \log \sum_{a \in \text{pred}(n)} w_{a,n} \exp(o_a)$, which we cannot formulate as a linear expression easily. When considering $w_{a,n} \exp(o_a) = \exp(o_a + \log w_{a,n})$, we can approximate $\log \sum \exp(z)$ by $\max z$. Specifically, let $z_a = o_a + \log w_{a,n}$ and we bound

$$\begin{aligned}
\max_{a \in \text{pred}(n)} z_a &= \log \exp(\max_{a \in \text{pred}(n)} z_a) \\
&\leq \log \sum_{a \in \text{pred}(n)} \exp(z_a) = o_n^* \\
&\leq \log \left(|\text{pred}(n)| \exp(\max_{a \in \text{pred}(n)} z_a) \right) = \log(|\text{pred}(n)|) + \max_{a \in \text{pred}(n)} z_a.
\end{aligned}$$

In other words, the approximate value o_n of a sum node n can be bound by the true value o_n^* as

$$o_n^* - \log(|\text{pred}(n)|) \leq o_n = \max_{a \in \text{pred}(n)} z_a \leq o_n^*,$$

meaning that our approximation is a lower bound of the true o_n^*, and the error in the estimate is at the most logarithm of the number of predecessors. If we wanted an upper bound, we could easily add $\log(|\text{pred}(n)|)$ to the value o_n. To formulate the max function, we can linearize it by introducing slack binary indicators $m_{a,n} \in \{0, 1\}$ for each predecessor a of sum node n

$$o_n \leq o_a + \log w_{a,n} + m_{a,n} \cdot T_n^{\text{LL}} \quad \forall n \in \mathcal{V}^{\Sigma}, \forall a \in \text{pred}(n) \tag{9}$$

$$\sum_{a \in \text{pred}(n)} m_{a,n} = |\text{pred}(n)| - 1 \quad \forall n \in \mathcal{V}^{\Sigma} \tag{10}$$

where T_n^{LL} is a big enough "big-M" constant (Wolsey, 2020). Constraint (10) ensures that constraint (9) is tight ($o_n \leq z_a$) for a single predecessor a for which $m_a = 0$. Since we maximize the likelihood, the value of o_n will be equal to the $\max_a z_a$.

5 Likely Counterfactual Explanations

As our main contribution, we present a novel formulation for Likely Counterfactual Explanations (LiCE), which finds plausible CEs (with high likelihood), while satisfying common desiderata. Since the optimization problem is written as MIO, the solution (CE) satisfies all constraints and is globally optimal. Throughout the section, we assume that all continuous values are normalized to the range $[0, 1]$.

We now describe how we formulate the input encoding, classification model, and various desiderata as MIO constraints. The potential of MIO to formulate similar constraints is well discussed in the literature (e.g., Russell, 2019; Kanamori et al., 2020; Mohammadi et al., 2021; Jiang et al., 2024), though the discussion rarely contains concrete formulations (Parmentier & Vidal, 2021). We discuss MIO formulations of the desiderata specific for the mixed polytope input encoding (Russell, 2019) in Appendix A.3.

Input encoding To encode the input vector, we utilize the mixed polytope formulation (Russell, 2019), as explained in Eqs. 1–5 on page 4. The mixed polytope encoding works for purely continuous values by setting $K_j = 0$. For fully categorical features, one must disregard the d_j^{cont} variable as explained in more detail in Appendix A.4.1.

The input to the classification model (and to the SPN) is then a set of all variables c_j and $d_{j,k}$ (but not d_j^{cont}) concatenated into a single vector. With some abuse of the notation, we denote this vector $\mathbf{x}'$. When there is no risk of confusion, we denote the space of encoded inputs as $\mathcal{X}$ and the number of features after encoding as P.

Model formulation We encode the classification model using OMLT library (Ceccon et al., 2022) which simplifies the formulation of various ML models, though we focus on Neural Networks. Linear combinations in layers are modeled directly, while ReLUs are modeled using big-M formulations, though other formulations are possible (Fischetti & Jo, 2018).

Validity Let $h^{\text{raw}} : \mathcal{X} \to \mathcal{Z}$ be the neural network model $h(\cdot)$ without activation at the output layer. Let $h^{\text{raw}}(\mathbf{x}')$ be the result obtained from the model implementation. Assuming that we have a binary classification task ($C = 2$), a neural network typically has a single output neuron ($\mathcal{Z} = \mathbb{R}$). A sample $\mathbf{x}$ is classified based on whether the raw output is above or below 0, i.e., $h(\mathbf{x}) = \mathbb{1}\{h^{\text{raw}}(\mathbf{x}) \geq 0\}$. Thus, depending on whether the factual is classified as 0 or 1, we set

$$h^{\text{raw}}(\mathbf{x}') \geq \tau \text{ or } h^{\text{raw}}(\mathbf{x}') \leq -\tau, \tag{11}$$

respectively, where $\tau \geq 0$ is a margin that can be set to ensure a higher certainty of the decision, improving the reliability of the CE. We present further specifications of validity for $C > 2$ in Appendix A.3.1.

Similarity and Sparsity To ensure similarity of the counterfactual, we follow Wachter et al. (2017) and Russell (2019) and use the somewhat non-standard $\|\cdot\|_{1,\text{MAD}}$ norm, weighed by inverse Median Absolute Deviation (MAD)

$$\|\mathbf{x}\|_{1,\text{MAD}} = \sum_{j=1}^{P} \left| \frac{x_j}{\text{MAD}_j} \right| \tag{12}$$

$$\text{MAD}_j = \text{median}_{(\mathbf{x},\cdot)\in\mathcal{D}} \left(|x_j - \text{median}_{(\mathbf{x},\cdot)\in\mathcal{D}}(x_j)| \right).$$

This metric also improves sparsity and adds scale invariance that is robust to outliers (Russell, 2019).

Actionability We call a CE actionable if it satisfies monotonicity and immutability constraints. For immutability, the constraint is simply $x_j = x'_j$ for each immutable feature j. We can also set the input value as a parameter instead of a variable, omitting the feature encoding. Modeling monotonicity, i.e., that a given value cannot decrease/increase, is done using a single inequality for continuous features, e.g., $l_j = 0$ for a non-decreasing feature. For ordinal values, we fix to zero all one-hot dimensions representing smaller ordinal values, for non-decreasing features. Similarly, we can enforce basic causality constraints. Details are provided in Appendix A.3.3.

Plausibility As explained in Section 4, fixed SPN fitted on data allows us to estimate likelihood within MIO formulation. The negative likelihood can be added to the minimization objective with some multiplicative coefficient $\alpha > 0$. Alternatively, the likelihood can be used in constraints to force all generated CEs to have likelihood above a certain threshold δ^{SPN}. Such constraint is simply

$$o_{n^{\text{root}}} \geq \delta^{\text{SPN}}, \tag{13}$$

where δ^{SPN} is a hyperparameter of our method, and $o_{n^{\text{root}}}$ is the likelihood estimated by the SPN.

Full LiCE model In summary, our method optimizes the following problem:

$$\begin{aligned} \arg\min_{\mathbf{l},\mathbf{u},\mathbf{d}} \; & (\mathbf{l}+\mathbf{u})^\top \mathbf{v}^{\text{cont}} + (\mathbf{d} - \mathbf{d}^{\text{fact}})^\top \mathbf{v}^{\text{bin}} - \alpha \cdot o_{n^{\text{root}}} \\ \text{s.t. } & \text{mixed polytope conditions (1–5) hold} \\ & \text{ML classifier constraints hold} \\ & \text{validity constraints (e.g., 11) hold} \\ & \text{SPN constraints (8–10) hold} \\ & \text{plausibility constraint (13) holds} \\ & \text{data-specific desiderata (e.g., actionability) constraints hold,} \end{aligned} \tag{14}$$

Table 2: Approximation quality of the SPN. The first row shows the mean likelihood of CEs, evaluated by the SPN. The second row is the mean output of the MIO formulation of the same SPN. The third row shows the mean difference.

	GMSC	Adult	Credit
True SPN output	-25.62 ± 4.42	-18.10 ± 3.77	-29.05 ± 3.32
MIO formulation output ($o_{n^{\text{root}}}$)	-25.71 ± 4.47	-18.62 ± 3.97	-29.28 ± 3.39
SPN approximation error	0.09 ± 0.35	0.52 ± 0.45	0.22 ± 0.23

where α is a parameter of LiCE, weighing the influence of log-likelihood in the objective, $\mathbf{l}$, $\mathbf{u}$ and $\mathbf{d}$ represent the vectors obtained by concatenation of the parameters in Eqs. 1–5. The vector $\mathbf{d}^{\text{fact}}$ is the vector of binary variables of the encoded factual $\mathbf{x}$. $\mathbf{v}^{\text{cont}}$ and $\mathbf{v}^{\text{bin}}$ represent weights for continuous and binary variables, respectively. The weights for feature j are $1/\text{MAD}_j$ and thus Eq. 14 (when $\alpha = 0$) corresponds to Eq. 12. Details about the data-specific constraints are in Appendix A.6.1.

6 Experiments

We first train a basic feed-forward Neural Network (NN) classifier with 2 hidden layers with ReLU activations. One could easily use one of the variety of ML models that can be formulated using MIO, including linear models, (gradient-boosted) trees, forests, or graph neural networks.

Secondly, we train an SPN to model the likelihood on the same training dataset. We include the class y of a sample $\mathbf{x}$ into the training since we have prior knowledge of the counterfactual class. SPNs have a variety of methods for training (Xia et al., 2023), out of which we use a variant of LearnSPN (Gens & Domingos, 2013) implemented in the SPFlow library (Molina et al., 2019).

Data We performed tests on the Give Me Some Credit (GMSC) dataset (Credit Fusion, 2011), Adult dataset (Becker & Kohavi, 1996) and German Credit (referred to as Credit) dataset (Hofmann, 1994). We dropped some outlier data and some less informative features (details in Appendix A.5) and performed all experiments in a 5-fold cross-validation setting.

LiCE variants The main proposed model reflects directly the formulation (14). We compare two variants, one with a lower-bound on log-likelihood (δ^{SPN}) at the median log-likelihood value on training data, similarly to Artelt & Hammer (2020). We also set $\alpha = 0$, to minimize purely the distance to factual. We refer to this as **LiCE (median)**. The other variant, **LiCE (optimize)**, is the opposite, i.e., we optimize a combination of distance and likelihood with $\alpha = 0.1$ and relax the plausibility constraint (Eq. 13). **MIO** represents our method without the SPN model directly formulated. We use all constraints described in Section 5, without the plausibility and SPN constraints. We use the SPN post hoc to select the most likely explanation.

MIO and LiCE are implemented using the open-source Pyomo modeling library (Bynum et al., 2021) that allows for simple use of (almost) any MIO solver. We use the Gurobi solver (Gurobi Optimization, LLC, 2023). We solve each formulation for up to 2 minutes, after which we recover (up to) 10 best solutions. The entire implementation, together with data, is part of the supplemental material and will be made open source.

Compared methods We compare our methods to the **C-CHVAE** (Pawelczyk et al., 2020), **FACE** (Poyiadzi et al., 2020) and **PROPLACE** (Jiang et al., 2024) methods described in Section 3. We use the implementations of FACE and C-CHVAE provided in the CARLA library (Pawelczyk et al., 2021). We run FACE in two variants, connecting samples within a given distance (ϵ) or by nearest neighbors (knn). For PROPLACE, we use the official implementation (Jiang et al., 2024). We omit PlaCE and DACE since their implementation does not support CE generation for Neural Networks.

In addition to those, we also compare to **DiCE** (Mothilal et al., 2020), a well-known method focusing on generating a diverse set of counterfactuals. **VAE** is a method using Variational Auto-Encoder. It is an implementation available in version 0.4 of the DiCE library based on the work of Mahajan et al. (2020). For DiCE and VAE, we select the most likely CE out of 10 generated CEs.

Table 3: The proportion of factual instances for which a given method generated a valid or actionable counterfactual. Actionable CEs satisfy the immutability and monotonicity of relevant features (see Section 2.1).

		GMSC (Credit Fusion, 2011)		**Adult** (Becker & Kohavi, 1996)		**Credit** (Hofmann, 1994)	
	Method	Valid	Actionable	Valid	Actionable	Valid	Actionable
	DiCE	**100.0%**	**100.0%**	99.8%	56.2%	98.4%	3.4%
	VAE	1.4%	0.2%	75.4%	10.2%	27.2%	0.0%
	C-CHVAE	98.6%	21.6%	16.8%	8.6%	11.0%	8.8%
	FACE (ϵ)	98.6%	13.2%	62.0%	19.6%	27.2%	10.0%
	FACE (knn)	98.6%	16.2%	79.4%	28.4%	27.2%	8.8%
	PROPLACE	98.6%	6.6%	79.4%	6.6%	27.2%	11.8%
ours	**MIO**	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**
ours	**LiCE** (optimize)	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**
ours	**LiCE** (median)	55.6%	55.6%	91.2%	91.2%	**100.0%**	**100.0%**

Table 4: Mean negative log-likelihood (NLL), $\|\cdot\|_{1,\text{MAD}}$ distance, and the number of changed features, measured on *valid* generated counterfactuals, with information about standard deviation. The log-likelihood is estimated by the SPN. The number of valid counterfactuals generated by a given method varies (see Table 3), so the direct comparison between methods is non-trivial. The (+spn) means that the given method generates 10 CEs from which we choose the likeliest valid counterfactual using the SPN. For all measures, a lower value is better.

		GMSC (Credit Fusion, 2011)			**Adult** (Becker & Kohavi, 1996)			**Credit** (Hofmann, 1994)		
	Method	NLL ↓	Similarity ↓	Sparsity ↓	NLL ↓	Similarity ↓	Sparsity ↓	NLL ↓	Similarity ↓	Sparsity ↓
	DiCE (+spn)	30.5 ±3.7	20.7 ±5.2	6.4 ±1.0	20.5 ±3.0	23.5 ±9.9	4.6 ±1.7	51.6 ±17.9	28.1 ±7.9	8.7 ±2.2
	VAE (+spn)	23.1 ±12.6	15.4 ±4.4	9.1 ±0.8	17.1 ±3.0	33.3 ±10.9	5.5 ±1.5	49.0 ±17.8	28.8 ±7.8	10.9 ±1.8
	C-CHVAE	24.9 ±2.4	17.4 ±4.7	9.0 ±0.0	17.9 ±3.2	8.6 ±6.3	3.0 ±1.0	34.0 ±6.5	13.5 ±4.9	7.6 ±1.0
	FACE (ϵ)	30.0 ±9.0	14.8 ±3.8	8.5 ±1.1	16.1 ±3.0	14.6 ±8.4	3.7 ±1.2	46.4 ±17.5	18.1 ±6.2	7.0 ±1.2
	FACE (knn)	29.5 ±8.2	14.6 ±3.8	8.4 ±1.1	15.6 ±3.1	14.1 ±8.0	3.7 ±1.2	44.4 ±17.8	18.5 ±6.2	7.1 ±1.3
	PROPLACE	27.3 ±5.7	12.1 ±3.1	7.4 ±1.1	17.6 ±3.3	22.0 ±8.0	4.8 ±1.2	41.2 ±17.0	24.5 ±6.3	8.7 ±1.3
ours	MIO (+spn)	27.8 ± 6.4	**5.9 ± 1.8**	**2.2 ± 0.8**	17.9 ± 3.7	5.7 ± 3.6	2.2 ± 0.9	47.6 ± 18.2	**4.4 ± 2.7**	2.2 ± 1.0
ours	LiCE (optim.)	25.6 ± 4.4	**5.9 ± 1.8**	2.6 ± 1.1	18.1 ± 3.8	**5.5 ± 3.6**	**2.0 ± 1.0**	**29.1 ± 3.3**	**4.4 ± 2.7**	2.1 ± 1.0
ours	LiCE (median)	**18.1 ± 2.6**	10.6 ± 3.5	4.4 ± 1.1	**12.9 ± 1.0**	9.7 ± 6.5	2.9 ± 1.3	29.8 ± 3.1	**4.4 ± 2.7**	**2.0 ± 1.1**

If a CE method requires any prior training, we use the default hyperparameters (or some reasonable values, details in Appendix A.6.2) and train it on the same training set. If a given method can take into account actionability constraints, we enforce them.

Experimental settings For all experiments, we assume the SPN and the NN are fitted and fixed. We generate CEs for 100 factuals by each method for each fold, summing up to 500 factuals per dataset. The factuals are randomly selected from both classes. Methods that can output more CEs (MIO, LiCE, DiCE, VAE) are set to find at most 10 CEs and we select valid CE with the highest likelihood (evaluated by the SPN) post-hoc. Further details about hyperparameters and experiment configurations are in Appendix A.6.

Results To assess the quality of the MIO approximation of the SPN, we compare the CE likelihood computed by the MIO solver and the true value computed by SPN in Table 2. The worst approximation error is at 0.52 on average, amounting to just 2.85%. We find this surprisingly tight. Moreover, considering the differences between methods (cf. Table 4), this is acceptable.

The comparison of the CE methods is non-trivial since factuals for which a given method successfully returned a valid counterfactual are not the same for all methods. See Table 3 for details on the success rate of the presented methods. For LiCE (median), the lower rates are caused by a failure to create a counterfactual candidate in time. For other methods, it is also a failure to follow the validity/actionability criteria, especially for the case when a valid CE exists but an actionable does not. Overall, these results show that MIO-based methods have a high success rate unless the constraints are too tight. Methods unconstrained by the likelihood, i.e., MIO and LiCE (optimize), have a 100% success rate. For our methods, all generated CEs are guaranteed to be both valid and actionable.

We now compare the CE methods on plausibility, similarity, and sparsity measured by negative log-likelihood (evaluated by the SPN), $\|\cdot\|_{1,\text{MAD}}$, and by the number of modified features, respectively, see Table 4. The results are difficult to interpret since not every method produced a valid CE for each factual. Nevertheless, MIO and LiCE have success rates among the highest (cf. Table 3) and

still perform best not only with regards to Likelihood but also in terms of similarity and sparsity. Results on a subset of factuals for which each method generated a valid CE paint a similar picture, see Table 11 in Appendix A.7.2.

The plausibility results of LiCE (median) seem to be unparalleled. Part of the improvement could be explained by the method finishing in time mainly on only the easier instances. Interestingly, the optimizing variant of LiCE achieves better mean objective value on Adult and GMSC than the median variant, despite the median version's objective function not accounting for the NLL. This is partly because LiCE (optimize) has a bigger feasible space, allowing it to generate closer CEs with a likelihood worse than the median of the training set. The fact LiCE (optimize) beats MIO on Adult on similarity (which MIO directly optimizes) is counterintuitive. It is caused by choosing the likeliest CE out of a set of 10. This set includes local optima that are farther from the factual but might have a higher likelihood.

While for some datasets, the plausibility results are comparable between multiple methods, the similarity and sparsity remain dominated by our methods. We must also point out that merely adding the SPN as a post-hoc evaluation to some existing method (e.g., DiCE) performs significantly worse. Further comparisons and discussion of the results are in Appendices A.7 and A.8.

Limitations Our method shares the limitations of all MIO methods with respect to scalability and computational complexity. The additional SPN formulation leads to some computational overhead, especially when using the likelihood threshold, as exemplified in the results on the GMSC dataset in Table 3.

Our method relies on an SPN to evaluate likelihood, i.e., plausibility. One may question using an SPN to model the data distribution accurately. We show empirically strong correlation between SPN likelihood and true probability in Appendix A.8.4. Furhter, in Appendix A.8.5, we show that the true probability of CEs generated by LiCE is comparable to the probability of CEs generated by other well performing methods.

7 Discussion and Conclusions

We have presented a comprehensive method for generating counterfactual explanations called LiCE. In Section 5, we show that our method satisfies the most common desiderata–namely validity, similarity, sparsity, actionability, and most importantly, *plausibility*.

Our method shows promising performance at the intersection of plausibility, similarity, and sparsity. It also reliably generates high-quality, valid, and actionable CEs. However, time concerns are relevant once the full SPN is formulated within the model.

In future work, the limitations of using MIO could be addressed by approximation algorithms. Additionally, other SPN-based models (e.g., Trapp et al., 2018) could be considered for estimating plausibility.

Last but not least, the MIO formulation of a Sum-Product Network can be of independent interest.

References

Ishaq Aden-Ali and Hassan Ashtiani. On the sample complexity of learning sum-product networks. In Silvia Chiappa and Roberto Calandra (eds.), *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 4508–4518. PMLR, 26–28 Aug 2020. URL https://proceedings.mlr.press/v108/aden-ali20a.html.

André Artelt and Barbara Hammer. Convex Density Constraints for Computing Plausible Counterfactual Explanations. In Igor Farkaš, Paolo Masulli, and Stefan Wermter (eds.), *Artificial Neural Networks and Machine Learning – ICANN 2020*, volume 12396, pp. 353–365. Springer International Publishing, Cham, 2020. ISBN 978-3-030-61608-3 978-3-030-61609-0. doi: 10.1007/978-3-030-61609-0_28.

Andre 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), pp. 01–09, Orlando, FL, USA, December 2021. IEEE. ISBN 978-1-72819-048-8. doi: 10.1109/SSCI50451.2021.9660058.

Barry Becker and Ronny Kohavi. Adult. UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5XW20.

Robert E Bixby. A brief history of linear and mixed-integer programming computation. *Documenta Mathematica*, 2012:107–121, 2012.

Francesco Bodria, Fosca Giannotti, Riccardo Guidotti, Francesca Naretto, Dino Pedreschi, and Salvatore Rinzivillo. Benchmarking and survey of explanation methods for black box models. *Data Mining and Knowledge Discovery*, pp. 1–60, 2023.

Dieter Brughmans, Lissa Melis, and David Martens. Disagreement amongst counterfactual explanations: How transparency can be misleading. *TOP*, May 2024. ISSN 1863-8279. doi: 10.1007/s11750-024-00670-2.

Nadia Burkart and Marco F Huber. A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research*, 70:245–317, 2021.

Michael L Bynum, Gabriel A Hackebeil, William E Hart, Carl D Laird, Bethany L Nicholson, John D Siirola, Jean-Paul Watson, David L Woodruff, et al. *Pyomo - Optimization Modeling in Python, 3rd Edition*, volume 67. Springer, 2021.

Ruth M. J. Byrne. *The Rational Imagination: How People Create Alternatives to Reality*. The MIT Press, June 2005. ISBN 978-0-262-26962-9. doi: 10.7551/mitpress/5756.001.0001.

F. Ceccon, J. Jalving, J. Haddad, A. Thebelt, C. Tsay, C. D Laird, and R. Misener. OMLT: Optimization & machine learning toolkit. *Journal of Machine Learning Research*, 23(349):1–8, 2022.

YooJung Choi, Antonio Vergari, and Guy Van den Broeck. Probabilistic circuits: A unifying framework for tractable probabilistic models. *UCLA*, October 2020.

Will Cukierski Credit Fusion. Give me some credit, 2011.

Zhicheng Cui, Wenlin Chen, Yujie He, and Yixin Chen. Optimal Action Extraction for Random Forests and Boosted Trees. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 179–188, Sydney NSW Australia, August 2015. ACM. ISBN 978-1-4503-3664-2. doi: 10.1145/2783258.2783281.

Rudresh Dwivedi, Devam Dave, Het Naik, Smiti Singhal, Rana Omer, Pankesh Patel, Bin Qian, Zhenyu Wen, Tejal Shah, Graham Morgan, and Rajiv Ranjan. Explainable AI (XAI): Core Ideas, Techniques, and Solutions. *ACM Computing Surveys*, 55(9):194:1–194:33, January 2023. ISSN 0360-0300. doi: 10.1145/3561048.

Equal Credit Opportunity Act (ECOA). Equal Credit Opportunity Act (ECOA). `https://www.law.cornell.edu/uscode/text/15/chapter-41/subchapter-IV`, 1974. Title 15 of the United States Code, Chapter 41, Subchapter IV, paragraph 1691 and following.

European Commission. Directive 2013/36/EU of the European Parliament and of the Council of 26 june 2013 on access to the activity of credit institutions and the prudential supervision of credit institutions and investment firms, amending Directive 2002/87/EC and repealing Directives 2006/48/EC and 2006/49/ec., 2016a. URL `https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex%3A32013L0036`. Accessed: 2023-04-30.

European Commission. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)., 2016b. URL `https://eur-lex.europa.eu/eli/reg/2016/679/oj`. Accessed: 2023-04-30.

Matteo Fischetti and Jason Jo. Deep neural networks and mixed integer linear optimization. *Constraints*, 23(3):296–309, July 2018. ISSN 1572-9354. doi: 10.1007/s10601-018-9285-6.

Robert Gens and Pedro Domingos. Learning the structure of sum-product networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pp. III–873–III–880. JMLR.org, 2013.

Noam Goldberg, Steffen Rebennack, Youngdae Kim, Vitaliy Krasko, and Sven Leyffer. MINLP formulations for continuous piecewise linear function fitting. *Computational Optimization and Applications*, 79(1):223–233, May 2021. ISSN 1573-2894. doi: 10.1007/s10589-021-00268-5.

Riccardo Guidotti. Counterfactual explanations and how to find them: Literature review and benchmarking. *Data Mining and Knowledge Discovery*, April 2022. ISSN 1573-756X. doi: 10.1007/s10618-022-00831-6.

David Gunning. Broad agency announcement explainable artificial intelligence (xai). Technical report, DARPA, August 2016.

Gurobi Optimization, LLC. Gurobi optimizer reference manual, 2023.

Hans Hofmann. Statlog (German Credit Data), 1994.

Joey Huchette and Juan Pablo Vielma. Nonconvex Piecewise Linear Functions: Advanced Formulations and Simple Modeling Tools. *Operations Research*, 71(5):1835–1856, September 2023. ISSN 0030-364X. doi: 10.1287/opre.2019.1973.

Joey Huchette, Gonzalo Muñoz, Thiago Serra, and Calvin Tsay. When deep learning meets polyhedral theory: A survey. *arXiv preprint arXiv:2305.00241*, 2023.

Junqi Jiang, Jianglin Lan, Francesco Leofante, Antonio Rago, and Francesca Toni. Provably Robust and Plausible Counterfactual Explanations for Neural Networks via Robust Optimisation. In *Proceedings of the 15th Asian Conference on Machine Learning*, pp. 582–597. PMLR, February 2024.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Learning in graphical models*, pp. 105–161, 1998.

Kentaro Kanamori, Takuya Takagi, Ken Kobayashi, and Hiroki Arimura. DACE: Distribution-Aware Counterfactual Explanation by Mixed-Integer Linear Optimization. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pp. 2855–2862, Yokohama, Japan, July 2020. International Joint Conferences on Artificial Intelligence Organization. ISBN 978-0-9992411-6-5. doi: 10.24963/ijcai.2020/395.

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.

Amir-Hossein Karimi, Bernhard Schölkopf, and Isabel Valera. Algorithmic recourse: from counterfactual explanations to interventions. In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pp. 353–362, 2021.

Bo Li, Peng Qi, Bo Liu, Shuai Di, Jingen Liu, Jiquan Pei, Jinfeng Yi, and Bowen Zhou. Trustworthy AI: From Principles to Practices. *ACM Computing Surveys*, 55(9):177:1–177:46, January 2023. ISSN 0360-0300. doi: 10.1145/3555803.

Divyat Mahajan, Chenhao Tan, and Amit Sharma. Preserving Causal Constraints in Counterfactual Explanations for Machine Learning Classifiers, June 2020.

Donato Maragno, Jannis Kurtz, Tabea E. Röber, Rob Goedhart, Ş Ilker Birbil, and Dick den Hertog. Finding Regions of Counterfactual Explanations via Robust Optimization, May 2023.

Kiarash Mohammadi, Amir-Hossein Karimi, Gilles Barthe, and Isabel Valera. Scaling Guarantees for Nearest Counterfactual Explanations. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '21, pp. 177–187, New York, NY, USA, July 2021. Association for Computing Machinery. ISBN 978-1-4503-8473-5. doi: 10.1145/3461702.3462514.

Alejandro Molina, Antonio Vergari, Karl Stelzner, Robert Peharz, Pranav Subramani, Nicola Di Mauro, Pascal Poupart, and Kristian Kersting. Spflow: An easy and extensible library for deep probabilistic learning using sum-product networks, 2019.

Ramaravind Kommiya 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*, pp. 607–617, January 2020. doi: 10.1145/3351095.3372850.

Hien D. Nguyen and Geoffrey McLachlan. On approximations via convolution-defined mixture models. *Communications in Statistics - Theory and Methods*, 48(16):3945–3955, 2019. doi: 10.1080/03610926.2018.1487069. URL https://doi.org/10.1080/03610926.2018.1487069.

Christos H Papadimitriou. On the complexity of integer programming. *Journal of the ACM (JACM)*, 28(4):765–768, 1981.

Axel Parmentier and Thibaut Vidal. Optimal Counterfactual Explanations in Tree Ensembles. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 8422–8431. PMLR, July 2021.

Martin Pawelczyk, Klaus Broelemann, and Gjergji Kasneci. Learning Model-Agnostic Counterfactual Explanations for Tabular Data. In *Proceedings of The Web Conference 2020*, WWW '20, pp. 3126–3132, New York, NY, USA, April 2020. Association for Computing Machinery. ISBN 978-1-4503-7023-3. doi: 10.1145/3366423.3380087.

Martin Pawelczyk, Sascha Bielawski, Johannes van den Heuvel, Tobias Richter, and Gjergji Kasneci. CARLA: A python library to benchmark algorithmic recourse and counterfactual explanation algorithms, 2021.

Hoifung Poon and Pedro Domingos. Sum-product networks: A new deep architecture. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pp. 689–690, November 2011. doi: 10.1109/ICCVW.2011.6130310.

Rafael Poyiadzi, Kacper Sokol, Raul Santos-Rodriguez, Tijl De Bie, and Peter Flach. FACE: Feasible and Actionable Counterfactual Explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pp. 344–350, February 2020. doi: 10.1145/3375627.3375850.

Chris Russell. Efficient Search for Diverse Coherent Explanations. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, pp. 20–28, New York, NY, USA, January 2019. Association for Computing Machinery. ISBN 978-1-4503-6125-5. doi: 10.1145/3287560.3287569.

Marco Scutari. Bnlearn: Bayesian Network Structure Learning, Parameter Learning and Inference, September 2007.

Erdogan Taskesen. Learning Bayesian Networks with the bnlearn Python Package., January 2020. URL https://erdogant.github.io/bnlearn.

Martin Trapp, Robert Peharz, Carl E Rasmussen, and Franz Pernkopf. Learning deep mixtures of gaussian process experts using sum-product networks. *arXiv preprint arXiv:1809.04400*, 2018.

Berk Ustun, Alexander Spangher, and Yang Liu. Actionable Recourse in Linear Classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pp. 10–19, Atlanta GA USA, January 2019. ACM. ISBN 978-1-4503-6125-5. doi: 10.1145/3287560.3287566.

Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual Explanations Without Opening the Black Box: Automated Decisions and the GDPR. *SSRN Electronic Journal*, 2017. ISSN 1556-5068. doi: 10.2139/ssrn.3063289.

H Paul Williams. *Model building in mathematical programming*. John Wiley & Sons, 2013.

Laurence A Wolsey. *Integer programming*. John Wiley & Sons, 2020.

Riting Xia, Yan Zhang, Xueyan Liu, and Bo Yang. A survey of sum–product networks structural learning. *Neural Networks*, 2023.

Table 5: General functions used

General function symbols	
$\lvert \cdot \rvert$	Absolute value (if scalar) or size of the set
$[\cdot]$	Set of integers, $[N] = \{1, 2, \ldots, N\}$
$\mathbb{1}\{\cdot\}$	Equal 1 if input is true, 0 otherwise
$\lVert \cdot \rVert_0$	ℓ_0 norm, number of non-zero elements
$2^{[P]}$	Set of all subsets of $[P]$

Table 6: Symbols used as indices

Indices	
j	Index of features, typically $j \in [P]$
(m)	Index of counterfactuals within a set $\mathcal{C}_{\mathbf{x}}$, typically $m \in [M]$
n	A node of the SPN, $n \in \mathcal{V}$
i	Index of bins of a histogram in a leaf node (n), typically $i \in [B_n]$
a	A predecessor node (of node n) in the SPN, usually $a \in \mathrm{pred}(n)$
k	A class ($k \in [C]$) or categorical value ($k \in [K_j]$) index
e	Index of the feature that is changed as an effect of causal relation R

A Appendix

A.1 Notation

Generally, notation follows these rules:

- Capital letters typically refer to amounts of something, as in classes, features, bins, etc. Exceptions are U, L, and F, which are taken from the original work (Russell, 2019).
- Caligraphic capital letters denote sets or continuous spaces.
- Small Latin letters are used as indices, variables, or parameters of the MIP formulation.
- Small Greek letters refer to hyperparameters of the LiCE formulation or parameters of the SPN (scope ψ, parameters θ).
- Subscript is used to specify the position of a scalar value in a matrix or a vector. When in parentheses, it specifies the index of a vector within a set.
- Superscript letters refer to a specification of a symbol with otherwise intuitively similar meaning. Except for $\mathbb{R}^P$, where P has the standard meaning of P-dimensional.
- A hat ($\hat{}$) symbol above an element means that the element is the output of the Neural Network $h(\cdot)$.
- A prime ($'$) symbol as a superscript of an element means that the element is a part of (or the output of) the counterfactual.
- In bold font are only vectors. When we work with a scalar value, the symbol is in regular font.

The specific meanings of symbols used in the article are shown in Tables 5 to 9. The symbols are divided into groups.

- Functions non-specific to our task (Table 5)
- Used indices (Table 6)
- LiCE (hyper)parameters that can be tuned (Table 7)
- Classification task and SPN symbols (Table 8)
- MIO formulation parameters and variables (Table 9)

Table 7: Input parameters into the LiCE formulation

	LiCE (hyper)parameters
τ	The minimal difference between counterfactual class ($h^{\text{raw}}(\mathbf{x}')_{\hat{y}'}$) and factual class ($h^{\text{raw}}(\mathbf{x}')_{\hat{y}}$) NN output value. Alternatively, for binary classification, it is the requirement for a minimal absolute value of the NN output before sigmoid activation ($h^{\text{raw}}(\mathbf{x}')$).
ρ	Limit for the relative difference of values of the objective function within the set of closest counterfactuals $\mathcal{C}_{\mathbf{x}}$.
α	Weight of negative log-likelihood in the objective function
ϵ_j	Minimal change in continuous value c_j of j-th feature. The absolute difference between x'_j and x_j is either 0, or at least ϵ_j.
δ^{SPN}	Lower bound on the estimated value of likelihood of the generated counterfactual.

Table 8: Symbols of the classification task, CE search, and SPNs

	Classification task symbols
P	Number of features
C	Number of classes
$\mathcal{D}$	The dataset, set of 2-tuples $(\mathbf{x}, y) \in \mathcal{D}$
$\mathcal{X}$	Input space $\mathcal{X} \subseteq \mathbb{R}^P$
$\mathbf{x}$	A (factual) sample $\mathbf{x} \in \mathcal{X}$
x_j	A j-th feature of sample $\mathbf{x}$
y	Ground truth of sample $\mathbf{x}$, $y \in [C]$
$h(\cdot)$	Classifier we are explaining $h : \mathcal{X} \to [C]$
$\hat{y}$	Classifier-predicted class $h(\mathbf{x}) = \hat{y} \in [C]$
$h^{\text{raw}}(\cdot)$	NN classifier output without activation $h^{\text{raw}} : \mathcal{X} \to \mathcal{Z}$
$\mathcal{Z}$	Output space of the NN classifier, without sigmoid/softmax activation
	Counterfactual generation symbols
$\lVert\cdot\rVert_{1,\text{MAD}}$	Counterfactual distance function (see Eq. 12)
$\mathcal{C}_{\mathbf{x}}$	Set of generated counterfactuals for factual $\mathbf{x}$
M	Number of sought counterfactuals, $M \geq \lvert\mathcal{C}_{\mathbf{x}}\rvert$
$\mathbf{x}'$	Counterfactual explanation of $\mathbf{x}$, $\mathbf{x}' \in \mathcal{C}_{\mathbf{x}}$
$\mathbf{x}'^*$	Optimal (closest) counterfactual
$\mathbf{x}'_{(m)}$	m-th counterfactual explanation of factual $\mathbf{x}$
x'_j	A value of j-th feature of the counterfactual
$\hat{y}'$	Predicted class of the counterfactual (can be a parameter of LiCE)
	Sum Product Network symbols
$\mathcal{V}$	Set of nodes of the SPN
$\mathcal{V}^{\text{L}}$	Set of leaf nodes
$\mathcal{V}^{\Sigma}$	Set of sum nodes
$\mathcal{V}^{\Pi}$	Set of product nodes
$\text{pred}(\cdot)$	Function returning children (predecessors) of a node
$\psi(\cdot)$	Scope function mapping nodes to their input features $\psi : \mathcal{V} \to 2^{[P]}$
θ	Parameters of the SPN
O_n	Output value of a node $n \in \mathcal{V}$
$w_{a,n}$	Weight of output value of predecessor node a in computing the value of sum node n.
n^{root}	Root node, its value is the value of the SPN

Table 9: Used variables and parameters in the MIO formulation

MIO formulation variables	
l_j	Decrease in continuous value of j-th feature.
$\mathbf{l}$	Concatenated vector of all l_j.
u_j	Increase in continuous value of j-th feature.
$\mathbf{u}$	Concatenated vector of all u_j.
c_j	Continuous value of j-th CE feature.
$d_{j,k}$	1 iff x'_j takes k-th categorical value $k \in K_j$.
$\mathbf{d}$	All variables $d_{j,k}$ concatenated into a vector.
d_j^{cont}	1 iff x'_j takes continuous value c_j.
$h^{\text{raw}}(\cdot)_k$	Value of h^{raw}, corresponding to class $k \in [C]$.
g_k	1 iff class $k \in [C]$ has higher h^{raw} value than the factual class.
s_j	1 iff j-the feature changed, i.e., $x_j \neq x'_j$.
r	1 iff causal relation R is activated, i.e., cause is satisfied and effect is enforced.
$\bar{b}_{n,i}$	1 iff x'_j does *not* belong to the i-th bin ($i \in [B_n]$), assuming j-th feature corresponds to node n, i.e., $\psi(n) = \{j\}$.
o_n	Estimated output value of SPN node $n \in \mathcal{V}$.
$m_{a,n}$	Binary slack indicator for sum node $n \in \mathcal{V}^{\Sigma}$ equal to 0 if output of predecessor a constrains output of n tightly.
MIO formulation parameters	
L_j	Lower bound on continuous values of j-th feature. In our implementation, equal to 0.
U_j	Upper bound on continuous values of j-th feature. In our implementation, equal to 1.
F_j	Default continuous value of j-th feature, equal to the value of the factual x_j, if it has continuous value. Otherwise equal to the median.
K_j	Number of categorical values of j-th feature.
f_j	Equal to x_j, if it has categorical value. If x_j is continuous, f_j is removed, and so are all constraints containing it.
S	Maximal number of feature value changes of $\mathbf{x}'$ compared to $\mathbf{x}$. Sparsity limit.
R	Example causal relation: if j-th feature increases, e-th feature must decrease.
B_n	Number of bins in the histogram of leaf node n.
$t_{n,i}$	Threshold between $i-1$-th and i-th bin in histogram of leaf node n
$q_{n,i}$	Likelihood value of i-th bin of node n.
$\mathbf{v}^{\text{bin}}$	Vector of respective $\lVert\cdot\rVert_{1,\text{MAD}}$ weights for binary one-hot encodings.
$\mathbf{v}^{\text{cont}}$	Vector of respective $\lVert\cdot\rVert_{1,\text{MAD}}$ weights for continuous values.
$\mathbf{d}^{\text{fact}}$	One-hot encoded vector of factual categorical values corresponding to $\mathbf{d}$.
T_n^{LL}	A "big-M" constant for sum node n, used for slack in the computation of $\max$.

A.2 Other Desiderata for CEs

We present more desiderata by Guidotti (2022) that we consider.

- *Diversity.* Each $\mathbf{x}'_{(m)} \in \mathcal{C}_{\mathbf{x}}$ should be as different as possible from any other CE in the set, ideally by proposing changes in different features. For example, one CE recommends increasing the income; another one should recommend decreasing the loan amount instead. An important example of a CE library aiming for diversity is DiCE (Mothilal et al., 2020). In MIO, this is usually achieved by adding constraints and resolving the formulation (Russell, 2019; Mohammadi et al., 2021).
- *Causality.* Given that we know some causal relationships between the features, the generated CEs should follow them. For example, if $\mathbf{x}'$ contains a decrease in the total loan amount, the number of payments or their amount should also decrease.

A.2.1 Our approach to these desiderata

Causality Like actionability, causality depends on prior knowledge of the data. In causality, the constraints are in the form of implications (Mahajan et al., 2020). We describe a way to model causal constraints where if one value changes in a certain direction, then another feature must change accordingly. Details are provided in Section A.3.3.

Diversity and Robustness The diversity of CEs generated by MIO is discussed in the literature (Russell, 2019; Mohammadi et al., 2021). Although their approach can be applied to our model too, here we simply generate a set of top-M counterfactuals closest to the global optimum. We can optionally limit the maximal distance relative to the optimal CE; see Section A.3.5. Regarding the robustness of the counterfactuals, Artelt et al. (2021) show that finding plausible CEs indirectly improves the robustness. Thus, we do not add any further constraints to the model despite this being a viable option (e.g., Maragno et al., 2023; Jiang et al., 2024).

A.3 MIO formulations of Desiderata

The following MIO formulations of the desiderata are novel in that we came up with them, and, to the best of our knowledge, they were not formalized before. They are not too complex, but we formulate them for completeness.

A.3.1 Validity

For $C > 2$ classes, the raw output has C dimensions ($\mathcal{Z} = \mathbb{R}^C$), and the classifier assigns the class equal to the index of the highest value, i.e., $h(\mathbf{x}) = \arg\max_{k\in[C]} h^{\text{raw}}(\mathbf{x})_k$. Let $\hat{y}'$ be the desired counterfactual class. The validity constraint, given that we specify the counterfactual class prior, is then

$$h^{\text{raw}}(\mathbf{x}')_{\hat{y}'} - h^{\text{raw}}(\mathbf{x}')_k \geq \tau \quad \forall k \in [C] \setminus \{\hat{y}'\}. \tag{15}$$

Note that we can also implement a version where we do not care about the counterfactual class $\hat{y}'$ in advance by the following

$$\begin{aligned} g_k = 1 \implies h^{\text{raw}}(\mathbf{x}')_k - h^{\text{raw}}(\mathbf{x}')_{\hat{y}} \geq \tau \quad \forall k \in [C] \setminus \{\hat{y}\} \\ g_k = 0 \implies h^{\text{raw}}(\mathbf{x}')_k - h^{\text{raw}}(\mathbf{x}')_{\hat{y}} \leq \tau \quad \forall k \in [C] \setminus \{\hat{y}\} \\ \sum_{k\in[C]\setminus\{\hat{y}\}} g_k \geq 1, \end{aligned} \tag{16}$$

where $\implies$ can be seen either as an indicator constraint or as an implication (Williams, 2013), g_k is equal to 1 if and only if class k has a higher value than the factual class $\hat{y}$ in the raw output. The sum then ensures that at least one other class has a higher value.

A wide variety of constraints ensuring validity are possible. For example, we can ensure that the factual class has the lowest score by setting $\sum_{k\in[C]\setminus\{\hat{y}\}} g_k \geq C - 1$, or we could enforce a custom order of classes.

A.3.2 SPARSITY

To constrain the sparsity further, we can set an upper bound S on the number of features changed

$$\begin{aligned}
&\sum_j s_j \leq S \\
&\quad s_j \geq 1 - d_{j,f_j} && \forall j \in [P] \\
&\quad s_j \geq d_{j,k} && \forall j \in [P],\ \forall k \in [K_j] \setminus \{f_j\} \\
&\quad s_j \geq l_j + u_j && \forall j \in [P] \\
&\quad s_j \in \{0, 1\} && \forall j \in [P],
\end{aligned} \tag{17}$$

where we use the binary value s_j that equals 1 if the j-th feature changed, the f_j is the categorical value of attribute j of the factual (if applicable).

Neither LiCE nor MIO use this constraint.

A.3.3 CAUSALITY

Consider the following example of a causal relation R. If feature j increases its value, another feature e must decrease. For continuous ranges, this is formulated as

$$\begin{aligned}
r &\geq u_j - l_j \\
l_e &\geq r\epsilon_e \\
u_e &\leq 1 - r \\
r &\in \{0, 1\},
\end{aligned} \tag{18}$$

where ϵ_e is a minimal change in the value of feature e and r equals 1 if the relation R is active. In the case when the features are ordinal, we can assume that their values are just variables representing categorical one-hot encoding, ordered by indices and use:

$$\begin{aligned}
r &\geq \sum_{k=f_j+1}^{K_j} d_{j,k} \\
r &\leq \sum_{k=1}^{f_e} d_{e,k} \\
r &\in \{0, 1\},
\end{aligned} \tag{19}$$

where f_j is the categorical value of the factual in feature j. Naturally, one can see that we can use any combination of increasing/decreasing values in continuous and categorical feature spaces. With these formulations, we can also model monotone values, such as age or education. We simply replace the variable r by 1.

One can formulate any directed graph composed of these causal relations by decomposing it into pairwise relations, one per edge. This way, we can encode commonly used Structural Causal Models that utilize directed graphs to express causality.

A.3.4 COMPLEX DATA

We use the umbrella term "Complex data" for tabular data with non-real continuous values. This includes categorical (e.g., race), binary (e.g., migrant status), ordinal (e.g., education), and discrete contiguous (e.g., number of children) values.

For binary, we use a simple 0-1 encoding; categorical data is encoded into one-hot vectors; and discrete features are discretized by fixing their value to an integer variable within the formulation. Since we normalize all values to the $[0, 1]$ range, we introduce a proxy integer variable z_j:

$$\begin{aligned}
(F_j - l_j + u_j) * \text{scale}_j + \text{shift}_j &= z_j \\
z_j &\in \mathbb{Z}
\end{aligned}$$

For ordinal variables, we use the same encoding as categorical values, with the addition of the one-hot encoding being sorted by value rank to allow for the causality/monotonicity to be enforced.

A.3.5 DIVERSITY

Instead of a single counterfactual, the solver returns (up to) M counterfactuals closest to the global optimum, optionally within some distance range. This range is defined in terms of the objective function, which is the distance of a counterfactual in our case. In other words, we search for a set $\mathcal{C}_{\mathbf{x}} = \{\mathbf{x}'_{(1)}, \ldots, \mathbf{x}'_{(M)}\}$ of counterfactuals that have a similar distance to the factual.

Let $\mathbf{x}'^*$ be the closest CE satisfying all other constraints; we can set a parameter ρ that represents the relative distance of all CEs to the $\mathbf{x}'^*$ leading to the generation of set

$$\mathcal{C}_{\mathbf{x}} = \{\mathbf{x}' \mid \|\mathbf{x} - \mathbf{x}'\|_{1,\text{MAD}} \leq (1+\rho) \cdot \|\mathbf{x} - \mathbf{x}'^*\|_{1,\text{MAD}}\}.$$

Nevertheless, we disregard the relative distance parameter and search for the M closest CEs. Later, we sift through the set $\mathcal{C}$ of top-M counterfactuals, looking for the most likely CEs. Here, one could perform any filtering.

A.4 OTHER MIO FORMULATIONS

A.4.1 MIXED POLYTOPE FORMULATION CORRECTION

For purely categorical features, the original mixed polytope (Russell, 2019) implementation contains an issue. The first categorical value (represented by zero) is mapped to the continuous variable. This seems to work fine for the logarithmic regression (Russell, 2019), but it failed on non-monotone neural networks, leading to non-binary outputs. This was corrected by replacing the continuous variable c_j with another binary decision variable, making it a standard one-hot encoding.

A.4.2 SPN HISTOGRAM FORMULATION

In practice, the probability distribution of a leaf $n \in \mathcal{V}^{\text{L}}$ trained on data is a histogram on a single feature j, i.e., $\psi(n) = \{j\}$. The interval of possible values of x'_j is split into B_n bins, delimited by $B_n + 1$ breakpoints denoted $t_{n,i},\ i \in [B_n + 1]$.

Because modeling that a value of a variable belongs to a union of intervals is simpler than an intersection, we consider variables $\bar{b}_{n,i}$ that equal 1 if and only if the value x'_j does *not* belong to the interval $[t_{n,i}, t_{n,i+1})$. This leads to a set of constraints

$$\bar{b}_{n,i} \geq t_{n,i} - x'_j \qquad \forall n \in \mathcal{V}^{\text{L}}, \forall i \in [B_n] \tag{20}$$

$$\bar{b}_{n,i} \geq x'_j + \epsilon_j - t_{n,i+1} \qquad \forall n \in \mathcal{V}^{\text{L}}, \forall i \in [B_n] \tag{21}$$

$$\sum_{i=1}^{B_n} \bar{b}_{n,i} = B_n - 1 \qquad \forall n \in \mathcal{V}^{\text{L}} \tag{22}$$

$$o_n = \sum_{i=1}^{B_n} (1 - \bar{b}_{n,i}) \log q_{n,i} \qquad \forall n \in \mathcal{V}^{\text{L}} \tag{23}$$

$$\bar{b}_{n,i} \in \{0, 1\} \qquad \forall n \in \mathcal{V}^{\text{L}}, \forall i \in [B_n], \tag{24}$$

where $q_{n,i}$ is the likelihood value in a bin i and o_n is the output value of the leaf node n. ϵ_j is again the minimal change in the feature j and ensures that we consider an open interval on one side. We use the fact that all values x_j (thus also $t_{n,i}$) are in the interval $[0, 1]$. Eq. 20 sets $\bar{b}_{n,i} = 1$ if $x'_j < t_{n,i}$ and Eq. 21 sets $\bar{b}_{n,i} = 1$ for values on the other side of the bin $x'_j \geq t_{n,i+1}$. Eq. 22 ensures that a single bin is chosen and Eq. 23 sets the output value to the log value of the bin that $\mathbf{x}'$ belongs to. This implementation of bin splitting is inspired by the formulation of interval splitting in piecewise function fitting of Goldberg et al. (2021).

We assume that the bins cover the entire space, which we can ensure by adding at most 2 bins on both sides of the interval.

A.5 DATA MODIFICATIONS

We remove samples with missing values. Optionally, we also remove some outlier data or uninformative features.

GMSC We do not remove any feature in GMSC, but we keep only data with reasonable values to avoid numerical issues within MIO. The thresholds for keeping the sample are as follows

- `MonthlyIncome` < 50000
- `RevolvingUtilizationOfUnsecuredLines` < 1
- `NumberOfTime30-59DaysPastDueNotWorse` < 10
- `DebtRatio` < 2
- `NumberOfOpenCreditLinesAndLoans` < 40
- `NumberOfTimes90DaysLate` < 10
- `NumberRealEstateLoansOrLines` < 10
- `NumberOfTime60-89DaysPastDueNotWorse` < 10
- `NumberOfDependents` < 10

this removes around 5.5% of data after data with missing values was removed. We could combat the same issues by taking a $\log$ of some of the features. In our "pruned" GMSC dataset, there are 113,595 samples and 10 features, none of which are categorical, 7 are discrete contiguous, and the remaining 3 are real continuous. Further details are in the preprocessing code.

Adult In the Adult dataset, we remove 5 features

- `fnlwgt` which equals the estimated number of people the data sample represents in the census, and is thus not actionable and difficult to obtain for new data, making it less useful for predictions,
- `education-num` because it can be substituted by ordinal feature `education`,
- `native-country` because it is again not actionable, less informative, and also heavily imbalanced,
- `capital-gain` and `capital-loss` because they contain few non-zero values.

It is not uncommon to remove the features we did, as some of them also have many missing values. We remove only about 2% of the data by removing samples with missing values. We are left with 47,876 samples and 9 features, 5 of which are categorical, 1 is binary, 1 ordinal, and the remaining 2 are discrete contiguous. Further details are in the preprocessing code.

Credit We do not remove any samples or features for the Credit dataset. The dataset contains 1,000 samples and 20 features, 10 of which are categorical, 2 are binary, 1 ordinal, 5 are discrete contiguous, and the remaining 2 are real continuous. Further details are in the preprocessing code.

All code used for the data preprocessing is in the supplementary material.

A.6 Experiment setup

Here, we describe the details of our experiments. The code used for experiments with examples is in the supplementary material and will be made available once the paper is accepted.

A.6.1 Additional Constraints

In addition to data type constraints described in Section A.5, we also constrain some features for immutability and causality.

GMSC

- **Immutable**: `NumberOfDependents`
- **Monotone**: `age` cannot decrease
- **Causal**: no constraints

Adult

- **Immutable**: `race` and `sex`
- **Monotone**: `age` cannot decrease and `education` cannot decrease
- **Causal**: `education` increases $\implies$ `age` increases

Credit

- **Immutable**: `Number of people being liable to provide maintenance for`, `Personal status and sex`, and `foreign worker`
- **Monotone**: `Age` cannot decrease
- **Causal**: `Present residence since` increases $\implies$ `Age` increases and `Present employment since` increases $\implies$ `Age` increases

A.6.2 Hyperparameter setup

The entire configuration can be found in the code, but we also present (most of) it here.

Neural Network We compare methods on a neural network with four layers, first with a size equal to the length of the encoded input, then 20 and 10 for hidden layers, and a single neuron as output. It trained with batch size 64 for 50 epochs. We compare all methods on this neural network architecture, trained separately five times for each training set (from the five folds).

SPN To create fewer nodes in the SPN (i.e., to not overtrain it), we set the `min_instances_slice` parameter to the number of samples divided by 20.

CE methods We used default parameters for most methods. In cases when there were no default values set, we used the following:

- *DiCE*: we use the gradient method of searching for CEs.
- *VAE*: we set the size of the model to copy the predictor model. We parametrize the hinge loss with a margin of 0.1 and multiply the validity loss by 10 to promote validity. We use learning rate 1e-3 and batch size 64. We use weight decay of 1e-4 and train for 20 epochs (200 for the Credit data since the dataset is small).
- *FACE*: we only configure the fraction of the dataset used to search for the CE, increasing it to 0.5 for the Credit dataset due to its size.
- *C-CHVAE*: we set the size of the model to copy the predictor model. For the Credit dataset, we increase the number of training epochs to 50.
- *PROPLACE*: We create the retrained NN models to reflect the same architecture and train them for 15 epochs. We set up 1 instance of PROPLACE per class and set its delta by starting at 0.025 and decreasing by 0.005 until we are able to recover enough samples.
- *LiCE + MIO*: For our methods, we configure a time limit of 2 minutes for MIO solving. These are high enough for MIO, but constrained LiCE struggles with increasing likelihood requirements. We generate 10 closest CEs, not using the relative distance parameter. We set the decision margin $\tau = 10^{-4}$ and we use one $\epsilon_j = 10^{-4}$ for all features j because they are normalized. In the SPNs, we use $T_n^{\text{LL}} = 100$ as a safe upper bound though this could be computed more tightly for an individual sum node. We choose δ^{SPN} equal to the median (or lower quartile) of likelihood on the dataset. For LiCE (optimize), we used $\alpha = 0.1$ since features are normalized to $[0, 1]$ and log-likelihood often takes values in the $[-100, -10]$ range.

A.6.3 Computational resources

Most experiments ran on a personal laptop with 32GB of RAM and 16 CPUs AMD Ryzen 7 PRO 6850U, but since the proposed methods had undergone wider experimentation, their experiments

Table 10: Comparison of LiCE variants. (optimize) means that we optimize the likelihood together with the distance, with coefficient $\alpha = 0.1$. (quartile) means that we constrain the CE to have the likelihood greater or equal to the lower quartile likelihood of training data. (median) is the same as (quartile), but we take the median instead of the quartile. Finally, (sample) is a relaxation of the (median) variant. It constrains the CE likelihood to be greater or equal to the likelihood of the factual sample or the median value, whichever is lower.

	GMSC			**Adult**			**Credit**		
Method	NLL	Similarity	Sparsity	NLL	Similarity	Sparsity	NLL	Similarity	Sparsity
MIO (+spn)	27.8 ±6.4	5.9 ±1.8	2.2 ±0.8	17.9 ±3.7	5.7 ±3.6	2.2 ±0.9	47.6 ±18.2	4.4 ±2.7	2.2 ±1.0
LiCE (optimize)	25.6 ±4.4	5.9 ±1.8	2.6 ±1.1	18.1 ±3.8	**5.5 ±3.6**	**2.0 ±1.0**	**29.1 ±3.3**	4.4 ±2.7	2.1 ±1.0
LiCE (quartile)	27.0 ±3.5	**5.8 ±1.8**	**1.9 ±0.8**	18.4 ±3.5	5.6 ±3.8	**2.0 ±1.0**	41.1 ±16.4	**4.3 ±2.7**	**1.9 ±1.0**
LiCE (median)	**18.1 ±2.6**	10.6 ±3.5	4.4 ±1.1	**12.9 ±1.0**	9.7 ±6.5	2.9 ±1.3	29.8 ±3.1	4.4 ±2.7	2.0 ±1.1
LiCE (sample)	20.4 ±3.8	9.4 ±3.7	4.2 ±1.3	14.4 ±2.7	8.4 ±5.6	2.7 ±1.2	31.3 ±7.0	4.4 ±2.7	1.9 ±1.1

were run on an internal cluster with assigned 32GB of RAM and 16 CPUs, some AMD EPYC 7543 and some Intel Xeon Scalable Gold 6146, based on their availability.

Regarding computational time, it is non-trivial to estimate. The time varies greatly for some methods since, for example, VAE retries generating a CE until a valid is found or a limit on tries is reached. Most methods we compared took a few hours for the 500 samples, including the method training. The MIO method takes, on average, a few seconds to generate an optimal counterfactual, while LiCE often reaches the 2-minute time limit.

Considering the tests presented in this paper, we estimate 200 hours of real-time was spent generating them, meaning approximately 3,200 CPU hours. If we include all preliminary testing, the compute time is estimated at around 20,000 CPU hours, though these are all inaccurate rough estimates, given that the hours were not tracked.

A.7 Further comparisons

In this section, we would like to discuss some results that could not fit into the article's main body.

A.7.1 LiCE variants

We tested multiple versions of using the SPN within LiCE. In Table 10, we show results for 2 more configurations.

One, called *(sample)*, is a relaxation of the (median) variant. It constrains the CE likelihood to be greater or equal to the likelihood of the factual sample (i.e., the counterfactual should have, at worst, the same likelihood as the factual) or the median value, whichever is lower. This increases the proportion of factuals for which the method returns a CE in time, though only by 10 percentage points at most. This suggests that the complexity might not depend on the likelihood of the factual, thus that there might be a notable difference in likelihood landscape for the opposite classes.

The *LiCE (quartile)* is a weaker variant of *LiCE (median)*, with the bound set to the first quartile instead of the median likelihood. This is enough to obtain CEs for 100% of factuals (and in good time, see Table 12). Its good performance w.r.t. similarity and sparsity is possibly caused by the method returning very close CEs with a "good enough" log-likelihood.

The results show that selecting the most likely CE out of 10 local optima given by MIO is quite strong. The two-stage setup can be quite performant. The results on similarity show that some of the MIO CEs are not globally optimal. This is because the SPN in the second phase selects some of the locally optimal (i.e., globally suboptimal) CEs.

A.7.2 Valid CEs on common factuals

Table 11 shows the results on the intersection of factuals for which all methods generated a valid CE. The proposed methods show similar differences in all metrics, as in Table 4.

Notice the comparability of DiCE results on negative log-likelihood. This suggests that the two-stage setting of generating a diverse set of CEs and then selecting the likeliest could be a viable option. On the other hand, compared to LiCE (or MIO), there is a major difference in all measures.

Table 11: Results in the same format as in Table 4, but we consider only valid CEs generated for the intersection of factuals for which all methods generated a valid CE. These results are more suitable for the comparison of methods between each other. The VAE was omitted from the evaluation on the GMSC dataset because the intersection of factuals would be empty if we included VAE.

	GMSC (272 factuals)			**Adult** (64 factuals)			**Credit** (55 factuals)		
Method	NLL	Similarity	Sparsity	NLL	Similarity	Sparsity	NLL	Similarity	Sparsity
DiCE (+spn)	30.6 ±3.5	21.9 ±5.8	6.4 ±1.0	19.9 ±2.2	23.0 ±9.0	4.3 ±1.5	36.1 ±2.4	22.3 ±4.9	7.6 ±1.9
VAE (+spn)	-	-	-	16.4 ±2.5	29.4 ±8.1	5.2 ±1.3	46.2 ±17.6	31.1 ±8.2	11.3 ±1.9
C-CHVAE	24.8 ±2.4	17.2 ±4.6	9.0 ±0.0	17.3 ±3.0	7.6 ±6.0	2.8 ±1.0	34.0 ±6.5	13.5 ±4.9	7.6 ±1.0
FACE (ϵ)	30.2 ±9.6	14.8 ±3.8	8.4 ±1.1	15.1 ±2.9	8.7 ±6.2	2.8 ±1.1	48.8 ±17.3	18.2 ±6.3	6.8 ±1.3
FACE (knn)	30.1 ±9.4	14.6 ±4.0	8.2 ±1.2	14.6 ±3.1	8.6 ±5.9	2.7 ±1.1	42.5 ±16.9	19.2 ±6.1	7.1 ±1.3
PROPLACE	27.4 ±6.9	12.3 ±3.0	7.2 ±1.1	17.6 ±3.2	18.2 ±6.7	4.1 ±1.2	39.4 ±15.7	25.2 ±6.2	8.7 ±1.3
MIO (+spn)	27.8 ±8.0	6.2 ±1.7	2.2 ±0.9	16.0 ±3.6	4.0 ±3.0	1.8 ±0.8	48.3 ±18.2	2.6 ±1.5	1.8 ±0.9
LiCE (optimize)	24.4 ±5.1	6.2 ±1.7	2.7 ±1.1	16.2 ±3.5	**3.7 ±3.1**	1.7 ±0.8	**29.2 ±2.7**	2.6 ±1.6	1.7 ±0.7
LiCE (quartile)	26.3 ±3.6	**6.1 ±1.7**	**1.9 ±0.8**	16.9 ±3.6	**3.7 ±3.1**	**1.6 ±0.8**	45.0 ±17.8	**2.5 ±1.5**	**1.6 ±0.8**
LiCE (median)	**18.1 ±2.7**	10.7 ±3.4	4.4 ±1.1	**12.8 ±1.1**	7.0 ±5.5	2.4 ±1.2	30.4 ±2.4	2.5 ±1.6	1.7 ±0.8

Table 12: Median time spent on the computation of a single CE. The values above 120 in the LiCE computation are caused by computational overhead in formulating the SPN. The time limit given to the solver was 120 seconds.

Method	**GMSC**	**Adult**	**Credit**
DiCE (+spn)	25.27s	17.04s	147.35s
VAE (+spn)	0.64s	0.97s	0.63s
C-CHVAE	0.43s	0.63s	0.59s
FACE (ϵ)	7.88s	7.46s	5.12s
FACE (knn)	5.65s	7.41s	5.28s
PROPLACE	**0.38s**	**0.24s**	**0.20s**
MIO (+spn)	0.76s	1.51s	1.56s
LiCE (optimize)	132.61s	36.06s	3.06s
LiCE (quartile)	20.02s	10.61s	2.78s
LiCE (median)	124.20s	15.29s	2.98s

A.7.3 TIME COMPLEXITY

Regarding the complexity of the SPN formulation, the number of variables is linearly dependent on the size of the SPN (real-valued variables). Additionally, each leaf node requires one binary variable for each bin of the histogram distribution. Sum nodes require one extra binary variable per predecessor; the total number is bounded by the number of all nodes from above, but it is typically less. The number of constraints is linearly dependent on the size of the SPN.

This is, however, difficult to translate to the algorithmic complexity of solving the MIO, which is exponential w.r.t. size of the formulation in general.

Table 12 shows the median number of seconds required to generate (or fail to generate) a CE. We see that there are stark differences between methods and also between datasets. For our methods (MIO and LiCE), we constrain the maximal optimization time to 120 seconds.

LiCE seems to be comparable on Adult as well as Credit datasets. Since MIO seems to be faster, we suggest that the main portion of the overhead is caused by solving the SPN formulation. Note that the optimizing variant of LiCE takes a long time partly to prove optimality. A (non-optimal) solution could likely be obtained even with a tighter time limit.

There also seems to be some computational overhead in constructing the formulation, which could likely be partly optimized away in the implementation.

A.7.4 Other plausibility metrics

We considered using Kernel Density Estimator (KDE) for the evaluation (similarly to (Artelt & Hammer, 2020)), but the KDE is not suitable for categorical data, so we decided against it.

A.8 Further comments

Given the limited size of the Credit dataset, it is unsurprising to see so many failures of some methods. There is not much data for some methods to support the training. This might be behind the low success rate of computing a valid CE.

Regarding the other results, it is possible that the VAE method has been misconfigured for GMSC, returning very few results.

The main disadvantage of LiCE is the time complexity of CE generation. We argue, however, that for some use cases, the user might be willing to wait to obtain a high-quality CE. We leave this decision to the user.

A.8.1 Omitted methods

It is not feasible to test against all CE methods, so we looked for a selection of methods that consider the plausibility of generated CEs. Two methods were, however, not tested for the following reasons:

- PlaCE (Artelt & Hammer, 2020) does not allow for explaining Neural Networks. It also cannot model categorical features well.
- DACE (Kanamori et al., 2020) does not have a public implementation that would allow for Neural Networks as models. It might also struggle with the size of datasets used here since they are an order of magnitude larger, and DACE computes the Local Outlier Factor, meaning that the formulation size increases linearly with the increase in the number of samples.

A.8.2 Potential negative consequences

Given the many CE methods for generating CEs, one must deal with the disagreement problem (Brughmans et al., 2024), where a user could be misled by the owner of an ML model who selects the CEs that align with their interests. We argue that our method does not severely contribute to this problem, since it is deterministic, thus resistant to re-generation attempts to obtain a more favorable CE. Our method also outperforms many other methods, making arguing for their use more difficult.

A.8.3 Suitability of SPNs for estimating the plausibility of CEs

We believe that SPNs are well suited for the problem, because (i) they naturally model distribution over continuous and discrete random variables; (ii) their simple formulation can be tightly approximated within MIO; (iii) and they are universal approximators (Nguyen & McLachlan, 2019).

Other options are

- *Gaussian Mixture Models (GMMs)* are designed only for continuous random variables. SPNs are a strict superset to GMMs.
- *Flow models* are very flexible, but they model only distribution on continuous random variables. Since they are parametrized by neural networks, they might be rather difficult to formulate within MIO, especially considering their block nature relying on smooth non-linear functions (exp, softplus).
- *Neural auto-regressive models* can model discrete and random variables, and they provide exact likelihood. But again, they use relatively large neural networks which might need non-linearities that are difficult to use within MIO (sigmoid, softmax).
- *Auto-encoders* have with respect to MIO similar advantages and disadvantages as neural auto-regressive models. Furthermore, they provide only lower-bound estimates of true likelihood in the form of ELBO.

	sprinkler	asia	sachs	child	water	alarm	win95pts	andes
Number of nodes	4	8	11	20	32	37	76	223
Number of edges	4	8	17	25	66	46	112	338
Number of parameters	9	18	178	230	10083	509	574	1157

Table 13: Size comparison of BNs used to generate the synthetic data.

	sprinkler	asia	sachs	child	water	alarm	win95pts	andes
Pearson coeff.	0.996	0.990	0.964	0.954	0.955	0.959	0.959	0.793
Kendall (τ-b) coeff.	1.000	0.992	0.860	0.853	0.828	0.892	0.891	0.600
Spearman coeff.	1.000	1.000	0.972	0.966	0.961	0.978	0.977	0.788
Total variation	0.017	0.073	-	-	-	-	-	-

Table 14: Evaluation of the fit by correlation coefficients for all 8 tested BNs. Total Variation was computed only for smaller BNs, where the computation was practical. Numbers are rounded to 3 decimal digits.

A.8.4 SPN AS A MODEL OF DATA DISTRIBUTION

Furthermore, we test the ability of an SPN to model the true data distribution empirically. We choose 8 Bayesian Networks (BNs) to model the data-generating process. For each of them, we generate (sample) training data, then fit an SPN to this data, and finally compare the SPN's likelihood estimates of test samples to their true probability, given by the BN.

More specifically, we utilize the `bnlearn` Python library (Taskesen, 2020) and select 7 Bayesian Networks of varying sizes from the Bayesian Network Repository (Scutari, 2007) (namely asia, sachs, child, water, alarm, win95pts, and andes). The eighth BN (sprinkler) is another standard BN, available directly in the `bnlearn` library. See Table 13 for parameters of the used networks.

To train the SPN, we sample 10,000 points using the BN and train the SPN in the same way as for LiCE, with default parameters. Then we sample 1,000 more samples and evaluate their likelihood using the trained SPN. We also compute their true probability from the BN and compare these pairs of values. We perform the above process 5 times with different seeds for each BN and select the best-performing SPN for comparison.

In Table 14, we show the correlation coefficients of the log-likelihood and log-probability computed on the 1,000 test samples. We also show the total variation for the smaller BNs, when the value can be computed in reasonable time. In Figure 2, we show scatter plots of the data on which the correlation coefficients were computed. The BNs are sorted in increasing order of the number of nodes.

The SPN performs quite well, with the exception of the biggest BN (andes), where the drop might be explained by 10,000 samples being too few to train the SPN precisely.

A.8.5 CE GENERATION WITH KNOWLEDGE OF THE TRUE DISTRIBUTION.

In this section, we would like to compare the CE generation methods using the true data distribution. While this distribution is generally unknown, we construct the following experiments to evaluate our method in such a scenario, by forming 3 synthetic datasets.

We utilize three of the Bayesian Networks (used in Section A.8.4) of varying size (asia, alarm, and win95pts), choose a target variable (`dysp`, `BP`, and `Problem1`, respectively) and sample a training dataset of 10,000 samples. On this training dataset, we train an SPN and a Neural Network model, that we then utilize to generate counterfactuals for a set of 100 factuals newly sampled from the BN. We perform this whole setup for 5 different seeds for each BN and aggregate the results.

In the tables below (Tables 15, 16, and 17 for asia, alarm, and win95pts, respectively), we evaluate the mean log-likelihood of generated CEs using the fitted SPN, the mean true probability, mean

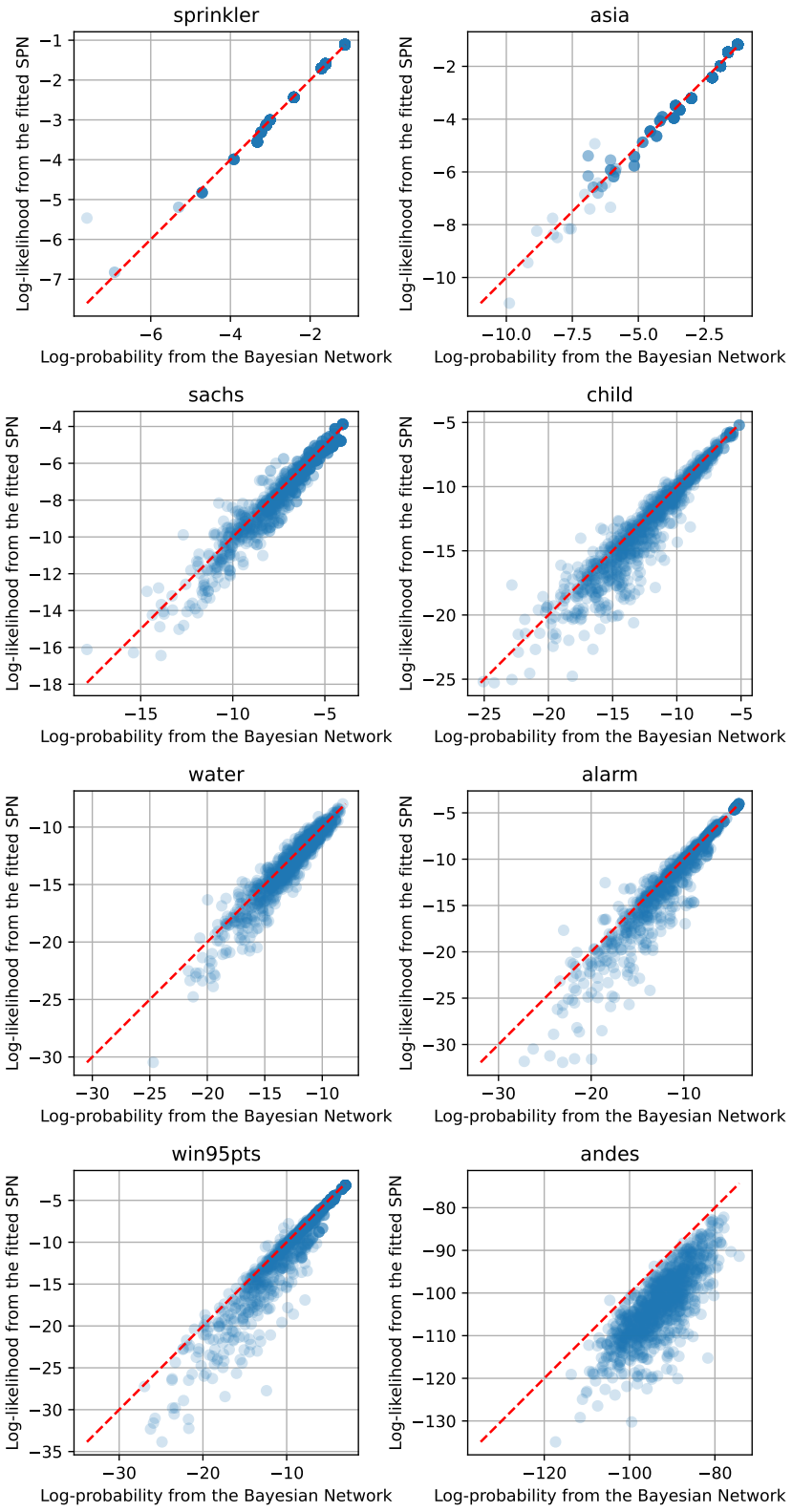

Figure 2: Visual comparison of correlation between the true probability of a sample and log-likelihood estimate given by an SPN. Each plot shows 1,000 points sampled using the given Bayesian Network, see their names in the titles. For numerical comparison, see Table 14.

asia	LL estimate (SPN) ↑	True probability (BN) ↑	Similarity ↓	Sparsity ↓	Time [s] ↓	% valid ↑
CH-CVAE	-2.46 ± 1.71	$1.6 \times 10^{-1} \pm 9.8 \times 10^{-2}$	2.46 ± 2.47	1.31 ± 0.72	0.74 ± 0.54 s	46.6 %
FACE (knn)	-2.44 ± 1.54	$1.6 \times 10^{-1} \pm 9.7 \times 10^{-2}$	2.42 ± 2.46	1.27 ± 0.65	0.27 ± 0.11 s	46.6 %
FACE (ϵ)	-3.75 ± 0.79	$3.8 \times 10^{-2} \pm 5.1 \times 10^{-2}$	7.69 ± 2.42	2.33 ± 0.47	0.32 ± 0.09 s	1.2 %
PROPLACE	-3.98 ± 1.87	$7.3 \times 10^{-2} \pm 8.3 \times 10^{-2}$	4.35 ± 3.06	1.79 ± 0.74	0.18 ± 0.10 s	46.6 %
MIO (+spn)	-1.53 ± 0.80	$2.3 \times 10^{-1} \pm 6.7 \times 10^{-2}$	2.89 ± 1.95	1.86 ± 0.74	1.11 ± 0.87 s	100 %
LiCE (med)	-1.33 ± 0.15	$2.4 \times 10^{-1} \pm 4.5 \times 10^{-2}$	3.09 ± 2.51	1.94 ± 0.91	0.68 ± 0.09 s	100 %
LiCE ($\alpha = 1$)	-1.78 ± 0.94	$1.9 \times 10^{-1} \pm 6.3 \times 10^{-2}$	2.42 ± 2.07	1.53 ± 0.80	0.85 ± 0.13 s	100 %

Table 15: Comparison on the asia BN. LL stands for log-likelihood. We show the mean probability directly, because of a few 0 probability counterfactuals.

alarm	LL estimate (SPN) ↑	True probability (BN) ↑	Similarity ↓	Sparsity ↓	Time [s] ↓	% valid ↑
CH-CVAE	-10.07 ± 4.69	$3.1 \times 10^{-3} \pm 5.9 \times 10^{-3}$	13.20 ± 8.86	3.78 ± 2.21	0.99 ± 0.86 s	26.8 %
FACE (knn)	-9.10 ± 3.92	$2.7 \times 10^{-3} \pm 5.0 \times 10^{-3}$	12.30 ± 8.62	3.54 ± 2.25	9.86 ± 2.67 s	55 %
FACE (ϵ)	-9.76 ± 3.48	$1.4 \times 10^{-3} \pm 3.5 \times 10^{-3}$	14.09 ± 8.51	4.08 ± 2.24	8.45 ± 2.28 s	41.4 %
PROPLACE	-10.90 ± 4.39	$9.5 \times 10^{-4} \pm 2.8 \times 10^{-3}$	16.11 ± 10.64	4.60 ± 2.74	0.80 ± 0.35 s	55 %
MIO (+spn)	-10.91 ± 5.30	$2.9 \times 10^{-3} \pm 5.6 \times 10^{-3}$	3.87 ± 1.39	1.37 ± 0.53	1.93 ± 0.14 s	100 %
LiCE (med)	-7.73 ± 1.87	$2.8 \times 10^{-3} \pm 5.3 \times 10^{-3}$	8.46 ± 7.52	2.54 ± 2.02	9.36 ± 9.65 s	100 %
LiCE ($\alpha = 1$)	-9.50 ± 4.53	$3.3 \times 10^{-3} \pm 5.8 \times 10^{-3}$	4.58 ± 2.53	1.51 ± 0.77	8.60 ± 2.61 s	100 %

Table 16: Comparison on the alarm BN. LL stands for log-likelihood. We show the mean probability directly, because of a few 0 probability counterfactuals.

distance (similarity), sparsity, and time spent generating the valid counterfactuals. Finally, we show the percentage of factuals for which a valid counterfactual was found by a given method.

We see that LiCE methods, especially the likelihood optimizing variant ($\alpha = 1$), perform comparably to other methods even when taking into account the true distribution.

Finally, note that:

- performance improvements in terms of distance and sparsity reflect experiments on real data;
- only MIO-based methods generate 100% of valid counterfactuals, other methods generate 55.8% CEs at best;
- the time complexity of LiCE is on par with other methods;
- while not perfect, SPN likelihood generally correlates with the true probability (see the discussion in Section A.8.4 for additional details).

win95pts	LL estimate (SPN) ↑	True probability (BN) ↑	Similarity ↓	Sparsity ↓	Time [s] ↓	% valid ↑
CH-CVAE	-7.33 ± 2.95	$3.2 \times 10^{-4} \pm 4.6 \times 10^{-4}$	8.03 ± 7.77	3.90 ± 2.90	1.28 ± 0.67 s	55.8 %
FACE (knn)	-8.69 ± 3.45	$1.4 \times 10^{-3} \pm 2.2 \times 10^{-3}$	7.20 ± 5.45	3.51 ± 1.90	9.64 ± 2.76 s	55.8 %
FACE (ϵ)	-10.08 ± 3.83	$6.7 \times 10^{-4} \pm 1.3 \times 10^{-3}$	9.30 ± 5.50	4.39 ± 1.90	7.92 ± 2.40 s	29.2 %
PROPLACE	-8.32 ± 3.01	$3.8 \times 10^{-4} \pm 8.1 \times 10^{-4}$	7.08 ± 6.63	3.58 ± 2.63	0.47 ± 0.14 s	55.8 %
MIO (+spn)	-10.86 ± 4.54	$9.1 \times 10^{-5} \pm 1.9 \times 10^{-4}$	2.43 ± 0.71	1.70 ± 0.46	1.71 ± 0.09 s	100 %
LiCE (med)	-7.77 ± 1.22	$5.2 \times 10^{-4} \pm 2.1 \times 10^{-3}$	7.42 ± 8.58	3.47 ± 3.15	5.19 ± 0.37 s	100 %
LiCE ($\alpha = 1$)	-9.93 ± 4.15	$1.3 \times 10^{-3} \pm 6.0 \times 10^{-3}$	2.49 ± 1.72	1.57 ± 0.83	5.35 ± 0.76 s	100 %

Table 17: Comparison on the win95pts BN. LL stands for log-likelihood. We show the mean probability directly, because of a few 0 probability counterfactuals.