Preserving Causal Constraints in Counterfactual Explanations for Machine Learning Classifiers

Divyat Mahajan
Microsoft Research
Bangalore, India
divyatmahajan@gmail.com

Chenhao Tan
University of Colorado Boulder
Boulder, USA
chenhao@chenhaot.com

Amit Sharma
Microsoft Research
Bangalore, India
amshar@microsoft.com

Abstract

Explaining the output of a complex machine learning (ML) model often requires approximation using a simpler model. To construct interpretable explanations that are also consistent with the original ML model, *counterfactual* examples — showing how the model's output changes with small perturbations to the input — have been proposed. This paper extends the work in counterfactual explanations by addressing the challenge of *feasibility* of such examples. For explanations of ML models in critical domains such as healthcare and finance, counterfactual examples are useful for an end-user only to the extent that perturbation of feature inputs is feasible in the real world. We formulate the problem of feasibility as preserving causal relationships among input features and present a method that uses (partial) structural causal models to generate actionable counterfactuals. When feasibility constraints may not be easily expressed, we propose an alternative method that optimizes for feasibility as people interact with its output and provide oracle-like feedback. Our experiments on synthetic Bayesian networks and the widely used `Adult` dataset show that our proposed methods can generate counterfactual explanations that satisfy feasibility constraints.

1 Introduction

Explanations for a machine learning model are important for people to interpret its output, especially in critical decision-making scenarios such as healthcare, governance, and finance. Techniques for explaining an ML model often involve a simpler surrogate model that yields interpretable information, such as feature importance scores [17]. However, these techniques suffer from an inherent fidelity-interpretability tradeoff due to their use of a simpler model for generating explanations. Highly interpretable explanations may end up approximating too much and be inconsistent with the original ML model (low fidelity), while high fidelity explanations may be as complex as the original ML model and thus less interpretable.

Counterfactual explanations [21] have been proposed as an alternative that are always consistent with the original ML model and arguably may also be interpretable. Counterfactual (CF) explanations present the perturbations in the original input features that could have led to a change in the prediction of the model. For example, consider a person whose loan application has been rejected by an ML classifier. For this person, a counterfactual explanation provides what-if scenarios wherein they would

CausalML: Machine Learning and Causal Inference for Improved Decision Making Workshop, 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

have their loan approved, e.g., *"your loan would have been approved if your income was $10000 more"*. Since the goal is to generate perturbations of an input that lead to a different outcome from the ML model, generation of CF explanations has parallels with adversarial examples [7]. However, counterfactual explanations have an additional goal of being *feasible* in the real world. Continuing with the loan example, the counterfactual changes in the input should follow some natural laws (e.g., age cannot decrease) and knowledge about interactions between features (e.g., changing education level without changing age is infeasible).

In this work, therefore, we focus on the problem of generating *feasible* counterfactuals. While current methods focus on generating counterfactuals that follow the same observed feature distribution as the training data [5], a common limitation of current method is that they perturb features independently to achieve the desired output class, but do not consider whether such perturbations are plausible [14, 5, 18]. A more realistic assumption is that features are causally related to each other, and changing one may invariably lead to changes in other features. Thus, generating feasible counterfactuals requires that we model these causal relationships between different features.

Our main contribution is to formally define feasibility for a counterfactual example and present a generative framework that can optimize for feasibility during counterfactual generation, rather than post-filtering as in [14]. We define two types of feasibility: *global* feasibility that must be satisfied by all counterfactual examples, and *local* feasibility that depends on an end-user's preferences. We show that there is a connection between global feasibility and causality — preserving causal constraints between features is a necessary condition for generating globally feasible counterfactual examples. However, local feasibility for an end user may depend on their personal constraints or preferences and may only be revealed by user feedback on generated CF examples.

To model feasibility, we propose a generative model for counterfactual examples, BaseGenerativeCF, and provide a variational bound, analogous to a variational auto-encoder [9]. Compared to past work on CF generation that requires solving an optimization problem for every input, a generative model provides significant computational advantage in generating multiple counterfactuals for inputs. We then propose extensions of this model based on the level of information available about feasibility. Ideally, if accurate information on causal relationships is available as a structural causal model, we utilize it to construct a *causal proximity regularizer* between a counterfactual and the original input and add it to the BaseGenerativeCF model. Notably, our method does not require knowledge of the full causal network; it can work with any set of known edges of the network to provide feasibility with respect to those causal edges. We also provide an extension to directly optimize for common unary and binary feasibility constraints. In general, however, optimizing for many complex constraints can be difficult, and in the case of user preferences, no such hard constraints may exist. We thus provide a general method that utilizes feedback from a feasibility oracle and iteratively learns to generate feasible CF examples.

Results on simulated data from synthetic Bayesian networks, and the Adult-Income dataset [10] show that our proposed methods can generate counterfactual examples that are more feasible than models that do not include causal assumptions or user feedback. Further, our novel generative model is much faster than existing approaches to generate CFs. To summarize, our contributions include:

- We formulate feasibility constraints in counterfactual generation into two components: 1) satisfying causal relationships between features (global); 2) accommodating user preferences (local).
- We propose an encoder-decoder framework for generating counterfactuals subject to feasibility constraints. When structural causal information is available, we show how to model causal relationships directly in the CF generation process.
- Using this framework, we also provide an method to incorporate causal constraints for counterfactual explanations using an oracle-based mechanism.

2 Feasibility of Counterfactual Explanations

We formally define feasibility for counterfactual examples by firsting defining a causal model. Throughout, we assume a machine learning classifier, $h : \mathcal{X} \rightarrow \mathcal{Y}$ where $\boldsymbol{x} \in \mathcal{X}$ are the features and $y \in \mathcal{Y}$ is a categorical output. A *valid* counterfactual example for an input $\boldsymbol{x}$ and outcome y is one that changes the outcome of h to the desired outcome y' [14].

Definition 2.1. ***Causal Model [16].*** *A causal model is a triplet $M = \langle U, V, F \rangle$ such that U is a set of exogenous variables, V is a set of endogenous variables that are determined by variables inside the model, and F is a set of functions that determine the value of each $v_i \in V$ (up to some independent noise) based on values of $U_i \bigcup Pa_i$ where $U_i \subseteq U$ and $Pa_i \subseteq V \setminus v_i$.*

Definition 2.2. ***Global Feasibility.*** *Let $\langle \boldsymbol{x}_i, y_i \rangle$ be the input features and the predicted outcome from h, and let y' be the desired output class. Let $M = \langle U, V, F \rangle$ be a causal model over $\mathcal{X}$ such that each feature is in $U \bigcup V$. Then, a counterfactual example $\langle \boldsymbol{x}_{\texttt{cf}}, y_{\texttt{cf}} \rangle$ is globally feasible if it is valid ($y_{\texttt{cf}} = y'$), the change from $\boldsymbol{x}_i$ to $\boldsymbol{x}_{cf}$ satisfies all constraints entailed by the causal model, and all exogenous variables $\boldsymbol{x}^{exog} = U$ lie within the input domain.*

For example, a CF example that changes an individual's age to 300 is infeasible since it violates the limits of the input domain of the age feature. A CF example that decreases age is infeasible since it violates the natural causal model/constraint that age can only increase with time. In general, constraints relating to the input domain can be learned from an i.i.d. sample of data by estimating the joint distribution of features in $\boldsymbol{x}$. Dhurandhar et al. [5] use an auto-encoder to align CF examples to the data-generating process. However, *causal constraints cannot be learned from data alone, and often need extra information* [16]. As we will show in Section 3, they can be provided in the form of structural causal models, or more practically, as constraints on how features can change.

More complex causal constraints can be defined over pairs or multiple variables. For example, in the loan decision example from above, we can consider v_{p1} as *education-level* and v as *age*, and posit a causal relationship that increasing *education-level* needs years to complete and thus causes *age* to increase. Thus, any counterfactual example that increases *education-level* without increasing *age* is infeasible. That said, a counterfactual example that increases *age* without changing *education-level* may still be feasible since we do not know the full set of causes that may increase *age*. While some of these feasibility constraints can be formulated in simple terms, causal relationships over multiple features can lead to complex constraints. In Section 3, we show how pairwise constraints can be derived given any structural causal model (SCM) and provide a method for generating feasible counterfactuals given these constraints from an SCM.

That said, for a particular user, a globally feasible CF example may still be infeasible due to an end-user's context or personal preferences. We therefore introduce local feasibility.

Definition 2.3. ***Local Feasibility****: A CF example is locally feasible for a user if it is globally feasible and satisfies user-level constraints.*

For example, a user may find it difficult to change their city because of family constraints. Thus, a counterfactual example may be locally infeasible due to many factors, which may vary from person to person. Preserving constraints entailed from a causal model provides necessary conditions for a feasible counterfactual, but customization is needed for local feasibility. In Section 4, we will discuss how an oracle-based method can personalize the feasibility for each person based on their feedback. As we will show, this method can also be a practical alternative to the SCM-based method for global constraints when information about causal edges is not known.

3 Satisfying Global Feasibility through Causal Constraints

Counterfactual generation is usually framed as solving an optimization problem that searches in the feature space to find perturbations that are proximal (close to the original data input) but lead to a different output class from the machine learning model. Wachter et al. [21] provide the following optimization formulation to generate a counterfactual example $\boldsymbol{x}^{cf}$ for an input instance $\boldsymbol{x}$ given a ML model f, where the target class is y':

$$\underset{\boldsymbol{x}^{cf}}{\operatorname{argmin}} \operatorname{Loss}(f(\boldsymbol{x}^{cf}), y') + \operatorname{Distance}(\boldsymbol{x}, \boldsymbol{x}^{cf}). \tag{1}$$

Loss refers to a classification loss (such as cross-entropy) and Distance refers to a distance metric (such as ℓ_2 distance). That is, we seek to generate counterfactual explanations that belong to a target class y' while still remaining proximal to the original feature. Note that under this formulation, features of the input $\boldsymbol{x}$ can be changed independently to construct $\boldsymbol{x}^{cf}$. A new optimization problem also needs to be solved for each new input. Extensions have been proposed that approximate the joint distribution over features using an auto-encoder term [5], but these methods do not provide any theoretical justification for the loss terms.

Below we provide a generative formulation of the counterfactual generation problem and provide a variational lower bound that can be used to derive a loss function. Being a generative model, it avoids separate, new optimizations for each input and can also be extended to handle causal constraints.

3.1 Base CF Explanation Objective

Here we present a model-based framework to generate counterfactuals. We define the problem of generating CF examples $\boldsymbol{x}^{cf}$ as building a model that maximizes $\Pr(\boldsymbol{x}^{cf}|y', \boldsymbol{x})$ such that $\boldsymbol{x}^{cf}$ belongs to class y'. Our approach is based on an encoder-decoder framework where the task of the encoder is to project input features to a suitable latent space and the task of the decoder is to generate a counterfactual from the latent representation given by the encoder. Analogous to a variational auto-encoder (VAE) [9], we first arrive at a latent representation $\boldsymbol{z}$ for the input instance $\boldsymbol{x}$ via the encoder $q(\boldsymbol{z}|\boldsymbol{x}, y')$ and then generate the corresponding counterfactual $\boldsymbol{x}^{cf}$ via the decoder $p(\boldsymbol{x}^{cf}|\boldsymbol{z}, y')$. Following the construction in VAEs [9], we first derive the evidence lower bound (ELBO) for generating CF explanations.

Theorem 1. *The evidence lower bound to optimize CF objective* $\Pr(\boldsymbol{x}^{cf}|y', \boldsymbol{x})$ *for global feasibility is:*

$$\ln \Pr(\boldsymbol{x}^{cf}|y', \boldsymbol{x}) \geq \mathbb{E}_{Q(\boldsymbol{z}|\boldsymbol{x},y')} \ln P(\boldsymbol{x}^{cf}|\boldsymbol{z}, y', \boldsymbol{x}) - \mathbb{KL}(Q(\boldsymbol{z}|\boldsymbol{x}, y')||P(\boldsymbol{z}|y', \boldsymbol{x}) \tag{2}$$

The proof is in the Supplementary Materials. The prior of the latent variable $\boldsymbol{z}$ is modulated by y' and $\boldsymbol{x}$, but following [19], we simply use $p(\boldsymbol{z}|y', \boldsymbol{x}) \sim \mathcal{N}(\mu_{y'}, \sigma^2_{y'})$, so the KL Divergence can be computed in closed form. $P(\boldsymbol{x}^{cf}|\boldsymbol{z}, y', \boldsymbol{x})$ represents the probability of the output $\boldsymbol{x}^{cf}$ given the desired class and latent variable $\boldsymbol{z}$. This can be empirically estimated by the ℓ_1/ℓ_2 loss or any general $\mathrm{Distance}$ metric between input $\boldsymbol{x}$ and $\boldsymbol{x}^{cf}$. That is, without additional assumptions, we are assuming that probability $P(\boldsymbol{x}^{cf})$ is highest near $\boldsymbol{x}$. In addition, this probability expression is conditioned on y', implying that $\boldsymbol{x}^{cf}$ is valid only if belongs to y' class when applied with h. We thus use a classification loss (e.g., hinge-loss) between $h(\boldsymbol{x}^{cf})$ and y', where y' represents the target class and β represents the margin. We obtain,

$$-\mathbb{E}_{Q(\boldsymbol{z}|\boldsymbol{x},y')} \ln P(\boldsymbol{x}^{cf}|\boldsymbol{z}, y', \boldsymbol{x}) \approx \mathbb{E}_{Q(\boldsymbol{z}|\boldsymbol{x},y')}[\mathrm{Distance}(\boldsymbol{x}, \boldsymbol{x}^{cf}) + \lambda\, \mathrm{HingeLoss}(h(\boldsymbol{x}^{cf}), y', \beta)]$$

where λ is a hyperparameter. To summarize, given the ML model h to be explained, we learn our proposed model by minimizing the following loss function:

$$\mathop{\mathbb{E}}_{Q(\boldsymbol{z}|\boldsymbol{x},y')}[\mathrm{Distance}(\boldsymbol{x}, \boldsymbol{x}^{cf}) + \lambda\, \mathrm{HingeLoss}(h(\boldsymbol{x}^{cf}), y', \beta)] + KL(Q(\boldsymbol{z}|\boldsymbol{x}, y')||P(\boldsymbol{z}|y', \boldsymbol{x})) \tag{3}$$

where y' is the target counterfactual class. Our loss formulation bears an intuitive resemblance with the standard counterfactual loss formulation (Eq. 1). $\mathrm{HingeLoss}(f(\boldsymbol{x}^{cf}), y', \beta)$ helps us to generate valid counterfactuals with respect to the ML model h, and $\mathrm{Distance}(\boldsymbol{x}, \boldsymbol{x}^{cf})$ helps us to generate counterfactuals that are close to the input feature. The additional third term in the loss function represents the KL divergence between the prior distribution $p(\boldsymbol{z}|y')$ and the latent space encoder $q(\boldsymbol{z}|\boldsymbol{x}, y')$, analogous to the loss term in a VAE [9]. Our encoder-decoder framework can be viewed as an adaptation of VAE for the task of generating counterfactuals.

The Hinge Loss function is defined as follows:

$$HingeLoss(h(\boldsymbol{x}^{cf}), y', \beta) = max\{[\max_{y!=y'}\{h(\boldsymbol{x}^{cf})_y\} - h(\boldsymbol{x}^{cf})_{y'}], -\beta\} \tag{4}$$

The above Hinge Loss formulation encourages classifier's score on target class to be higher than any other class by at least a margin of β. Typically, the $\mathrm{Distance}$ function can be defined as the ℓ_1 distance between the input $\boldsymbol{x}$ and the counterfactual $\boldsymbol{x}^{cf}$: $\mathrm{Distance}(\boldsymbol{x}, \boldsymbol{x}^{cf}) = \|\boldsymbol{x} - \boldsymbol{x}^{cf}\|_1$. However, as we will show next, causal knowledge can be used to define a more appropriate $\mathrm{Distance}$ function.

We refer to this approach as **BaseGenCF** in the rest of the paper. The major advantage of our approach is that we end up learning a generative model $p(\boldsymbol{x}^{cf}|\boldsymbol{z}, y')$ for the counterfactuals given the input x. It makes our approach computationally attractive for generating counterfactuals for a series of inputs as compared to past works [5, 14] that solve an optimization problem for each input independently. We can directly sample any number of counterfactuals from the generative model $p(\boldsymbol{x}^{cf}|\boldsymbol{z}, y')$.

3.1.1 Augmenting Likelihood Information

Based on [5], we add another baseline **AEGenCF**, that adds a separate pre trained auto encoder loss term to **BaseGenCF**. We first train a (variational) auto encoder on the training dataset and use it to penalise counterfactuals that don't obey the training distribution. We add the following term to the loss function in Eq:(3). The training loss is as follows:

$$\text{BaseGenCFLoss}(\boldsymbol{x}) + \mathop{\mathbb{E}}_{Q(\boldsymbol{z}|\boldsymbol{x},y')} [\lambda_{ae} * \|Distance(\boldsymbol{x}^{cf}, AE(\boldsymbol{x}^{cf}))\|]$$

where AE represents the pre-trained Auto Encoder and λ_{ae} is a hyperparamter.

3.2 Modeling Causality through a SCM

We now define a feasibility-compatible notion of distance to utilize the causal knowledge about features and thus constrain independent perturbations over features. We want the counterfactual to be proximal to the data sample not only based on the Euclidean distance between them, but also based on the causal relationships between features.

Suppose we are provided with the structural causal model [16] for the observed data, including the causal graph G over $U \bigcup V$ and the functional relationships between variables. V is the set of all endogenous nodes that have at least one parent in the graph. Now for each node $v \in V$, we know the generating mechanism of v conditioned on its parents, i.e., $v = f(v_{p1}, .., v_{pk}) + \epsilon$ where ϵ denotes independent random noise. We can use the causal knowledge to define a feasibility-compatible notion of $Distance$ for the nodes $v \in V$:

$$\text{DistCausal}_{\text{v}}(\boldsymbol{x}_v, \boldsymbol{x}_v^{cf}) = \text{Distance}(\boldsymbol{x}_v^{cf}, f(\boldsymbol{x}_{v_{p1}}^{cf}, .., \boldsymbol{x}_{v_{pk}}^{cf})).$$

This distance metric indicates that the perturbation for the feature v should be related to the perturbations in its parents via the mapping f associated with the conditional distribution of the feature v, which implies that the distance is not just dependent on the original feature value $\boldsymbol{x}_v$. This addresses the problem of independent perturbations in different components of a feature and generates perturbations that follow the underlying causal distribution. That is, a counterfactual is preferable if it is close to the input instance and also preserves the associated change distribution due to the causal structure over its features. For exogenous features U, we still use the ℓ_1/ℓ_2 distance. Hence, the Distance term in BaseGenCF loss function (Eq. 3) can be modified as follows to generate counterfactuals that preserve causal constraints, where U are the exogenous nodes (i.e., nodes without any parents in the causal graph) and V are the remaining features.

$$\text{Distance}(\boldsymbol{x}, \boldsymbol{x}^{cf}) = \sum_{v \in U} \text{Distance}(\boldsymbol{x}_v^{cf}, \boldsymbol{x}_v) + \sum_{v \in V} \text{DistCausal}_{\text{v}}(\boldsymbol{x}_v, \boldsymbol{x}_v^{cf}) \tag{5}$$

Since knowing the full causal graph is often impractical, the above approach can also work whenever we have partial knowledge of the causal structure (e.g., some edges in the causal graph). From this partial causal knowledge, we construct a set of nodes V for which we know the generating mechanism for each node $v \in V$ conditioned on its parents and consider the rest of the variables in U. We refer to this approach as **SCMGenCF**.

4 Practical Methods for Learning Feasibility of CF Examples

While the SCMGenCF provides a precise formulation of the feasibility of CFs in terms of causal edges, it requires strong assumptions on knowing the causal edges and the functional form of relationships between features. Hence, we provide alternative approaches for the case when we cannot assume access to causal knowledge about the data generating process.

4.1 Approximating Feasibility Constraints

When the exact functional causal mechanism for a variable is unknown, we present an approximation of the `SCMGenCF` method that uses knowledge of certain constraints based on domain knowledge. For example, one may require that Age of a person cannot decrease, or that Education-level shares a monotonic causal relationship with Age, without knowing the true functional form. Here we propose a simple extension to **BaseGenCF** to add constraints for encoding unary and binary constraints:

Unary constraints We consider unary constraints that stipulate whether a feature can increase or decrease. This is added to the loss function eq (3) as Hinge Loss on the feature of interest. As an example, for the case that a feature can only increase, the Hinge Loss would be as follows: $-\min(0, \boldsymbol{x}_v^{cf} - \boldsymbol{x}_v)$.

Binary constraints Binary constraints capture the nature of causal relationship between two features. One of the most common are monotonic constraints, which we approximate by learning an appropriate linear model for each binary constraint. Let x_1 and x_2 be two features where x_1 causes x_2 and we have a monotonically increasing trend between them. We capture this monotonic trend by learning a linear model between x_1 and x_2, under the constraint that the parameter that relates x_1 to x_2 should be positive (or negative depending on the nature of monotonicity) . This can be learnt by minimizing the following loss function over training data: $+(\boldsymbol{x}_{v_2} - \alpha - \beta \boldsymbol{x}_{v_1}) - \min(0, \beta)$, where α and β are parameters that can be learned from training data. After we have learnt the linear model, we can proceed in the exact same fashion as we did for **SCMGenCF**, as now we know the approximate distribution f between the features. Hence, this method acts an an approximation to the SCMGenCF approach for constraints when we do not have access to the structural causal model. We call this **ModelApproxGenCF**.

4.2 Oracle Based Approach

The **ModelApproxGenCF** approach is limited since it might difficult to approximate complex constraints directly. Further, in the case of local feasibility, there may be no explicit constraints that users can easily provide. However, a user may be able to provide yes/no feedback to the generated CF examples. In this section, we thus consider learning feasibility constraints from oracle feedback on generated CFs. Specifically, we assume blackbox access to an Oracle that provides a feasibility score for each (input, counterfactual) pair. We assume that the Oracle exposes implicit user preferences or causal knowledge through a black-box interface. Similar ideas of capturing knowledge implicitly using oracles can be found in Karaletsos et al. [8].

Let us represent the Oracle O's scoring mechanism as follows: Given any input pair $(\boldsymbol{x}, \boldsymbol{x}^{cf})$ the oracle outputs 1 if the CF example is (locally) feasible, otherwise it outputs 0. Hence, in order to generate feasible counterfactuals, our task is to maximize the Oracle score. Consider g as any general mechanism to generate counterfactuals over a dataset, i.e., $\forall \boldsymbol{x} \in \mathcal{X}, \boldsymbol{x}_{cf} = g(\boldsymbol{x})$, then the task to generate feasible counterfactuals is: $\max_g \sum_{\boldsymbol{x} \in \mathcal{X}} O(\boldsymbol{x}, g(\boldsymbol{x}))$.

The oracle can be interpreted as an expert who guides a method towards generating feasible counterfactuals. However, we only have black-box access to the Oracle and there would usually be a high cost to query the oracle, especially where the oracle represents human judgment. It is thus desirable to maximize the oracle score on generated counterfactuals using as few oracle queries as possible.

Our BaseGenCF approach provides us with a convenient way to learn the oracle's knowledge using a model (o) that generates a score for a counterfactual.

$$o(\boldsymbol{x}, \boldsymbol{x}^{cf}) = \mathop{\mathbb{E}}_{Q(\boldsymbol{z}|\boldsymbol{x}, y')} [\exp\left(-(\boldsymbol{x}^{cf} - \mu)^T (\boldsymbol{x}^{cf} - \mu)\right)] \tag{6}$$

where μ represents the output of the counterfactual generator/decoder $p(\mu|\boldsymbol{z}, y')$ in our framework, with $\boldsymbol{x}$ as the input to the encoder $q(z|\boldsymbol{x}, y')$. The key task is to train the BaseGenCF framework such that the decoder becomes a better approximation of the oracle: we expect $o(x, x^{cf})$ to be higher whenever $O(x, x^{cf}) = 1$, and lower when $O(x, x^{cf}) = 0$.

4.2.1 Proposed Algorithm

We now provide the complete training algorithm, called **OracleGenCF**. It has two training phases:

1. **Base Training Phase**: Since we do not have any access to $(\boldsymbol{x}, \boldsymbol{x}^{cf})$ query pairs apriori, we train our BaseGenCF and sample counterfactuals q^{cf} from the decoder $p(\boldsymbol{x}^{cf}|\boldsymbol{z}, y')$ to create queries for the oracle O. We refer to this set of queries as Q: $\{(x, q^{cf})\}$.
2. **Oracle Modeling Phase**: We fine-tune the BaseGenCF generative framework by minimising the loss (Eq. 6) to arrive at a better approximation for Oracle O. We also add the BaseGenCF loss (Eq. 3) to ensure the counterfactuals we generate are still valid and proximal. This ensures that we do not solely focus on the task of learning a good model of the Oracle, because that

would only ensure feasibility but may violate proximity to the original input and whether the counterfactual belongs to the desired class. We also add a hyperparameter λ_o that controls the tradeoff between modeling valid counterfactuals and modeling the oracle. The exact loss function used for fine-tuning the framework is:

$$min \sum_{i \in Q} [\text{BaseGenCFLoss}(\boldsymbol{x}_i) + \lambda_o * \| O(\boldsymbol{x}_i, q_i^{cf}) - o(\boldsymbol{x}_i, q_i^{cf}) \|]$$

4.2.2 Capturing Global & Local feasibility

The OracleGenCF can be used to capture personalized user-specific constraints with a user as the oracle, thus representing human perception. Different users could be modeled using different oracles. It can represent any user-specific constraint by simply stating a counterfactual as feasible ($O(\boldsymbol{x}, \boldsymbol{x}^{cf}) = 1$) or infeasible ($O(\boldsymbol{x}, \boldsymbol{x}^{cf}) = 0$).

In addition, the method is flexible and can be used to capture a variety of feasibility constraints, including global constraints. Consider an example of feature v and its causes $(v_1, \ldots, v_p)$, with a positive Individual Treatment Effect (ITE) of each cause on the feature v. This can be captured implicitly by a simple Oracle O.

$$O(\boldsymbol{x}, \boldsymbol{x}^{cf}) = \begin{cases} 1, & \text{if (} \forall \text{i } \{\boldsymbol{x}^{cf}_{v_{pi}} > \boldsymbol{x}_{v_{pi}}\} \implies \boldsymbol{x}^{cf}_v > \boldsymbol{x}_v \text{) or (} \forall \text{i } \{\boldsymbol{x}^{cf}_{v_{pi}} < \boldsymbol{x}_{v_{pi}}\} \implies \boldsymbol{x}^{cf}_v < \boldsymbol{x}_v \text{)} \\ 0, & \text{otherwise} \end{cases}$$

5 Empirical Evaluation

We evaluate the proposed methods on a simulated dataset and a Bayesian network from past work, where the true causal model is known. We then evaluate the OracleGenerativeCF on a real world dataset (`Adult`) [10]. As we have mentioned before, in general, a counterfactual could be infeasible due to many reasons, but for our evaluation, we assume we are given a constraint that completely captures the feasibility of a counterfactual. To design this constraint, we either infer it from the causal model (simulated datasets) or from domain knowledge (real-world dataset).

5.1 Data and Feasibility Constraints

Simple-BN. We consider a toy dataset of 10,000 samples with three features $(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3)$ and one outcome variable (y). The causal relationships between them are modeled as follows:

$$p(x_1) \sim N(\mu_1, \sigma_1); \qquad p(\boldsymbol{x}_2) \sim N(\mu_2, \sigma_2)$$
$$p(x_3|x_1, x_2) \sim N(k_1 * (x_1 + x_2)^2 + b_1, \sigma_3); k_1 > 0, b_1 > 0;$$
$$p(y|x_1, x_2, x_3) \sim Bernoulli(\sigma(k_2 * (x_1 * x_2) + b_2 - x_3)); k_2 > 0, b_2 > 0$$

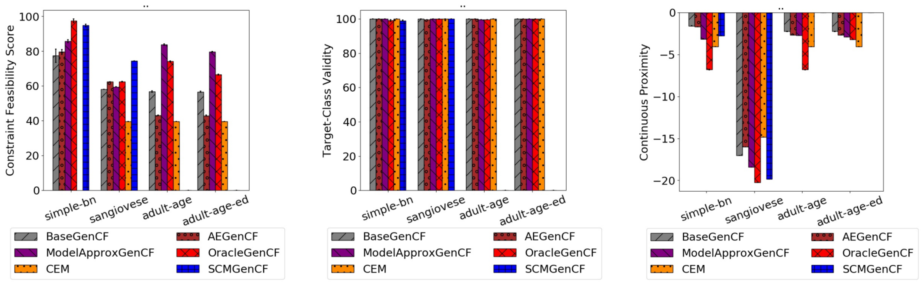
Note that we require x_1 and x_2 to be positive, so they follow a truncated normal distribution. The intuition is that $\boldsymbol{x}_3$ is determined by a monotonically increasing function of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. At the same time, y is positively affected by an increase in $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ but negatively affected by $\boldsymbol{x}_3$. Thus, a naive counterfactual method may not satisfy the monotonic constraint on $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ and $\boldsymbol{x}_3$. Specifically, the global monotonicity constraint is defined as:

C: ($\boldsymbol{x}_1, \boldsymbol{x}_2$ increase $\implies$ $\boldsymbol{x}_3$ increases) AND ($\boldsymbol{x}_1, \boldsymbol{x}_2$ decrease $\implies$ $\boldsymbol{x}_3$ decrease)

We report results for a hand-crafted set of parameters such that there is a strong tradeoff between proximity and monotonic feasibility constraint: $\mu_1 = 50$, $\mu_2 = 50$, $\sigma_1 = 15$ $\sigma_2 = 17$, $\sigma_3 = 0.5$, $k_1 = 0.0003$, $k_2 = 0.0013$, $b_1 = 10$, $b_2 = 10$.

Sangiovese [13]. This is a conditional linear Bayesian network on the effects of different agronomic settings on quality of Sangiovese grapes [2]. It has 14 features and a categorical output for quality, with a sample size of 10,000. The true causal model is known. The features are all continuous except Treatment which has 16 levels. For simplicity, we remove the categorical variable Treatment since it leads to 16 different linear functions. For feasibility, we test a monotonic constraint over two variables, $BunchN$ and $SproutN$. Specifically, the global monotonicity constraint is defined as:

C: ($SproutN$ increase $\implies$ $BunchN$ increases) AND ($SproutN$ decrease $\implies$ $BunchN$ decrease)



(a) (b) (c)

Figure 1: Constraint-Feasibility, Target-Class Validity and Cont-Proximity metrics for the three datasets.

Adult [10]. We consider the Adult Dataset to validate our approach on a real world dataset. The outcome y is binary $y = 0$ (Low Income Group), and $y = 1$ (High Income Group), with a sample size of 32561. Since we do not have the true causal model, we design constraints that capture feasibility using domain knowledge.

C1: $\boldsymbol{x}^{cf}_{Age} \geq \boldsymbol{x}_{Age}$; C2: $(\boldsymbol{x}^{cf}_{Ed} > \boldsymbol{x}_{Ed} \implies \boldsymbol{x}^{cf}_{Age} > \boldsymbol{x}_{Age})$ AND $(\boldsymbol{x}^{cf}_{Ed} = \boldsymbol{x}_{Ed} \implies \boldsymbol{x}^{cf}_{Age} \geq \boldsymbol{x}_{Age})$

C1 represents a unary constraint that Age cannot decrease in counterfactual explanations. C2 represents a monotonic constraint that increase in Educational level should increase Age, and if Educational level remains the same, age should not decrease. C2 also includes an additional constraint that Education level cannot decrease. Hence, if Education level decreases then its an infeasible counterfactual, regardless of the change in Age. We designed it this way to make the constraint practical, since in practice Education level cannot normally decrease too. To make the counterfactual generation task more challenging, we sample data points with the value of feature Age greater than 35 and outcome class $y = 0$ and data points with the value of feature Age less than 45 and the outcome class $y = 1$. This creates a setup in which higher age data points are more correlated with the low income class group and lower age data points are more correlated with the high income class group. With this sampling strategy, we obtain a dataset of size 15691 and consider the task of generating counterfactuals with the target class as $y = 1$ or the high income group. That is, $y' = 1$ is the desired output class.

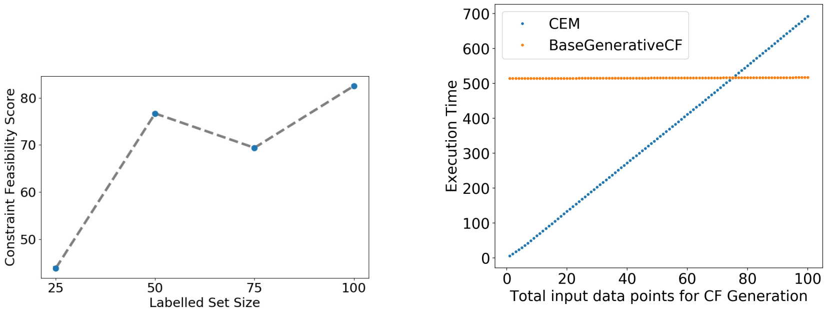
5.2 Setup

For all the experiments,the machine learning classifier h is implemented as a neural network with two hidden layers, with non linear activation (ReLU) on the first hidden layer. It is trained to minimize the cross entropy loss using Adam optimizer on the dataset with a 80-10-10 split for train/validation/test. Also, the continuous features are scaled to (0-1) range and categorical features are represented as one-hot encoded vectors. Each proposed method for counterfactual generation is trained using a 80-10-10% training, validation and test dataset respectively. For the OracleGenCF method, we additionally generate the query set Q using 10% of the training dataset with 10 counterfactuals per data point.

We report results for all proposed methods (BaseGenCF, AEGenCF, ModelApproxCF, SCMGenCF and OracleGenCF), except for the Adult dataset where we cannot apply SCMGenCF due to lack of causal knowledge about the dataset. While the OracleGenCF is designed to handle local constraints, we compare its performance on global constraints to enable a fair comparison with other methods. To see the exact modelling of constraints in ModelApproxGenCF for different datasets, please refer to the Supplementary Materials. Throughout, we also compare these methods to the Contrastive Explanations method (CEM) proposed in [5].

Evaluation Metrics. We define the following metrics to evaluate counterfactual examples:

- Target-Class Validity: Percentage of counterfactuals whose predicted class by the ML classifier is the same as the target class.



(a) (b)

Figure 2: Constraint-Feasibility score as the number of labelled examples is increased for global constraints in the Adult Dataset (a) (Other metrics are in the Supplementary Materials). (b): Time taken to generate CF examples.

- `Cont-Proximity`: We define proximity for continuous features as the ℓ_1-distance between $\boldsymbol{x}^{cf}$ and $\boldsymbol{x}$ in units of median absolute deviation for each feature, and averaged across all features [14]. It is multiplied by (-1) so that higher values are better.
- `Cat-Proximity`: We define proximity for categorical features as the total number of mismatches on categorical value between $\boldsymbol{x}^{cf}$ and $\boldsymbol{x}$ for each feature, and averaged across all features [14]. It is multiplied by (-1) so that higher values are better.
- `Constraint Feasibility Score`: For the Simple-BN and Sangiovese dataset, the constraint mentioned above can be observed as a combination of two sub constraints: X1 and X2. Hence, to ensure good performance at satisfying both the sub-constraints, we define the following metric for constraint feasibility: $\frac{2*S1*S2}{S1+S2}$ where S1, S2 represent the percentage of Counterfactuals satisfying the sub constraints X1, X2 repsectively.

  However in the case of Adult dataset, due to the additional non monotonic constraint of Education level cannot decrease, we simply report the percentage of Counterfactual satisfying the complete constraint `C2` on the Adult dataset.
- `Causal-Edge Score`: Log Likelihood of Counterfactuals w.r.t. a given causal edge distribution.
- `Causal-Graph Score`: Log Likelihood of Counterfactuals with respect to the full causal model.

Causal-Edge-Score and Causal-Graph-Score are defined only for `SimpleBN` and `Sangiovese` where the true causal model is known.

HyperParameter tuning. For a fair comparison between methods, we optimize hyperparameters using the validation set and use random search for 100 iterations. Since an ideal counterfactual needs to satisfy feasibility, target-class validity, and proximity, we select the hyperparameters that lead to maximum fasibility, while still obtaining more than 90% target class validity and proximity at least τ. τ was conservatively selected to remove models that result in much lower proximity than the BaseGenerativeCF method.

5.3 Results

Figure 1 shows `Target-Class Validity`, `Cont-Proximity` and `Constraint-Feasibility Score` for different methods, averaged over 10 runs (other metrics are in the Supplementary Materials). For Simple-BN dataset, `OracleGenCF` achieves the highest score, `SCMGenCF` achieve the highest Constraint Feasibility score on the Sangiovese dataset, while the `ModelApproxGenCF` achieves the highest Constraint Feasibility score on the Adult dataset.

Also, across the three datasets, the methods designed to preserve feasibility (`SCMGenCF`, `OracleGenCF` and `ModelApproxGenCF`) perform better than the methods `BaseGenCF`, `AEGenCF` and `CEM`. `CEM` performs the lowest on Constraint Feasibility score across all datasets, it achieves a score of zero on simple-bn dataset and around 40 percent on the Sangiovese and Adult dataset.

That said, increasing feasibility induces a tradeoff with the continuous proximity. All the methods achieve perfect score on validity across different datasets. However, with respect to proximity, higher feasibility scores for `OracleGenCF`, `ModelApproxGenCF` and `SCMGenCF` lead to a decrease in proximity compared to `BaseGenCF`, from 5% to 13% less for `SCMGenCF`, 44% to 250% less for `OracleGenCF` and 8% to 50% less for `ModelApproxGenCF` based on the dataset.

Thus, depending on the dataset and underlying causal model, achieving high fraction of feasibile CF examples requires considering a tradeoff with proximity to tune the hyperparameters.

Number of labelled CFs required from Oracle. A key question for the Oracle-based method is the number of labelled CF examples it needs. Using the Adult dataset and the non-decreasing Age constraint, we show the `Constraint-Feasibility Score` of `OracleGenCF` as we increase the number of labelled CF examples (Figure 2 (a)). For the global constraint, we find that the Feasibility Score increases with labelled inputs, reaching nearly 80% with 100 labels.

Computational Advantage. Besides feasibility, `BaseGenCF` is computationally faster than past methods like `CEM`, since it uses a generative model. In Figure 2 (b), we show the time required to generate counterfactual examples for k inputs for the Adult dataset and find that `BaseGenCF` takes less time per CF example as k increases. `CEM` has a similar execution time for each input while `BaseGenCF` takes time for initial training but then negligible time for every new input.

6 Related Work

Our work builds upon the literature on explainable ML [17, 12] by focusing on a specific type of explanation through counterfactual examples [21] and tackling the problem of feasibility. In recent work, Mothilal et al. [14] highlight causality as an important factor for preserving feasibility in counterfactuals. Dhurandhar et al. [5] and Looveren et al. [20] use an auto-encoder loss term to generate counterfactuals that follow the data distribution. Liu et al. [11] proposes the use of Generative Adversarial Networks [6] as a generative framework for counterfactuals. Prior work, however, does not consider providing a method for satisfying causal constraints in the CF generation method. Our paper extends this line of work by formally defining feasibility, providing a theoretical justification of the counterfactual loss and proposing a VAE-based generative model that can preserve causal constraints. Concurrent to this work, Parafita et al. [15] propose a method on generating CF explanations as interventions on a structural causal model but their approach depends on full knowledge of structural causal model.

More generally, there has been increasing work to incorporate causal knowledge in machine learning, such as by causally invariant representations [3] or modelling causal knowledge for generalization [4]. Our work highlights how causal knowledge about features can be used to explain machine learning models.

7 Conclusion

Feasibility in counterfactual explanations is hard to quantify due to multiple reasons that can render a counterfactual infeasible. In this work, we provided a general framework to tackle the issue of feasibility in CF explanations. We highlighted the connections between causality and feasibility in counterfactual explanations, and presented methods that can learn to preserve feasibility through causal knowledge or oracle feedback.

References

[1] Algorithms for monitoring and explaining machine learning models. `https://docs.seldon.io/projects/alibi` alibi.

[2] Bayesian network repository. `http://www.bnlearn.com/bnrepository/` sangiovese.

[3] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

[4] Yoshua Bengio, Tristan Deleu, Nasim Rahaman, Rosemary Ke, Sébastien Lachapelle, Olexa Bilaniuk, Anirudh Goyal, and Christopher Pal. A meta-transfer objective for learning to disentangle causal mechanisms. *arXiv preprint arXiv:1901.10912*, 2019.

[5] Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Paishun Ting, Karthikeyan Shanmugam, and Payel Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In *Advances in Neural Information Processing Systems*, pages 592–603, 2018.

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[7] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[8] Theofanis Karaletsos, Serge Belongie, and Gunnar Rätsch. Bayesian representation learning with oracle constraints. *arXiv preprint arXiv:1506.05011*, 2015.

[9] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[10] Ronny Kohavi and Barry Becker. Uci machine learning repository. *https://archive.ics.uci.edu/ml/datasets/adult*, 1996.

[11] Shusen Liu, Bhavya Kailkhura, Donald Loveland, and Yong Han. Generative counterfactual introspection for explainable deep learning. *arXiv preprint arXiv:1907.03077*, 2019.

[12] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.

[13] Alessandro Magrini, Stefano Di Blasi, and Federico Mattia Stefanini. A conditional linear gaussian network to assess the impact of several agronomic settings on the quality of tuscan sangiovese grapes. *Biometrical Letters*, 2017.

[14] Ramaravind Kommiya Mothilal, Amit Sharma, and Chenhao Tan. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the ACM FAT* conference (to appear)*, 2020.

[15] Álvaro Parafita and Jordi Vitrià. Explaining visual models by causal attribution. *arXiv preprint arXiv:1909.08891*, 2019.

[16] Judea Pearl. *Causality*. Cambridge University Press, 2009.

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

[18] Chris Russell. Efficient search for diverse coherent explanations. In *Proceedings of FAT**, 2019.

[19] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pages 3483–3491, 2015.

[20] Arnaud Van Looveren and Janis Klaise. Interpretable counterfactual explanations guided by prototypes. *arXiv preprint arXiv:1907.02584*, 2019.

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

A Supplementary Materials: Theorem 1 Proof

Theorem 2. *The evidence lower bound to optimize CF objective* $\Pr(\boldsymbol{x}^{cf}|y', \boldsymbol{x})$ *for global feasibility is:*

$$\begin{aligned}\ln \Pr(\boldsymbol{x}^{cf}|y', \boldsymbol{x}) \geq \mathbb{E}_{Q(\boldsymbol{z}|\boldsymbol{x},y')} \ln P(\boldsymbol{x}^{cf}|\boldsymbol{z}, y', \boldsymbol{x}) \\ -\mathbb{KL}(Q(\boldsymbol{z}|\boldsymbol{x}, y')||P(\boldsymbol{z}|y', \boldsymbol{x}))\end{aligned} \tag{7}$$

Proof. An ideal counterfactual generation model approximates $\boldsymbol{x}$ (proximity) and generates $\boldsymbol{x}^{cf}$ that are valid w.r.t desired class y'. Thus for a model we seek to maximize $P(\boldsymbol{x}^{cf}|y', \boldsymbol{x})$ where P is the underlying probability distribution over $\mathcal{X}$.

$$\begin{aligned}&\ln P(\boldsymbol{x}^{cf}|y', \boldsymbol{x}) \\ &= \ln \int P(\boldsymbol{x}^{cf}, \boldsymbol{z}|y', \boldsymbol{x}) d\boldsymbol{z} \\ &= \ln \int Q(\boldsymbol{z}|\boldsymbol{x}, y') \frac{P(\boldsymbol{x}^{cf}, \boldsymbol{z}|y', \boldsymbol{x})}{Q(\boldsymbol{z}|\boldsymbol{x}, y')} dz \\ &\geq \int Q(\boldsymbol{z}|\boldsymbol{x}, y') \ln \frac{P(\boldsymbol{x}^{cf}, \boldsymbol{z}|y', \boldsymbol{x})}{Q(\boldsymbol{z}|\boldsymbol{x}, y')} dz \\ &= \mathbb{E}_{Q(\boldsymbol{z}|\boldsymbol{x},y')} \ln \frac{P(\boldsymbol{x}^{cf}, \boldsymbol{z}|y', \boldsymbol{x})}{Q(\boldsymbol{z}|\boldsymbol{x}, y')} \\ &= \mathbb{E}_{Q(\boldsymbol{z}|\boldsymbol{x},y')} \ln P(\boldsymbol{x}^{cf}|\boldsymbol{z}, y', \boldsymbol{x}) - \mathbb{E}_{Q(\boldsymbol{z}|\boldsymbol{x},y')} \ln \frac{Q(\boldsymbol{z}|\boldsymbol{x}, y')}{P(\boldsymbol{z}|y', \boldsymbol{x})}\end{aligned} \tag{8}$$

Where the inequality above is due to Jensen's inequality. Using the definition of KL-Divergence,

$$\begin{aligned}\ln P(\boldsymbol{x}^{cf}|y', \boldsymbol{x}) \geq \mathbb{E}_{Q(\boldsymbol{z}|\boldsymbol{x},y')} \ln P(\boldsymbol{x}^{cf}|\boldsymbol{z}, y', \boldsymbol{x}) \\ - \mathbb{KL}(Q(\boldsymbol{z}|\boldsymbol{x}, y')||P(\boldsymbol{z}|y', \boldsymbol{x}))\end{aligned}$$

□

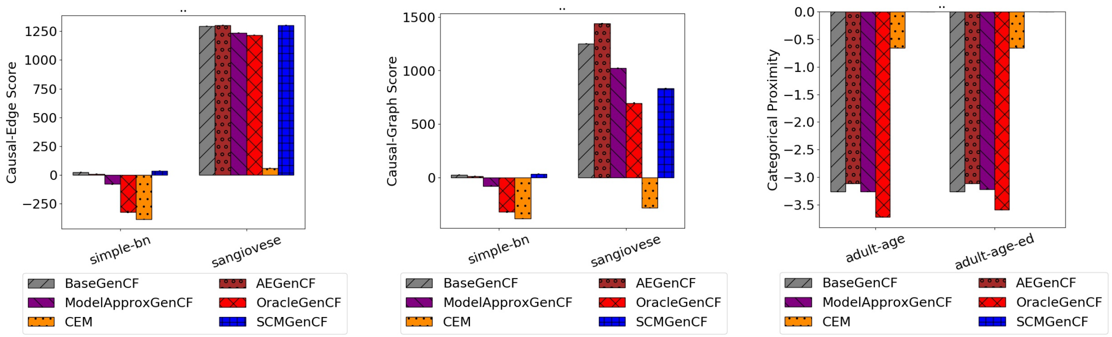
B Supplementary Materials: Additional Results

Figure 3 (a), shows the methods evaluated on the Causal Edge Score metric. We cannot evalute methods on the `Adult` dataset under this metric due to lack of causal knowledge about the dataset. `SCMGenCF` performs the best on both the `Simple-BN` and `Sangiovese` dataset.

Figure 3 (b) shows the methods evaluated on the Causal Graph Score metric. Note that the Causal Graph Score for `Sangiovese` dataset has been scaled down by a factor of 10. This was done in order to match with the range of Causal Graph score for the `Simple-BN` dataset. An interesting observation on Sangiovese dataset in this plot is that `SCMGenCF` performs worse than `BaseGenCF`, `AEGenCF` and `ModelApproxGenCF`, despite being better than them on the Causal-Edge Score, Figure 3 (a). Also, the performance of `ModelApproxCF` and `OracleGenCF` w.r.t to the performance of `BaseGenCF` and `AEGenCF` is much worse on Causal-Graph Score, as compared on the Causal-Edge Score for the Sangiovese dataset. This can be explained due to low Continuous Proximity score for them. `ModelApproxGenCF`, `OracleGenCF` and `SCMGenCF` have lower proximity than `BaseGenCF` and `AEGenCF` for `Sangiovese` dataset (Figure 1 (c)). Proximity affects the Causal Graph Score as methods with a lower proximity score would lead to lower likelihood score for corresponding feature distributions.

Figure 3 (c) shows the methods evaluated on the Cat-Proximity metric. The dataset `Simple-BN` and `Sangiovese` do not contain any categorical variables, hence we do not include them for this analysis. `CEM` performs quite well than other methods on Categorical proximity in both the cases of age (C1) and age-ed (C2) constraint.

Figure (4) shows the results on Target-Class Validity and Proximity for the analysis of Oracle performance at preserving global constraints in the `Adult` dataset with increasing labelled CF examples.



(a) (b) (c)

Figure 3: Causal-Edge Score, Causal-Graph Score and Categorical Proximity metrics for the three datasets

C Supplementary Materials: Implementation Details

C.1 Constraint Modelling in ModelApproxGenCF

For the case of Simple-BN and Sangiovese dataset, the feasibility constraint is monotonic, hence we use the *Binary Constraints* formulation of `ModelApproxGenCF`, as described in Section: 4.1. For the case of Adult Dataset, the constraint C1 is modelled using the *Unary Constraints* formluation, with a Hinge Loss on the feautre Age.

The constraint C2 in Adult Dataset is more complex than the previous constraints, since the feature Education is categorical. We model the constraint C2 under the *Unary Constraints* formulation, since it can be viewed as combinations of two unary constraints: Age cannot decrease and Education cannot decrease. The Hinge Loss on categorical variable Education is implemented by converting the embedding of categorical variable Education into a continuous value. We rank different education levels with increasing score and take a weighted sum of the categorical embedding with the scores assigned for each education category. Hence, we get a continuous score for education feature embedding which is representative of the level/rank of education. Now, we can apply the same Hinge Loss on the continuous values of education feature to put penalty on counterfactuals that decrease the level of education.

The vector we used for ranking different educations levels is as follows:

- HS-Grad, School: 0
- Bachelors, Assoc, Some-college: 1
- Masters:2
- Prof-school, Doctorate: 3

C.2 Base Encoder Decoder Architecture

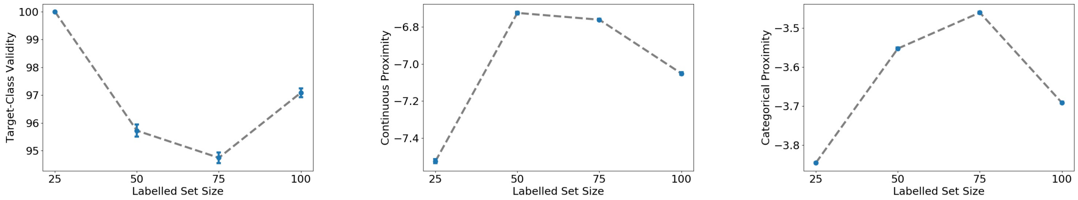
Here we provide the implementation details of the base variational encoder decoder used in all our different methods. Both the encoder and decoder are modeled as Neural Networks (NN) with multiple hidden layers and non linear activations. Encoder comprises of two Neural Networks: one NN is used to estimate the mean and other NN is used to estimate the variance of posterior distribution $q(z|x, y_k)$. Similarly, decoder comprises of a neural network to estimate the counterfactual from the latent encoding and the target class.

The latent space dimension is set to 10 for all the different methods and datasets. Both the encoder and the decoder are conditioned on the target counterfactual class.

Encoder Architecture:

Neural Network estimating mean of the latent distribution:

- Hidden Layer 1(DataSize, 20), BatchNorm, Dropout(0.1), ReLU



(a) (b) (c)

Figure 4: Target-Class Validity (a) and Proximity (b), (c) as the number of labelled examples is increased for global constraints in the Adult dataset

- Hidden Layer 2(20, 16), BatchNorm, Dropout(0.1), ReLU
- Hidden Layer 3(16, 14), BatchNorm, Dropout(0.1), ReLU
- Hidden Layer 4(14, 12), BatchNorm, Dropout(0.1), ReLU
- Hidden Layer 5(12, LatentDim)

Neural Network estimating variance of the latent distribution:

- Hidden Layer 1(DataSize, 20), BatchNorm, Dropout(0.1), ReLU
- Hidden Layer 2(20, 16), BatchNorm, Dropout(0.1), ReLU
- Hidden Layer 3(16, 14), BatchNorm, Dropout(0.1), ReLU
- Hidden Layer 4(14, 12), BatchNorm, Dropout(0.1), ReLU
- Hidden Layer 5(12, LatentDim), Sigmoid

Decoder Architecture:

The following Neural Network architecture is used for estimating the counterfactual:

- Hidden Layer 1(LatentDim, 12), BatchNorm, Dropout(0.1), ReLU
- Hidden Layer 2(12, 14), BatchNorm, Dropout(0.1), ReLU
- Hidden Layer 3(14, 16), BatchNorm, Dropout(0.1), ReLU
- Hidden Layer 4(16, 20), BatchNorm, Dropout(0.1), ReLU
- Hidden Layer 5(20, DataSize), Sigmoid

C.3 Contrastive Explanantions

For experiments involving the Contrastive Explanations (CEM) method [5], we used the implementation provided by the open source library ALIBI [1]. Since the choice of auto encoder is not specified in ALIBI, we trained our own auto encoder with the same architecture as defined in the previous subsection (C.2)