DISENTANGLING ALGORITHMIC RECOURSE

Martin Pawelczyk
University of Tübingen
first.last@uni-tuebingen.de

Lea Tiyavorabun
University of Amsterdam
l.ty@posteo.net

Gjergji Kasneci
University of Tübingen
first.last@uni-tuebingen.de

ABSTRACT

The goal of algorithmic recourse is to reverse unfavorable decisions under automated decision making by suggesting actionable changes (e.g., reduce the number of credit cards). Such changes allow individuals to achieve favorable outcomes (e.g., loan approval) under low costs for the affected individual. To suggest low-cost recourse, several recourse methods have been proposed in recent literature. These techniques usually generate recourses under the assumption that the features are independently manipulable. This, however, can be misleading since the omission of feature dependencies comes with the risk that some required recourse changes are overlooked. In this work, we propose a novel, theory-driven framework, DisEntangling Algorithmic Recourse (DEAR), that suggests a solution to this problem by leveraging disentangled representations to find interpretable, small-cost recourse actions under input dependencies. Our framework addresses the independently manipulable feature (IMF) assumption by dissecting recourse actions into direct and indirect actionable changes.

1 INTRODUCTION

Counterfactual explanations provide a means for actionable model explanations at feature level. Such explanations, which have become popular among legal and technical communities, provide both an explanation and an instruction: the former emphasizes why a certain machine learning (ML) prediction was produced; the latter gives an instruction on how to act to arrive at a desirable outcome.

Several approaches in recent literature tackled the problem of providing recourses by generating counterfactual explanations (Wachter et al., 2018; Ustun et al., 2019). For instance, Wachter et al. (2018) proposed a gradient based approach which finds the nearest counterfactual resulting in the desired prediction. Pawelczyk et al. (2020) proposed a method which uses a generative model to find recourses in dense regions of the input space. More recently, Karimi et al. (2021) advocated for considering causal structure of the underlying data when generating recourses. However, these works either assume that the features can be independently manipulated (Wachter et al., 2018), narrowly focus on manifold constraints (Pawelczyk et al., 2020) or require a correct specification of the causal graph in combination with the correct structural equation models (Karimi et al., 2021). For example, for many practical use cases, such strong causal assumptions constitute the limiting factor when it comes to the deployment of these recourse methods.

On the other hand, most of the practical approaches implicitly make the *independently manipulable feature* (IMF) assumption. Therefore, critiques of counterfactual explanations and algorithmic recourse have highlighted the feature dependency issue (Barocas et al., 2020; Venkatasubramanian & Alfano, 2020): in a nutshell, *changing one feature will likely change others*. For instance, a recourse system might ask to increase the feature 'income' for a loan approval. However, there might be several ways of achieving the same desired outcome of loan approval: either one could increase 'income' through a promotion or one could find a new role in a different company. In the former case, the value of the variable reflecting 'time on job' would go up, which would likely amplify the model's output towards the desirable outcome. In the latter case, however, the model's output would

likely swing towards a loan rejection, since the short 'time on job' opposes the positive influence of the 'income' increase.

Thus, for an individual who is subjected to an automated decision system the recourse suggestion is crucial. These suggestions, however, can be misleading, especially for approaches that make the IMF assumption, since the omission of feature dependencies comes with the risk that required recourse changes are not reliably captured. Across different works, various scholars have formulated different requirements on what it means for a recourse suggestion to be reliable (Wachter et al., 2018; Karimi et al., 2020a; Keane & Smyth, 2020; Joshi et al., 2019; Pawelczyk et al., 2020; Antorán et al., 2021; Karimi et al., 2021; 2020b; Verma et al., 2020): (**R1**) Recourses should adhere to interdependencies between the input features; (**R2**) Recourses should lie in dense regions of the data distribution; (**R3**) Recourses should be attainable at low cost by the individual; (**R4**) Recourse costs should be controllable by separating important from unimportant features.

Combining requirements R1-R4 in one recourse system poses a severe challenge to making algorithmic recourse practicable in the real world. In this work, we address the critical problem of algorithmic recourse in the face of these four challenges. More specifically, we make the following key contributions: (i) we formulate the problem of algorithmic recourse in the face of data dependencies using generative models (R1, R2); (ii) we propose a new framework called `DEAR` (**D**is**E**ntangling **A**lgorithmic **R**ecourse) which generates recourses by disentangling the latent representation of covarying features from a subset of promising recourse features (R1). As a byproduct of our framework, we show that recourse actions can be divided into two types of actions: *direct* and *indirect actions*, which can be exploited to lower the cost of recourse (R4); (iii) Through theoretical analysis we show that our framework captures recourse costs by decoupling direct recourse costs from indirect costs. The indirect costs result from the *direct* actions and emerge only in the presence of feature dependencies (R1, R2). To the best of our knowledge, this work is the first to address R1-R4 using one unifying framework, which will bridge a critical gap in the recourse literature, thereby, paving the way for real-world deployment of algorithmic recourse.

2 Background

Notation Before we introduce our framework, we note $\|\cdot\|$ refers to the 2-norm of a vector, $h(f(\mathbf{x}))$ denotes the probabilistic output of the trained classifier, where $f : \mathbb{R}^d \to \mathbb{R}$ is a differentiable scoring function (e.g., logit scoring function) and $h : \mathbb{R} \to [0, 1]$ is an activation function (e.g., sigmoid) that maps scores to continuous probability outputs. We denote the set of outcomes by $y \in \{0, 1\}$, where $y = 0$ is the undesirable outcome (e.g., loan rejection) and $y = 1$ indicates the desirable outcome (e.g., loan approval). Moreover, $\hat{y} = \mathbb{I}[h(f(\mathbf{x})) > \theta] = \mathbb{I}[f(\mathbf{x}) > s]$ is the predicted class, where $\mathbb{I}[\cdot]$ denotes the indicator function and θ is a threshold rule in probability space (e.g., $\theta = 0.5$), with corresponding threshold rule s in scoring space (e.g., $s = 0$ when a sigmoid activation is used).

The Recourse Formulation Counterfactual explanation methods provide recourses by identifying which attributes to change for reversing an unfavorable model prediction. We now describe the generic formulation leveraged by several state-of-the-art recourse methods. The goal is to find a set of actionable changes in order to improve the outcomes of instances $\mathbf{x}$ which are assigned an undesirable prediction under f. Moreover, one typically defines a cost measure in input space $c : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$. Typical choices are the ℓ_1 or ℓ_2 norms. Then the recourse problem is set up as follows:

$$\boldsymbol{\delta}^* = \arg\min_{\boldsymbol{\delta}} c(\mathbf{x}, \check{\mathbf{x}}) \text{ s.t. } \check{\mathbf{x}} = \mathbf{x} + \boldsymbol{\delta},\ \check{\mathbf{x}} \in \mathcal{A}_d,\ f(\check{\mathbf{x}}) \geq s. \tag{1}$$

The objective in (1) seeks to minimize the recourse costs $c(\mathbf{x}, \check{\mathbf{x}})$ subject to the constraint that the predicted label $\hat{y}$ flips from 0 (i.e., $f(\check{\mathbf{x}}) < s$) to 1 (i.e., $f(\check{\mathbf{x}}) \geq s$), and $\mathcal{A}_d$ represents a set of constraints ensuring that only admissible changes are made to the factual input x. For example, $\mathcal{A}_d$ could specify that no changes to protected attributes such as 'sex' can be made. The assumption underlying (1) is that each feature can be independently manipulated regardless of existing feature dependencies. Under this so-called independently manipulable feature (IMF) assumption, existing popular approaches use gradient based optimization techniques (Wachter et al., 2018; Pawelczyk et al., 2022), random search (Laugel et al., 2017), or integer programming (Ustun et al., 2019; Karimi et al., 2020a; Rawal & Lakkaraju, 2020) to find recourses.

3 OUR FRAMEWORK: DEAR

Here, we present our novel recourse framework, **Dis**E**n**tangling **A**lgorithmic **R**ecourse (DEAR), which will fill the gap between the strong causal recourse assumptions and the strong IMF assumption. More specifically: 1) We introduce the generative model required to generate recourses under input dependencies (R1, R2), 2) using our model we then show that disentangled representations need to be learned to provide accurate recourse cost estimates (R3), 3) using the disentangled representations we suggest our objective function to find recourses under input dependencies and provide a closed-form expression for the recourses output by our framework (R4) and finally 4) we provide a detailed discussion on how to operationalize and optimize our objective effectively (R4).

3.1 OUR GENERATIVE MODEL AND DISENTANGLED REPRESENTATIONS

The Generative Model On a high level, our framework consists of separating the latent code of a generative model into (i) observable features $\mathbf{x}_\mathcal{S}$ – that we wish to perform direct recourse actions on – and (ii) latent space features $\mathbf{v}$ that have been trained to become disentangled of the observable features (i.e., $\mathbf{v} \perp\!\!\!\perp \mathbf{x}_\mathcal{S}$). A direct recourse action has two effects: a direct effect on the input features that have to be changed, and an indirect effect on other, dependent features. The strength of the indirect effect is then determined by a generative model (see Figure 1). To formalize this intuition, let the input $\mathbf{x}$ be produced by the following generative model:

$$\mathbf{x} = [\mathbf{x}_\mathcal{S}, \mathbf{x}_{\mathcal{S}^c}] = [g_{\mathbf{x}_\mathcal{S}}(\mathbf{v}, \mathbf{x}_\mathcal{S}), g_{\mathbf{x}_{\mathcal{S}^c}}(\mathbf{v}, \mathbf{x}_\mathcal{S})] = g(\mathbf{v}, \mathbf{x}_\mathcal{S}), \tag{2}$$

where $g : \mathbb{R}^k \to \mathbb{R}^d$, $\mathbf{v} \in \mathbb{R}^{k-|\mathcal{S}|}$ refers to the the latent code and $\mathbf{x}_\mathcal{S}$ corresponds to a subset of the input features where $S \subset \{1, \dots, d\}$, and the complement set is $S^c = \{1, \dots, d\} \setminus S$.

The Importance of Disentanglement Learning Here, we use our generative model from (2) and consider recourse cost estimates under this model. In the following Proposition, we consider this setting to obtain an intuition on how the generative model required for our framework has to be trained.

Proposition 1 (Recourse costs)**.** *Under the generative model in (2), the cost of recourse* $\|\boldsymbol{\delta}_x\|^2 = \|\mathbf{x} - \check{\mathbf{x}}\|^2$ *is given by:*

$$\|\boldsymbol{\delta}_x\|^2 \approx \underbrace{\mathbf{d}_\mathcal{S}^\top \left(\mathbf{J}_{\mathbf{x}_\mathcal{S}}^{(\mathbf{x}_\mathcal{S})^\top} \mathbf{J}_{\mathbf{x}_\mathcal{S}}^{(\mathbf{x}_\mathcal{S})}\right) \mathbf{d}_\mathcal{S}}_{\textit{Direct Costs}} + \underbrace{\mathbf{d}_\mathcal{S}^\top \left(\mathbf{J}_{\mathbf{x}_\mathcal{S}}^{(\mathbf{x}_{\mathcal{S}^c})^\top} \mathbf{J}_{\mathbf{x}_\mathcal{S}}^{(\mathbf{x}_{\mathcal{S}^c})}\right) \mathbf{d}_\mathcal{S}}_{\textit{Indirect Costs}},$$

where :

$$\mathbf{J}_{\mathbf{x}_\mathcal{S}}^{(\mathbf{x}_\mathcal{S})} = \underbrace{\frac{\partial g_{\mathbf{x}_\mathcal{S}}(\mathbf{v}, \mathbf{x}_\mathcal{S})}{\partial \mathbf{v}} \frac{\partial \mathbf{v}}{\partial \mathbf{x}_\mathcal{S}}}_{\textit{Entanglement costs}} + \underbrace{\frac{\partial g_{\mathbf{x}_\mathcal{S}}(\mathbf{v}, \mathbf{x}_\mathcal{S})}{\partial \mathbf{x}_\mathcal{S}}}_{\textit{Identity Mapping}}$$

$$\mathbf{J}_{\mathbf{x}_\mathcal{S}}^{(\mathbf{x}_{\mathcal{S}^c})} = \underbrace{\frac{\partial g_{\mathbf{x}_{\mathcal{S}^c}}(\mathbf{v}, \mathbf{x}_\mathcal{S})}{\partial \mathbf{v}} \frac{\partial \mathbf{v}}{\partial \mathbf{x}_\mathcal{S}}}_{\textit{Entanglement costs}} + \underbrace{\frac{\partial g_{\mathbf{x}_{\mathcal{S}^c}}(\mathbf{v}, \mathbf{x}_\mathcal{S})}{\partial \mathbf{x}_\mathcal{S}}}_{\textit{Elasticity of } g_{\mathbf{x}_{\mathcal{S}^c}} \textit{ w.r.t } \mathbf{x}_\mathcal{S}}}.$$

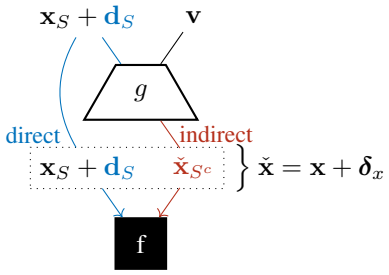


Figure 1: Finding recourse for input $\mathbf{x}$ with DEAR. For an input $\mathbf{x}$ with $f(\mathbf{x}) < s$, we encode $[\mathbf{x}_\mathcal{S}, \mathbf{v}]{=}e(\mathbf{x})$. Then, we find direct actions $\mathbf{d}_\mathcal{S}$, which generate recourse, i.e. we find a $\check{\mathbf{x}}$ such that $f(\check{\mathbf{x}}) > s$, where $\check{\mathbf{x}} = [\mathbf{d}_\mathcal{S} + \mathbf{x}_\mathcal{S}, \check{\mathbf{x}}_{\mathcal{S}^c}(\mathbf{v}, \mathbf{d}_\mathcal{S} + \mathbf{x}_\mathcal{S})]$. The direct action $\mathbf{d}_\mathcal{S}$ has two effects: 1) it changes the features in $\mathcal{S}$ directly (i.e., $\mathbf{d}_\mathcal{S} + \mathbf{x}_\mathcal{S}$), and 2) it changes the features in $\mathcal{S}^c$ indirectly (i.e., $\check{\mathbf{x}}_{\mathcal{S}^c}(\mathbf{v}, \mathbf{d}_\mathcal{S} + \mathbf{x}_\mathcal{S})$). The strength of the indirect change $\check{\mathbf{x}}_{\mathcal{S}^c}$ is determined by the elasticity of $g_{\mathbf{x}_{\mathcal{S}^c}}$ w.r.t. $\mathbf{x}_\mathcal{S}$.

The result of Proposition 1 is intuitive. It says that we can control the recourse costs using the actions $\mathbf{d}_\mathcal{S}$. We provide the proof in Appendix A.1, and make two noteworthy observations. First, the result reveals that the costs have to be partitioned into direct and indirect costs. The direct costs correspond to the costs that one would have obtained from algorithms that use the IMF assumption when searching for recourses (e.g., (Wachter et al., 2018; Pawelczyk et al., 2022; Ustun et al., 2019)). The indirect costs are due to feature dependencies of $\mathbf{x}_\mathcal{S}$ with $\mathbf{x}_{\mathcal{S}^c}$. If $\mathbf{x}_\mathcal{S}$ is independent of $\mathbf{x}_{\mathcal{S}^c}$ (i.e., the elasticity of $g_{\mathbf{x}_{\mathcal{S}^c}}$ w.r.t $\mathbf{x}_\mathcal{S}$ is $\mathbf{0}$), then a change in $\mathbf{x}_\mathcal{S}$ will not alter $\mathbf{x}_{\mathcal{S}^c}$ and the only cost remaining is the direct cost (we refer to Figure 1 for a schematic overview of the mechanism). Second, we observe that the costs can be *inflated*, if the latent space variables $\mathbf{v}$ depend on $\mathbf{x}_\mathcal{S}$. This is expressed through the *entanglement cost* terms in Proposition 1.

3.2 Our Objective

So far, we have ignored the predictive model f in our considerations. Now, we introduce the predictive model and we rewrite the recourse problem from (1) as follows:

$$\mathbf{d}_{\mathcal{S}}^{*} = \arg\min_{\mathbf{d}_{\mathcal{S}}} c(\mathbf{x}, \check{\mathbf{x}}) \text{ s.t. } \check{\mathbf{x}} = g(\mathbf{v}, \mathbf{x}_{\mathcal{S}} + \mathbf{d}_{\mathcal{S}}),\ \check{\mathbf{x}} \in \mathcal{A}_d,\ \mathbf{x}_{\mathcal{S}} \perp\!\!\!\perp \mathbf{v},\ f(\check{\mathbf{x}}) > s, \tag{3}$$

where we have used the insight that $\mathbf{x}_{\mathcal{S}} \perp\!\!\!\perp \mathbf{v}$ derived from Proposition 1. Relative to the objective from (1), our objective in (3) uses our generative model to capture input dependencies. Instead of finding recourse actions across the whole input space, we find recourse actions for the inputs in $\mathcal{S}$. We make recommendation on the choice of $\mathcal{S}$ in the next Section. Our reformulation has several advantages compared to existing recourse methods from the literature: i) relative to manifold-based recourse methods (Joshi et al., 2019; Pawelczyk et al., 2020; Antorán et al., 2021) the actions are applied to input space variables as opposed to latent space variables, and thus they are inherently interpretable; ii) relative to manifold-based recourse methods and recourse methods which use the IMF assumption (Wachter et al., 2018; Laugel et al., 2017), we can sharply separate the direct effect, which $\mathbf{d}_{\mathcal{S}}$ has on $\check{\mathbf{x}}$ via $\mathbf{x}_{\mathcal{S}}$, from its indirect effect, which $\mathbf{d}_{\mathcal{S}}$ has on $\check{\mathbf{x}}$ determined by the generative model when it is dependent of $\mathbf{x}_{\mathcal{S}^c}$ (recall Proposition 1); iii) relative to causal recourse methods (Karimi et al., 2021; 2020b), we neither assumed causal graphical models nor did we assume structural equation models to incorporate input dependencies.

3.3 A Closed-Form Expression for DEAR's Recourses

In this Section, we derive an approximate closed-form solution for the objective in (3) which uses our insights from Proposition 1 (i.e., $\mathbf{v} \perp\!\!\!\perp \mathbf{x}_{\mathcal{S}}$). We then use this solution inform the choice of the set $\mathcal{S}$ in the absence of strong prior information or preferences expressed by the end-user.

Proposition 2 (Recourses by DEAR)**.** *Suppose* $\mathbf{v} \perp\!\!\!\perp \mathbf{x}_{\mathcal{S}}$. *Then a first-order approximation* $\tilde{\mathbf{d}}_{\mathcal{S}}^{*}$ *to the optimal recourse* $\mathbf{d}_{\mathcal{S}}^{*}$ *from the objective in* (3) *using* $c = \|\mathbf{x} - \check{\mathbf{x}}\|^2$ *is given by:*

$$\boldsymbol{\delta}_x^{*} \approx \mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})} \tilde{\mathbf{d}}_{\mathcal{S}}^{*} = \frac{m}{\lambda + \|\mathbf{w}\|_2^2} \cdot \mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})} \mathbf{w}, \tag{4}$$

where $\mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})} := \frac{\partial g(\mathbf{v}, \mathbf{x}_{\mathcal{S}})}{\partial \mathbf{x}_{\mathcal{S}}}\Big|_{\mathbf{v}=\mathbf{v}, \mathbf{x}_{\mathcal{S}}=\mathbf{x}_{\mathcal{S}}}$, *$s$ is the target score in logit space,* $m = s - f(\mathbf{x})$ *is the logit gap,* $\mathbf{w} = {\mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})}}^{\top} \nabla f(\mathbf{x})$ *and* λ *is the trade-off parameter.*

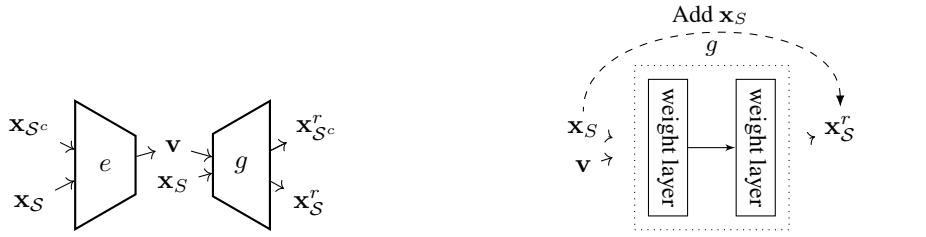
The above result is intuitive. The optimal action $\tilde{\mathbf{d}}_{\mathcal{S}}^{*}$ applied to the inputs $\mathbf{x}_{\mathcal{S}}$ is being transformed by the generator Jacobian $\mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})}$ to yield the optimal action in input space $\boldsymbol{\delta}_x^{*}$. The generator Jacobian, in turn, measures the influence that the features in $\mathcal{S}$ have on the input $\mathbf{x}$.

In the absence of strong prior information or preferences on which sets of variables $\mathcal{S}$ the actions should be performed, we suggest to use *singletons*. The insight from Proposition 2 becomes more clear, if $\mathcal{S}$ are singletons. Then, $\mathbf{w}$ from (4) becomes a scalar. Therefore, to make most progress towards the desired outcome at first order, $\mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})}$ should have a high similarity to the model gradient $\nabla f(\mathbf{x})$ in the dot-product sense. To see this, consider a first-order approximation of $f(\mathbf{x} + \boldsymbol{\delta}_x^{*}) \approx f(\mathbf{x}) + \nabla f(\mathbf{x})^{\top} \boldsymbol{\delta}_x^{*} = f(\mathbf{x}) + \frac{m \cdot w}{\lambda + \|\mathbf{w}\|_2^2} \cdot \nabla f(\mathbf{x})^{\top} \mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})}$. To push the score $f(\mathbf{x})$ towards the target score $s \geq 0$, $\mathcal{S}$ should be chosen such that the dot product $\nabla f(\mathbf{x})^{\top} \mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})}$ is high. In other words, i) the feature from $\mathcal{S}$ ideally has high discriminative power with respect to the predictive model f, and ii) the feature from $\mathcal{S}$ should influence other features from $\mathcal{S}^c$ that have high discriminative power, too. In the absence of strong prior information on which variables to perform the direct actions, it seems natural to use a local attribution method (e.g., Lundberg & Lee (2017)) to determine feature relevance. We encourage interested readers to take a look at Appendix 4 to find details on how to operationalize our method. Finally, appendix 5 contains experimental evaluations.

4 Optimizing Our Objective

Motivated by the insights from Propositions 1 and 2, we present an algorithmic procedure to compute minimal cost recourses under feature dependencies using a penalty term during autoencoder training

that encourages disentanglement of $\mathbf{x}_\mathcal{S}$ and $\mathbf{v}$ in order to keep the *entanglement costs* low. In summary, DEAR requires two steps: first, we need to obtain a latent space representation $\mathbf{v}$, which is independent of $\mathbf{x}_\mathcal{S}$. Second, we require an optimization procedure to identify the nearest counterfactual.



(a) Conditional autoencoder

(b) ResNet component within the decoder

Figure 2: DEAR's autoencoder architecture. In (a), we show our conditional autoencoder architecture, which we train subject to the Hessian penalty described in Section 4. In (b), we show how we achieve the identity mapping between $\mathbf{x}_\mathcal{S}$ and the reconstructed $\mathbf{x}^r_\mathcal{S}$ using a ResNet component (He et al., 2016a;b): we add $\mathbf{x}_\mathcal{S}$ to $\mathbf{x}^r_\mathcal{S}$ before passing the arguments to the loss $\mathcal{L}_R$ from (5).

Step 1: Training the Generative Model The main idea is to train a generative model, in which $\mathbf{x}_\mathcal{S}$ is independent of the latent variable $\mathbf{v}$, while providing high-quality reconstruction of the input $\mathbf{x}$. Thus, the training loss for the generative model consists of two components. First, it consists of both an encoder network e and decoder network g, for which the reconstruction loss,

$$\mathcal{L}_R(g, e; \mathbf{x}_\mathcal{S}) = \|g(e(\mathbf{x}), \mathbf{x}_\mathcal{S}) - \mathbf{x}\|_2^2, \tag{5}$$

guides both networks towards a good reconstruction of $\mathbf{x}$. Second, we want to drive the *entanglement costs* to 0, for which we need the decoder g to be disentangled with respect to the latent space, i.e., each component of $\mathbf{z} = [\mathbf{v}, \mathbf{x}_\mathcal{S}]$ should ideally control a single factor of variation in the output of g. To formalize this intuition, recall that $g(\mathbf{x}_\mathcal{S}, \mathbf{v}) = \mathbf{x} \in \mathbb{R}^d$, where each output $g_j = x_j$ for $1 \leq j \leq d$ has its own $|\mathbf{x}_\mathcal{S}| \times |\mathbf{v}|$ Hessian matrix $\mathbf{H}^{(j)}$. We refer to the collections of the d Hessian matrices as $\mathbf{H}$. Thus, the second loss we seek to minimize is given by:

$$\mathcal{L}_\mathbf{H}(g; \mathbf{x}_\mathcal{S}) = \sum_{j=1}^{d} \left(\sum_{k=1}^{|\mathbf{v}|} \sum_{l=1}^{|\mathbf{x}_\mathcal{S}|} H_{kl}^{(j)} \right), \tag{6}$$

which is also known as the Hessian penalty (Peebles et al., 2020). We illustrate the intuition of this objective on the j-th output x_j: we regularize the Hessian matrix $\mathbf{H}^{(j)} = \frac{\partial}{\partial \mathbf{v}} \frac{\partial g_j}{\partial \mathbf{x}_\mathcal{S}}$ and encourage its off-diagonal terms to become 0. Driving the off-diagonal terms to 0 implies that $\frac{\partial g_j}{\partial \mathbf{x}_\mathcal{S}}$ is not a function of $\mathbf{v}$ and thus $\mathbf{v}$ plays no role for the output of g_j when searching for minimum cost actions using $\mathbf{x}_\mathcal{S}$. We use the Hessian penalty from (Peebles et al., 2020) in our implementation. Finally, Proposition 1 requires the identity mapping between the latent space $\mathbf{x}_\mathcal{S}$ and the reconstructed $\mathbf{x}^r_\mathcal{S}$. We encourage our generator g to learn this mapping by using a ResNet architecture (He et al., 2016a;b) as shown in Figure 2.

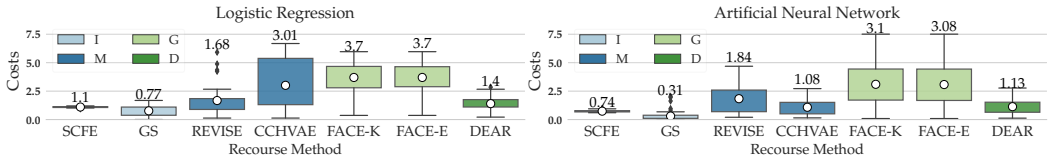
Step 2: Finding Minimal Cost Actions $\mathbf{d}_\mathcal{S}$ Given our trained generative model from step 1, we rewrite the problem in (3) using a Lagrangian with trade-off parameter λ. For a given encoded input instance $e(\mathbf{x}) = [\mathbf{v}, \mathbf{x}_\mathcal{S}]$, our objective function reads:

$$\mathbf{d}^*_\mathcal{S} = \underset{\mathbf{d}_\mathcal{S},\, \check{\mathbf{x}} \in \mathcal{A}_d}{\arg\min}\ \mathcal{L} = \underset{\mathbf{d}_\mathcal{S},\, \check{\mathbf{x}} \in \mathcal{A}_d}{\arg\min}\ \ell\big(f(\check{\mathbf{x}}(\mathbf{d}_\mathcal{S})), s\big) + \lambda \|\mathbf{x} - \check{\mathbf{x}}(\mathbf{d}_\mathcal{S})\|_1, \tag{7}$$

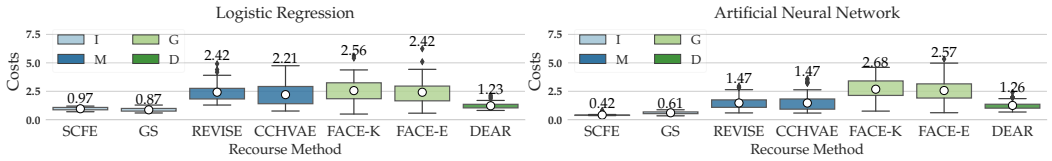
where $\check{\mathbf{x}}(\mathbf{d}_\mathcal{S}) = g(\mathbf{v}, \mathbf{x}_\mathcal{S} + \mathbf{d}_\mathcal{S})$ is a potential counterfactual in input space, $\ell(\cdot, \cdot)$ denotes the MSE loss, and $s \geq 0$ is the target score in logit space. The term on the right side encourages the counterfactual $g(\mathbf{v}, \mathbf{x}_\mathcal{S} + \mathbf{d}_\mathcal{S}) = \check{\mathbf{x}}$ to be close to the given input point $\mathbf{x}$, while the left hand side encourages the predictions to be pushed from the factual output $f(\mathbf{x})$ towards s. We do gradient descent iteratively on the loss function in (7) until the class label changes from $y = 0$ to $y = 1$. Algorithm 1 summarizes our optimization procedure. Below we further discuss how other constraints can be included into our objective.

Algorithm 1 DEAR

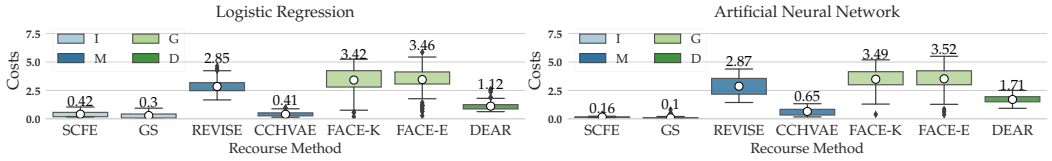
Input: f, $\mathbf{x}$ s.t. $f(\mathbf{x})<0$, g, e, $\lambda > 0$, Learning rate: $\alpha > 0$, $s \geq 0$, $\mathcal{S}$
Initialize: $\mathbf{d}_{\mathcal{S}} = \mathbf{0}$, $e(\mathbf{x}) = [\mathbf{v}, \mathbf{x}_{\mathcal{S}}]$, $\check{\mathbf{x}} = g(\mathbf{v}, \mathbf{x}_{\mathcal{S}} + \mathbf{d}_{\mathcal{S}})$
while $f(\check{\mathbf{x}}) < s$ **do**
  $\mathbf{d}_{\mathcal{S}} = \mathbf{d}_{\mathcal{S}} - \alpha \cdot \nabla_{\mathbf{d}_{\mathcal{S}}} \mathcal{L}(\check{\mathbf{x}}(\mathbf{d}_{\mathcal{S}}); f, s, \lambda)$ {Optimize (7)}
  $\check{\mathbf{x}} = g(\mathbf{v}, \mathbf{x}_{\mathcal{S}} + \mathbf{d}_{\mathcal{S}})$
end while {Class changed, i.e., $f(\check{\mathbf{x}}) \geq s$}
Return: $\check{\mathbf{x}}^* = \check{\mathbf{x}}$



(a) Adult



(b) COMPAS



(c) Give Me Credit

Figure 3: Measuring the cost of algorithmic recourse across different recourse methods using the ℓ_1 cost. We use boxplots to show the distribution of recourse costs across all individuals from the test set who require algorithmic recourse. The numbers above the maximum values correspond to the white dots (= median recourse costs). The color of the boxplots represent the type of the recourse method: The methods with 'I' use the IMF assumption, the methods with 'M' purely focus on the data manifold constraints, and methods with 'G' use a graphical model to generate recourses through dense paths. 'D' refers to our method which takes both manifold and input dependencies into account. Section 5.2 provides a detailed discussion of the results.

5 Experimental Results

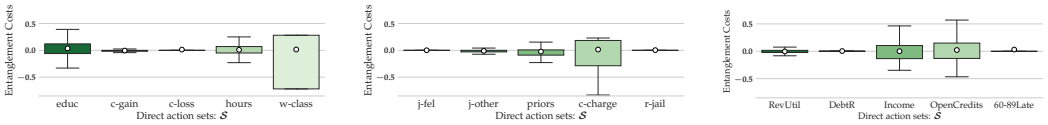
Encoding Monotonicity Constraints In the presence of strong prior knowledge on how certain features are allowed to change (e.g., 'years of schooling' (yos) or 'age' can only go up) one can add Hinge-losses (Mahajan et al., 2019) to encourage monotonicity constraints. Let x_{yos} correspond to the schooling feature. Then we can add $-\min(0, \check{x}_{yos} - x_{yos})$ to the loss function in (7) to ensure that the counterfactual $\check{x}_{yos}$ should increase, where $\check{x}_{yos}$ is the corresponding entry from $g(\mathbf{v}, \mathbf{x}_S + \mathbf{d}_{\mathcal{S}})$.

Handling Categorical Variables Using DEAR, one can easily handle (high-cardinality) categorical features. We can turn all categorical features into numeric features by standard one-hot encoding. For each categorical feature, we can then use a softmax-layer after the final output layer of the decoder. For the purpose of the one-hot-encoded reconstruction, we apply the argmax.

In this Section, we conduct extensive quantitative and qualitative evaluations to analyze DEAR's performance using our conceptual insights from the previous Section. Quantitatively, we conduct a *baseline comparison* contrasting our framework DEAR with state-of-the-art recourse methods

		Adult						COMPAS						GMC					
		LR			ANN			LR			ANN			LR			ANN		
	Method	SR (↑)	CV (↓)	YNN (↑)	SR (↑)	CV (↓)	YNN (↑)	SR (↑)	CV (↓)	YNN (↑)	SR (↑)	CV (↓)	YNN (↑)	SR (↑)	CV (↓)	YNN (↑)	SR (↑)	CV (↓)	YNN (↑)
I	GS	**1.00**	**0.00**	0.55	**1.00**	**0.00**	0.34	**1.00**	**0.00**	0.93	**1.00**	**0.00**	0.99	**1.00**	NA	0.19	**1.00**	NA	0.64
	SCFE	0.73	**0.00**	0.81	0.84	**0.00**	0.67	0.08	**0.00**	0.91	0.92	**0.00**	0.97	**1.00**	NA	0.23	0.98	NA	0.65
M	REVISE	0.35	**0.00**	0.37	**1.00**	0.12	**0.72**	0.63	0.11	**1.00**	0.99	0.12	0.98	0.99	NA	**1.00**	**1.00**	NA	0.95
	CCHVAE	0.54	0.17	0.53	**1.00**	0.07	0.61	**1.00**	0.32	**1.00**	**1.00**	0.17	0.96	**1.00**	NA	0.24	**1.00**	NA	0.80
G	FACE-K	0.99	0.36	0.71	**1.00**	0.29	0.57	0.99	0.38	**1.00**	0.60	0.40	**1.00**	**1.00**	NA	0.95	**1.00**	NA	0.96
	FACE-E	0.74	0.38	0.70	0.99	0.30	0.58	0.99	0.41	**1.00**	0.39	0.40	**1.00**	**1.00**	NA	0.94	**1.00**	NA	**0.97**
D	DEAR	**1.00**	**0.00**	**0.84**	**1.00**	**0.00**	0.70	**1.00**	**0.00**	**1.00**	**1.00**	0.01	**1.00**	**1.00**	NA	0.91	**1.00**	NA	0.94

Table 1: Measuring the reliability of algorithmic recourse for the ANN and LR models on all data sets. The *success rate* (SR), *constraint violation* (CV) and *y-nearest neighbors* (YNN) measures are described in Section 5. For GMC, there were no immutable features and therefore we are reporting NA. Our method (i.e., DEAR) usually performs on par or better relative to other recourse methods.



(a) Adult (b) COMPAS (c) GMC

Figure 4: Evaluating DEAR's entanglement costs on all data sets. At the end of autoencoder training, we compute the Hessians' off-diagonal elements of the decoder and average them (see Section 5.1 for more details). We then plot these averaged values using boxplots across test instances. The feature names indicate the sets $\mathcal{S}$, that we perform direct recourse actions on. The white dots indicate the median entanglement costs, and the box indicates the interquartile range. In line with Proposition 1, the costs are pushed to 0.

(Pawelczyk et al., 2021) using common evaluation measures from the recourse literature such as recourse costs and reliability measures. Qualitatively, we consider three aspects: (i) the *entanglement costs*, (ii) the structure of the *cost splits* (i.e., direct vs. indirect costs) and (iii) a *case study* to showcase the advantages of our new framework.

5.1 Details on Experiments

Real–world Data Our first data set is the *Adult* data set taken from the UCI repository. This data set consists of approximately $48K$ samples with demographic (e.g., race, sex), education and employment (e.g., degree, occupation, hours-per-week), personal (e.g., marital status, relationship), financial (capital gain/loss) features where the label predicts whether an individual's income exceeds $50K\$$ per year ($y = 1$). Our second data set, *COMPAS* (Correctional Offender Management Profiling for Alternative Sanctions) consists of defendants' criminal history, jail and prison time, demographics and the goal is to predict recidivism risk for defendants from Broward County, Florida. Our third data set is the *Give Me Credit* data set from 2011 Kaggle competition. It is is a credit scoring data set, consisting of 150,000 observations and 11 features. The classification task consists of deciding whether an instance will experience financial distress within the next two years (*SeriousDlqin2yrs* is 0).

Prediction Models For all our experiments, we obtain counterfactual explanations for two classification models, for which we provide additional details in Appendix B:

LR: Logistic regression: This is a binary classifier that was trained without regularization.

ANN: Artificial Neural Network: This is binary classifier with a two-layer network that was trained with ReLU activation functions.

Recourse Methods For all data sets, recourses are generated in order to flip the prediction label from the unfavorable class ($y = 0$) to the favorable class ($y = 1$). We partition the data set into 80-20 train-test splits, and do the model training and testing on these splits. We used the recourse implementations from the open-source library CARLA (Pawelczyk et al., 2021) to generate recourses. The library includes state-of-the-art recourse methods. We use the following six methods as our baselines for comparison:

I: `GS` (Laugel et al., 2017): Under the *IMF assumption*, this is a classifier-agnostic method that generates recourses by conducting random search in input space.

I: `SCFE` (Wachter et al., 2018): Under the *IMF assumption*, this method generates recourses by solving a variant of (1) using iterative gradient descent.

M: `REVISE` (Joshi et al., 2019): To find recourses that lie on the *data manifold*, this method utilizes a trained autoencoder to transform the input space into a latent embedding space. `REVISE` then uses gradient descent in latent space to find recourses that lie on the data manifold.

M: `CCHVAE` (Pawelczyk et al., 2020): This is a classifier-agnostic method to find recourses that lie on the *data manifold*. `CCHVAE` also uses a trained autoencoder to transform the input space into a latent embedding space. The latent representation is then randomly perturbed to find recourses.

G: `FACE-K` & `FACE-E` (Poyiadzi et al., 2020): This is classifier-agnostic method that finds recourses that lie on paths along dense regions. These methods construct *neighbourhood graphs* to find paths through dense regions. The graph is either an ϵ-graph (FACE-E) or a k-nearest neighbour graph (FACE-K).

Note that 'I' abbreviates methods which use the IMF assumption, 'M' abbreviates methods which generate recourses that lie on the data manifold, and 'G' abbreviates methods, which use a graphical model to generate recourses that lead through dense paths. 'D' refers to our method (i.e., `DEAR`), which takes input dependencies into account. To allow for a fair comparison across the explanation models, which use autoencoders, we use similar base architectures for `DEAR`. Appendix B provides implementation details for the recourse methods and of all used autoencoder models. We compute all evaluation measures by using the min-max normalized inputs used for training the classification and generative models. Below we describe the evaluation measures.

Recourse Costs Since we are interested in generating small cost recourses, we define a notion of distance from the counterfactual explanation to the input point. As all methods under consideration minimize the ℓ_1 norm, we use this measure and compare the ℓ_1-costs across the methods.

Entanglement Costs For a fixed set $\mathcal{S}$, for every instance at the end of training, we obtain d Hessian matrices $\mathbf{H}^{(j)} = \frac{\partial^2 g}{\partial \mathbf{v} \partial x_S}$ for $1 \leq j \leq d$. We then average the Hessian off-diagonal elements across all j and plot their distribution across all training instances. We can only do this for our recourse method `DEAR`.

Constraint Violations (CV) We set the protected attributes 'sex' and 'race' to be immutable for the Adult and COMPAS data sets, and count how often each of the explanation models suggests changes to these protected features. GMC has no protected attribute.

Label Neighborhood (YNN) We also use a measure that evaluates how much data support recourses have from positively classified instances (Pawelczyk et al., 2021). Ideally, recourses should be close to correct positively classified individuals, which is a desideratum formulated by the authors of (Laugel et al., 2019). Values of YNN close to 1 imply that the neighbourhoods around the recourses consists of points with the same predicted label, indicating that the neighborhoods around these points have already been reached by positively classified instances.

Success Rate (SR) Some generated recoures do not alter the predicted label of the instance as anticipated. We keep track of how often the suggested counterfactual yield successful recourse by counting the fraction of the respective methods' correctly determined counterfactuals.

5.2 Evaluation

Recourse Costs The baseline comparisons regarding the cost of recourse are shown in Figure 3. Relative to methods that use the IMF assumption (i.e., `GS`, `SCFE`), `DEAR`'s median costs are higher. This is to be expected since `SCFE` and `GS` *ignore feature dependencies*, essentially providing a lower bound for `DEAR`'s recourse costs. Relative to manifold-based recourse methods (i.e, `REVISE` and

(a) Logistic Regression

(b) Artificial Neural Network

Figure 5: Cost splits as suggested by Proposition 1 on both classifiers across all data sets. The direct costs corresponds to the direct action $\mathbf{d}_S$ and are measured as $\|\mathbf{d}_S\|_1$. The indirect costs are measured as $\|\mathbf{x}_{S^c} - \check{\mathbf{x}}_{S^c}(\mathbf{v}, \mathbf{d}_S + \mathbf{x}_S)\|_1$.

CCHVAE), DEAR usually performs more favourably ensuring up to 50 percent less costly median recourse costs. This is due to the fact that DEAR can use the most discriminative features in input space – as opposed to latent space – to search for recourses. Relative to the graph-based methods (i.e., FACE-K and FACE-E) our method performs significantly better. Since FACE has to ensure connected paths, their costs are usually the highest.

Reliability of Recourse We measure the reliability of recourse using SR, CV and YNN presented in the previous Section. The results across all methods, data sets and classifiers are shown in Table 1. We see that DEAR has the highest SRs across all data sets and classifiers, among the highest YNN scores, and one of the lowest constraint violation rates. Compared to the manifold-based recourse methods, DEAR's success rate is up to 45 percentage points higher. This is due to the fact that the lower dimensional data manifold can end before the decision boundary is reached and thus the manifold-based methods, which search for recourse in latent space, sometimes get stuck before they find a counterfactual instance (see (Downs et al., 2020) for a detailed analysis of this phenomenon). Antorán et al. (2021, Appendix) report a similar finding. A similar reason probably prevents FACE from reaching high SRs. Our model does not suffer from this shortcoming since it primarily uses the most discriminative features in input space (Proposition 2) to search for recourses, resulting in SRs of 1.

Qualitative Analysis Finally, we analyze our recourse model qualitatively. We start by analyzing the *entanglement costs*. As required by Proposition 1, we require these costs to be pushed to 0. We plot the distribution of the averaged off-diagonal terms in Figure 4. The results show that the entanglement cost is consistently pushed to 0 (most medians are at 0). The only exception occurs for the feature 'workclass' for which our autoencoder network performs less well at concentrating the costs at 0 (right bar in the left panel). In summary, however, our mechanism is very well aligned with Proposition 1's requirement of disentangled $\mathbf{v}$ and $\mathbf{x}_S$.

6 Conclusion

In this work, we considered the problem of generating algorithmic recourse in the presence of feature dependencies – a previously overlooked problem. We developed DEAR (**D**is**E**ntangling **A**lgorithmic **R**ecourse), a novel recourse method that generates recourses by disentangling the latent representation of co-varying features from a subset of promising recourse features to capture some of the main practical desiderata: recourses should adhere to (i) feature dependencies and (ii) data manifold constraints, while providing (iii) low recourse costs.

REFERENCES

Javier Antorán, Umang Bhatt, Tameem Adel, Adrian Weller, and José Miguel Hernández-Lobato. Getting a clue: A method for explaining uncertainty estimates. *International Conference on Learning Representations (ICLR)*, 2021.

Solon Barocas, Andrew D. Selbst, and Manish Raghavan. The hidden assumptions behind counterfactual explanations and principal reasons. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT*)*, 2020.

Michael Downs, Jonathan L. Chu, Yaniv Yacoby, Finale Doshi-Velez, and Weiwei Pan. Cruds: Counterfactual recourse using disentangled subspaces. *ICML Workshop on Human Interpretability in Machine Learning (WHI 2020)*, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision (ECCV)*, 2016b.

Shalmali Joshi, Oluwasanmi Koyejo, Warut Vijitbenjaronk, Been Kim, and Joydeep Ghosh. Towards realistic individual recourse and actionable explanations in black-box decision making systems. *arXiv preprint arXiv:1907.09615*, 2019.

Amir-Hossein Karimi, Gilles Barthe, Borja Balle, and Isabel Valera. Model-agnostic counterfactual explanations for consequential decisions. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, 2020a.

Amir-Hossein Karimi, Julius von Kügelgen, Bernhard Schölkopf, and Isabel Valera. Algorithmic recourse under imperfect causal knowledge: a probabilistic approach. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020b.

Amir-Hossein Karimi, Bernhard Schölkopf, and Isabel Valera. Algorithmic recourse: from counterfactual explanations to interventions. In *Proceedings Conference on Fairness, Accountability, and Transparency (FAccT)*, 2021.

Mark T Keane and Barry Smyth. Good counterfactuals and where to find them: A case-based technique for generating counterfactuals for explainable ai (xai). In *International Conference on Case-Based Reasoning*, 2020.

Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. Inverse classification for comparison-based interpretability in machine learning. *arXiv preprint arXiv:1712.08443*, 2017.

Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. The dangers of post-hoc interpretability: Unjustified counterfactual explanations. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Divyat Mahajan, Chenhao Tan, and Amit Sharma. Preserving causal constraints in counterfactual explanations for machine learning classifiers. *arXiv preprint arXiv:1912.03277*, 2019.

Martin Pawelczyk, Klaus Broelemann, and Gjergji Kasneci. Learning model-agnostic counterfactual explanations for tabular data. In *Proceedings of The Web Conference 2020 (WWW)*, New York, NY, USA, 2020. ACM.

Martin Pawelczyk, Sascha Bielawski, Johannes van den Heuvel, Tobias Richter, and Gjergji Kasneci. Carla: A python library to benchmark algorithmic recourse and counterfactual explanation algorithms. In *Advances in Neural Information Processing Systems 34 (NeurIPS) Datasets and Benchmark Track*, 2021.

Martin Pawelczyk, Chirag Agarwal, Shalmali Joshi, Sohini Upadhyay, and Himabindu Lakkaraju. Exploring counterfactual explanations through the lens of adversarial examples: A theoretical and empirical analysis. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2022.

William Peebles, John Peebles, Jun-Yan Zhu, Alexei A. Efros, and Antonio Torralba. The hessian penalty: A weak prior for unsupervised disentanglement. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2020.

Rafael Poyiadzi, Kacper Sokol, Raul Santos-Rodriguez, Tijl De Bie, and Peter Flach. Face: Feasible and actionable counterfactual explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society (AIES)*. ACM, 2020.

Kaivalya Rawal and Himabindu Lakkaraju. Beyond individualized recourse: Interpretable and interactive summaries of actionable recourses. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020.

Berk Ustun, Alexander Spangher, and Yang Liu. Actionable recourse in linear classification. In *Proceedings of the 2019 Conference on Fairness, Accountability, and Transparency (FAT*)*. ACM, 2019.

Suresh Venkatasubramanian and Mark Alfano. The philosophical basis of algorithmic recourse. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT*)*, 2020.

Sahil Verma, John Dickerson, and Keegan Hines. Counterfactual explanations for machine learning: A review. *arXiv preprint arXiv:2010.10596*, 2020.

Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: automated decisions and the gdpr. *Harvard Journal of Law & Technology*, 31(2), 2018.

A Theoretical Analysis

A.1 Proof of Proposition 1

Proof (Recourse Costs by DEAR) First, we note that $\mathbf{v}$ is usually obtained via some kind of training procedure, and thus it could be a function of $\mathbf{x}_S$. Next, we partition both $\mathbf{z} = [\mathbf{v} \;\; \mathbf{x}_S]^\top$ and $\mathbf{d}_z = [\mathbf{d}_v \;\; \mathbf{d}_S]^\top$. Moreover, we partition $g(\mathbf{v}, \mathbf{x}_S) = [g_{\mathbf{x}_S}(\mathbf{v}, \mathbf{x}_S) \;\; g_{\mathbf{x}_{S^c}}(\mathbf{v}, \mathbf{x}_S)]^\top = [\mathbf{x}_S \; \mathbf{x}_{S^c}]^\top$. Then, the matrix of derivatives can be partitioned as follows:

$$\mathbf{J}_z^{(\mathbf{x})} = \begin{bmatrix} \mathbf{J}_{\mathbf{v}}^{(\mathbf{x}_{S^c})} & \mathbf{J}_{x_S}^{(\mathbf{x}_{S^c})} \\ \mathbf{J}_{\mathbf{v}}^{(\mathbf{x}_S)} & \mathbf{J}_{\mathbf{x}_S}^{(\mathbf{x}_S)} \end{bmatrix} := \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}.$$

Since we are not interested in applied changes to $\mathbf{v}$ we set $\mathbf{d}_v{:=}\mathbf{0}$. By Lemma 1 we know $\|\mathbf{d}_x\|^2{=}\mathbf{d}_z^\top \big({\mathbf{J}_z^{(\mathbf{x})}}^\top \mathbf{J}_{\mathbf{z}}^{(\mathbf{x})}\big)\mathbf{d}_z$. A direct computation with $\mathbf{d}_z{:=}[\mathbf{0} \quad \mathbf{d}_S]^\top$ yields:

$$\begin{aligned} \|\mathbf{d}_x\|^2 &\approx [\mathbf{0} \quad \mathbf{d}_S] \begin{bmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{B} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ \mathbf{d}_S \end{bmatrix} \\ &= \mathbf{d}_S^\top \mathbf{B}^\top \mathbf{B} \mathbf{d}_S + \mathbf{d}_S^\top \mathbf{D}^\top \mathbf{D} \mathbf{d}_S \\ &= \underbrace{\mathbf{d}_S^\top \big({\mathbf{J}_{\mathbf{x}_S}^{(\mathbf{x}_{S^c})}}^\top \mathbf{J}_{\mathbf{x}_S}^{(\mathbf{x}_{S^c})}\big)\mathbf{d}_S}_{\text{Indirect Costs}} + \underbrace{\mathbf{d}_S^\top \big({\mathbf{J}_{\mathbf{x}_S}^{(\mathbf{x}_S)}}^\top \mathbf{J}_{\mathbf{x}_S}^{(\mathbf{x}_S)}\big)\mathbf{d}_S}_{\text{Direct Costs}}. \end{aligned}$$

By the chain rule of multivariate calculus (recall that $\mathbf{v}$ and $\mathbf{x}_S$ need not be independent), note that we can write out the above terms as follows:

$$\begin{aligned}
\mathbf{J}_{\mathbf{x}_S}^{(\mathbf{x}_{S^c})} &= \frac{\partial g_{\mathbf{x}_{S^c}}(\mathbf{v}, \mathbf{x}_S)}{\partial \mathbf{v}} \frac{\partial \mathbf{v}}{\partial \mathbf{x}_S} + \frac{\partial g_{\mathbf{x}_{S^c}}(\mathbf{v}, \mathbf{x}_S)}{\partial \mathbf{x}_S} \frac{\partial \mathbf{x}_S}{\partial \mathbf{x}_S} \\
&= \underbrace{\frac{\partial g_{\mathbf{x}_{S^c}}(\mathbf{v}, \mathbf{x}_S)}{\partial \mathbf{v}} \frac{\partial \mathbf{v}}{\partial \mathbf{x}_S}}_{\text{Entanglement costs}} + \underbrace{\frac{\partial g_{\mathbf{x}_{S^c}}(\mathbf{v}, \mathbf{x}_S)}{\partial \mathbf{x}_S}}_{\text{Elasticity of } g(\mathbf{x}_{S^c}) \text{ w.r.t to } \mathbf{x}_S} \\
\mathbf{J}_{\mathbf{x}_S}^{(\mathbf{x}_S)} &= \frac{\partial g_{\mathbf{x}_S}(\mathbf{v}, \mathbf{x}_S)}{\partial \mathbf{v}} \frac{\partial \mathbf{v}}{\partial \mathbf{x}_S} + \frac{\partial g_{\mathbf{x}_S}(\mathbf{v}, \mathbf{x}_S)}{\partial \mathbf{x}_S} \frac{\partial \mathbf{x}_S}{\partial \mathbf{x}_S} \\
&= \underbrace{\frac{\partial g_{\mathbf{x}_S}(\mathbf{v}, \mathbf{x}_S)}{\partial \mathbf{v}} \frac{\partial \mathbf{v}}{\partial \mathbf{x}_S}}_{\text{Entanglement costs}} + \underbrace{\frac{\partial g_{\mathbf{x}_S}(\mathbf{v}, \mathbf{x}_S)}{\partial \mathbf{x}_S}}_{\text{Identity Mapping}}.
\end{aligned}$$

□

Let us consider what this implies intuitively. For the direct costs, notice that g would achieve the best reconstruction of $\mathbf{x}_S$ by using the identify mapping. Recall, in Section 4, we suggested to use a `ResNet` component within the decoder to enforce this identity mapping during training of our autoencoder model. Hence, under perfect disentanglement the disentanglement costs are 0, and the $\mathbf{J}_{\mathbf{x}_S}^{(\mathbf{x}_S)}$=$\mathbf{1}$: thus, the direct cost would ideally be given by $\mathbf{d}_S^\top \mathbf{d}_S$. This is the squared ℓ_2 norm of $\mathbf{d}_S$.

The indirect costs, on the other hand, depend on the sensitivity of $\mathbf{x}_{S^c}$ with respect to $\mathbf{x}_S$, that is, $\mathbf{J}_{\mathbf{x}_S}^{(\mathbf{x}_{S^c})}$. Again, we consider the case of perfect disentanglement first: Suppose $\mathbf{x}_S$ was a variable that was unrelated to the remaining variables $\mathbf{x}_{S^c}$, while still being predictive of the outcome: Then $\mathbf{J}_{x_S}^{(x_{S^c})} = \mathbf{0}$, and a change $\mathbf{d}_S$ would only have a direct impact on the outcome, and thus the indirect costs would disappear. In this case, the recourse cost for independence–based and dependence–based methods coincide. On the other extreme, suppose $\mathbf{x}_S$ was almost a copy of $\mathbf{x}_{S^c}$, then $\mathbf{J}_{\mathbf{x}_S}^{(\mathbf{x}_{S^c})} \approx \mathbf{1}$, and changing $\mathbf{x}_S$ clearly impacts the remaining variables $\mathbf{x}_{S^c}$. In this case, an independence–based method would not reliably capture the recourse costs.

A.2 Proof of Proposition 2

Proof of Proposition 2. Recall the problem in (3):

$$\mathbf{d}_{\mathcal{S}}^* = \arg\min_{\mathbf{d}_{\mathcal{S}}} \mathcal{L} = \arg\min_{\mathbf{d}_{\mathcal{S}}} \lambda \|\mathbf{d}_{\mathcal{S}}\|^2 + \|s - f\big(g(\mathbf{v}, \mathbf{x}_{\mathcal{S}} + \mathbf{d}_{\mathcal{S}})\big)\|^2. \tag{8}$$

We use the following first-order approximation to $f\big(g(\mathbf{v}, \mathbf{x}_{\mathcal{S}} + \mathbf{d}_{\mathcal{S}})\big) \approx f(\mathbf{x}) + \nabla f(\mathbf{x})^\top \mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})} \mathbf{d}_{\mathcal{S}}$, where we have substituted $g(\mathbf{v}, \mathbf{x}_{\mathcal{S}}) = \mathbf{x}$ and used that $\mathbf{v} \perp\!\!\!\perp \mathbf{x}_{\mathcal{S}}$ by design of the generative model. Then, we can derive a surrogate loss to the loss from (8):

$$\mathcal{L} \approx \tilde{\mathcal{L}} = \lambda \cdot \mathbf{d}_{\mathcal{S}}^\top \mathbf{d}_{\mathcal{S}} + \big(m - \nabla f(\mathbf{x})^\top \mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})} \mathbf{d}_{\mathcal{S}}\big)^\top \big(m - \nabla f(\mathbf{x})^\top \mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})} \mathbf{d}_{\mathcal{S}}\big), \tag{9}$$

where $m = s - f(\mathbf{x})$. The second term on the right in (9) can be written as:

$$m^2 - 2m \nabla f(\mathbf{x})^\top \mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})} \mathbf{d}_{\mathcal{S}} + \mathbf{d}_{\mathcal{S}}^\top {\mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})}}^\top \nabla f(\mathbf{x}) \nabla f(\mathbf{x})^\top \mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})} \mathbf{d}_{\mathcal{S}}.$$

By solving $\arg\min_{\mathbf{d}_{\mathcal{S}}} \tilde{\mathcal{L}}$ we find the optimal change required in the features $\mathbf{x}_{\mathcal{S}}$ as follows:

$$\tilde{\mathbf{d}}_{\mathcal{S}}^* = \boldsymbol{M}^{-1} \boldsymbol{\eta}, \tag{10}$$

where

$$\boldsymbol{M} = \lambda \cdot \mathbf{I} + {\mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})}}^\top \nabla f(\mathbf{x}) \nabla f(\mathbf{x})^\top \mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})} \qquad \boldsymbol{\eta} = m \cdot \nabla f(\mathbf{x})^\top \mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})}. \tag{11}$$

Next, we define $\mathbf{w} = {\mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})}}^\top \nabla f(\mathbf{x})$. Note that $\mathbf{w}\mathbf{w}^\top$ is a rank-1 matrix. Thus, by the well-known Sherman-Morrison-Woodbury formula, $\mathbf{M}$ can be inverted as follows:

$$\mathbf{M}^{-1} = \frac{1}{\lambda}\left(\mathbf{I} - \frac{\mathbf{w}\mathbf{w}^\top}{\lambda + \|\mathbf{w}\|_2^2}\right). \tag{12}$$

As a consequence, after substituting (12) into (11) we obtain that:

$$\tilde{\mathbf{d}}_{\mathcal{S}}^{*} = \frac{m}{\lambda + \|\mathbf{w}\|_2^2}\mathbf{w}. \tag{13}$$

Further, note that $\boldsymbol{\delta} = g(\mathbf{z} + \mathbf{d}_{\mathcal{S}}) - g(\mathbf{z}) \approx \mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})}\mathbf{d}_{\mathcal{S}}$, where we have used that $\mathbf{v} \perp\!\!\!\perp \mathbf{x}_{\mathcal{S}}$. Therefore, we obtain a first-order approximation to the optimal recourse in input space:

$$\boldsymbol{\delta}_x^{*} \approx \mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})}\tilde{\mathbf{d}}_{\mathcal{S}}^{*} = \frac{m}{\lambda + \|\mathbf{w}\|_2^2} \cdot \mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})}\mathbf{w}, \tag{14}$$

as claimed. □

A.3 Proof of Lemma 1

Lemma 1 (Recourse costs in terms latent space quantities)**.** *Given a latent representation* $\mathbf{z}$ *of a sample* $\mathbf{x} = g(\mathbf{z})$ *and a generated counterfactual* $\check{\mathbf{x}} = g(\check{\mathbf{z}})$ *with* $\check{\mathbf{z}} = \mathbf{z} + \mathbf{d}_z$*, the cost of recourse* $\|\mathbf{x} - \check{\mathbf{x}}\|^2$ *can be expressed in terms of latent space quantities:*

$$\|\boldsymbol{\delta}_x\|^2 \approx \mathbf{d}_z^{\top}\big(\mathbf{J}_z^{(\mathbf{x})^{\top}}\mathbf{J}_z^{(x)}\big)\mathbf{d}_z,$$

where the matrix of derivatives with respect to output $\mathbf{x} = g(\mathbf{z})$ *is given by* $\mathbf{J}_z^{(\mathbf{x})} := \frac{\partial g(\mathbf{z})}{\partial \mathbf{z}}\Big|_{\mathbf{z}=\mathbf{z}}$.

Proof. We use the cost of recourse, and a first-order Taylor series approximation for $g(\mathbf{z} + \mathbf{d}_z)$ at $\mathbf{z}$ to arrive at:

$$\begin{aligned}
\|\boldsymbol{\delta}_x\|^2 &= \|g(\mathbf{z}) - g(\mathbf{z} + \mathbf{d}_z)\|^2 \\
&\approx \|g(\mathbf{z}) - (g(\mathbf{z}) + \mathbf{J}_{\mathbf{z}}^{(\mathbf{x})}\mathbf{d}_z)\|^2 \\
&= \mathbf{d}_z^{\top}\big(\mathbf{J}_z^{(\mathbf{x})^{\top}}\mathbf{J}_z^{(\mathbf{x})}\big)\mathbf{d}_z,
\end{aligned}$$

where $\mathbf{J}_{\mathbf{z}}^{(\mathbf{x})} := \frac{\partial g(\mathbf{z})}{\partial \mathbf{z}}\Big|_{\mathbf{z}=\mathbf{z}}$. □

On an intuitive level, Lemma 1 measures how the cost – measured in input space quantities – depends on perturbations of each component of the generative latent space $\mathbf{z}$.

B Implementation Details

B.1 Recourse Methods

For all data sets, the features are binary-encoded and the data is scaled to lie between 0 and 1. We partition the data sets into train-test splits. The training set is used to train the classification models for which recourses are generated. Recourses are generated for all samples in the test split for the fixed classification model. In particular, we use the following algorithms to generate recourses. Specifically,

- SCFE As suggested in Wachter et al. (2018), an Adam optimizer is used to oprimize (1). We obtain recourses using an ℓ_1 distance function, and the binary cross entropy loss between the counterfactual label and the target.
- GS The explanation model uses a counterfactual search algorithm in the input space. Particularly, instances are sampled within an ℓ_1-norm ball with search radius search radius r_i until recourse is successfully obtained. The search radius of the norm ball is increased until recourse is found.
- C-CHVAE An autoencoder is additionally trained to model the data-manifold. The explanation model uses a counterfactual search algorithm in the latent space of the AE. Particularly, a latent sample within an ℓ_1-norm ball with search radius r_l is used until recourse is successfully obtained. The search radius of the norm ball is increased until recourse is found. The architecture of the generative model are provided in Appendix B.3.

- REVISE As with the recourse model of Pawelczyk et al. (2020), an autoencoder is additionally trained to model the data-manifold. The explanation model uses a gradient-based search algorithm in the latent space of the AE. For a fixed weight on the distance component, we allow up to 500 gradient steps until recourse is successfully obtained. Moreover, we iteratively search for the weight leading up to minimum cost recourse. The architectures of the generative model are provided in Appendix B.3.

We describe architecture and training details in the following.

B.2 Supervised Classification Models

All models are implemented in PyTorch and use a $80 - 20$ train-test split for model training and evaluation. We evaluate model quality based on the model accuracy. All models are trained with the same architectures across the data sets:

	Neural Network	Logistic Regression
Units	[Input dim, 18, 9, 3, 1]	[Input dim, 1]
Type	Fully connected	Fully connected
Intermediate activations	ReLU	N/A
Last layer activations	Sigmoid	Sigmoid

Table 2: Classification Model Details

		Adult	COMPAS	Give Me Credit
Batch-size	ANN	512	32	64
	Logistic Regression	512	32	64
Epochs	ANN	50	40	30
	Logistic Regression	50	40	30
Learning rate	ANN	0.002	0.002	0.001
	Logistic Regression	0.002	0.002	0.001

Table 3: Training details

	Adult	COMPAS	Give Me Credit
Logistic Regression	0.83	0.84	0.92
Neural Network	0.84	0.85	0.93

Table 4: Performance of classification models used for generating algorithmic recourse.

B.3 Generative Model Architectures used for DEAR, CCHVAE and REVISE

For all experiments, we use the following architectures.

Additionally, for DEAR all generative models use the Hessian Penalty (Peebles et al., 2020) and a residual block, which we both described in more detail in Section 4 of the main text.

	Adult	COMPAS	Give Me Credit
Encoder layers	[input dim, 16, 32, 10]	[input dim, 8, 10, 5]	[input dim, 8, 10, 5]
Decoder layers	[10, 16, 32, input dim]	[5, 10, 8, input dim]	[5, 10, 8, input dim]
Type	Fully connected	Fully connected	Fully connected
Loss function	MSE	MSE	MSE

Table 5: Autoencoder details