Ordered Counterfactual Explanation by Mixed-Integer Linear Optimization

Kentaro Kanamori,[1] **Takuya Takagi,** [2] **Ken Kobayashi,** [2,3]
Yuichi Ike, [2] **Kento Uemura,** [2] **Hiroki Arimura** [1]
[1]Hokkaido University,
[2]Fujitsu Laboratories Ltd.,
[3]Tokyo Institute of Technology,
{kanamori, arim}@ist.hokudai.ac.jp, {takagi.takuya, ken-kobayashi, ike.yuichi, uemura.kento}@fujitsu.com

Abstract

Post-hoc explanation methods for machine learning models have been widely used to support decision-making. One of the popular methods is *Counterfactual Explanation (CE)*, also known as *Actionable Recourse*, which provides a user with a perturbation vector of features that alters the prediction result. Given a perturbation vector, a user can interpret it as an "*action*" for obtaining one's desired decision result. In practice, however, showing only a perturbation vector is often insufficient for users to execute the action. The reason is that if there is an asymmetric interaction among features, such as causality, the total cost of the action is expected to depend on the order of changing features. Therefore, practical CE methods are required to provide an appropriate order of changing features in addition to a perturbation vector. For this purpose, we propose a new framework called *Ordered Counterfactual Explanation (OrdCE)*. We introduce a new objective function that evaluates a pair of an action and an order based on feature interaction. To extract an optimal pair, we propose a mixed-integer linear optimization approach with our objective function. Numerical experiments on real datasets demonstrated the effectiveness of our OrdCE in comparison with unordered CE methods.

Introduction

Complex machine learning models such as neural networks and tree ensembles are widely used in critical decision-making tasks (e.g., medical diagnosis and loan approval). Thus, post-hoc methods for extracting explanations from an individual prediction of these models have been attracting much attention for the last few years (Ribeiro, Singh, and Guestrin 2016; Lundberg and Lee 2017; Koh and Liang 2017; Ribeiro, Singh, and Guestrin 2018). To provide a user with a better insight into future improvement, a post-hoc method needs to show not only why undesirable predictions are given, but also how to act to obtain a desired prediction result (Doshi-Velez and Kim 2017; Miller 2019).

One of the post-hoc methods that show an action to obtain a desired outcome is *Counterfactual Explanation (CE)* (Wachter, Mittelstadt, and Russell 2018), also known as *Actionable Recourse* (Ustun, Spangher, and Liu 2019). Consider an example of a synthetic credit loan approval dataset shown in Figure 1. Imagine a situation that a user receives an undesired prediction $H(\hat{x}) \neq y^\star$ for a target label $y^\star$ from a trained model H, which means denial of credit loan, on an instance $\hat{x}$ related to one's current livelihood. We want to provide the user with advice a on changes of features such as "Income" and "JobSkill" so that the user can change one's current status $\hat{x}$ to obtain the desired outcome $H(\hat{x} + a) = y^\star$ (Ustun, Spangher, and Liu 2019).

To achieve this goal, most of the existing CE methods find a perturbation vector $a^\star \in \mathcal{A}$, called an *action*, as an optimal solution of the following optimization problem:

$$a^\star := \operatorname*{arg\,min}_{a \in \mathcal{A}} \; C_{\mathrm{dist}}(a \mid \hat{x}) \quad \text{subject to } H(\hat{x} + a) = y^\star,$$

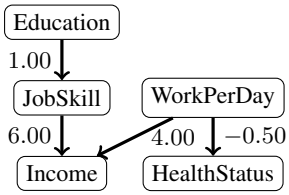
where $\mathcal{A}$ is a set of feasible perturbation vectors, and C_{dist} is a cost function that measures the required efforts of a, such as TLPS (Ustun, Spangher, and Liu 2019) and DACE (Kanamori et al. 2020). Table 1 presents examples of actions extracted from a logistic regression classifier trained on our credit approval dataset in Figure 1. A user can obtain one's desired outcome by changing each feature according to the suggested perturbation vectors in Table 1.

In practice, however, showing only a perturbation vector $a^\star$ is often insufficient for users to execute the action due to interaction among features (Poyiadzi et al. 2020). In the previous example, as shown in the causal DAG in Figure 1(b), we have an asymmetric interaction "JobSkill" $\rightarrow$ "Income", which means that increasing one's "JobSkill" has a positive effect on increasing "Income" while the opposite does not. From these observations, we see that it is more reasonable to increase first "JobSkill" and then "Income" than the reverse order. Thus, practical CE methods are required to provide an appropriate order of changing features in addition to a perturbation vector $a^\star$.

To achieve this requirement, we propose a novel CE framework that returns a pair $(a^\star, \sigma^\star)$, called an *ordered action*, of a perturbation vector $a^\star$ and a permutation $\sigma^\star$ of features that advises a user to change features in that order. We assume that the feature interaction is represented by an *interaction matrix* M, whose element indicates the linear interaction between two features, such as correlations, causal effects, or given by some prior knowledge. Roughly speak-

	Name	Range
x_1	"Education"	$1, \ldots, 5$
x_2	"JobSkill"	$1, \ldots, 10$
x_3	"Income"	$10, \ldots, 100$
x_4	"WorkPerDay"	$4, \ldots, 10$
x_5	"HealthStatus"	$1, \ldots, 10$

(a) Features



(b) Causal DAG

Figure 1: Features and the causal DAG of our synthetic credit loan approval dataset. The task is to predict whether an individual's credit loan will be approved. We labeled each individual depending on one's values of "Income" and "HealthStatus".

ing, we consider the following optimization problem:

$$(a^\star, \sigma^\star) := \underset{a \in \mathcal{A}, \sigma \in \Sigma}{\arg\min}\ C_{\text{dist}}(a \mid \hat{x}) + \gamma \cdot C_{\text{ord}}(a, \sigma \mid M)$$
$$\text{subject to } H(\hat{x} + a) = y^\star,$$

where C_{ord} is a new cost function for determining an order of changing features, Σ is the set of all permutations of the features perturbed by a, and $\gamma > 0$ is a trade-off parameter.

Our Contributions

Our goal is to extend CE framework so that it provides an ordered action by taking into account feature interaction. Our contributions are summarized as follows:

1. As a new framework for CE, we propose *Ordered Counterfactual Explanation (OrdCE)* that provides an *ordered action*, i.e., a pair of a perturbation vector and an order of changing features. For that purpose, we introduce a new objective function that evaluates the cost of an ordered action based on a given *interaction matrix* M.
2. We formulate the problem of finding an optimal ordered action as a *mixed-integer linear optimization (MILO)* problem, which can be efficiently solved by modern MILO solvers such as CPLEX (IBM 2018). Our formulation works on popular classifiers, such as linear models, tree ensembles, and multilayer perceptrons. In addition, the formulation can be combined with any existing cost function used in MILO-based CE methods, such as TLPS (Ustun, Spangher, and Liu 2019) or DACE (Kanamori et al. 2020).
3. We conducted numerical experiments on real datasets and compared the performance of our OrdCE with previous CE methods. We confirmed that (i) our MILO approach obtained better ordered actions than baselines within practical computation time, and (ii) the obtained orders were reasonable from the perspective of feature interaction.

Table 2 presents examples of ordered actions extracted by OrdCE on the credit approval dataset in Figure 1. These orders of changing features are consistent with the causal DAG shown in Figure 1(b). For example, the ordered action extracted by OrdCE + DACE indicates increasing "WorkPerDay" before "Income". This order is more reasonable than its reverse order because "WorkPerDay" has a positive effect on "Income". In addition, in the result of OrdCE + TLPS, the total number of the changing features increases from that of the unordered TLPS in Table 1. However, because "JobSkill" has an effect 6.00 on "Income" as shown in Figure 1(b), changing "JobSkill" by $+1$ naturally causes an increase of "Income" by $+6$. Thus, it is expected that the user completes the ordered action suggested by OrdCE + TLPS with only changing "JobSkill" at the 1st step. Our OrdCE can find such an appropriate order by optimizing a perturbation vector and an order simultaneously. In summary, we see that our method presents a reasonable ordered action, which helps a user act to obtain the desired outcome.

Method	Action		
	"Income"	"WorkPerDay"	"HealthStatus"
DACE	+4	+1	+3
TLPS	+5	0	0

Table 1: Examples of actions extracted by the existing CE methods on the credit approval dataset in Figure 1.

Method	Order	Feature	Action
OrdCE + DACE	1st	"HealthStatus"	+3
	2nd	"WorkPerDay"	+1
	3rd	"Income"	+4
OrdCE + TLPS	1st	"JobSkill"	+1
	2nd	"Income"	+6

Table 2: Examples of ordered actions extracted by our OrdCE on the credit approval dataset in Figure 1.

Related Work

The existing CE methods can be categorized into gradient-based (Wachter, Mittelstadt, and Russell 2018; Moore, Hammerla, and Watkins 2019; Karimi et al. 2020b), autoencoder (Dhurandhar et al. 2018; Mahajan, Tan, and Sharma 2019), SAT (Karimi et al. 2020a), or mixed-integer linear optimization (MILO) (Cui et al. 2015; Ustun, Spangher, and Liu 2019; Russell 2019; Kanamori et al. 2020). Since our cost function is non-differentiable due to the discrete nature of a permutation σ over features, we focus on MILO-based methods, which can directly handle such functions.

Most of CE methods provide only a perturbation vector as an action. To the best of our knowledge, FACE (Poyiadzi et al. 2020) and OAS (Ramakrishnan, Lee, and Albarghouthi 2020) are the only exceptions that consider a sequence of actions. FACE provides a sequence of training instances as a path from a given instance to the target instances in a neighborhood graph. However, FACE does not take into account feature interaction and not determine the execution order of features. On the other hands, OAS provides a sequence of actions, which is related to classical planning (Nau, Ghallab, and Traverso 2004). There, the costs of candidate actions are static and irrelevant to their order, while in OrdCE, the costs dynamically depend on the previously chosen actions due to

their interaction. It is also noteworthy that (Bertsimas et al. 2019) studied how to determine an order of features to improve explainability in linear regression.

Preliminaries

For a positive integer $n \in \mathbb{N}$, we write $[n] := \{1, \dots, n\}$. For a proposition ψ, $\mathbb{I}[\psi]$ denotes the indicator of ψ, i.e., $\mathbb{I}[\psi] = 1$ if ψ is true, and $\mathbb{I}[\psi] = 0$ if ψ is false.

Throughout this work, we consider a *binary classification problem* as a prediction task, which is sufficient for CE. For a multi-class classification problem, we can reduce it to a binary one between the target class and the other classes. We denote input and output domains by $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_D \subseteq \mathbb{R}^D$ and $\mathcal{Y} = \{-1, +1\}$, respectively. An *instance* is a vector $x = (x_1, \dots, x_D) \in \mathcal{X}$. A *classifier* is a function $H \colon \mathcal{X} \to \mathcal{Y}$.

Additive Classifiers

In this paper, we focus on CE with *additive classifiers (AC)* $H \colon \mathcal{X} \to \mathcal{Y}$ expressed in the following additive form (Hastie, Tibshirani, and Friedman 2009):

$$H(x) = \operatorname{sgn}\left(\sum\nolimits_{t=1}^{T} w_t \cdot h_t(x) - b\right),$$

where $T \in \mathbb{N}$ is the total number of base learners, $h_t \colon \mathcal{X} \to \mathbb{R}$ is a base learner, $w_t \in \mathbb{R}$ is a weight value of h_t for $t \in [T]$, and $b \in \mathbb{R}$ is an intercept.

Linear Models *Linear models (LM)*, such as Logistic Regression and Linear Support Vector Machines, are one of the most popular classifiers (Hastie, Tibshirani, and Friedman 2009). We remark that an LM is a special case of AC such that $T = D$ and $h_d(x) = x_d$ for $d \in [D]$.

Tree Ensembles *Tree ensembles (TE)*, such as Random Forests (Breiman 2001), are renowned for their high prediction performances in machine learning competitions. Each base learner h_t of a TE is a *decision tree*, which is a classifier consisting of a set of if-then-else rules expressed in the form of a binary tree. A decision tree $h_t \colon \mathcal{X} \to \mathcal{Y}$ with L_t leaves represents a partition $r_{t,1}, \dots, r_{t,L_t}$ of the input domain $\mathcal{X}$ (Hastie, Tibshirani, and Friedman 2009). Then, it can be expressed as the linear combination $h_t(x) = \sum_{l=1}^{L_t} \hat{y}_{t,l} \cdot \mathbb{I}[x \in r_{t,l}]$, where $\hat{y}_{t,l} \in \mathcal{Y}$ is a predictive label of the leaf $l \in [L_t]$.

Multilayer Perceptrons *Multilayer perceptrons (MLP)*, or neural networks, have become increasingly more common over the past decade (Goodfellow, Bengio, and Courville 2016). For simplicity, we consider two-layer ReLU networks, i.e., MLP with one hidden layer and the rectified linear unit (ReLU) function $g(x) = \max(0, x)$ as an activation function. Our proposed methods presented later can be extended to general multilayer ReLU networks. In MLP, each base learner h_t is an output of a neuron in its hidden layers with the ReLU function g. It can be expressed as $h_t(x) = g(w^{(t)} \cdot x + b^{(t)})$, where $w^{(t)} \in \mathbb{R}^D$ and $b^{(t)} \in \mathbb{R}$ are weight values and an intercept with respect to the t-th neuron in the hidden layer, respectively.

Problem Statement

Action and Ordered Action

Let $H \colon \mathcal{X} \to \mathcal{Y}$ and $\hat{x} = (\hat{x}_1, \dots, \hat{x}_D) \in \mathcal{X}$ be a classifier and a given instance such that $H(\hat{x}) = -1$, respectively. We define an *action* for $\hat{x}$ as a perturbation vector $a \in \mathbb{R}^D$ such that $H(\hat{x} + a) = +1$. An *action set* $\mathcal{A} = A_1 \times \cdots \times A_D$ is a finite set of feasible actions such that $0 \in A_d$ and $A_d \subseteq \{a_d \in \mathbb{R} \mid \hat{x}_d + a_d \in \mathcal{X}_d\}$ for $d \in [D]$. We can determine each A_d depending on the type and constraint of the feature $d \in [D]$. For example, a feature representing "Age" must be a positive integer and cannot be decreased. We define the *perturbing features* of an action a as $\operatorname{supp}(a) := \{d \in [D] \mid a_d \neq 0\}$. For $K \in [D]$, we write $\mathcal{A}_{\leq K} := \{a \in \mathcal{A} \mid |\operatorname{supp}(a)| \leq K\}$.

We introduce an *ordered action* for OrdCE. An ordered action is a pair of a perturbation vector $a \in \mathcal{A}_{=K}$ for some $K \in [D]$ and a permutation $\sigma = (\sigma_1, \dots, \sigma_K) \in [D]^K$ of the perturbing features $\operatorname{supp}(a)$, which suggests perturbing the features $\operatorname{supp}(a)$ in that order. We denote by $\Sigma(a)$ the set of all permutations of $\operatorname{supp}(a)$, and call $\sigma \in \Sigma(a)$ a *perturbing order* of a.

Interaction Matrix

Practically, causal relationships are usually unknown in advance, and we need to estimate them. Since linear causal models can be estimated properly in practical settings where hidden common causes are included (Shimizu et al. 2011), we assume the feature interaction is linear. We assume that the feature interaction is represented by a matrix $M = (M_{i,j})_{1 \leq i,j \leq D}$, which we call an *interaction matrix*. Each element $M_{i,j}$ represents the linear interaction from i to j, that is, when we perturb a feature x_i to $x_i + a_i$, then x_j is perturbed to $x_j + M_{i,j} a_i$. We can compute $M_{i,j}$ explicitly with causal effect estimation methods (Pearl 2009; Shimizu et al. 2011; Janzing et al. 2013) or some prior knowledge of domain experts.

From a given causal DAG estimated by, for example, DirectLiNGAM (Shimizu et al. 2011; Hyvärinen and Smith 2013), we can compute an interaction matrix as follows. Let $B = (B_{i,j})_{1 \leq i,j \leq D}$ be the adjacency matrix of the estimated causal DAG. By reordering the order of the nodes, we can assume that B is a strictly upper triangular matrix. Here, LiNGAM considers a model expressed the following structural equations: $x_j = \sum_{i \in \mathrm{pa}_B(j)} B_{i,j} x_i + e_j$, where e_j is a continuous random variable that has non-Gaussian distributions and is independent of each other, and $\mathrm{pa}_B(j) \subseteq [D]$ is the set of features that are the ancestors of j on the estimated causal DAG. Then, we obtain

$$M = I + \sum\nolimits_{k=1}^{D-1} B^k.$$

Cost Function

As a score to evaluate the required effort of an ordered action (a, σ), we introduce a new cost function C_{OrdCE} as follows. Given an input instance $\hat{x} \in \mathcal{X}$, an interaction matrix M, and a trade-off parameter $\gamma \geq 0$, we define C_{OrdCE} for a

pair of a perturbation $a \in \mathcal{A}$ and its order $\sigma \in \Sigma(a)$ as

$$C_{\mathrm{OrdCE}}(a, \sigma \mid \hat{x}, M, \gamma) := C_{\mathrm{dist}}(a \mid \hat{x}) + \gamma \cdot C_{\mathrm{ord}}(a, \sigma \mid M),$$

where C_{dist} and C_{ord} are *distance-based* and *ordering cost functions*, respectively. The former evaluates the required effort of a perturbation vector a, and the latter determines a perturbing order σ of a.

Distance-based Cost Function As with the existing CE methods, we utilize a distance-based cost function C_{dist} to evaluate the required effort of an entire perturbation a. For simplicity, we assume C_{dist} as the following form:

$$C_{\mathrm{dist}}(a \mid \hat{x}) = \sum\nolimits_{d=1}^{D} \mathrm{dist}_d(a_d \mid \hat{x}_d),$$

where $\mathrm{dist}_d \colon A_d \to \mathbb{R}_{\geq 0}$ is a cost measure of the feature d that evaluates the effort to change $\hat{x}_d$ to $\hat{x}_d + a_d$, such as the total-log percentile shift (Ustun, Spangher, and Liu 2019). Note that our optimization approach can adapt to other types of existing cost functions, such as ℓ_1-Mahalanobis' distance (Kanamori et al. 2020). While these useful distance-based cost functions have been proposed, they do not deal with a perturbing order $\sigma \in \Sigma(a)$.

Ordering Cost Function To extend CE so as to deal with a perturbing order $\sigma = (\sigma_1, \ldots, \sigma_K) \in \Sigma(a)$, we introduce an ordering cost function C_{ord} as the following form:

$$C_{\mathrm{ord}}(a, \sigma \mid M) = \sum\nolimits_{k=1}^{K} \mathrm{cost}^{(k)}(a_{\sigma_1,\ldots,\sigma_k} \mid M),$$

where $a_{\sigma_1,\ldots,\sigma_k} := (a_{\sigma_1}, \ldots, a_{\sigma_k})$ and $\mathrm{cost}^{(k)}$ is a cost for each k-th perturbation a_{σ_k}. We define $\mathrm{cost}^{(k)}$ as a function that depends not only on the perturbation of σ_k at the k-th step but also on the previously perturbed features $\sigma_1, \ldots, \sigma_{k-1} \in \mathrm{supp}(a)$ since the actual amount of a perturbation of a feature is affected by the values of other features that interact with it. Note that we assume that the previously perturbed features are unaffected by the following perturbation, which is based on the *intervention* in causal models (Pearl 2009).

To define $\mathrm{cost}^{(k)}$, we introduce a parameter $\Delta^{(k)} = \Delta^{(k)}(a_{\sigma_1,\ldots,\sigma_k} \mid M)$, called the *actual perturbation*, as the amount of change on x_{σ_k} that we actually need to obtain a desired perturbation. Then, the resulting perturbation a_{σ_k} is equal to the sum of $\Delta^{(k)}$ and the effect of the previous perturbations $\Delta^{(1)}, \ldots, \Delta^{(k-1)}$. Formalizing this idea, we have the following conditions on $\Delta^{(k)}$ for $k = 1, 2, 3$:

$$\begin{aligned} a_{\sigma_1} &= \Delta^{(1)}, \\ a_{\sigma_2} &= \Delta^{(2)} + M_{\sigma_1,\sigma_2} \cdot \Delta^{(1)}, \\ a_{\sigma_3} &= \Delta^{(3)} + M_{\sigma_2,\sigma_3} \cdot \Delta^{(2)} + M_{\sigma_1,\sigma_3} \cdot \Delta^{(1)}. \end{aligned}$$

Generally, we have $a_{\sigma_k} = \Delta^{(k)} + \sum_{l=1}^{k-1} M_{\sigma_l,\sigma_k} \cdot \Delta^{(l)}$. For any $k \in [K]$, we inductively obtain

$$\Delta^{(k)}(a_{\sigma_1,\ldots,\sigma_k} \mid M) = a_{\sigma_k} - \sum\nolimits_{l=1}^{k-1} M_{\sigma_l,\sigma_k} \cdot \Delta^{(l)}(a_{\sigma_1,\ldots,\sigma_l} \mid M).$$

By using $\Delta^{(k)}$, we define $\mathrm{cost}^{(k)}$ as follows:

$$\mathrm{cost}^{(k)}(a_{\sigma_1,\ldots,\sigma_k} \mid M) := \left| \Delta^{(k)}(a_{\sigma_1,\ldots,\sigma_k} \mid M) \right|.$$

In practice, since each feature has different scale, we multiply each $\Delta^{(k)}$ by a scaling factor $s_{\sigma_k} > 0$, such as the inverse of its standard deviation.

Problem Definition

Our aim is to find an ordered action (a, σ) that minimizes the cost C_{OrdCE}. This problem can be defined as follows.

Problem 1. Given an additive classifier $H \colon \mathcal{X} \to \mathcal{Y}$, an input instance $\hat{x} \in \mathcal{X}$ such that $H(\hat{x}) = -1$, an action set $\mathcal{A}$, an interaction matrix $M \in \mathbb{R}^{D \times D}$, $K \in [D]$, and $\gamma \geq 0$, find an ordered action (a, σ) that is an optimal solution for the following optimization problem:

$$\begin{array}{ll} \underset{a \in \mathcal{A}_{\leq K}, \sigma \in \Sigma(a)}{\text{minimize}} & C_{\mathrm{OrdCE}}(a, \sigma \mid \hat{x}, M, \gamma) \\ \text{subject to} & H(\hat{x} + a) = +1. \end{array}$$

By solving the above optimization problem, we obtain an ordered action (a, σ) that accords with feature interaction.

Concrete Examples

To study the cost function C_{ord} and objective function C_{OrdCE}, we present concrete examples to observe behavior of these functions in the same setting as Introduction. Our synthetic credit loan approval dataset consists of five features $x_1, \ldots x_5$ presented in Figure 1. As an interaction matrix, we take the matrix M whose element $M_{i,j}$ represents the average causal effect from a feature i to j. As described previously, $M = (M_{i,j})_{1 \leq i,j \leq 5}$ is calculated from the adjacency matrix of the causal DAG in Figure 1(b) as follows:

$$M = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 1 & 1 & 6 & 0 & 0 \\ 0 & 1 & 6 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 4 & 1 & -0.5 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}.$$

In the examples below, we use C_{ord} with scaling factors $s_d > 0$ ($d \in [D]$):

$$C_{\mathrm{ord}}(a, \sigma \mid M) = \sum\nolimits_{k=1}^{K} s_{\sigma_k} \cdot \left| \Delta^{(k)}(a_{\sigma_1,\ldots,\sigma_k} \mid M) \right|.$$

First, we show an example to observe behavior of the ordering cost function C_{ord} in the following Example 1.

Example 1. Consider a perturbation $a = (0, 0, 4, 1, 3)$ and its perturbing orders $\sigma = (4, 3, 5)$ and $\sigma^\circ = (5, 4, 3)$. We compare the values of our ordering cost function C_{ord} for the two ordered actions (a, σ) and (a, σ°).

For (a, σ), the actual perturbations $\Delta^{(k)}$ can be calculated as follows:

$$\begin{aligned} \Delta^{(1)} &= 1 - 0 = 1, \\ \Delta^{(2)} &= 4 - M_{4,3} \cdot \Delta^{(1)} = 0, \\ \Delta^{(3)} &= 3 - M_{3,5} \cdot \Delta^{(2)} - M_{4,5} \cdot \Delta^{(1)} = 3.5. \end{aligned}$$

Thus, the value of C_{ord} for (a, σ) can be calculated as

$$C_{\mathrm{ord}}(a, \sigma \mid M) = s_{\sigma_1} \cdot |\Delta^{(1)}| + s_{\sigma_2} \cdot |\Delta^{(2)}| + s_{\sigma_3} \cdot |\Delta^{(3)}| = s_4 + 3.5 s_5.$$

Similarly, the value of C_{ord} for (a, σ°) is

$$C_{\mathrm{ord}}(a, \sigma^\circ \mid M) = s_{\sigma_1^\circ} \cdot |\Delta^{(1)}| + s_{\sigma_2^\circ} \cdot |\Delta^{(2)}| + s_{\sigma_3^\circ} \cdot |\Delta^{(3)}| = s_4 + 3 s_5.$$

Because $s_d > 0$ for all $d \in \{1, \ldots, 5\}$, $C_{\mathrm{ord}}(a, \sigma \mid M) > C_{\mathrm{ord}}(a, \sigma^\circ \mid M)$ holds. □

In the above example, the ordered action (a, σ) suggests increasing the values of "WorkPerDay", "Income", and "HealthStatus" in this order. The other ordered action (a, σ°) suggests increasing the values of "HealthStatus", "WorkPerDay", and "Income" in this order. Since "WorkPerDay" has a negative causal effect to "HealthStatus", the perturbing order σ° is more reasonable than σ from the perspective of the feature interaction. The above example indicates that we can obtain an appropriate order of its perturbing features by minimizing C_{ord}.

Next, we show an example to observe behavior of the objective cost function C_{OrdCE} in the following Example 2.

Example 2. Consider the same setting as Example 1 and two feasible ordered actions (a, σ) and (a°, σ°) with

$$a = (0, 0, 6, 0, 0), \quad \sigma = (3),$$
$$a^\circ = (0, 1, 6, 0, 0), \quad \sigma^\circ = (2, 3).$$

We compare the values of our objective function C_{OrdCE} for the two ordered actions (a, σ) and (a°, σ°).

For (a, σ), the actual perturbations $\Delta^{(1)}$ can be calculated as $\Delta^{(1)} = 6 - 0 = 6$. Thus, the value of C_{OrdCE} for (a, σ) can be calculated as

$$C_{\mathrm{OrdCE}}(a, \sigma \mid \hat{x}, M, \gamma) = c_3 + \gamma \cdot (s_{\sigma_1} \cdot |\Delta^{(1)}|) = c_3 + \gamma \cdot 6 s_3,$$

where $c_3 = \mathrm{dist}_3(6 \mid \hat{x}_3) > 0$.

Similarly, the value of C_{OrdCE} for (a°, σ°) is

$$\begin{aligned} &C_{\mathrm{OrdCE}}(a^\circ, \sigma^\circ \mid \hat{x}, M, \gamma) \\ &= c_2 + c_3 + \gamma \cdot (s_{\sigma_1^\circ} \cdot |\Delta^{(1)}| + s_{\sigma_2^\circ} \cdot |\Delta^{(2)}|) \\ &= c_2 + c_3 + \gamma \cdot s_2, \end{aligned}$$

where $c_2 = \mathrm{dist}_2(1 \mid \hat{x}_2) > 0$.

Now we assume $6 s_3 - s_2 > 0$. Then, we obtain

$$\begin{aligned} &C_{\mathrm{OrdCE}}(a, \sigma \mid \hat{x}, M, \gamma) > C_{\mathrm{OrdCE}}(a^\circ, \sigma^\circ \mid \hat{x}, M, \gamma) \\ &\iff \gamma \cdot (6 s_3 - s_2) - c_2 > 0 \\ &\iff \gamma > \frac{c_2}{6 s_3 - s_2}. \end{aligned}$$

Hence, if $6 s_3 - s_2 > 0$ and $\gamma > c_2 / (6 s_3 - s_2)$, then $C_{\mathrm{OrdCE}}(a, \sigma \mid \hat{x}, M, \gamma) > C_{\mathrm{OrdCE}}(a^\circ, \sigma^\circ \mid \hat{x}, M, \gamma)$. □

In the above example, the ordered action (a, σ) suggests increasing only "Income". The other ordered action (a°, σ°) suggests increasing the values of "JobSkill" and "Income" in this order. The total number of the changing features of the latter ordered action is greater than that of the former. However, a user is expected to complete the latter ordered action with only changing "JobSkill" since increasing one's "JobSkill" has a positive effect on increasing "Income" as mentioned in Introduction. From the above example, we see that we can adjust a trade-off between the required effort of an entire perturbation a and each step $\Delta^{(k)}$ by tuning the parameter γ.

Optimization Framework

Basic Constraints

For $d \in [D]$, we set $A_d = \{a_{d,1}, \ldots, a_{d,I_d}\}$ and $a_{d,1} = 0$. First, we introduce binary variables $\pi_{d,i} \in \{0, 1\}$ for $d \in [D]$ and $i \in [I_d]$, which indicate that $a_{d,i} \in A_d$ is selected ($\pi_{d,i} = 1$) or not ($\pi_{d,i} = 0$) as in the previous MILO-based methods (Ustun, Spangher, and Liu 2019; Kanamori et al. 2020). Then, $\pi_{d,i}$ must satisfy the following constraints:

$$\sum\nolimits_{i=1}^{I_d} \pi_{d,i} = 1, \forall d \in [D]. \tag{1}$$

By using $\pi_{d,i}$, we can express a perturbation for each feature d as $a_d = \sum_{i=1}^{I_d} a_{d,i} \cdot \pi_{d,i}$. Note that $\pi_{d,1} = 1$ indicates that a feature d is not perturbed since $a_{d,1} = 0$.

To express an order of perturbing features, we introduce binary variables $\pi_{d,i}^{(k)} \in \{0, 1\}$ for $d \in [D], i \in [I_d]$, and $k \in [K]$, which indicate $a_{d,i} \in A_d$ is selected as the k-th perturbation; that is, $\pi_{d,i}^{(k)} = 1$ if $a_{d,i} \in A_d$ is selected in the k-th step, and $\pi_{d,i}^{(k)} = 0$ otherwise. Then, $\pi_{d,i}^{(k)}$ must satisfy the following constraints:

$$\pi_{d,i} = \sum\nolimits_{k=1}^{K} \pi_{d,i}^{(k)}, \forall d \in [D], \forall i \in [I_d]. \tag{2}$$

We also introduce binary variables $\sigma_{k,d}$ for $k \in [K]$ and $d \in [D]$ that indicate whether the feature d is perturbed in the k-th step. We impose the following constraints on $\sigma_{k,d}$:

$$\sigma_{k,d} = 1 - \pi_{d,1}^{(k)}, \forall k \in [K], \forall d \in [D], \tag{3}$$

$$\sum\nolimits_{k=1}^{K} \sigma_{k,d} \leq 1, \forall d \in [D], \tag{4}$$

$$\sum\nolimits_{d=1}^{D} \sigma_{k,d} \leq 1, \forall k \in [K], \tag{5}$$

$$\sum\nolimits_{d=1}^{D} \sigma_{k,d} \geq \sum\nolimits_{d=1}^{D} \sigma_{k+1,d}, \forall k \in [K-1]. \tag{6}$$

Constraint (4) allows that we can perturb each feature at most once. Constraint (5) imposes that we can perturb at most one feature in each step. Constraint (6) is a symmetry breaking constraint on $\sigma_{k,d}$.

Objective Function

Our objective function C_{OrdCE} consists of C_{dist} and C_{ord}, which we express with the program variables $\pi_{d,i}^{(k)}$ and $\sigma_{k,d}$.

Distance-based Cost Function From our assumption of C_{dist}, it can be expressed as follows:

$$C_{\mathrm{dist}}(a \mid \hat{x}) = \sum\nolimits_{d=1}^{D} \sum\nolimits_{i=1}^{I_d} \sum\nolimits_{k=1}^{K} c_{d,i} \cdot \pi_{d,i}^{(k)},$$

where $c_{d,i}$ is a constant value such that $c_{d,i} = \mathrm{dist}_d(a_{d,i} \mid \hat{x}_d)$. Note that our MILO formulation can adapt to other existing cost functions such as DACE (Kanamori et al. 2020) and SCM (Mahajan, Tan, and Sharma 2019).

Ordering Cost Function Since C_{ord} is non-linear due to a permutation σ, we need to express it by linear constraints of the variables. We introduce variables ζ_k for $k \in [K]$ such that $\zeta_k = |\Delta^{(k)}|$. Then, C_{ord} can be expressed as follows:

$$C_{\mathrm{ord}}(a, \sigma \mid M) = \sum\nolimits_{k=1}^{K} \zeta_k.$$

Moreover, for $k \in [K]$ and $d \in [D]$, we introduce variables $\delta_{k,d} \in \mathbb{R}$ to express $\zeta_k = |\sum_{d=1}^{D} \delta_{k,d}|$. Then, $\delta_{k,d}$ must satisfy $\delta_{k,d} = \Delta^{(k)}$ if $\sigma_{k,d} = 1$, and $\delta_{k,d} = 0$ if $\sigma_{k,d} = 0$. We can linearize these non-linear constraints as follows:

$$\delta_{k,d} \geq \sum\nolimits_{i=1}^{I_d} a_{d,i} \cdot \pi_{d,i}^{(k)} - \varepsilon_{k,d} - U_{k,d} \cdot (1 - \sigma_{k,d}), \quad \forall k \in [K], \forall d \in [D], \quad (7)$$

$$\delta_{k,d} \leq \sum\nolimits_{i=1}^{I_d} a_{d,i} \cdot \pi_{d,i}^{(k)} - \varepsilon_{k,d} - L_{k,d} \cdot (1 - \sigma_{k,d}), \quad \forall k \in [K], \forall d \in [D], \quad (8)$$

$$L_{k,d} \cdot \sigma_{k,d} \leq \delta_{k,d} \leq U_{k,d} \cdot \sigma_{k,d}, \forall k \in [K], \forall d \in [D], \quad (9)$$

$$\varepsilon_{k,d} = \sum\nolimits_{l=1}^{k-1} \sum\nolimits_{d'=1}^{D} M_{d',d} \cdot \delta_{l,d'}, \quad \forall k \in [K], \forall d \in [D], \quad (10)$$

$$-\zeta_k \leq \sum\nolimits_{d=1}^{D} \delta_{k,d} \leq \zeta_k, \forall k \in [K], \quad (11)$$

where $\varepsilon_{k,d}$ is an auxiliary variable such that $\varepsilon_{k,d} = \sum_{l=1}^{k-1} M_{\sigma_l,d} \cdot \Delta^{(l)}$ for $k \in [K]$ and $d \in [D]$. The constant values $L_{k,d}$ and $U_{k,d}$ are the lower and upper bounds of $\delta_{k,d}$. These values can be recursively computed from the interaction matrix M and the action set $\mathcal{A}$ as follows:

$$L_{k+1,d} = L_{k,d} - \max_{d' \in [D] \setminus \{d\}} \max_{\Delta \in \{L_{k,d'}, U_{k,d'}\}} M_{d',d} \cdot \Delta,$$

$$U_{k+1,d} = U_{k,d} - \min_{d' \in [D] \setminus \{d\}} \min_{\Delta \in \{L_{k,d'}, U_{k,d'}\}} M_{d',d} \cdot \Delta,$$

where $L_{1,d} = \min_{a_d \in A_d} a_d$ and $U_{1,d} = \max_{a_d \in A_d} a_d$.

Base Learner Constraints

We introduce variables $\xi_t \in \mathbb{R}$ for $t \in [T]$ such that $\xi_t = h_t(\hat{x} + a)$, where h_t is the t-th base learner of H. From the definition of additive classifiers, the constraint $H(\hat{x} + a) = +1$ is equivalent to the following linear constraint of ξ_t:

$$\sum\nolimits_{t=1}^{T} w_t \cdot \xi_t \geq b. \quad (12)$$

We express the constraint $\xi_t = h_t(\hat{x} + a)$ by linear constraints of ξ_t and $\pi_{d,i}$ because $h_t(\hat{x} + a)$ depends on the value of a, i.e., the variables $\pi_{d,i}$. In the following, we show how to express ξ_t when H is a linear model (LM), tree ensemble (TE), or multilayer perceptron (MLP).

Linear Models From the definition of LM, $T = D$ and $h_d(\hat{x} + a) = \hat{x}_d + a_d$ for $d \in [D]$. Hence, we can simply express the base learner of the LM as follows:

$$\xi_d = \hat{x}_d + \sum\nolimits_{i=1}^{I_d} a_{d,i} \cdot \pi_{d,i}, \forall d \in [D]. \quad (13)$$

Tree Ensembles Each base learner h_t of the TE is a decision tree. To express $\xi_t = h_t(\hat{x} + a)$, we can utilize the following *decision logic constraint* (Cui et al. 2015; Kanamori et al. 2020):

$$\phi_{t,l} \in \{0, 1\}, \forall t \in [T], \forall l \in [L_t], \quad (14)$$

$$\sum\nolimits_{l=1}^{L_t} \phi_{t,l} = 1, \forall t \in [T], \quad (15)$$

$$D \cdot \phi_{t,l} \leq \sum\nolimits_{d=1}^{D} \sum\nolimits_{i \in I_{t,l}^{(d)}} \pi_{d,i}, \forall t \in [T], \forall l \in [L_t], \quad (16)$$

$$\xi_t = \sum\nolimits_{l=1}^{L_t} \hat{y}_{t,l} \cdot \phi_{t,l}, \forall t \in [T], \quad (17)$$

where $I_{t,l}^{(d)} = \{i \in [I_d] \mid \hat{x}_d + a_{d,i} \in r_{t,l}^{(d)}\}$ and $r_{t,l}^{(d)}$ is the subspace of $\mathcal{X}_d$ such that $r_{t,l} = r_{t,l}^{(1)} \times \cdots \times r_{t,l}^{(D)}$.

Multilayer Perceptrons Each base learner h_t of the MLP is an output of the t-th neuron with the ReLU activation function, i.e., $h_t(x + a) = \max\{0, w^{(t)}(x + a) + b^{(t)}\}$. Hence, we need to extract the positive part of $w^{(t)}(x + a) + b^{(t)}$ as the output of the t-th base learner ξ_t. To express it, we can utilize the following constraints proposed by (Serra, Tjandraatmadja, and Ramalingam 2018):

$$\nu_t \in \{0, 1\}, \bar{\xi}_t \geq 0, \forall t \in [T], \quad (18)$$

$$\xi_t \leq H_t \cdot \nu_t, \forall t \in [T], \quad (19)$$

$$\bar{\xi}_t \leq \bar{H}_t \cdot (1 - \nu_t), \forall t \in [T], \quad (20)$$

$$\xi_t = \bar{\xi}_t + \sum\nolimits_{d=1}^{D} w_d^{(t)} \sum\nolimits_{i=1}^{I_d} a_{d,i} \cdot \pi_{d,i} + F_t, \forall t \in [T], \quad (21)$$

where F_t, H_t, and $\bar{H}_t$ are constants such that $F_t = w^{(t)}\hat{x} + b^{(t)}$, $H_t \geq \max_{a \in \mathcal{A}} w^{(t)}(x + a) + b^{(t)}$, and $\bar{H}_t \geq -\min_{a \in \mathcal{A}} w^{(t)}(x + a) + b^{(t)}$. The variable ν_t indicates whether $w^{(t)}(x+a)+b^{(t)}$ is positive, and $\bar{\xi}_t$ represents negative part of $w^{(t)}(x+a)+b^{(t)}$. Note that the above constraints can be extended to a general MLP with more than two hidden layers (Serra, Tjandraatmadja, and Ramalingam 2018).

Overall Formulation

Finally, we show our overall formulation as follows:

$$\begin{aligned}
\text{minimize} \quad & \textstyle\sum_{d=1}^{D} \sum_{i=1}^{I_d} \sum_{k=1}^{K} c_{d,i} \pi_{d,i}^{(k)} + \gamma \cdot \sum_{k=1}^{K} \zeta_k \\
\text{subject to} \quad & \text{Constraint (1–12)}, \\
& \begin{cases} \text{Constraint (13)}, & \text{if } H \text{ is a LM}, \\ \text{Constraint (14–17)}, & \text{if } H \text{ is a TE}, \\ \text{Constraint (18–21)}, & \text{if } H \text{ is a MLP}, \end{cases} \\
& \pi_{d,i}^{(k)} \in \{0, 1\}, \forall k \in [K], \forall d \in [D], \forall i \in [I_d], \\
& \sigma_{k,d} \in \{0, 1\}, \forall k \in [K], \forall d \in [D], \\
& \delta_{k,d}, \zeta_k \in \mathbb{R}, \forall k \in [K], \forall d \in [D].
\end{aligned} \quad (22)$$

As with the existing MILO-based methods (Ustun, Spangher, and Liu 2019; Russell 2019; Kanamori et al. 2020), our formulation can be (i) handled by off-the-shelf

MILO solvers, such as CPLEX (IBM 2018), and (ii) customized by additional user-defined constraints, such as one-hot encoded categorical features and hard constraints for ordering (e.g., *precondition* (Ramakrishnan, Lee, and Albarghouthi 2020)). In summary, we can obtain ordered actions that satisfy user-defined constraints without implementing designated algorithms.

For computational complexity, Problem (22) includes K times more variables and constraints than in the unordered one. Thus, solving (22) is equal to or mode difficult than unordered ones. However, in the context of CE, sparse actions are preferred from the perspective of interpretability (Wachter, Mittelstadt, and Russell 2018). Therefore, it is sufficient to choose small K for obtaining sparse actions.

Post-processing and Partially Ordered Actions

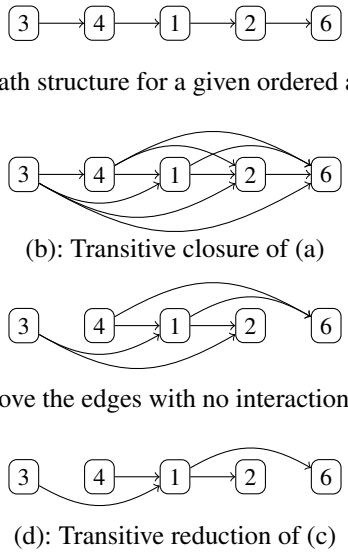
When some perturbing features have no interaction, changing the order of such features does not affect the cost C_{ord}. For example in Figure 1(b), the cost of the ordered action ["Education" → "WorkPerDay" → "JobSkill"] is the same as that of ["WorkPerDay" → "Education" → "JobSkill"] because "WorkPerDay" has no effect to the others. Thus the ordered action can be reduced to the partially ordered action ["WorkPerDay"] and ["Education" → "JobSkill"]. Suggesting such a partial order helps a user to execute an ordered action. We provide a post-processing algorithm that computes a partial order of the perturbing features from an ordered action and an interaction matrix.

An ordered action may be reduced to a *partially ordered action*, which is a pair $(a, \leq)$ of a perturbation vector $a \in \mathcal{A}$ and a partial order $\leq$ on $\mathrm{supp}(a)$. Here, we give a procedure to construct a partially ordered structure $\leq$ from an interaction matrix M and an ordered action (a, σ). If a perturbing order σ' of a is consistent with the obtained partial order $\leq$, the ordered action (a, σ') has the same ordering cost C_{ord} with that of (a, σ).

An ordered action can be expressed in the form of a path structure like Figure 2(a), where each node indicates a perturbing feature. In this path structure, changing a feature i should be executed after changing all its ancestors $\mathrm{pa}(i)$ and not after any its descendant. A partially ordered action is expressed in the form of a DAG like Figure 2(d), and a change in a feature satisfies the same condition as above. Note that even if a causal DAG is given, the DAG of a partially ordered action is not necessarily a subgraph of the causal DAG.

Algorithm We give an algorithm to obtain a partially ordered action from an ordered action and an interaction matrix M. The procedure is as follows:

1. Construct a path that represents the perturbing order of a given ordered action.
2. Compute the transitive closure that represents the total order of the perturbing features.
3. Remove an edge from i to j if there is no interaction between i and j, that is, $M_{i,j} = M_{j,i} = 0$.
4. Compute the transitive reduction that represents a partially ordered action.



(a): Path structure for a given ordered action

(b): Transitive closure of (a)

(c): Remove the edges with no interaction from (b)

(d): Transitive reduction of (c)

Figure 2: Algorithm for obtaining a partially ordered action.

Here, a *transitive closure* of a directed graph G is a directed graph that has an edge from i to j if and only if there is a directed path from i to j in G. Also, a *transitive reduction* of a directed graph G is a directed graph with the fewest number of edges whose transitive closure is the same as the transitive closure of G. For a finite DAG, its transitive closure and its transitive reduction are uniquely determined and can be computed in polynomial time (Munro 1971; Aho, Garey, and Ullman 1972; Gries et al. 1989).

We show that our algorithm can output a desired partial order of perturbing features for a given ordered action. For this purpose, we see that the ordering cost C_{ord} of an ordered action only depends on its partially ordered structure obtained by the above procedure. If i is not an ancestor of j on the DAG of a partially ordered action, then $M_{i,j} = M_{j,i} = 0$. Thus, the actual perturbation in the k-th step is calculated as follows:

$$\begin{aligned}
&\Delta^{(k)}(a_{\sigma_1,\ldots,\sigma_k} \mid M) \\
&\quad = a_{\sigma_k} - \sum\nolimits_{l=1}^{k-1} M_{\sigma_l,\sigma_k} \cdot \Delta^{(l)}(a_{\sigma_1,\ldots,\sigma_l} \mid M) \\
&\quad = a_{\sigma_k} - \sum\nolimits_{\substack{l=1,\ldots,k-1 \\ \sigma_l \in \mathrm{pa}(\sigma_k)}} M_{\sigma_l,\sigma_k} \cdot \Delta^{(l)}(a_{\sigma_1,\ldots,\sigma_l} \mid M),
\end{aligned}$$

where $\mathrm{pa}(j) \subseteq \mathrm{supp}(a)$ denotes the set of ancestors of a feature j on the DAG of the partially ordered action. That is, the ordered action (a, σ') has the same ordering cost C_{ord} as that of (a, σ) if σ' is consistent with the partially ordered structure obtained from (a, σ).

Experiments

In this section, we conducted experiments on real datasets to investigate the effectiveness and behavior of our **OrdCE**. All the code was implemented in Python 3.7 with scikit-learn and IBM ILOG CPLEX v12.10[1]. All the experiments

[1]All the code is available at https://github.com/kelicht/ordce.

C_{dist}	Dataset	Logistic Regression		Random Forest		Multilayer Perceptron	
		Greedy	**OrdCE**	**Greedy**	**OrdCE**	**Greedy**	**OrdCE**
TLPS	FICO	3.96 ± 2.6	$\mathbf{3.27 \pm 2.1}$	3.26 ± 2.9	$\mathbf{3.22 \pm 3.2}$	3.35 ± 3.0	$\mathbf{1.57 \pm 1.4}$
	German	4.9 ± 5.8	$\mathbf{4.81 \pm 5.7}$	3.23 ± 2.9	$\mathbf{3.2 \pm 2.9}$	5.38 ± 4.7	$\mathbf{5.03 \pm 4.5}$
	WineQuality	1.78 ± 1.8	$\mathbf{1.57 \pm 1.5}$	0.901 ± 0.55	$\mathbf{0.875 \pm 0.52}$	0.969 ± 0.83	$\mathbf{0.761 \pm 0.61}$
	Diabetes	2.91 ± 2.5	$\mathbf{2.47 \pm 2.0}$	2.3 ± 1.8	$\mathbf{2.26 \pm 1.8}$	1.12 ± 1.5	$\mathbf{0.668 \pm 0.99}$
DACE	FICO	10.6 ± 7.3	$\mathbf{9.61 \pm 6.7}$	6.78 ± 4.8	$\mathbf{6.67 \pm 4.7}$	3.5 ± 3.5	$\mathbf{3.41 \pm 3.3}$
	German	6.19 ± 5.3	$\mathbf{5.88 \pm 4.9}$	5.54 ± 4.6	$\mathbf{5.42 \pm 4.5}$	7.0 ± 5.8	$\mathbf{6.7 \pm 5.4}$
	WineQuality	2.93 ± 2.0	$\mathbf{2.42 \pm 1.6}$	1.65 ± 1.2	$\mathbf{1.51 \pm 1.1}$	1.91 ± 1.5	$\mathbf{1.66 \pm 1.3}$
	Diabetes	2.56 ± 1.7	$\mathbf{2.43 \pm 1.6}$	2.38 ± 1.7	$\mathbf{2.21 \pm 1.6}$	0.832 ± 1.2	$\mathbf{0.766 \pm 1.1}$

(a) Objective Function C_{OrdCE}

C_{dist}	Dataset	Logistic Regression		Random Forest		Multilayer Perceptron	
		Greedy	**OrdCE**	**Greedy**	**OrdCE**	**Greedy**	**OrdCE**
TLPS	FICO	2.21 ± 1.6	$\mathbf{1.33 \pm 0.87}$	1.72 ± 1.5	$\mathbf{1.49 \pm 1.2}$	3.05 ± 2.8	$\mathbf{0.84 \pm 0.74}$
	German	1.85 ± 1.5	$\mathbf{1.71 \pm 1.4}$	1.5 ± 1.2	$\mathbf{1.47 \pm 1.2}$	2.27 ± 1.8	$\mathbf{1.76 \pm 1.6}$
	WineQuality	1.0 ± 0.98	$\mathbf{0.765 \pm 0.59}$	0.475 ± 0.31	$\mathbf{0.439 \pm 0.29}$	0.69 ± 0.63	$\mathbf{0.446 \pm 0.35}$
	Diabetes	1.74 ± 1.6	$\mathbf{1.01 \pm 0.81}$	0.939 ± 0.67	$\mathbf{0.883 \pm 0.64}$	0.862 ± 1.3	$\mathbf{0.318 \pm 0.58}$
DACE	FICO	3.79 ± 2.6	$\mathbf{2.41 \pm 1.8}$	2.14 ± 1.5	$\mathbf{1.59 \pm 1.2}$	1.24 ± 1.2	$\mathbf{0.918 \pm 0.86}$
	German	1.92 ± 1.8	$\mathbf{1.43 \pm 1.1}$	1.6 ± 1.3	$\mathbf{1.46 \pm 1.2}$	2.23 ± 2.0	$\mathbf{1.87 \pm 1.5}$
	WineQuality	1.31 ± 0.94	$\mathbf{0.781 \pm 0.53}$	0.716 ± 0.54	$\mathbf{0.503 \pm 0.34}$	0.796 ± 0.69	$\mathbf{0.509 \pm 0.41}$
	Diabetes	1.2 ± 0.83	$\mathbf{1.02 \pm 0.7}$	1.13 ± 0.84	$\mathbf{0.912 \pm 0.67}$	0.425 ± 0.63	$\mathbf{0.322 \pm 0.47}$

(b) Ordering Cost Function C_{ord}

Table 3: Experimental results on the real datasets.

were conducted on 64-bit macOS Catalina 10.15.6 with Intel Core i9 2.4GHz CPU and 64GB memory, and we imposed a 300 second time limit for solving.

Experimental Setting

We randomly split each dataset into train (75%) and test (25%) instances, and trained ℓ_2-regularized logistic regression classifiers (LR), random forest classifiers (RF) with $T = 100$ decision trees, and two-layer ReLU network classifiers (MLP) with $T = 200$ neurons, on each training dataset. Then, we extracted ordered actions for test instances that had been received undesired prediction results, such as predicted as "high risk of default" from each classifier.

Distance-based Cost Functions As a distance-based cost function C_{dist}, we used four existing cost functions: the total log-percentile shift (TLPS) (Ustun, Spangher, and Liu 2019), weighted ℓ_1-norm of median absolute deviation (MAD) (Wachter, Mittelstadt, and Russell 2018; Russell 2019), ℓ_1-Mahalanobis' distance (DACE) (Kanamori et al. 2020), and distance based on structural causal models (SCM) (Mahajan, Tan, and Sharma 2019). The former two are *norm-based* cost functions that evaluate actions for each perturbing feature independently. The latter two are *interaction-aware* cost functions that evaluate actions by considering feature-correlation and causality, respectively.

Baseline Method To the best of our knowledge, there is no existing method that determines a minimum-cost perturbing order. Even if an ordered action is consistent with a given causal DAG, it is not necessarily optimal with respect to $C_{\mathrm{OrdCE}} = C_{\mathrm{dist}} + \gamma \cdot C_{\mathrm{ord}}$, since C_{ord} depends not only on the causal direction but also on the amount of the resultant perturbation. As a baseline, we proposed a greedy algorithm (**Greedy**), which consists of the following two steps:

1. Extract a perturbation vector $a^\star$ by optimizing C_{dist}.
2. Determine a perturbing order σ of $a^\star$ by solving the following optimization problem for k iteratively:

$$\sigma_k = \underset{d \in \mathrm{supp}(a^\star) \setminus \{\sigma_1, \ldots, \sigma_{k-1}\}}{\arg\min} \left| a^\star_d - \sum\nolimits_{l=1}^{k-1} M_{\sigma_l, d} \cdot \Delta^{(l)} \right|.$$

This procedure greedily selects a perturbing feature that has the smallest cost in each step.

To compare **OrdCE** with **Greedy**, we measured the average values of the distance-based cost C_{dist}, the ordering cost C_{ord}, and the objective function C_{OrdCE} over the ordered actions obtained by those two methods.

Experimental Results

We show experimental results here with TLPS and DACE owing to page limitation[2].

Comparison with Baseline We used four real datasets: FICO ($D = 23$) (FICO et al. 2018), German ($D = 40$),

[2]For the full results, see https://arxiv.org/abs/2012.11782.

Method	Order	Feature	Action	C_{dist}	C_{ord}
Greedy	1st	"BMI"	-6.25	**0.778**	0.828
OrdCE	1st	"Glucose"	-3.0	0.825	**0.749**
	2nd	"BMI"	-5.05		

(a) TLPS (Ustun, Spangher, and Liu 2019)

Method	Order	Feature	Action	C_{dist}	C_{ord}
Greedy	1st	"BMI"	-0.8	**0.716**	0.825
	2nd	"SkinTHK"	-2.5		
	3rd	"Glucose"	-8.5		
	4th	"Insulin"	-32.0		
OrdCE	1st	"Insulin"	-32.0	**0.716**	**0.528**
	2nd	"Glucose"	-8.5		
	3rd	"SkinTHK"	-2.5		
	4th	"BMI"	-0.8		

(b) DACE (Kanamori et al. 2020)

Table 4: Examples of ordered actions extracted from the RF classifier on the Diabetes dataset.

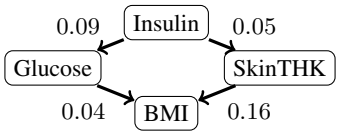


Figure 3: Subgraph of the causal DAG of the Diabetes dataset estimated by the DirectLiNGAM algorithm (Shimizu et al. 2011; Hyvärinen and Smith 2013).

WineQuality ($D = 12$), and Diabetes ($D = 8$) (Dua and Graff 2017) datasets, where D is the number of features. For German dataset, we transformed each categorical feature into a one-hot encoded vector. For each dataset, we estimated the adjacency matrix of a causal DAG by the DirectLiNGAM algorithm (Shimizu et al. 2011; Hyvärinen and Smith 2013), and computed an interaction matrix M from the adjacency matrix. We set $\gamma = 1.0$ and $K = 4$.

Table 3(a) and Table 3(b) present the average of the objective function C_{OrdCE} and the ordering cost C_{ord} for extracted ordered actions, respectively. From Table 3, we can observe that **OrdCE** always achieved lower C_{OrdCE} and C_{ord} than **Greedy** for all datasets and classifiers. Especially, in MLP and TLPS on FICO dataset, the averages of C_{OrdCE} and C_{dist} given by **OrdCE** are 1.57 and 0.84, respectively, which were less than half of those obtained by **Greedy**.

Next, we examine the ordered actions given by **OrdCE** to confirm the practicality. Table 4 presents examples of ordered actions extracted from the RF classifier on the Diabetes dataset, and Figure 3 presents a subgraph of the estimated causal DAG of the dataset. In both cases of TLPS and DACE, our **OrdCE** output ordered actions that accords with the directed edges in the causal DAG in Figure 3. On the other hand, the action extracted by **Greedy** with DACE, the order is not consistent with the causal DAG. From these results, we confirmed that **OrdCE** succeeded in obtaining

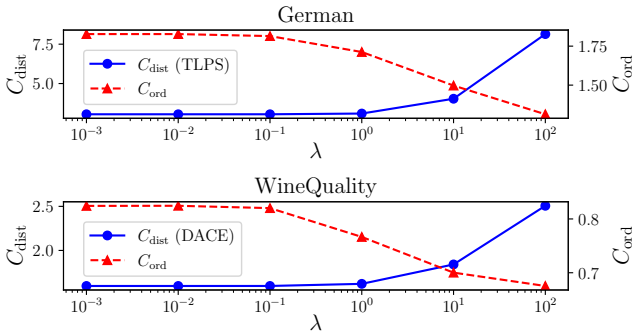


Figure 4: Sensitivity analyses of the trade-off parameter γ of OrdCE between the average C_{dist} and C_{ord}.

a reasonable perturbing order from the perspective of the causal DAG. In addition, for TLPS, the perturbation given by **OrdCE** is different from that given by **Greedy**. This difference is caused by the effect that **OrdCE** optimizes a perturbation vector and its order simultaneously.

Regarding the computation time, **OrdCE** was certainly slower than **Greedy** because **OrdCE** exactly solved Problem 1. In MLP and TLPS on FICO dataset, the average computation times of **OrdCE** and **Greedy** are 183 and 12.6 seconds, respectively. For other datasets, **OrdCE** is within 1.38–120 times slower than **Greedy**. However, Tables 3 and 4 indicate that **OrdCE** found better ordered actions in terms of C_{OrdCE} and C_{ord} than **Greedy** within 300 seconds, which is a reasonable computation time.

Sensitivity Analysis of Trade-off Parameter To examine the sensitivity of the trade-off parameter γ, we observed C_{dist} and C_{ord} of ordered actions extracted from LR classifiers by varying γ. The above (resp. below) figure in Figure 4 presents the average C_{dist} (resp. C_{ord}) for each γ on the German and WineQuality datasets. We can see trade-off relationship between C_{dist} and C_{ord}. As mentioned in (Wachter, Mittelstadt, and Russell 2018; Mothilal, Sharma, and Tan 2020), suggesting multiple actions is helpful for diversity. By varying γ, we can obtain several distinct ordered actions that have diverse characteristics in terms of C_{dist} and C_{ord}.

Conclusion

We proposed Ordered Counterfactual Explanation (OrdCE) that provides an optimal pair of a perturbation vector and an order of the features to be perturbed. We introduced a new objective function that evaluates the required cost of a perturbation vector and an order, and proposed a MILO formulation for optimizing it. By experiments on real datasets, we confirmed the effectiveness of our method by comparing it with a greedy method. As future work, we plan to conduct user-experiments to evaluate the usability of our OrdCE. In this study, while we assume the causal relationship is linear, our cost function has a potential to deal with non-linear relationships, which sometimes appear in the real world (Pearl 2009). Therefore, it is also interesting future work to develop a method for optimizing our cost function with non-linear causal relationships.

Acknowledgments

We wish to thank Kunihiro Wasa and Kazuhiro Kurita for making a number of valuable suggestions. We also thank anonymous reviewers for their insightful comments. This work was supported in part by JSPS KAKENHI Grant-in-Aid for JSPS Research Fellow 20J20654, Scientific Research (A) 20H00595, and JST CREST JPMJCR18K3.

References

Aho, A. V.; Garey, M. R.; and Ullman, J. D. 1972. The Transitive Reduction of a Directed Graph. *SIAM Journal on Computation* 1(2): 131–137.

Bertsimas, D.; Delarue, A.; Jaillet, P.; and Martin, S. 2019. Optimal Explanations of Linear Models. *arXiv,* arXiv:1907.04669 .

Breiman, L. 2001. Random Forests. *Machine Learning* 45(1): 5–32.

Cui, Z.; Chen, W.; He, Y.; and Chen, Y. 2015. Optimal Action Extraction for Random Forests and Boosted Trees. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 179–188.

Dhurandhar, A.; Chen, P.-Y.; Luss, R.; Tu, C.-C.; Ting, P.; Shanmugam, K.; and Das, P. 2018. Explanations Based on the Missing: Towards Contrastive Explanations with Pertinent Negatives. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 590–601.

Doshi-Velez, F.; and Kim, B. 2017. Towards A Rigorous Science of Interpretable Machine Learning. *arXiv,* arXiv:1702.08608 .

Dua, D.; and Graff, C. 2017. UCI Machine Learning Repository. URL http://archive.ics.uci.edu/ml. Accessed Sep. 9th, 2020.

FICO; Google; Imperial College London; MIT; University of Oxford; UC Irvine; and UC Berkeley. 2018. Explainable Machine Learning Challenge. URL https://community.fico.com/s/explainable-machine-learning-challenge. Accessed Sep. 9th, 2020.

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep Learning*. MIT Press.

Gries, D.; Martin, A. J.; van de Snepscheut, J. L.; and Udding, J. T. 1989. An algorithm for transitive reduction of an acyclic graph. *Science of Computer Programming* 12(2): 151 – 155.

Hastie, T.; Tibshirani, R.; and Friedman, J. H. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition*. Springer Series in Statistics. Springer.

Hyvärinen, A.; and Smith, S. M. 2013. Pairwise Likelihood Ratios for Estimation of Non-Gaussian Structural Equation Models. *Journal of Machine Learning Research* 14(1): 111–152.

IBM. 2018. CPLEX Optimizer — IBM. URL https://www.ibm.com/analytics/cplex-optimizer. Accessed Sep. 9th, 2020.

Janzing, D.; Balduzzi, D.; Grosse-Wentrup, M.; and Schölkopf, B. 2013. Quantifying causal influences. *Annals of Statistics* 41(5): 2324–2358.

Kanamori, K.; Takagi, T.; Kobayashi, K.; and Arimura, H. 2020. DACE: Distribution-Aware Counterfactual Explanation by Mixed-Integer Linear Optimization. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, 2855–2862.

Karimi, A.-H.; Barthe, G.; Balle, B.; and Valera, I. 2020a. Model-Agnostic Counterfactual Explanations for Consequential Decisions. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*, volume 108, 895–905.

Karimi, A.-H.; von Kügelgen, J.; Schölkopf, B.; and Valera, I. 2020b. Algorithmic recourse under imperfect causal knowledge: a probabilistic approach. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 265–277.

Koh, P. W.; and Liang, P. 2017. Understanding Black-Box Predictions via Influence Functions. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, 1885–1894.

Lundberg, S. M.; and Lee, S.-I. 2017. A Unified Approach to Interpreting Model Predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 4765–4774.

Mahajan, D.; Tan, C.; and Sharma, A. 2019. Preserving Causal Constraints in Counterfactual Explanations for Machine Learning Classifiers. In *CausalML: Machine Learning and Causal Inference for Improved Decision Making Workshop, NeurIPS 2019*.

Miller, T. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence* 267: 1 – 38.

Moore, J.; Hammerla, N.; and Watkins, C. 2019. Explaining Deep Learning Models with Constrained Adversarial Examples. In *Proceedings of the 16th Pacific Rim International Conference on Artificial Intelligence*, 43–56.

Mothilal, R. K.; Sharma, A.; and Tan, C. 2020. Explaining Machine Learning Classifiers through Diverse Counterfactual Explanations. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 607–617.

Munro, I. 1971. Efficient determination of the transitive closure of a directed graph. *Information Processing Letters* 1(2): 56 – 58.

Nau, D.; Ghallab, M.; and Traverso, P. 2004. *Automated Planning: Theory & Practice*. Morgan Kaufmann Publishers Inc.

Pearl, J. 2009. *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2nd edition.

Poyiadzi, R.; Sokol, K.; Santos-Rodriguez, R.; De Bie, T.; and Flach, P. 2020. FACE: Feasible and Actionable Counterfactual Explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 344–350.

Ramakrishnan, G.; Lee, Y. C.; and Albarghouthi, A. 2020. Synthesizing Action Sequences for Modifying Model Decisions. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*, 5462–5469.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1135–1144.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2018. Anchors: High-Precision Model-Agnostic Explanations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 1527–1535.

Russell, C. 2019. Efficient Search for Diverse Coherent Explanations. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 20–28.

Serra, T.; Tjandraatmadja, C.; and Ramalingam, S. 2018. Bounding and Counting Linear Regions of Deep Neural Networks. In *Proceedings of the 35th International Conference on Machine Learning - Volume 80*, 4558–4566.

Shimizu, S.; Inazumi, T.; Sogawa, Y.; Hyvärinen, A.; Kawahara, Y.; Washio, T.; Hoyer, P. O.; and Bollen, K. 2011. DirectLiNGAM: A Direct Method for Learning a Linear Non-Gaussian Structural Equation Model. *Journal of Machine Learning Research* 12: 1225–1248.

Ustun, B.; Spangher, A.; and Liu, Y. 2019. Actionable Recourse in Linear Classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 10–19.

Wachter, S.; Mittelstadt, B.; and Russell, C. 2018. Counterfactual Explanations Without Opening the Black Box: Automated Decisions and the GDPR. *Harvard Journal of Law & Technology* 31: 841–887.