Learning Model-Agnostic Counterfactual Explanations for Tabular Data

Martin Pawelczyk
University of Tuebingen
Tuebingen, Germany
martin.pawelczyk@uni-tuebingen.de

Klaus Broelemann
Schufa Holding AG
Wiesbaden, Germany
klaus.broelemann@schufa.de

Gjergji Kasneci
University of Tuebingen
Tuebingen, Germany
gjergji.kasneci@uni-tuebinge.de

ABSTRACT

Counterfactual explanations can be obtained by identifying the smallest change made to an input vector to influence a prediction in a positive way from a user's viewpoint; for example, from 'loan rejected' to 'awarded' or from 'high risk of cardiovascular disease' to 'low risk'. Previous approaches would not ensure that the produced counterfactuals be *proximate* (i.e., not local outliers) and *connected* to regions with substantial data density (i.e., close to correctly classified observations), two requirements known as *counterfactual faithfulness*. Our contribution is twofold. First, drawing ideas from the manifold learning literature, we develop a framework, called C-CHVAE, that generates *faithful counterfactuals*. Second, we suggest to complement the catalog of counterfactual quality measures using a criterion to quantify the *degree of difficulty* for a certain counterfactual suggestion. Our real world experiments suggest that *faithful counterfactuals* come at the cost of higher *degrees of difficulty*.

KEYWORDS

Transparency, Counterfactual explanations, Interpretability

ACM Reference Format:
Martin Pawelczyk, Klaus Broelemann, and Gjergji Kasneci. 2020. Learning Model-Agnostic Counterfactual Explanations for Tabular Data. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3366423.3380087

1 INTRODUCTORY REMARKS

Machine learning models are increasingly being deployed to automate high-stake decisions in industrial applications, e.g., financial, employment, medical or public services. Wachter et al. [22] discuss to establish a legally binding right to request explanations on any prediction that is made based on personal data of an individual. In fact, the EU General Data Protection Regulation (GDPR) includes a right to request "meaningful information about the logic involved, as well as the significance and the envisaged consequences" [22] of automated decisions.

As people are increasingly being affected by these automated decisions, it is natural to ask how those affected can be empowered to receive desired results in the future. To this end, Wachter et al. [22] suggest using counterfactual explanations. In this context, a counterfactual is defined as a small change made to the input vector to influence a classifier's decision in favor of the person represented by the input vector.

WWW '20, April 20–24, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380087

1.1 A step towards user empowerment

The "close world" desideratum. At a high level, Wachter et al. [22] formulated the desideratum that counterfactuals should come from a 'possible world' which is 'close' to the user's starting point. Laugel et al. [11] formalized the *close world* desideratum and split it into two measurable criteria, *proximity* and *connectedness*. Proximity describes that counterfactuals should not be local outliers and connectedness quantifies whether counterfactuals are close to correctly classified observations. We shortly review both criteria in section 5. To these two criteria, we add a third one based on percentile shifts of the cumulative distribution function (CDF) of the inputs, as a measure for the *degree of difficulty*. Intuitively, all criteria help quantify how *attainable* suggested counterfactuals are.

The C-CHVAE. In this work, our main contribution is a general-purpose framework, the *Counterfactual Conditional Heterogeneous Autoencoder*, C-CHVAE, which allows finding (multiple) counterfactual feature sets while generating counterfactuals with high occurrence probability. This is a fundamental requirement towards *attainability* of counterfactuals. In particular, our framework is compatible with a multitude of autoencoder (AE) architectures as long as the AE allows both modelling of heterogeneous data and approximating the conditional log likelihood of the the mutable/free inputs given the immutable/protected ones. Moreover, the C-CHVAE does not require access to a distance function (for the input space) and is classifier agnostic. Part of this work was previously published as a *NeurIPS HCML workshop paper* [15]. Our source code can be found at: https://github.com/MartinPawel/c-chvae.

1.2 Challenges for counterfactuals

Attainability. Intuitively, a counterfactual is attainable, if it is jointly (1) a 'close' suggestion that is not a local outlier, (2) similar to correctly classified observations and (3) associated with low total CDF percentile shifts. Hence, in our point of view, *attainability* is a composition of faithful counterfactuals ((1) and (2)) which are at the same time not too difficult to attain (3). To reach a better understanding, let us translate conditions (1), (2) and (3) into the following synthetic bank loan setting: a client applies for a loan and a bank employs a counterfactual empowerment tool. Under these circumstances, we focus on one problematic aspect. The tool could make suggestions that '*lie outside of a client's wheelhouse*', that is to say, it is not reasonable to suggest counterfactuals that (a) one would typically not observe in the data, (b) that are not typical for the subgroup of users the client belongs to, and that (c) are extremely difficult to attain, where difficulty is measured in terms

of the percentiles of the CDF of the given inputs. For example, in table 1, the suggestion made by the second method is likely not attainable given her age and education level.

Similarity via latent distance. Additionally, in health, banking or credit scoring contexts we often face continuous, ordinal and nominal inputs concurrently. This is also known as *heterogeneous* or *tabular* data. For this type of data, it can sometimes be difficult to measure distance in a meaningful way (e. g. measuring distance between different occupations). Furthermore, existing methods leave the elicitation of appropriate distance/cost functions up to (expert) opinions [4, 10, 12, 21, 22], which can vary considerably across individuals [5]. Therefore, we suggest measuring similarity between the input feature $\boldsymbol{x}_i$ and a potential counterfactual $\tilde{\boldsymbol{x}}_i$ as follows.

Definition 1 (Latent distance). *Let $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathbb{R}^n$ be two observations in input space with corresponding lower dimensional representations $\boldsymbol{z}_i, \boldsymbol{z}_j \in \mathbb{R}^k$ with $k < n$, in latent space. Then the distance $d_L(\boldsymbol{x}_i, \boldsymbol{x}_j) := \|z_i - z_j\|_p$ is called the latent distance of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.*

1.3 Overview

Learning faithful counterfactuals via the C-CHVAE. We suggest embedding counterfactual search into a data density approximator, here a variational autoencoder (VAE) [9]. The idea is to use the VAE as a *search device* to find counterfactuals that are *proximate* and *connected* to the input data. The intuition of this approach becomes apparent by considering each part of the VAE in turn. As opposed to classical generative model contexts, the encoder part is not discarded at *test time/generation time*. Indeed, it is the *trained encoder* that plays a crucial role: given the original heterogeneous data, the encoder specifies a lower dimensional, real-valued and dense representation of that data, $\boldsymbol{z}$. Therefore, it is the encoder that determines which low-dimensional neighbourhood we should look to for potential counterfactuals. Next, we perturb the low dimensional data representation, $\boldsymbol{z} + \boldsymbol{\delta}$, and feed the perturbed representation into the decoder. For small perturbations the decoder gives a potential counterfactual by reconstructing the input data from the perturbed representation. This counterfactual is likely to occur. Next, the potential counterfactual is passed to the pretrained classifier, which we ask whether the prediction was altered. Figure 1 represents this mechanism.

Consistent search for heterogeneous data. While we aim to avoid altering immutable inputs, such as *age* or *education*, it is reasonable to believe that the immutable inputs can have an impact on what is attainable to the individual. Thus, the immutable inputs should influence the neighbourhood search for counterfactuals. For example, certain drugs can have different treatment effects, depending on whether a patient is male or female [16]. Hence, we wish to generate *conditionally consistent* counterfactuals.

Again, consider Figure 1 for an intuition of counterfactual search in the presence of immutable inputs. Unlike in vanilla VAEs, we assume a Gaussian mixture prior on the latent variables where each mixture component is also estimated by the immutable inputs. This helps cluster the latent space and has the advantage that we look for counterfactuals among *semantically similar* alternatives.

Contribution. The C-CHVAE is a general-purpose framework that generates counterfactuals. Its main merits are:

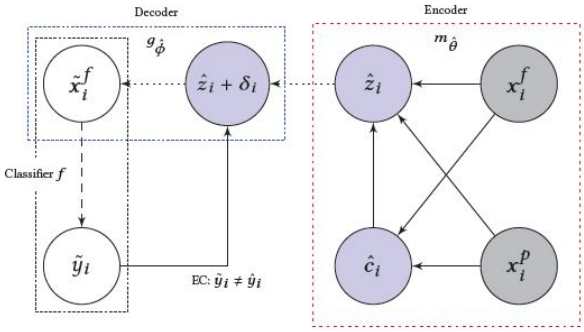


Figure 1: Autoencoding Counterfactual search. The learned encoder, $m_{\hat{\theta}}$, maps heterogeneous protected and free features, $\boldsymbol{x}^p$ and $\boldsymbol{x}^f$, and latent mixture components, $\hat{c}$, into a latent representation, $\hat{z}$. The learned decoder, $g_{\hat{\phi}}$, reconstructs the free inputs $\boldsymbol{x}^f$ from the perturbed representation, providing a potential counterfactual, $\tilde{\boldsymbol{x}} = (\boldsymbol{x}^p, \tilde{\boldsymbol{x}}^f)$. The counterfactual acts like a typical observation from the data distribution. Next, we feed the potential counterfactual $\tilde{\boldsymbol{x}}$ to the classifier, f. We stop the search, if the EC condition is met.

- **Faithful counterfactuals.** The generated counterfactuals are *proximate* and *connected* to regions of high data density and therefore likely attainable, addressing the most important desiderata in the literature on counterfactuals [11, 22];
- **Suitable for tabular data and classifier agnostic.** The data distribution is modelled by an autoencoder that handles heterogeneous data and interval constraints by choosing appropriate likelihood models. It can also be combined with a multitude of autoencoder architectures [7, 9, 13, 14, 18, 20];
- **No ad-hoc distance measures for *input* data.** The C-CHVAE does not require ad-hoc predefined distance measures for input data to generate counterfactuals. This is can be an advantage over existing work, since it can be difficult to devise meaningful distance measures for tabular data.

2 RELATED LITERATURE

Explainability through counterfactuals. At a meta level, the major difference separating our work from previous approaches is that we *learn* a separate model to learn similarity in latent space and use this model to generate counterfactual recommendations. Doing this allows us to generate counterfactuals that lie on the data manifold.

Approaches dealing with heterogeneous data rely on integer programming optimization [17, 21]. To produce counterfactuals that take on reasonable values (e. g. non negative values for wage income) one directly specifies the set of features and their respective support subject to change. The C-CHVAE also allows for such constraints by choosing the likelihood functions for each feature appropriately (see Section 4.2 and our github repo.).

A closely related collection of approaches assumes that distances or costs between any two points can be measured in a meaningful way [4, 10, 12, 21, 22]. The C-CHVAE, however, does not rely on

Method	ID	Input subset	Current	Percentile	Counterfactual	Percentile	Shift	Tot. shift	L. Outlier	Connected
I	1	`credit card debt`	5000	**55**	3500	**75**	**20**	40	No	Yes
		`saving account`	200	**45**	600	**65**	**20**			
II	1	`monthly income ($)`	2500	**40**	10000	**95**	**55**	75	Yes	No
		`# loans elsewhere`	5	**85**	2	**65**	**20**			

Table 1: Hypothetical counterfactuals for the same 22 year old individual without a college degree, who was denied credit. Suggestions were made by two different methods for a given classifier f. The rows suggest how a subset of free inputs would need to change to obtain credit, i. e. from $\hat{y} = 0$ to $\hat{y} = 1$. For the first empowerment technology, the suggestion might be reasonable whereas for the second one, the suggestion looks atypical and could be difficult to attain measured in terms of connectedness to existing knowledge, an outlier measure and the total percentile shift.

Method	Train	Classifier agnostic	Classifier	Tabular data
AR [21]	No	No	Lin. Models	No
HCLS [10]	No	No	SVM	No
GS [12]	No	Yes	All	No
FT [19]	No	No	Trees	No
C-CHVAE (ours)	Yes	Yes	All	Yes

Table 2: Overview of existing counterfactual generation methods. 'Train' indicates that a method requires training. 'Classifier agnostic' means whether a method can be combined with any black-box classifier.

task-specific, predefined similarity functions between the inputs and counterfactuals. For a *given autoencoder architecture*, we learn similarity between inputs and counterfactuals from the data.

Other approaches strongly rely on the pretrained classifier f and make use of restrictive assumptions, e. g. that f stems from a certain hypothesis class. For example, Ustun et al. [21] and Tolomei et al. [19] assume the pretrained classifiers to be linear or tree based respectively, which can restrict usefulness.

In independent work from our's, Joshi et al. [8] suggest a similar explanation model, however, they focus on causal models and are less concerned with the issue of evaluating counterfactual explanations.

Adversarial perturbations. Since counterfactuals are often generated independently of the underlying classification model, they are related to universal adversarial attacks (see for example Brown et al. [3]). While adversarial examples aim to alter the prediction a deep neural network makes on a data point via small and *imperceptible* changes, counterfactuals aim to alter data points to suggest *impactful* changes to individuals. Notice that counterfactuals do not fool a classifier in a classical sense, since individuals need to *exert real-world effort* to achieve the desired prediction. Since a review of the entire literature on adversarial attacks goes beyond the scope of this work, we refer the reader to the survey by Akhtar and Mian [2]. For an overview of counterfactual generation methods consider table 2.

Notation. In the remainder of this work, we denote the D dimensional feature space as $\mathcal{X} = \mathbb{R}^D$ and the feature vector for observation i by $\boldsymbol{x}_i \in \mathcal{X}$. We split the feature space into two disjoint feature subspaces of immutable (i. e. protected) and free features denoted by $\mathcal{X}_p = \mathbb{R}^{D_p}$ and $\mathcal{X}_f = \mathbb{R}^{D_f}$ respectively such that w.l.o.g. $\mathcal{X} = \mathcal{X}_p \times \mathcal{X}_f$ and $\boldsymbol{x}_i = (\boldsymbol{x}_i^p, \boldsymbol{x}_i^f)$. This means in particular that the d-th free feature of x_i is given by $x_{d,i}^f = x_{d,i}$ and the d-th protected feature is given by $x_{d,i}^p = x_{d+D_f,i}$. Let $\boldsymbol{z} \in \mathcal{Z} = \mathbb{R}^k$ denote the latent space representation of $\boldsymbol{x}$. The labels corresponding to the i'th observation are denoted by $y_i \in \mathcal{Y} = \{0, 1\}$. Moreover, we assume a given pretrained classifier $f : \mathcal{X} \to \mathcal{Y}$. Further, we introduce the following sets: $H^- = \{\boldsymbol{x} \in \mathcal{X} : f(\boldsymbol{x}) = 0\}, H^+ = \{\boldsymbol{x} \in \mathcal{X} : f(\boldsymbol{x}) = 1\}, D^+ = \{\boldsymbol{x}_i \in \mathcal{X} : y_i = 1\}$. We attempt to find an explainer $E : \mathcal{X} \to \mathcal{X}$, generating counterfactuals $E(\boldsymbol{x}) = \tilde{\boldsymbol{x}}$, such that $f(\boldsymbol{x}) \neq f(E(\boldsymbol{x}))$. Finally, values with $\hat{\cdot}$ usually denote estimated quantities, values carrying $\tilde{\cdot}$ denote candidate values and values with $\cdot^*$ denote the best value among a number of candidate values.

3 BACKGROUND

3.1 (Conditional) Variational Autoencoder

The simple VAE is often accompanied by an isotropic Gaussian prior $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. We then aim to optimize the following objective known as the Evidence Lower Bound (ELBO),

$$\mathrm{L}_{VAE}(p, q) = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x}^f)}[\log p(\boldsymbol{x}^f|\boldsymbol{z})] - \mathrm{D}_{KL}[q(\boldsymbol{z}|\boldsymbol{x}^f)||p(\boldsymbol{z})].$$

This objective bounds the data log likelihood, $\log\ p(\boldsymbol{x}^f)$, from below. In the simple model, the decoder and the encoder are chosen to be Gaussians, that is, $q(\boldsymbol{z}|\boldsymbol{x}^f) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}_q, \Sigma_q)$ and $p(\boldsymbol{x}^f|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}^f|\boldsymbol{\mu}_p, \Sigma_p)$, where the distributional parameters $\boldsymbol{\mu}(\cdot)$ and $\Sigma(\cdot)$ are estimated by neural networks. If all inputs were binary instead, one could use a Bernoulli decoder, $p(\boldsymbol{x}^f|\boldsymbol{z}) = Ber(\boldsymbol{x}^f|\varrho_p(\boldsymbol{z}))$.

Conditioning on a set of inputs, say $\boldsymbol{x}^p$, the objective that bounds the conditional log likelihood, $\log\ p(\boldsymbol{x}^f|\boldsymbol{x}^p)$, can be written as [18],

$$\begin{aligned}\mathrm{L}_{CVAE}(p, q) = &\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x}^f, \boldsymbol{x}^p)}[\log p(\boldsymbol{x}^f|\boldsymbol{z}, \boldsymbol{x}^p)] - \\ &\mathrm{D}_{KL}[q(\boldsymbol{z}|\boldsymbol{x}^f, \boldsymbol{x}^p)||p(\boldsymbol{z}|\boldsymbol{x}^p)],\end{aligned} \tag{1}$$

where one assumes that the prior $p(\boldsymbol{z}|\boldsymbol{x}^p)$ is still an isotropic Gaussian, i.e. $\boldsymbol{z}|\boldsymbol{x}^p \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. We will refer to this model as the CVAE.

4 C-CHVAE

In this part, we present both our objective function and the CHVAE architecture in Sections 4.1 and 4.2, respectively.

4.1 The C- in C-CHVAE

We take the pretrained, potentially non-linear, classifier $f(\cdot)$ as given, which can also be a training time fairness constraint classifier [1, 23]. Let us denote the encoder function, parameterized by θ, by $m_\theta(\cdot; \boldsymbol{x}^p))$, taking arguments $\boldsymbol{x}^f$. The decoder function, parametrized by ϕ, is denoted by $g_\phi(\cdot)$. It has inputs $m_\theta(\boldsymbol{x}^f; \boldsymbol{x}^p) = z \in \mathbb{R}^k$. Then our objective reads as follows,

$$\min_{\delta \in \mathcal{Z}} ||\boldsymbol{\delta}|| \text{ subject to} \tag{2}$$

$$f(g_{\hat{\phi}}(m_{\hat{\theta}}(\boldsymbol{x}) + \boldsymbol{\delta}), \boldsymbol{x}^p) \neq f(\boldsymbol{x}^f, \boldsymbol{x}^p) \; \& \tag{3}$$

$$\min_{\phi, \theta} \ell(\boldsymbol{x}, g_\phi(m_\theta(\boldsymbol{x}^f; \boldsymbol{x}^p))) - \Omega(m_\theta(\boldsymbol{x}^f; \boldsymbol{x}^p)), \tag{4}$$

where $\Omega(\cdot)$ is a regularizer on the latent space and $\|\cdot\|$ denotes the p-norm. The idea behind the objective is as follows. First, (4) approximates the conditional log likelihood, $p(\boldsymbol{x}^f|\boldsymbol{x}^p)$, while learning a lower dimensional latent representation. Subsequently, we use this latent representation, $\hat{\boldsymbol{z}} = m_{\hat{\theta}}(\boldsymbol{x}^f; \boldsymbol{x}^p)$, to search for counterfactuals ((2) and (3)). If the perturbation on $\hat{\boldsymbol{z}}$ is small enough, the trained decoder $g_{\hat{\phi}}$ gives a reconstruction $\tilde{\boldsymbol{x}}^{f*}$ that (a) is similar to $\boldsymbol{x}^f$, (b) satisfies the empowerment condition (3), and (c) lies in regions where we would usually expect data. Also, notice that this regularizer effectively plays the role of the distance function. It determines the neighbourhood of $\boldsymbol{x}$ in which we search for counterfactuals.

4.2 CHVAE

To solve the above optimization problem defined in (2)-(4), it is crucial to elicit an appropriate autoencoder architecture. We adjust the HVAE [14] so that it approximates conditional densities.

Factorized decoder. We suggest using the following hierarchical model to accommodate the generation of counterfactuals conditional on some immutable attributes. The factorized decoder with a conditional uniform Gaussian mixture prior ((5) and (6)) with parameters $\pi_l = 1/L$ for all mixture components l reads:

$$p(\boldsymbol{c}_i|\boldsymbol{x}_i^p) = Cat(\boldsymbol{\pi}) \tag{5}$$

$$p(\boldsymbol{z}_i|\boldsymbol{c}_i, \boldsymbol{x}_i^p) = \mathcal{N}(\boldsymbol{\mu}_p(\boldsymbol{c}_i), \boldsymbol{I}_K) \tag{6}$$

$$\begin{aligned} p(\boldsymbol{z}_i, \boldsymbol{x}_i^f, \boldsymbol{c}_i|\boldsymbol{x}_i^p) &= p(\boldsymbol{z}_i, \boldsymbol{c}_i|\boldsymbol{x}_i^p) \prod_{d=1}^{D_f} p(\boldsymbol{x}_{d,i}^f|\boldsymbol{z}_i, \boldsymbol{c}_i, \boldsymbol{x}_i^p) \\ &= p(\boldsymbol{z}_i|\boldsymbol{c}_i, \boldsymbol{x}_i^p) p(\boldsymbol{c}_i|\boldsymbol{x}_i^p) \cdot \prod_{d=1}^{D_f} p(\boldsymbol{x}_{d,i}^f|\boldsymbol{z}_i, \boldsymbol{c}_i, \boldsymbol{x}_i^p), \end{aligned} \tag{7}$$

where $\boldsymbol{z}_i \in \mathbb{R}^k$ is the continuous latent vector and $\boldsymbol{c}_i \in \mathbb{R}^C$ is a vector indicating mixture components, generating the instance $\boldsymbol{x}_i^f \in \mathbb{R}^{D_f}$. Note that (5) and (6), where we assume independence between $\boldsymbol{c}_i$ and $\boldsymbol{x}_i^p$, are analogous to the prior on $\boldsymbol{z}$ in the CVAE above, (1). Moreover, the intuition behind the mixture prior is to facilitate clustering of the latent space in a meaningful way.

Since the factorized decoder in (7) is a composition of various likelihood models, we can use one likelihood function per input, giving rise to modelling data with real-valued, positive real valued, count, categorical and ordinal values, concurrently. Additionally, the modelling framework lets us specify a variety of interval constraints by choosing likelihoods appropriately (e.g. truncated normal distribution or Beta distribution for interval data).

Factorized encoder. Then the factorized encoder is given by:

$$\begin{aligned} q(\boldsymbol{c}_i|\boldsymbol{x}_i^p, \boldsymbol{x}_i^f) &= Cat(\boldsymbol{\pi}(\boldsymbol{x}_i^p, \boldsymbol{x}_i^f)) \\ q(\boldsymbol{z}_i|\boldsymbol{x}_i^f, \boldsymbol{x}_i^p, \boldsymbol{c}_i) &= \mathcal{N}(\boldsymbol{\mu}_q(\boldsymbol{x}_i^f, \boldsymbol{x}_i^p, \boldsymbol{c}_i), \Sigma_q(\boldsymbol{x}_i^f, \boldsymbol{x}_i^p, \boldsymbol{c}_i)) \\ q(\boldsymbol{z}_i, \boldsymbol{x}_i^f, \boldsymbol{c}_i|\boldsymbol{x}_i^p) &= q(\boldsymbol{z}_i, \boldsymbol{c}_i|\boldsymbol{x}_i^p) \prod_{d=1}^{D_f} p(\boldsymbol{x}_{d,i}^f|\boldsymbol{z}_i, \boldsymbol{c}_i, \boldsymbol{x}_i^p) \\ &= q(\boldsymbol{z}_i|\boldsymbol{c}_i, \boldsymbol{x}_i^p) q(\boldsymbol{c}_i|\boldsymbol{x}_i^p) \cdot \prod_{d=1}^{D_f} p(\boldsymbol{x}_{d,i}^f|\boldsymbol{z}_i, \boldsymbol{c}_i, \boldsymbol{x}_i^p). \end{aligned} \tag{8}$$

Parameter sharing and likelihood models. Unlike in the vanilla CVAE in (1), which is only suitable for one data type at the time, the decoder was factorized into multiple likelihood models. In practice, one needs to carefully specify one likelihood model per input dimension $p(\boldsymbol{x}_{d,i}^f|\boldsymbol{z}_i, \boldsymbol{c}_i)$. In our github repository, we describe more details of the model architecture and which likelihood models we have chosen.

ELBO. The evidence lower bound (ELBO) can be derived as:

$$\begin{aligned} \log p(\boldsymbol{x}^f|\boldsymbol{x}^p) \geq \; & \mathbb{E}_{q(\boldsymbol{c}_i, \boldsymbol{z}_i|\boldsymbol{x}_i^f, \boldsymbol{x}_i^p)} \sum_{d=1}^{D_f} \log p(\boldsymbol{x}_{d,i}^f|\boldsymbol{z}_i, \boldsymbol{c}_i, \boldsymbol{x}_i^p) \\ & - \sum_i \mathbb{E}_{q(\boldsymbol{c}_i|\boldsymbol{x}_i^f, \boldsymbol{x}_i^p)} \mathrm{D}_{KL}[q(\boldsymbol{z}_i|\boldsymbol{c}_i, \boldsymbol{x}_i^f, \boldsymbol{x}_i^p)||p(\boldsymbol{z}_i|\boldsymbol{c}_i, \boldsymbol{x}_i^p)] \\ & - \sum_i \mathrm{D}_{KL}[q(\boldsymbol{c}_i|\boldsymbol{x}_i^f, \boldsymbol{x}_i^p)||p(\boldsymbol{c}_i|\boldsymbol{x}_i^p)] \end{aligned}$$

where we recognize the influence of the factorized decoder in the first line, effectively allowing us to model complex, heterogeneous data distributions.

4.3 Counterfactual search algorithm

As inputs, our algorithm requires any pretrained classifier f and the trained decoder and encoder from the CHVAE. It returns the closest $E(\boldsymbol{x})$ due to a nearest neighbour style search in the latent space. The details can be found in our github repository, but it uses a standard procedure to generate random numbers distributed uniformly over a sphere [6, 12] around the latent observation $\hat{\boldsymbol{z}}$. Thus, we sample observations $\tilde{\boldsymbol{z}}$ in l_p-spheres around the point $\hat{\boldsymbol{z}}$ until we find a counterfactual explanation $\tilde{\boldsymbol{x}}^*$.

5 EVALUATING ATTAINABILITY OF COUNTERFACTUALS

To quantify faithfulness, [11] suggest two measures, which we shortly review here since they do not belong to the catalog of commonly used evaluation measures (such as for example accuracy). Their two suggested measures quantify *proximity* (i.e. whether $E(\boldsymbol{x})$ is a local outlier) and *connectedness* (i.e. whether $E(\boldsymbol{x})$ is connected to other correctly classified observations from the same class). However, these measures do not indicate the *degree of difficulty* for the individual to attain a certain counterfactual given the current state. We suggest two appropriate measures in 5.2.

5.1 Counterfactual faithfulness

Proximity. Ideally, the distance between a counterfactual explanation $E(\boldsymbol{x})$ and its closest, non-counterfactual neighbour $\boldsymbol{a}_0 \in H^+ \cap D^+$ should be small:

$$\boldsymbol{a}_0 = \underset{\boldsymbol{x} \in H^+ \cap D^+}{\arg\min} \; d(E(\boldsymbol{x}), \boldsymbol{x}).$$

Moreover, it is required that the observation resembling our counterfactual, $\boldsymbol{a}_0$, be close to the rest of the data, which gives rise to the following relative metric:

$$P(E(\boldsymbol{x})) = \frac{d(E(\boldsymbol{x}), \boldsymbol{a}_0)}{\min\limits_{\boldsymbol{x} \neq \boldsymbol{a}_0 \in H^+ \cap D^+} d(\boldsymbol{a}_0, \boldsymbol{x})}.$$

The intuition behind this measure is to help evaluate whether counterfactuals are outliers relative to correctly classified observations.

Connectedness. We say that that a counterfactual $\boldsymbol{e}$ and an observation $\boldsymbol{a}$ are ϵ-chained, with $\epsilon > 0$, if there exists a sequence $\boldsymbol{e}_0, \boldsymbol{e}_1, ..., \boldsymbol{e}_N \in \mathcal{X}$ such that $\boldsymbol{e}_0 = \boldsymbol{e}, \boldsymbol{e}_N = \boldsymbol{a}$ and $\forall i < N, d(\boldsymbol{e}_i, \boldsymbol{e}_{i+1}) < \epsilon$ and $f(\boldsymbol{e}) = f(\boldsymbol{a})$. Now, given an appropriate value for ϵ, we can evaluate the connectedness of a counterfactual $E(\boldsymbol{x})$ using a binary score: $C(E(\boldsymbol{x})) = 1$, if $E(\boldsymbol{x})$ is ϵ-connected to $\boldsymbol{a} \in H^+ \cap D^+$ and $C(E(\boldsymbol{x})) = 0$, otherwise.

5.2 Degree of difficulty

Individual costs of counterfactuals. We suggest to measure the degree of difficulty of a certain counterfactual suggestion $\tilde{\boldsymbol{x}}$ in terms of the percentiles of $x_d^f = \{x_{i,d}\}_{i=1}^N$ and $\tilde{x}_d^{f*}$: $Q_f(\tilde{x}_d^{f*})$ and $Q_d(x_d^f)$ where $Q_d(\cdot)$ is the cumulative density function of x_d^f. As an example, a cost of p suggests changing a free feature by at least p percentiles to receive a desired result.

We suggest two measures with the following properties: (a) $cost(x_d^f; x_d^f) = \mathbf{0}_{N\times 1}$, implying that staying at the current state is costless and (b) $cost(\tilde{x}^f + \nu \mathbf{1}_{N\times 1}; x^f) \geq cost(\tilde{x}^f; x^f)$ with $\nu \geq 0$, that is, the further from the current state, the more difficulties we have to incur to achieve the suggestion. The *difficulty measures* then read as follows:

$$cost_1(\tilde{x}^{f*}; x^f) = \sum_{d=1}^{D_f} |(Q_d(\tilde{x}_d^{f*}) - Q_d(x_d^f)|, \tag{9}$$

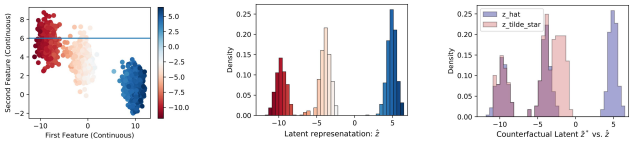
$$cost_2(\tilde{x}^{f*}; x^f) = \max_d |Q_d(\tilde{x}_d^{f*}) - Q_d(x_d^f)|. \tag{10}$$

The total percentile shift (TS) in (9) can be thought of as a baseline measure for how attainable a certain counterfactual suggestion might be. The maximum percentile shift (MS) in (10) across all free features reflects the maximum difficulty across all mutable features.

6 EXPERIMENTS

6.1 Synthetic experiments

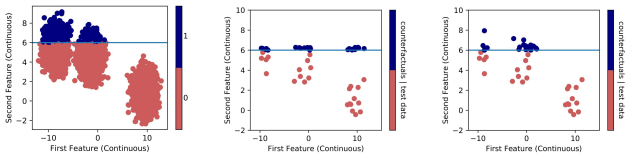
Homogeneous features. We begin by describing a data generating processes (DGP) for which it can be difficult to identify faithful counterfactuals. Example 1 corresponds to the case when all features are numerical. We generate 10000 observations from this DGP. We assume that the constant classifier $I(x_2 > 6)$ is given to us and our goal is to find counterfactuals for observations with 0-labels.



(a) Reconstructed train data generated by different $\hat{z}$ (coloured).

(b) Density of $\hat{z}$. Colours aligned so that left red $\hat{z}$ generates left $\hat{x}$ in 2(a).

(c) Density of $\hat{z}$ from 2(b) in blue. Density of $\tilde{z}^*$ (red) belongs to E(x) from 2(f) .



(d) True DGP.

(e) Test data and $E(\boldsymbol{x})$ by GS/AR (not shown). Upper right $E(\boldsymbol{x})$ lie where no data is expected.

(f) Test data and $E(\boldsymbol{x})$ by our cchvae. Most $E(\boldsymbol{x})$ lie in high-density areas and are connected.

Figure 2: Example 1. Homogeneous features. Figure 2(c) shows that generating close and meaningful counterfactuals amounts to finding the closest latent code from only 2 of the 3 modes of the latent distribution.

Figure 2(a) shows the reconstructed training data. The true DGP is shown in figure 2(d).

Example 1 (***make blobs***). *We generate* $\boldsymbol{x} = [x_1, x_2]$ *from a mixture of 3 Gaussians with* $\boldsymbol{\mu} = [\mu_1, \mu_2, \mu_3]$ *and* $\boldsymbol{\sigma} = [\sigma_1, \sigma_2, \sigma_3] = [1, 1, 1]$ *with a fixed seed. The response* y *is generated from* $Pr(y = 1|X) = I(x_2 > 6)$*, where* $I(\cdot)$ *denotes the indicator function.*

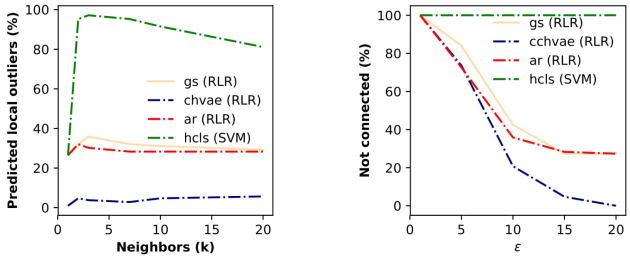
Figure 2(e) shows test data and their generated counterfactuals from AR and GS. For values from the lower right (blue) cluster in figure 2(a), both AR and GS suggest $E(\boldsymbol{x})$ that lie in the top right corner (figure 2(e)). Since both GS and AR generate almost identical values, we report the results for GS only. AR and GS favour sparse $E(\boldsymbol{x})$, meaning they only require changes along the second feature axis. However, it is apparent that the upper right corner $E(\boldsymbol{x})$ are not attainable – according to the DGP no data lives in this region. In contrast, our C-CHVAE suggests $E(\boldsymbol{x})$ that lie in regions of high data density, figure 2(f).

To gain a better understanding of our method consider figure 2(b). It shows the density of the estimated latent variable $\hat{z}$. The colours correspond to the clusters in the reconstructed data of figure 2(a). In figure 2(c), the counterfactual latent density $\tilde{z}^*$, i.e. the density of the latent variables from the counterfactuals $\tilde{x}^*$, is depicted on top of the density of $\hat{z}$. It shows that the density of $\tilde{z}^*$ is concentrated on the two modes which generate data that lies close to the decision boundary of the DGP.

6.2 Real world data sets

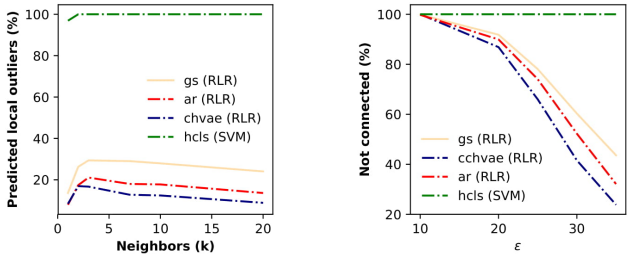
For our real world experiments we choose 2 credit data sets; a processed version of the "`Give me some credit`" data set and the *Home Equity Line of Credit* (`HELOC`) data set.[1][2] For the former, the

[1] https://www.kaggle.com/brycecf/give-me-some-credit-dataset.
[2] https://community.fico.com/s/explainable-machine-learning-challenge.



(a) Local outlier factor score (b) Connectedness score

Figure 3: Faithfulnes relative to $x \in H^+ \cap D^+$ for GMSC.



(a) Local outlier factor score (b) Connectedness score

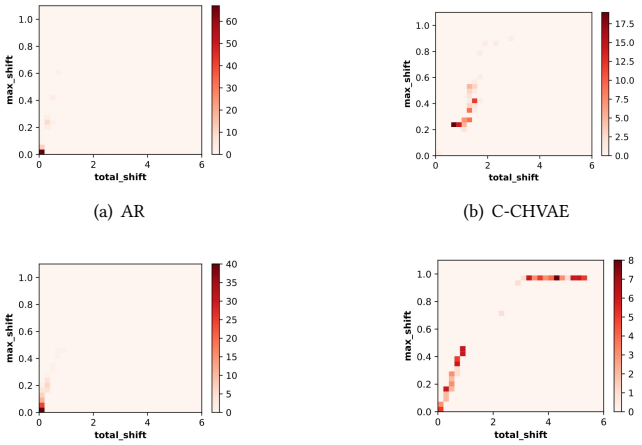
Figure 4: Faithfulness relative to $x \in H^+ \cap D^+$ for HELOC data.

target variable records whether individuals experience financial distress within a period of two years, in the latter case one uses the applicants' information from credit reports to predict whether they will repay the HELOC account within a fixed time window. Both data sets are standard in the literature [4, 17, 21] and are described in more detail in our github repository.

While GS works for different classifiers, the AR and HCLS algorithms do not. To also compare our results with AR we follow Ustun et al. [21] and choose an ℓ_2-penalized logistic regression model. For HCLS, we use SVM with a linear kernel when possible.

"Give Me Some Credit" *(GMSC).* For this data set, GS and AR produce very similar results in terms of faithfulness. HCLS performs worst and C-CHVAE (our's) outperforms all other methods. In terms of the local outlier score, the difference gets as high as 20 percentage points (figure 3(a)). With respect to the connectedness score the difference grows larger for large ϵ (figure 3(b)). In terms of *difficulty*, it the C-CHVAE's faithfully generated counterfactuals come at the cost of greater TS and MS (figure 5).

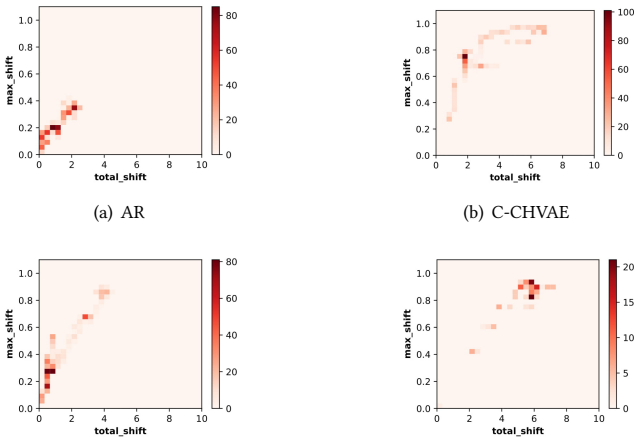
HELOC. With respect to *counterfactual faithfulness*, the C-CHVAE outperforms all other methods for both measures and all parameter choices (figure 4). Again, HCLS is not performing well; one reason could lie in the fact that one needs to specify the directions in which all free features are allowed to change. This seems to require very careful choices. Moreover, it is likely to restrict the counterfactual suggestions, leading to counterfactuals that might look less typical, which is what *faithfulness* measures. In terms of *difficulty*, the pattern is similar to the one above (see figure 6). The C-CHVAE tends



(a) AR (b) C-CHVAE

(c) GS (d) HCLS

Figure 5: Total shift vs. max. shift for $E(x)$ on GMSC data.



(a) AR (b) C-CHVAE

(c) GS (d) HCLS

Figure 6: Total shift vs. max. shift for $E(x)$ on HELOC data.

to make suggestions with higher MS. This time, to obtain *faithful counterfactuals* we are paying a price in terms of higher MS.

7 CONCLUSION AND FUTURE WORK

We have introduced a general-purpose framework for generating counterfactuals; in particular, the fact that our method works for tabular data without the specification of distance or cost functions in the input space allows practitioners and researchers to adapt this work to a wide variety of applications. To do so, several avenues for future work open up. First, all existing methods make recommendations of how features would need to be altered to receive a desired result, but none of these methods give associated input importance. And second, it would be desirable to formalize the tradeoff between the autoencoder capacity and counterfactual faithulness.

ACKNOWLEDGMENTS

We would like to thank Lars Holdijk and Michael Lohaus for insightful discussions and Alfredo Nazabal for his assistance in running the HVAE.

REFERENCES

[1] Alekh Agarwal, Alina Beygelzimer, Miroslav Dudík, John Langford, and Hanna Wallach. 2019. A reductions approach to fair classification. In *ICML*.
[2] Naveed Akhtar and Ajmal Mian. 2018. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access* 6 (2018), 14410–14430.
[3] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. 2017. Adversarial patch. *arXiv preprint arXiv:1712.09665* (2017).
[4] Rory Mc Grath, Luca Costabello, Chan Le Van, Paul Sweeney, Farbod Kamiab, Zhao Shen, and Freddy Lecue. 2018. Interpretable Credit Application Predictions With Counterfactual Explanations. *NeurIPS workshop: Challenges and Opportunities for AI in Financial Services: the Impact of Fairness, Explainability, Accuracy, and Privacy* (2018).
[5] Nina Grgic-Hlaca, Elissa M Redmiles, Krishna P Gummadi, and Adrian Weller. 2018. Human perceptions of fairness in algorithmic decision making: A case study of criminal risk prediction. In *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 903–912.
[6] Radoslav Harman and Vladimír Lacko. 2010. On decompositional algorithms for uniform sampling from n-spheres and n-balls. *Journal of Multivariate Analysis* 101, 10 (2010), 2297–2304.
[7] Oleg Ivanov, Michael Figurnov, and Dmitry Vetrov. 2018. Variational Autoencoder with Arbitrary Conditioning. *arXiv preprint arXiv:1806.02382* (2018).
[8] Shalmali Joshi, Oluwasanmi Koyejo, Warut Vijitbenjaronk, Been Kim, and Joydeep Ghosh. 2019. Towards Realistic Individual Recourse and Actionable Explanations in Black-Box Decision Making Systems. *arXiv preprint arXiv:1907.09615* (2019).
[9] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *Proceedings of the 2nd International Conference on Learning Representations (ICLR)* (2013).
[10] Michael T Lash, Qihang Lin, Nick Street, Jennifer G Robinson, and Jeffrey Ohlmann. 2017. Generalized inverse classification. In *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 162–170.
[11] Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, and Marcin Detyniecki. 2019. Issues with post-hoc counterfactual explanations: a discussion. *ICML Workshop on Human in the Loop Learning* (2019).
[12] Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. 2017. Inverse Classification for Comparison-based Interpretability in Machine Learning. *arXiv preprint arXiv:1712.08443* (2017).
[13] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. 2015. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644* (2015).
[14] Alfredo Nazabal, Pablo M Olmos, Zoubin Ghahramani, and Isabel Valera. 2018. Handling incomplete heterogeneous data using VAEs. *arXiv preprint arXiv:1807.03653* (2018).
[15] Martin Pawelczyk, Johannes Haug, Klaus Broelemann, and Gjergji Kasneci. 2019. Towards User Empowerment. *NeurIPS Workshop on Human-Centric Machine Learning* (2019).
[16] Vera Regitz-Zagrosek. 2012. Sex and gender differences in health. *EMBO reports* 13, 7 (2012), 596–603.
[17] Christopher Russell. 2019. Efficient Search for Diverse Coherent Explanations. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. ACM FAT, 20–28.
[18] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*. 3483–3491.
[19] Gabriele Tolomei, Fabrizio Silvestri, Andrew Haines, and Mounia Lalmas. 2017. Interpretable predictions of tree-based ensembles via actionable feature tweaking. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 465–474.
[20] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. 2017. Wasserstein auto-encoders. *arXiv preprint arXiv:1711.01558* (2017).
[21] Berk Ustun, Alexander Spangher, and Yang Liu. 2019. Actionable recourse in linear classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. ACM, 10–19.
[22] Sandra Wachter, Brent Mittelstadt, and Chris Russell. 2017. Counterfactual explanations without opening the black box: automated decisions and the GDPR. *Harvard Journal of Law & Technology* 31, 2 (2017), 2018.
[23] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P Gummadi. 2017. Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1171–1180.

A COUNTERFACTUAL SEARCH – ALGORITHMS

A.1 Counterfactual search algorithm

As inputs, the algorithm requires any pretrained classifier f and the trained decoder and encoder from the CHVAE. It returns the closest counterfactual. We note algorithm 1 uses a standard procedure to generate random numbers distributed uniformly over a sphere. Laugel et al. [12] use a similar algorithm, but relative to their work, we look for the smallest change in the *latent representation* $\boldsymbol{z}$ (not in input space) that would lead to a change in the predicted label. Thus, we sample observations $\tilde{\boldsymbol{z}}$ in l_p-spheres around the point $\hat{\boldsymbol{z}}$ until we find a counterfactual $\tilde{\boldsymbol{x}}^*$. For positive numbers r_1 and r_2, we define a (r_1, r_2)-sphere around $\hat{\boldsymbol{z}}$:

$$S(\hat{\boldsymbol{z}}, r_1, r_2) = \{\tilde{\boldsymbol{z}} \in \mathcal{Z} : r_1 \leq \|\hat{\boldsymbol{z}} - \tilde{\boldsymbol{z}}\| \leq r_2\}. \tag{11}$$

In order to generate uniform random numbers over a sphere, we also use the YPHL algorithm [6]. Their algorithm allows us to generate observations uniformly distributed over the unit-sphere. Next, one draws observations uniformly from $U[r_1, r_2]$, which are in turn used to rescale the distance between uniform sphere values and $\hat{\boldsymbol{z}}$. Eventually, we arrive at observations $\tilde{\boldsymbol{z}}$ that are uniformly distributed over $S(\hat{\boldsymbol{z}}, r_1, r_2)$.

Algorithm 1 shows a counterfactual search procedure when the latent variable has a dense distribution. It is straightforward to adjust the algorithm to scenarios when one desires to generate multiple counterfactual examples, also known as *flip sets* [17, 21]. The idea is that the user can choose one counterfactual from a menu of different counterfactuals, which fits her preferences best.

Algorithm 1 Stochastic Counterfactual Search For Latent Space

Input: X_{train}: training data; $x_{i,test}$: test observation; f: classifier trained on X_{train}; $m_{\hat{\theta}}$, $g_{\hat{\phi}}$: CHVAE encoder and decoder trained on X_{train}; S: number search samples; Δr: search radius.
Initialize: $f(x_{i,test}) = \hat{y}_{i,test}$; $m_{\hat{\theta}}(x_{i,test}) = \hat{z}_{i,test}$; $r = 0$; $C = \varnothing$; $\hat{z}_{train,min} = \min_i \hat{z}_{i,train}$; $\hat{z}_{train,max} = \max_i \hat{z}_{i,train}$.

while $C = \varnothing \ \wedge \tilde{z}_{i,test} \in [\hat{z}_{train,min}, \hat{z}_{train,max}]$ **do**
  for $j = 1$ **to** J **do**
    sample $\tilde{z}^j_{i,test}$ from $S(\hat{z}_{i,test}, r, \Delta r)$ in (11) {Perturbed representation}
    $\tilde{x}^j_{i,test} = g_{\hat{\phi}}(\tilde{z}^j_{i,test})$ {Potential counterfactual}
    $\tilde{y}^j_{i,test} = f(\tilde{x}^j_{i,test})$
    if $\tilde{y}^j_{i,test} \neq \hat{y}_{i,test}$ **then**
      $C \leftarrow (\tilde{z}^j_{i,test}, \tilde{x}^j_{i,test}, \tilde{y}^j_{i,test})$
    end if
  end for
  if $C = \varnothing$ **then**
    $r = r + \Delta r$ {Push search range outward}
  else if $\tilde{z}_{i,test} \notin [\hat{z}_{train,min}, \hat{z}_{train,max}]$ **then**
    Return: {No counterfactual consistent with data distribution}
    $C = \varnothing$
  else
    Return: {Find 'closest' counterfactual}
    $\tilde{z}^*_{i,test} = \text{argmin}_{\tilde{z}_{test} \in C} ||\tilde{z}_{test} - \hat{z}_{test}||$
    $\tilde{x}^*_{i,test} = g_{\hat{\phi}}(\tilde{z}^*_{i,test})$, $\tilde{y}^*_{i,test} = f(\tilde{x}^*_{i,test})$
  end if
end while

B COMMON LIKELIHOOD MODELS

For the sake of completeness we enumerate a list of commonly used likelihood models for numerical and nominal features [14]:

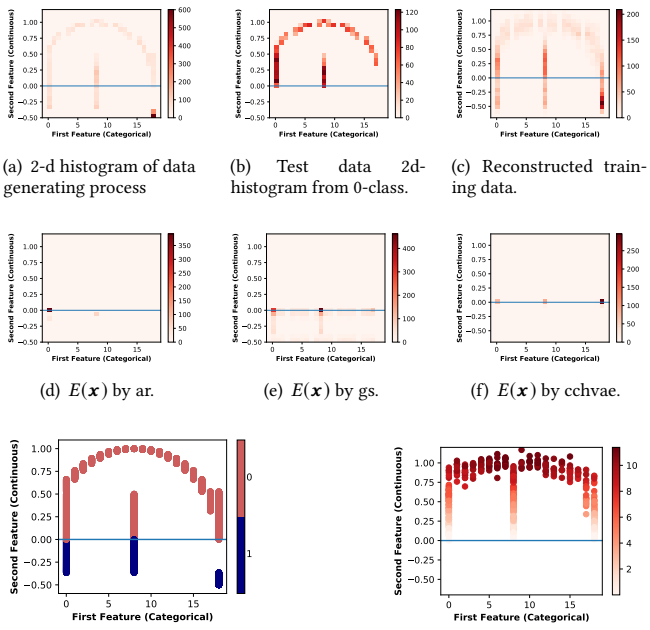
- **Real-valued data**. For real valued data, one usually assumes a Gaussian likelihood model such as,
$$p(\boldsymbol{x}_{i,d}|\boldsymbol{\gamma}_{i,d}) = \mathcal{N}(\mu_d(\boldsymbol{z}_i), \sigma^2_d(\boldsymbol{z}_i)),$$
where $\boldsymbol{\gamma}_{i,d} = \{\mu_d(\boldsymbol{z}_i), \sigma^2_d(\boldsymbol{z}_i)\}$ are modelled by the outputs of a DNN with inputs $\boldsymbol{z}$.
- **Positive real-valued data**. For positive real valued data, one can assume a log normal likelihood model such as,
$$p(\boldsymbol{x}_{i,d}|\boldsymbol{\gamma}_{i,d}) = \log \mathcal{N}(\mu_d(\boldsymbol{z}_i), \sigma^2_d(\boldsymbol{z}_i)),$$
where $\boldsymbol{\gamma}_{i,d} = \{\mu_d(\boldsymbol{z}_i), \sigma^2_d(\boldsymbol{z}_i)\}$.
- **Count data**. For count data, one can assume a Poisson likelihood model such as,
$$p(\boldsymbol{x}_{i,d}|\boldsymbol{\gamma}_{i,d}) = Poisson(\lambda_d(\boldsymbol{z}_i)),$$
where $\boldsymbol{\gamma}_{i,d} = \{\lambda_d(\boldsymbol{z}_i)\}$.
- **Ordinal data**. For ordinal valued data, we use the same procedure as in [14].
- **Categorical data.** For categorical data one can assume a multinomial logit model, where the probability of every category r is given by
$$p(\boldsymbol{x}_{i,d} = r|\boldsymbol{\gamma}_{i,d}) = \frac{\exp^{(h_{d_r}(\boldsymbol{z}_i))}}{\sum_{r=1}^{R} \exp^{(h_{d_r}(\boldsymbol{z}_i))}},$$
with parameters $\boldsymbol{\gamma}_{i,d} = \{h_{d_0}(\boldsymbol{z}_i), h_{d_1}(\boldsymbol{z}_i), ..., h_{d_{R-1}}(\boldsymbol{z}_i)\}$ and $h_{d_0}(\boldsymbol{z}_i) = 0$ to ensure identifiablity.

C SYNTHETIC EXAMPLE

EXAMPLE 2 (***DISCRETIZED MAKE MOONS***). *We generate the upper have circle $[x_1, x_2]$-pairs by cos(i) for $i \in (0, \pi)$, which we round to the closest decimal. The lower have circle $[x_1, x_2]$-pairs are then generated by 1 - sin(i) for $i \in (0, \pi)$, where we round sin(i) to the closest integer. Both the upper and the lower half contain half of the observations each and x_2 is treated as categorical with 19 categories. The response y is then generated from $Pr(y = 1|X) = I(x_1 > 0)$.*

Heterogeneous features. Figures 7(g), 7(a) and 7(b) depict the true data generating process with corresponding distribution of labels, corresponding 2d-histogram and test observations from the 0-class. Figure 7(c) depicts the 2d-histogram from the reconstructed test data. Despite the simplistic class assignment, finding attainable counterfactuals might not be trivial in this case since the data density is very fragmented.

Next, figures 7(d)-7(f) depict 25 counterfactuals generated using AR, HCLS and our C-CHVAE, respectively. For AR, counterfactuals appear in the 1st and the 9th category. For GS, counterfactuals appear in all categories. For our C-CHVAE, counterfactuals appear



(a) 2-d histogram of data generating process

(b) Test data 2d-histogram from 0-class.

(c) Reconstructed training data.

(d) $E(\boldsymbol{x})$ by ar.

(e) $E(\boldsymbol{x})$ by gs.

(f) $E(\boldsymbol{x})$ by cchvae.

(g) Data generating process.

(h) Test data cond. on Δz.

Figure 7: Example 2. Heterogeneous features. AR generates counterfactuals $E(\boldsymbol{x})$ from 1st and 9th category. GS generates counterfactuals from all categories. CCHVAE generates counterfactuals from 1st, 9th and 19th category. Figure 7(h) shows test data conditional on $\Delta z = \hat{z}^* - \hat{z}$, which indicates how much $\hat{z}$ needs to change to alter the prediction.

in the 1st, 9th and 19th category. The model correctly produces counterfactuals for all categories.

D DATA

D.1 Real world example: "Give Me Some Credit"

In the following, we list the specified pretrained classification models as well as the parameter specification used for the experiments. We use 80 percent of the data as our training set and the remaining part is used as the holdout test set. Additionally, we allow f access to all features, i.e. $f(\boldsymbol{x}^f, \boldsymbol{x}^p)$. The state of features can be found in table 3.

***AR** [21].* The AR algorithm requires to choose both an action set and free and immutable features. The implementation can be found here: https://github.com/ustunb/actionable-recourse. We specify that the *DebtRatio* feature can only move downward [21]. The AR implementation has a default decision boundary at 0 and therefore one needs to shift the boundary. We choose $p_{AR} = 0.50$, adjusting the boundary appropriately. Finally, we set the linear programming optimizer to *cbc*, which is based on an open-access `python` implementation. As f, we choose the l_2-regularized logistic regression model.

Feature	Free	Model	Dir. (HCLS)
Revolving Utilization Of Unsecured Lines	Y	log Normal	↓
Age	N	Poisson	
Number Of Times 30-59 Days Past Due Not Worse	Y	Poisson	↓
Debt Ratio	Y	log Normal	↓
Monthly Income	Y	log Normal	↑
Number Open Credit Lines And Loans	Y	Poisson	↓
Number Of Times 90 days Late	Y	Poisson	indirect
Number Real Estate Loans Or Lines	Y	Poisson	↓
Number Of Times 60-89 Days Past Due Not Worse	Y	Poisson	↓
Number Of Dependents	N	Poisson	

Table 3: "Give Me Some Credit": State of features and likelihood models.

***GS** [12].* GS is based on a version of the YPHL algorithm described above. As such we have to choose appropriate step sizes in our implementation to generate new observations from the sphere around $\boldsymbol{x}$. We choose a step size of 0.1. As f, we choose the l_2-regularized logistic regression model.

***HCLS** [10].* In our experiment we used their baseline `MATLAB` implementation, which can be found here: github.com/michael-lash/BCIC. HCLS requires us to choose a budget, which we set to 10. It also requires to choose a cost associated with changing each feature. We set it equal to 1 for all features. As f, we choose SVM with the Gaussian kernel, which delivered good results in reasonable time. Initially, we tried to choose the linear kernel, but after training for several hours with no convergence, we decided against it. We also experimented with different standardization forms (minmax standardization, z-score standardization), which did not help. For the evaluation metric, we choose accuracy and we used a *balance* option that weighs each individual sample inversely proportional to class frequencies in the training data. We had to specify an indirectly changeable feature, which we set to *NumberOfTimes90daysLate*. Finally, we had to choose the direction (Dir.(HCLS) in table 3) in which *every* free feature is allowed to move.

***C-CHVAE** (ours).* For our algorithm we made the following choices. We set the latent space dimension of both $\boldsymbol{s}$ and $\boldsymbol{z}$ to 5 and 6, respectively. For training, we used 50 epochs. Table 3 gives details about the chosen likelihood model for each feature. For count features, we use the Poisson likelihood model, while for features with a support on the positive part of the real line we choose log normal distributions. As f, we choose the l_2-regularized logistic regression model.

D.2 Real world example: HELOC

The *Home Equity Line of Credit (HELOC)* data set consists of credit applications made by homeowners in the US, which can be obtained from the FICO community.[3] The task is to use the applicant's information within the credit report to predict whether they will repay the HELOC account within 2 years. Table 4 gives an overview of the available features and the corresponding assumed likelihood models.

[3] https://community.fico.com/s/explainable-machine-learning-challenge?tabset-3158a=2.

Feature	Free	Model	Dir. (HCLS)
MSinceOldestTradeOpen	N	Poisson	
AverageMInFile	N	log Normal	
NumSatisfactoryTrades	Y	Poissonl	↑
NumTrades60Ever/DerogPubRec	Y	log Normal	↓
NumTrades90Ever/DerogPubRec	Y	log Normal	indirect
NumTotalTrades	Y	Poisson	↓
PercentInstallTrades	Y	log Normal	↑
MSinceMostRecentInqexcl7days	Y	Poisson	↓
NumInqLast6M	Y	Poisson	↓
NetFractionRevolvingBurden	Y	log Normal	↓
NumRevolvingTradesWBalance	Y	Poisson	↑
NumBank/NatlTradesWHighUtilization	Y	log Normal	↑
ExternalRiskEstimate	N	log Normal	
MPercentTradesNeverDelq	Y	log Normal	↓
MaxDelq2PublicRecLast12M	Y	Poisson	↓
MaxDelqEver	Y	Poisson	↓
NumTradesOpeninLast12M	Y	Poisson	↓
NumInqLast6Mexcl7days	Y	Poisson	↓
NetFractionRevolvingBurden	Y	Poisson	↓
NumInstallTradesWBalance	Y	Poisson	↑
NumBank2NatlTradesWHighUtilization	Y	Poisson	↓
PercentTradesWBalance	Y	log Normal	↑

Table 4: HELOC: State of features and likelihood models.

AR and GS. As before. Additionally, we do not specify how features have to move.

HCLS. As f, we choose SVM with the linear kernel. We specified *NumTrades90Ever/DerogPubRec* as the indirect feature. Again, we had to specfiy which directions features move, which we indicated in the 'Direction' column of table 4.

***C-CHAVE** (ours).* For our algorithm we made the following choices. We set the latent space dimension of both $\boldsymbol{s}$ and $\boldsymbol{z}$ to 1 and 10, respectively. For training, we used 60 epochs. Table 4 gives details about the chosen likelihood model for each feature. The rest remains as before.