Generating Algorithmic Recourse Keeping in Mind User Preferences

A

Thesis Submitted

in Partial Fulfillment of the Requirements

for the Degree of

MASTER OF TECHNOLOGY

By

SOUMYA SARKAR

Under the Supervision of

Dr. SHWETA JAIN

Department of Computer Science & Engineering

Indian Institute of Technology Ropar

May 8, 2024

Declaration

This is to certify that the thesis entitled "**Generating Algorithmic Recourse Keeping in Mind User Preferences**", submitted by me to the *Indian Institute of Technology Ropar*, for the award of the degree of Master of Technology, is a bonafide work carried out by me under the supervision of Dr. Shweta Jain. The content of this thesis, in full or in parts, have not been submitted to any other University or Institute for the award of any degree or diploma. I also wish to state that to the best of my knowledge and understanding nothing in this report amounts to plagiarism.

Sign:____

Soumya Sarkar
Department of Computer Science & Engineering,
Indian Institute of Technology Ropar,
Rupnagar-140001, Punjab, India.

Date:____

Certificate

This is to certify that the thesis entitled "**Generating Algorithmic Recourse Keeping in Mind User Preferences**", submitted by Soumya Sarkar (2022CSM1013), a master's student in the *Department of Computer Science & Engineering*, *Indian Institute of Technology Ropar*, for the award of the degree of Master of Technology, is a record of an original research work carried out by the candidate under my supervision and guidance. The thesis has fulfilled all requirements as per the regulations of the institute and in my opinion has reached the standard worthy of the award of the degree. The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Sign:____

Supervisor: Dr. Shweta Jain
Department of Computer Science & Engineering,
Indian Institute of Technology Ropar,
Rupnagar-140001, Punjab, India.

Date:____

Acknowledgements

I would like to thank, first and foremost, my guide Dr. Shweta Jain for guiding me in this work. I would also like to thank my friends and family for not just helping me in times of need, but also keeping me going even during difficult times.

Sincerely
Soumya Sarkar

Abstract

Black-box models such as deep neural networks have found increasing level of usage and control over critical decisions for major companies in essential domains such as banks, government bodies, etc. In such a scenario, it is critical to know how such models take decisions in order to validate the correctness and fairness of the model, as well as to tend to individual requests in response to a negative classification. Explainable AI (XAI) techniques have been used to deal with such scenarios and have gained prominence of late. One such XAI technique which uses "what if?" scenarios to explain a decision are called counterfactual. Using counterfactuals not only allows one to get explanations, but to get suggestions to improve the current scenario. Several works have tried to address this issue, but most of them do not consider user preferences, which can result in the suggestions being less personalized to the user. In this work, we try to generate counterfactuals keeping in mind user preferences and try to compare it with some baselines.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

BLACK BOX MODELS have found increasing usage in our day-to-day lives in a diverse variety of domains such as banking [8], agriculture [9], genomic prediction [10], among others. More importantly they are being increasingly trusted to make business-critical decisions that have real world effects. One such usage is the rise in number of black-box models being employed to automate the process of loan acceptance and rejection [11]. While this may be convenient for the entity employing such models, it leads to new kind of issues for end users. For example, it can lead to introduction of discriminatory behaviour against loan applicants [12] and job applicants [13], biased advertisements [14], and racial inequality in criminal justice [15].

As such, there is a need to know how such models work. In this aspect there has been an increasing call for methods to provide explanations for such decisions. This has led to the increasing popularity of Explainable AI techniques such as SHAP [16], LIME [17], etc. They can either give explanations on an individual basis (local explanations), or try to explain the working of a model as a whole (global explanations).

One interesting way of tackling such explanations is using "what if?" scenarios. For example, one explanation for why an applicants resume got rejected by a firm's model could be of the form "If you had worked for 2 years more and attained certification in Azure". This is a popular paradigm of thinking known as counterfactual thinking. It has long been studied in philosophy [18] and has found applications in economics [19], game theory [20], law [21], etc.

One of the most recent applications have been to use it to generate explanations for black-box models, and it has been shown not only to be an effective and intuitive way to explain outcomes, but also in compliance with regulations regarding black box models introduced in Europe (known as the EU General Data Protection Regulation or GDPR)[22].

1.1 Algorithmic Recourse

Counterfactuals can be used to provide suggestions which can help overturn the decision made by the black box system. This is known as algorithmic recourse and has many potential applications in order to alleviate the situations of many end users and applicants.

While generating algorithmic recourse, several things need to be kept in mind:

- Several features cannot be modified. For example, telling a 40 year person to change their birthplace to Canada is not a viable suggestion. These are called immutable features.
- There can be situations where features can only be modified in an increasing/decreasing function. For example, age can only increase.

There have been several methods which have been formulated to generate such algorithmic recourse, which we discuss in detail in Chapter 2.

1.2 A Need for Causality

Recent research by Karimi [3] has suggested that causality is an important consideration when it comes to such algorithmic recourses - if we fail to consider causality, then we can end up with sub-optimal recourse.

Causality has long been a topic of debate among statisticians, and for good reason - causality is much harder to measure than correlation. Till date randomized trials are the only way to generate data for true causal inference. While such a negative outlook towards causation stems from people like Pearson abandoning causality as a mere nuance in favour of the easier to use correlation, Judea Pearl argues that causal relationships are the fundamental building blocks both of physical reality as well as the human understanding of that reality [23].

In recent times, causality has found an increasing amount of usage in various fields, including machine learning [24], medicine [25], crime studies [26], economics [27], and many others. Of particular interest are its applications in machine learning, leading to more interpretable outcomes and models [28] and finding signifcant usage in Explainable AI techniques.

1.3 Personalized Recommendations

Recourse is useful when it provides the least costly solution to overturning decisions. However, cost is something that is uniquely personalized, and it is important that a blanket solution is not offered to every person seeking recourse. Yadav [5] says that a global cost function is likely to poorly represent different parts of a diverse population. Thus, it is necessary to elicitate the preference of the user and generate recourse accordingly.

Preference learning has long been studied in recommender systems, with varying approaches either tracking user's past history using knowledge-based recommender systems [29], or by using cold start techniques [30]. Cold start refers to the situation where we have new users for which we do not have any past data. This is particularly relevant to recourse generation, as users tend to be one-time users only, and they may not provide any details about themselves in keeping with privacy concerns. This usually results in recourse methods preferring to elicit preferences from users using choice [31]. The concept is grounded on choice and utility theory, where real choices replace ratings as the key data to learn the decision-maker's preferences as well as to make recommendations.

1.4 Duelling Bandits

One of the most popular ways of taking user choice is via pairwise comparisons. Bradley-Terry models [32] are, in general, the most popular choice when it comes to preference elicitation via pairwise comparisons due to its simple construction and useful properties.

Duelling bandits [7] uses either the Bradley-Terry model or Gaussian models and uses it to choose the best arm in a k-arm bandit problem. It is a useful technique when it comes to choosing the best in a list of items (arms) and shows theoretical guarantees in terms of user regret.

1.5 Our Work

To counter the problem of effectively incorporating user preferences in our recourse generation, we propose a novel method based on the duelling bandit concept which returns the best subset of features that a user would like to change. In keeping with the idea behind generalized additive independence, which states that the contribution of each attribute to the utility of a choice is affected by the attribute subset being considered [33], we return best subset instead of just list of features, as subset elements are a more realistic expression of latent human preferences than mere choices.

In our context, this is easily true if we consider causality. For example, one might prefer to change the education status from a M.Tech to a PhD if no other factors are considered. But if one considers family income levels and age, then the preference instead becomes joining a job and saving money. Thus, the subset ['job', 'save money] becomes the preferred choice over ['get a PhD'] in such a context.

Another example could be that one could prefer chocolate icecream over vanilla, but if we consider a pairing with chocolate waffles, one could end up flipping the choice entirely, as chocolate icecream on top of chocolate waffles might be "too chocolatey".

This shows a subset concept is more realistic a choice for modelling user preferences. We try to estimate the best subset for the user using pairwise comparisons of recourses generated using these subsets, keeping in mind causal changes. These causal changes are assumed to be natural effects of our actual actions, and thus they should be "effortless", and thus contribute no cost to the end result.

We compare our results with some baselines and go more in-depth in our methodology in the following chapters.

Chapter 2

Related Works

This chapter outlines the existing literature on the topic of algorithmic recourse and preference elicitation, highlighting some specific papers in detail.

2.1 Current Literature

Counterfactual as an XAI technique had been mooted since the landmark paper by Wachter [22]. It takes a very intuitive approach to choosing the cost of counterfactuals trying to minimize the amount of change along with the classification loss. To find the counterfactual x' with classification of $\mathrm{f}_{\mathrm{w}}(\mathrm{x}')$ for some data point x_{i} with the label y_{i} which we want to change to the new target y', we minimize the following objective:

$$\arg\min_{\mathrm{x}'}\max_{\lambda} \lambda\mathbb{L}(\mathrm{f}_{\mathrm{w}}(\mathrm{x}') - \mathrm{y}') + \mathrm{d}(\mathrm{x}_{\mathrm{i}}, \mathrm{x}') \tag{2.1}$$

where $\mathrm{d}(.,.)$ is a measure of distance and $\mathbb{L}(.,.)$ is some classification loss function.

Since then there have been several works which have tried to generate counterfactuals and recourse in other ways. Ustun et al. [34], for example, defines counterfactual cost in terms of the number of changed features and formalizes recourse generation as an Integer

Linear Program. DICE [35] provides users with a set of counterfactuals while maintaining a balance between proximity and feature diversity, both of which are objectives which are distance-based.

The most important considerations in such works is in terms of the cost function and the corresponding optimization equation. As we show, many papers follow different methodologies.

Karimi's paper on algorithmic recourse [3] shows how causality is necessary, and ignoring it can lead to suboptimal recourse. Thus, we no longer can look at the changes as a whole, and must now think in terms of "actions", aka single feature changes.

Further works take this action-based approach to generate counterfactuals. In particular, Naumann et al.[6] shows how actions taken sequentially can lead to optimal counterfactual solutions. Another interesting work is by Rawal et al. [4] which derives a cost function for global counterfactual explanation.

When it comes to learning user preference, there are fewer works which tackle this issue. Yadav et al.[5] uses a cost function distribution using a convex combination of percentile shift as well as linear cost and feature preference scores. De Toni et al. [2] takes a different approach in that they ask the user iteratively which counterfactual they prefer, learning about their preferences along the way. This is the work that is closest to ours in scope, although it doesn't use an automated pipeline for it's causal models.

We also look at the original duelling bandits paper in section 2.8 since it is the source of inspiration for our algorithm.

2.2 Algorithmic Recourse based on User's Feature-order Preference [1]

This paper is about algorithmic recourse aka counterfactual explanations that consider user preferences and try to change or modify features based on users ranking of the features. This is done by optimising an objective function that takes classification loss as well as a function c(.,.) to ensure the resulting counterfactual is as close as possible to the original one i.e minimum number of features are modified, and their modification order is based on user preference. To do this they do |M| set of updates, where 1st set allows only top 1 feature to be changed, 2nd set allows top 2 features to be changed, and so on till |M| sets. List of valid counterfactuals in each set is maintained and the one with least number of violations gets returned.

It is compared to Wachter and WachterM methods, and showed that on their defined metrics, it gave:

- Best user preference score (UPS)
- Lowest cost of changing the output (least L_1 distance from original output)
- Least number of features changed
- Better validity aka percentage of users for which counterfactual was possible than WachterM

2.3 Personalized Algorithmic Recourse with Preference Elicitation [2]

This paper introduces PEAR- a new methodology to generate recourse with user preference. PEAR builds on insights using Bayesian Preference Elicitation to iteratively refine an estimate of the costs of actions. It does so by asking the target user to choose from a set of interventions called as the choice set. The interventions are computed by maximizing the Expected Utility of Selection, a measure of information gain accounting for uncertainty on both the cost estimate and the user's responses. PEAR integrates the user choice and feeds it into a Reinforcement Learning agent coupled with Monte Carlo Tree Search to quickly identify and narrow down promising recourse plans.

The basic idea is shown in the figure 2.1 below:

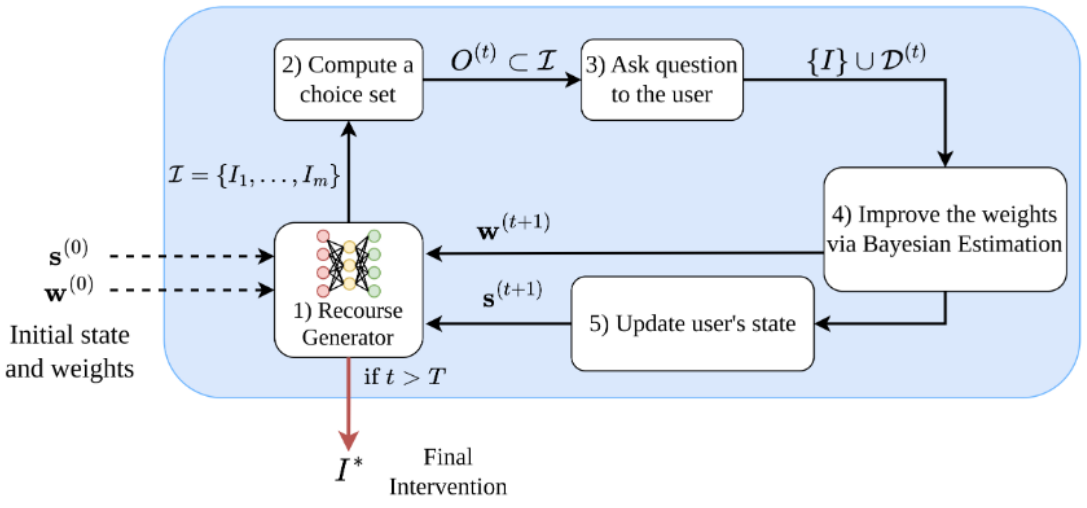

2) Compute a choice set
$O^{(t)} \subset \mathcal{I}$
3) Ask question to the user
$\{I\} \cup \mathcal{D}^{(t)}$
$\mathcal{I} = \{I_1, \dots, I_m\}$
4) Improve the weights via Bayesian Estimation
$\mathbf{w}^{(t+1)}$
$\mathbf{s}^{(0)}$
$\mathbf{w}^{(0)}$
Initial state and weights
1) Recourse Generator
$\mathbf{s}^{(t+1)}$
5) Update user's state
if $t > T$
I^*
Final Intervention

Figure 2.1: PEAR methodology

We initially provide the initial state[1] ($s^{(0)}$) and weights [2] ($w^{(0)}$), and from that a set of interventions - each I_i an ordered set of actions[3] $I_i = \{a^{(0)}, a^{(1)}, a^{(2)}, ...a^{(|I_i|)}\}$ from which counterfactuals can be generated - are created. Each I_i represents the set of actions needed to generate some counterfactual. From this a subset called choice set $O^{(t)}$ is created. Now the user selects the counterfactual they like best, then a new $w^{(t+1)}$ is computed using Bayesian Estimation, and user's state $s^{(t+1)}$ is updated. After T rounds the estimated weights are used to compute the final intervention I^*.

2.4 Algorithmic Recourse: from Counterfactual Explanations to Interventions [3]

The translation of counterfactual explanations to recourse actions was first explored by Ustun et al. [34], where additional feasibility constraints were levied to support the concept of actionable features (for example, asking the individual to reduce their age or change their race would not be allowed). While a step in the right direction, this work and others that followed implicitly assume that the set of actions resulting in the desired

[1]The user state $s \in S \subseteq \mathbb{R}^d$ is a vector of d categorical and real-valued features encoding, e.g., instruction level and income

[2]This is the vector of weights associated with the binary classifier $h : S \to \{0, 1\}$ such that given the user state s we get the undesirable classification $y = h(s)$. The goal of the counterfactual is to generate the state s' such that $y \neq h(s')$

[3]An action $a \in A$ is a map that takes a state s and changes a single feature, yielding a new state $s' = a(s)$ and expresses a recommendation of the form "Increase your income by $100."

output would directly follow from the counterfactual explanation. The authors of this paper challenge this assumption and show how existing approaches are inferior due to their lack of consideration for real-world properties - especifically the causal relationships which govern the world in which the actions will be performed. Two examples are given to this end.

Example 1: An individual has applied for a loan and was rejected. He has an annual earning of $\$75,000$ and account balance of $\$25,000$. The algorithm suggest increasing earning to $\$100,000$ ($+33\%$ increase) OR have a bank balance of $\$30,000$ ($+20\%$ increase). A better solution would be a salary increase to $\$85,000$ ($+14\%$ increase) because 30% of that salary is already being saved in the bank balance due to some actions in the real world.

Example 2: The algorithm suggests that to increase rice paddy output we increase the elevaton by some percent, withotu considering how that increase will affect the other variables (it assumes independence of the factors that affect paddy output).

Karimi defines the cost function as:

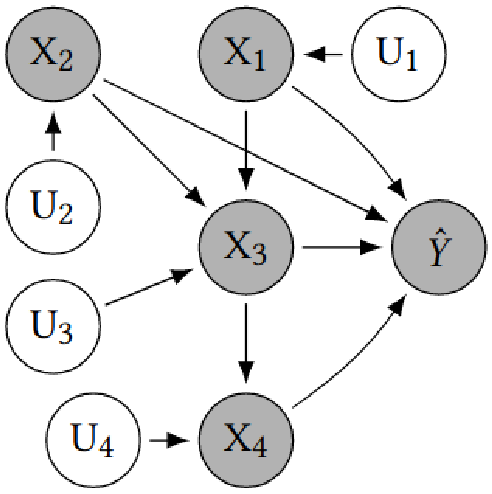
$$\begin{aligned} \mathrm{A}^* \in \arg\min_{\mathrm{A}} \;\; \mathrm{cost}(\mathrm{A};\mathrm{x}^{\mathrm{F}}) \;\; \text{s.t.} \;\; \mathrm{h}(\mathrm{x}^{\mathrm{SCF}}) \neq \mathrm{h}(\mathrm{x}^{\mathrm{F}}), \\ \mathrm{x}^{\mathrm{SCF}} = \mathbb{F}_{\mathrm{A}}(\mathbb{F}^{-1}(\mathrm{x}^{\mathrm{F}})), \\ \mathrm{x}^{\mathrm{SCF}} \in \mathcal{P}, \;\; \mathrm{A} \in \mathcal{F} \end{aligned} \tag{2.2}$$

where A refers to the set of actions, h refers to the model, $\mathrm{x}^{\mathrm{SCF}}$ refers to the structural counterfactual (aka the true nearest counterfactual based on causality constraints).

Thus, for a causal model which is known, and a set of feasible interventions, the optimization problem in Equation 2.2 yields algorithmic recourse through Minimal Interventions (MINT). Generating minimal interventions through solving 2.2 requires that we be able to compute the structural counterfactual, $\mathrm{x}^{\mathrm{SCF}}$, of the individual x^{F} in world $\mathcal{M}$, given any feasible action, A. To this end, the authors consider that the Structural Causal Model (SCM) $\mathcal{M}$ falls in the class of additive noise models (ANM)[4], so that one can deterministically compute the counterfactual $\mathrm{x}^{\mathrm{SCF}} = \mathbb{F}_{\mathrm{A}}(\mathbb{F}^{-1}(\mathrm{x}^{\mathrm{F}}))$ by performing the

[4]ANM have pairwise independent noise variables U_{i}

Abduction-Action-Prediction steps proposed by Pearl et al [36].



$$\left.\begin{aligned} X_1 &:= U_1 \\ X_2 &:= U_2 \\ X_3 &:= f_3(X_1, X_2) + U_3 \\ X_4 &:= f_4(X_3) + U_4 \end{aligned}\right\} \mathcal{M}$$

$$\hat{Y} = h\big(\{X_i\}_{i=1}^{4}\big)$$

FIGURE 2.2: Example of an SCM

They further propose to follow an optimisation problem which involves using "actions" aka single-feature changes, instead of changes to the counterfactual as a whole. This prevents suboptimal solutions and is more attuned to the real world. The paper, however, does not explain how to derive such a causal structure.

2.5 Beyond Individualized Recourse: Interpretable and Interactive Summaries of Actionable Recourses [4]

Most counterfactual algorithms provide solutions to local instances i.e., on an individual basis. But it is important to check if globally a model is fair and unbiased so that at a high-level the stakeholders are aware of what they are deploying and/or using. This is where this paper steps in. This is useful to decide if a model is good enough *before* deployment, unlike the local algorithms which work *after* deployment. In this work the authors propose a novel model-agnostic framework called Actionable Recourse Summaries (AReS) to learn global counterfactual explanations which can provide interpretable and accurate summary of recourses for the entire population with emphasis on specific sub-groups of interest (like race, sex, etc. - see Fig 2.3). The goal of their work is to enable decision makers to answer important questions about the generated recourse on a global scope. For example, how does the recourse differ across various racial subgroups? They also demonstrate theoretically that several of the prior approaches proposed to generate recourses for individuals are special cases of our framework. Furthermore, unlike prior research, they do not make the unrealistic assumption that one has access to real valued recourse costs. Instead, they develop a framework which leverages Bradley-Terry model to learn these costs from pairwise comparisons of features provided by end users. It also shows experimental results that work on global explanations as well as local explanations - the latter on par with SOTA baselines.

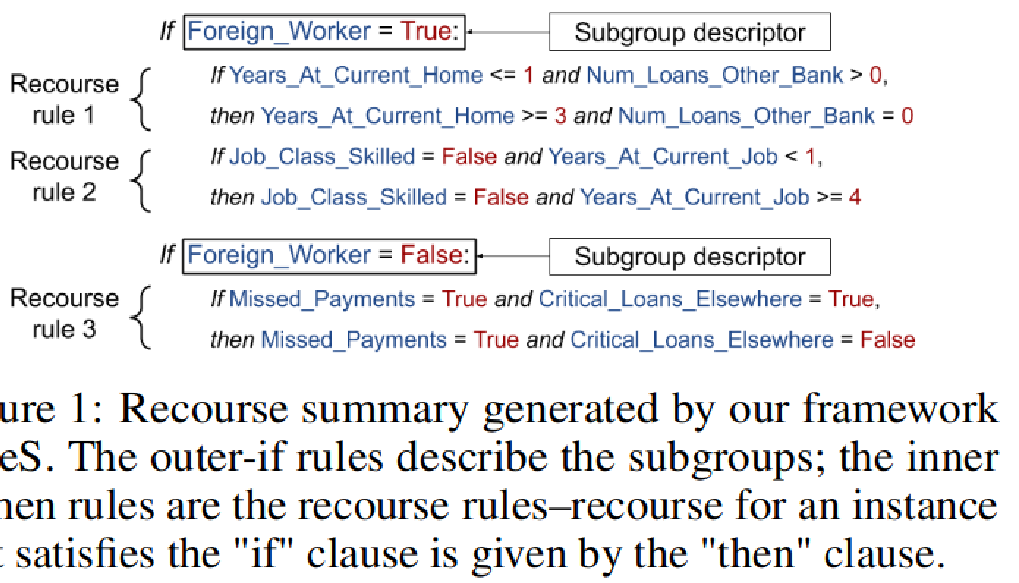

If Foreign_Worker = True:
Subgroup descriptor
Recourse rule 1
If Years_At_Current_Home <= 1 and Num_Loans_Other_Bank > 0,
then Years_At_Current_Home >= 3 and Num_Loans_Other_Bank = 0
Recourse rule 2
If Job_Class_Skilled = False and Years_At_Current_Job < 1,
then Job_Class_Skilled = False and Years_At_Current_Job >= 4
If Foreign_Worker = False:
Subgroup descriptor
Recourse rule 3
If Missed_Payments = True and Critical_Loans_Elsewhere = True,
then Missed_Payments = True and Critical_Loans_Elsewhere = False

Figure 1: Recourse summary generated by our framework AReS. The outer-if rules describe the subgroups; the inner if-then rules are the recourse rules–recourse for an instance that satisfies the "if" clause is given by the "then" clause.

FIGURE 2.3: Example of Recourse Rule

2.6 Low-cost Algorithmic Recourse For Users With Uncertain Cost Functions [5]

The authors argue that cost functions should be based on actionable recourses and must be tailored to the individual, rather than based on the entire populace. Even sparsity is not the end-all solution, and feature-diversity must be kept in mind. The assumption is that if users are provided with solutions that change different subset of features, then they are more likely to find at least one feasible solution.

In this work, they propose a way to identify a user-specific recourse set that contains at least one good solution for the user. As users' true cost functions are not known, they treat them as unknowns for the recourse method and assume they follow some underlying cost function distribution. They model this cost distribution using a highly flexible cost sampling procedure which makes as few assumptions as possible regarding user preferences. In addition, they also provide users with the option to specify their preferred editable features or even provide the complete cost function detailing the costs to transition between features values.

2.7 Consequence-aware Sequential Counterfactual Generation [6]

Even if a lot of the papers have moved on from previous distance-based approaches to action-based approaches, they still consider the impact of actions at the same time. This paper by Naumann and Ntoutsi shows how the ordering of the actions matter and show how counterfactuals generated by a sequential process improve the quality of the counterfactuals. One such example is given in figure 2.4

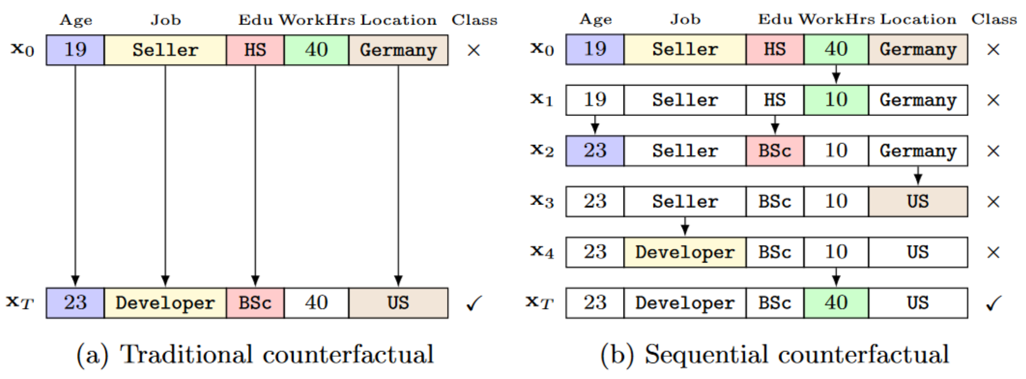

Age
Job
Edu
WorkHrs
Location
Class
x_0 19 Seller HS 40 Germany ×
x_T 23 Developer BSc 40 US ✓
(a) Traditional counterfactual
x_0 19 Seller HS 40 Germany ×
x_1 19 Seller HS 10 Germany ×
x_2 23 Seller BSc 10 Germany ×
x_3 23 Seller BSc 10 US ×
x_4 23 Developer BSc 10 US ×
x_T 23 Developer BSc 40 US ✓
(b) Sequential counterfactual

Fig. 1. The difference between traditional counterfactual generation (a) and the sequential approach (b). Although the generated counterfactual $\mathbf{x}_T$ is the same, the process and implied knowledge/information is different.

FIGURE 2.4: Example of Sequential Counterfactual Generation

They define the cost of an action a_i as consisting of two parts: $c_i(\cdot) = b_i(\cdot) \cdot g_i(\cdot)$, where b_i represents the direct effort, whereas g_i acts like a discount in order to incorporate the beneficial consequences of prior actions.

They formulate the problem as a multi-objective function:

$$\begin{aligned} \min_{\mathcal{S}}(& \underbrace{o_1}_{\text{Sequence Cost}}, \underbrace{o_2}_{\text{Gower Distance}}, \underbrace{o_{2+1}, o_{2+2}, o_{2+3}, ..., o_{2+d}}_{\text{Feature Tweaking frequencies}}) \\ & \text{s.t. } f(x_T) = \text{accept and } \bigwedge_{(a_i,v_i)\in\mathcal{S}} \mathbb{C}_i \end{aligned} \tag{2.3}$$

where sequence cost refers to the cost of the sequence, gower distance is a measure of the distance between the counterfactual and the original data point, and the third term is a representation of how much each feature changes - for example, if the change for 5 features is $1, 1, 1, 2, 1$ as in the case of figure 2.4, then in $(o_3, ..., o_7)$, o_{2+h} represents the number of times feature $\dddot{\mathcal{X}}_h$ changed i.e., $o_{2+h} = \#(\dddot{\mathcal{X}}_h \in \mathcal{I}_{a_i} \forall a_i \in \mathcal{S})$. They use a Biased Random Key Genetic Algorithm (BRKGA) with non-dominated sorting to solve this multi-objective function.

2.8 The K-armed Dueling Bandits Problem [7]

More often than not, in multi-armed bandit situations it is assumed the regret values can be calculated based on the utility of choosing each arm. However, often we face the situation of comparing arms which don't have utility values we can compare - rather, their comparison is qualitative. For example, choosing the best ice-cream flavour cannot be done in terms of quantitative values, but rather comparitively. Duelling bandits deals with this particular situation.

The authors postulate that using two kinds of models to calculate the probability of choosing an arm - the Bradley-Terry model as well as the Gaussian model - leads to several properties which are desirable such as strong stochastic transitivity as well as stochastic triangle inequality. They define the probability of choosing an arm $\hat{b}$ over another arm b as:

$$P(\hat{b} > b) = \epsilon(b, \hat{b}) + \frac{1}{2} \tag{2.4}$$

where $\epsilon(\mathrm{b}, \hat{\mathrm{b}}) \in (-\frac{1}{2}, \frac{1}{2})$.

Using this, they define two algorithms which uses pairwise comparisons and chooses the best arm among the k bandits. They show that their first algorithm IF1 with a time horizon of $\mathrm{T}(\mathrm{T} \geq \mathrm{K})$ achieves a regret bound of:

$$\mathbf{E}[R_T] \leq \mathcal{O}\left(\frac{K \log K}{\epsilon_{1,2}} \log \mathrm{T}\right) \tag{2.5}$$

Their second algorithm IF2, which uses a more aggressive pruning strategy, achieves a regret bound of:

$$\mathbf{E}[R_T] \leq \mathcal{O}\left(\frac{K}{\epsilon_{1,2}} \log \mathrm{T}\right) \tag{2.6}$$

Chapter 3

Methodology

Our work is the first work that provides an end-to-end system to generate counterfactuals keeping in mind user preference and causality. There are three distinct modules which we develop:

1. Preference Elicitation
2. Causal Discovery
3. Counterfactual Generation

In this section we describe each module in the upcoming sections, as well as describe the code in Section 3.4. The entire algorithm is also shown in Section 3.4.We also describe two alternate methods we use as a baseline in Section 3.5 and which were our earlier approaches to this problem.

3.1 Preference Elicitation

We employ the use of a modified version of the IF1 algorithm from the duelling bandits paper by Yue et al. [7]. The proposed algorithm is given below:

Algorithm 1: IFC1: User Preference using Duelling Bandit

Input: *Ft* All non-fixed features of size *N*, *T* time horizon, *model*, *x*

```
1  $\beta$ = Generate subsets of $Ft$
2  $\delta = \frac{1}{\mathrm{T}(2^{\mathrm{N}}-1)}$
3  Choose some random subset $\hat{\mathrm{b}}$
4  $\mathrm{W} = \beta - \hat{\mathrm{b}}$
5  Choose some random subset $\hat{\mathrm{b}}$
6  Initialize: Calculate $\hat{\mathrm{P}}_{\hat{\mathrm{b}},\mathrm{b}} = \mathrm{P}(\hat{\mathrm{b}} > \mathrm{b})$ and $\hat{\mathrm{C}}_{\hat{\mathrm{b}},\mathrm{b}}$
7  while W is not empty do
8      Next Candidate nc = None
9      foreach b in W do
10         Generate counterfactuals for $\hat{\mathrm{b}}$ and b subsets using →gen_ cf(model, x,
            features_ to_ vary)
11         Ask the user which one they prefer and update $\hat{\mathrm{T}} = \hat{\mathrm{T}} + 1$
12         Update $\hat{\mathrm{P}}_{\hat{\mathrm{b}},\mathrm{b}}$ and $\hat{\mathrm{C}}_{\hat{\mathrm{b}},\mathrm{b}}$ and calculate costs with reduction factor
13         if $\hat{\mathrm{P}}_{\hat{\mathrm{b}},\mathrm{b}} > 0.5$ and $\hat{\mathrm{C}}_{\hat{\mathrm{b}},\mathrm{b}} \geq 1 - \delta$ then
14             Remove b from W
15         end
16         else if $\hat{\mathrm{P}}_{\hat{\mathrm{b}},\mathrm{b}} < 0.5$ and $\hat{\mathrm{C}}_{\hat{\mathrm{b}},\mathrm{b}} \geq 1 - \delta$ then
17             if $\hat{\mathrm{P}}_{\hat{\mathrm{nc}},\mathrm{b}}$ then
18                 nc = b
19             end
20         end
21     end
22     if $\mathrm{nc} \neq \hat{\mathrm{b}}$ then
23         $\hat{\mathrm{b}} = \mathrm{nc}$
24         $\mathrm{W} = \mathrm{W} - \hat{\mathrm{b}}$
25         Recalculate $\hat{\mathrm{P}}_{\hat{\mathrm{b}},\mathrm{b}}$ and $\hat{\mathrm{C}}_{\hat{\mathrm{b}},\mathrm{b}}$ for remaining elements in W
26     end
27 end
28 Return: Best arm counterfactual and number of comparisons
```

AS this algorithm closely resembles the IF1 algorithm, we suspect that the regret guarantees for IF1 hold for IFC1 as well (C for counterfactual). Thus, we suspect that regret will be bounded by Equation 2.5.

When it comes to some specifics in the algorithm, such as updation of probabilities and confidence intervals, there was no mention of how such updation could be achieved. Keeping that in mind, we do a simple additive-based approach, where we add or subtract a constant quantity based on the user's response. If the user response doesn't match the probability i.e., $\mathrm{P}(\hat{\mathrm{b}} > \mathrm{b}) > 0.5$ but user chooses subset b, then we gate the probability to atmost 0.5. The constant quantity increases in magnitude as we repeat the same comparison again and again, ensuring such repeats are quickly dealt with.

The probabilities $\mathrm{P}_{\hat{\mathrm{b}},\mathrm{b}}$ are calculated using the Bradley-Terry model. We use the following Bradley-Terry formula:

$$\mathrm{P}(\hat{\mathrm{b}} > \mathrm{b}) = \frac{\exp(-|\hat{\mathrm{b}}|)}{\exp(-|\hat{\mathrm{b}}|) + \exp(-|\mathrm{b}|)} \tag{3.1}$$

We base it on length as we would like to favour smaller subsets, implying less amount of user-acted changes.

As with the IF1 algorithm, in order to ensure a close resemblance to it we use the same criteria for calculating $\hat{\mathrm{C}}_{\hat{\mathrm{b}},\mathrm{b}}$:

$$\hat{\mathrm{C}}_{\hat{\mathrm{b}},\mathrm{b}} = \sqrt{\frac{1}{\mathrm{t}}\log(\frac{1}{\delta})} \tag{3.2}$$

where t refers to the number of rounds of comparisons undertaken so far.

3.2 Causal Discovery

Causal discovery refers to the discovery of causal structures between features using historical data. While there are several methods capable of doing this, we choose the method given by Shimizu et al. [37] - ICA-based LiNGAM.

One of the primary benefits of using this method is that it always provides a Directed Acyclic Graph (DAG). This is beneficial since we always know the causal direction between any two features and there are no cycles, which would be detrimental to our approach. Another benefit is that it provides a simple linear model which means that causal inference (aka learning the effect of one feature on another) time would be less.

We utilise a python implementation of the model provided by causal-learn [38] and dowhy [39] packages in python.

3.3 Counterfactual Generation

Algorithm 2: gen_cf() Algorithm for generating counterfactuals

```
Input: model, x original datapoint, features_to_vary list of features which are
       allowed to be varied
1  Create a mask tensor based on which features need to be varies
2  while i<maximum iterations allowed do
3      Calculate the loss by using a combination of $L_2$ loss and classification loss
4      Backpropagate to find gradients and add them
5      foreach feature $f_1$ in features_to_vary do
6          foreach feature $f_2 \neq f_1$ do
7              Find causal effect $f_1 \rightarrow f_2$ with original value of $f_1$ as control and new
                value of $f_1$ as treatment
8          end
9      end
10 end
11 Return: Counterfactual and other metrics like cost
```

Manan et al. [1] utilizes a combination of the L_2 distance between the generated counterfactual and the original data point as well as the classification loss in order to calculate final loss. Going with that theme, we too utilize the same terms in our costs

In addition to that, we use a reduction factor (R_{factor}) which is used to discount the cost of a counterfactual (irrespective of its actual cost) based on how often the subset gets chosen by the user. We define it as approximately 85% of the original cost for each time it gets chosen in a pairwise comparison - there is an added factor which considers how close we get to the target prediction of 1.

However, in keeping with standard causal literature, we utilise an action-based approach. We calculate the cost of an action as the sum of the L_2 distance (but in this case, it is only with respect to a single feature being modified by the action) as well as the classification loss. It is important to note we do not include the overall L_2 distance since we actually only apply effort to change one single feature, and the causal effects of that change are assumed to come without any effort from the user. This is true in real life as well - for example, the effort needed by a person might be in converting his highest education degree from a bachelors to a masters, but the corresponding effect of increasing the job salary is not considered as an added cost, as it is a benefit of getting a masters degree.

The optimization equation thus becomes a matter of finding the least cost intervention as follows:

$$\begin{gathered} \mathcal{I}^* = \arg\min_{\mathcal{I}} \mathbb{C}(\mathcal{I}, x) \\ \text{such that } h(\mathcal{I}^*(x)) \neq h(x) \end{gathered} \tag{3.3}$$

where we define the cost function $\mathbb{C}(\mathcal{I}, x)$ as:

$$\begin{gathered} \mathbb{C}(\mathcal{I}, x) = \sum_{a_i \in \mathcal{I}} C(a_i, x) * R_{factor} \\ \text{where } C(a_i, x) = \lambda \|a_i(x_i) - x_i\|_2 + L_{classification}(a_i(x)) \\ \text{and } R_{factor} = 0.85(1 - 0.3 * y_{pred}) \end{gathered} \tag{3.4}$$

The method of optimizing this equation is mentioned in Section 3.4. This way of defining the loss closely follows the Wachter [22] loss function.

3.4 Code

Algorithm 3: Complete Algorithm

```
Input: index_x Index of datapoint, dataset, model
1 Do standard data-preprocessing on dataset
2 Find the datapoint x using index_x
3 Find best subset and counterfactual from IFC1(Ft, N, T, model, x)
4 Return: Best counterfactual and other metrics like cost, number of
    comparisons, etc.
```

Manan's work was a base paper for us, and thus we had to utilise it as a baseline for comparison. However, unfortunately, the code for Manan's paper was lost. The code, thus, had to be written from scratch.

In order to code Manan's paper, we used pytorch adn designed a modular codebase to utilise and call the algorithm. The idea is that the user would provide a preference order initially, and then using that via subset-wise calls we generate counterfactuals.

We used a neural network-free approach, choosing to modify the original source datapoint via gradient backflow. To affect the features selectively, we use gradient masking.

For our work, we tried to re-use the code given by Manan, but it led to plenty of issues. Two main issues kept cropping up:

- Vanishing gradients
- $\mathrm{y_{pred}} = 0$, leading to generation of NaN values.

We initially tried adding a default value of $\mathrm{y_{pred}} = 10^{-9}$, but as is the case with pytorch, this led to the computational graph being cutoff. The ultimate solution we came up involved perturbing the counterfactual for the current feature (since at a time only one feature got modified) till we reached some non-zero value or a limit on number of tries is reached.

To deal with vanishing gradients, we tried to manually inflate the gradient by adding a value derived from the uniform distribution $\mathrm{val_i} = \mathrm{U(0, L_{classification}(a_i(x)))}$ in the direction of the original gradient.

We take the additional step of flipping the gradient direction in case the loss increases from one iteration to the next.

These adjustments ensured that we got a guaranteed flip in the model classification, but it usually came at the cost of L_2 loss. This is why, as we show later on, we get costly counterfactuals.

The causal inference step is usually carried out after the gradients are added during backpropagation. We utilise a dictionary which helps to store causal relations that have been explored before, and uses it to calculate new causal relations. For example, if the relation "age"$\rightarrow$"savings" has been explored before, we utilise the previous values to calculate the new ones. This is made possible due to the linear nature of LiNGAM.

Let us study the causal effect of feature f_1 on f_2 as an example. Initially when we first explore the causal relation $f_1 \rightarrow f_2$, let f_1 take the counterfactual value of x_2 and its original value x_1 be the control value. Let the causal effect of f_1 on f_2 be y in this scenario. We then store this discovery in our dictionary. When we next encounter the causal relation $f_1 \rightarrow f_2$, where f_1 takes the counterfactual value of x_4 and its original value x_3 is the control, we can calculate the causal effect as shown below:

$$f_2 = y\frac{x_4 - x_3}{|x_2 - x_1|} \tag{3.5}$$

This optimisation leads to significant improvement in performance.

3.5 Other Ideas

3.5.1 Custom Method 1:

In this method, we use a regularization approach to the problem. For the Wachter cost function, we optimize as:

$$\arg\min_{x'}\max_{\lambda} \lambda\mathbb{L}(f_w(x') - y') + d(x_i, x') \tag{3.6}$$

where d(.,.) is a measure of distance. For the distance measure, we use L_1 or L_2 norm. In this approach, we use a weighted version of L_1 norm where the weights signify the importance of the feature i.e., its precedence.

$$d(x_i, x') = \sum_{i=0} \alpha_i \|x_i - x\|_1 \tag{3.7}$$

where $\alpha_{i+1} = c.\alpha_i$, and $\alpha_1 = \frac{\beta_0(c-1)}{c^n-1}$. Here c is known as the alpha multiplier, and $\sum_i \alpha_i = \beta_0$ is the sum of all alphas.

β_0 can have any value such as:

- $\beta_0 = 1$
- $\beta_0 = \max(1,\ \mathbb{L}(f_w(x') - y'))$
- $\beta_0 = \beta$ for some $\beta \in \mathbb{R}$

Here we use a multiplicative factor, but we could have used additive factor or exponent factors. For our experiments, we use as is mentioned in Equation 3.7.

3.5.2 Custom Method 2:

In this method, we use the same method as we used for preference learning, but we do not include causality in it. We treat it the same way as we would for our normal method.

Chapter 4

Experiments and Results

In this chapter we define some experiments we run to test our method and show the corresponding results and observations.

4.1 Experiment Description

4.1.1 Experiment on Synthetic Dataset

For comparison with some of the original baselines such as Manan et al. [1], we generate a synthetic dataset which involves the use of causality.

The dataset consists of 5 features X_1, X_2, X_3, X_4, and X_5 with the output as Y. They follow the causal graph shown in Figure 4.1 and are generated as per Equation 4.1.

Here, U_i refers to exogenous variables, whose causes are external and which explains the variables or outcomes of our model. We assume these external factors for each feature are independent of each other.

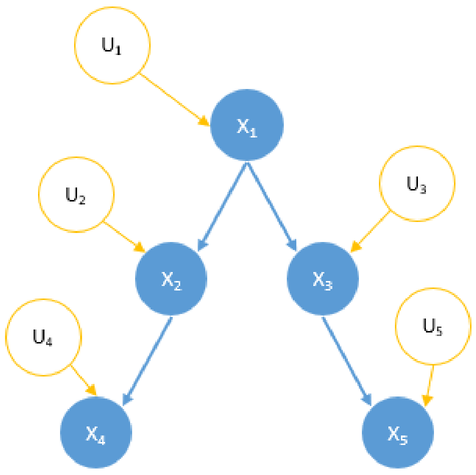


FIGURE 4.1: Structural Causal Model for Custom Dataset

$$\begin{aligned} X_1 &:= U_1 \\ X_2 &:= 2X_1 + U_2 \\ X_3 &:= 3X_1 + U_3 \\ X_4 &:= X_2 - 2X_3 + U_4 \\ X_5 &:= 2X_3 - 2X_1 + U_5 \end{aligned} \tag{4.1}$$

The output Y is given as below:

$$y = \begin{cases} 1 & \text{if sigmoid}(\sum_{i=1}^{5} x_i) \geq 0.5 \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

Hyperparamaters	**Custom Dataset**	**Give Me Credit**
Number of users	100	50
Size of dataset sampled	1000	5000
Learning Rate	10^{-3}	10^{-3}
Normalization	Z-score	Z-score
Maximum number of permutations of subset	3	3
(Custom 1 only) $\sum \alpha$	1000	1000
(Custom 1 only) α multiplier	100	100

TABLE 4.1: Hyperparameters of all experiments

We run this dataset on 4 methods:

- Manan's base work *(MANAN)* (Section 2.2)
- Manan's work with causality involved *(MANAN CAUSAL)*
- Custom method with different loss function *(CUSTOM 1)* (Section 3.5.1)

- Custom method which learns the preference similar to our method but does not use causality *(CUSTOM 2)* (Section 3.5.2)
- Our Method *(OUR METHOD)*

We do not assume any fixed features as Manan's method doesn't have provisions for it at the moment. We generate a randomized order everytime - for example, the ordering could be $X_5 > X_4 > X_2 > X_3 > X_1$ for one round, and $X_3 > X_4 > X_2 > X_1 > X_5$ for the next. For our method, we take a subset of these features. For example, if the order is $X_5 > X_4 > X_2 > X_3 > X_1$, then the possible user preferred subsets would be $\{X_5\}$, $\{X_5, X_4\}$, $\{X_5, X_4, X_3\}$, $\{X_5, X_4, X_3, X_2\}$ or $\{X_5, X_4, X_3, X_2, X_1\}$. It is to be noted that the strict ordering over these subsets does not hold, as our purpose is entirely different to what Manan's method tries to achieve. But we do choose a subset which respects the order somewhat as shown above - a subset like $\{X_5, X_1\}$ is not going to be chosen.

We provide the same hyperparameters for all the methods - these are mentioned in Table 4.1.

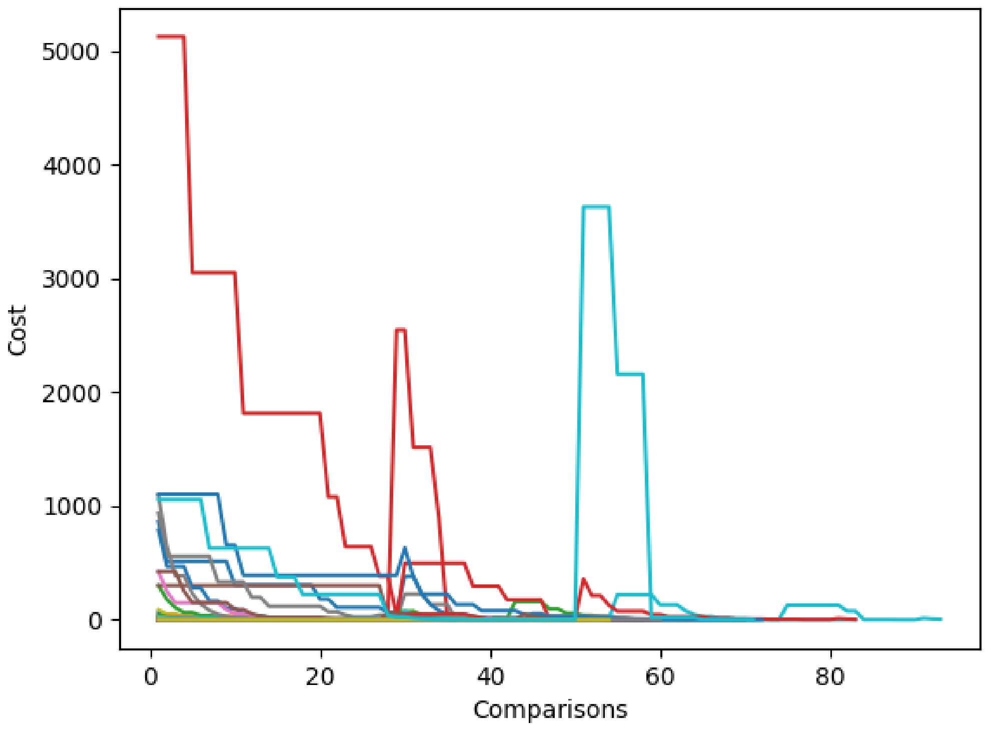


FIGURE 4.2: Counterfactual cost v/s Comparisons for Custom Dataset for Each User

4.2 Results

4.2.1 Custom Dataset

We show the results for all the methods in Table 4.2 below:

Paramater	**Manan**	**Manan with Causality**	**Custom 1**	**Custom 2**	**Our Method**
Average Number of Features Changed	1.08	1	5	1	3.42
Average L_2 Cost of Counterfactuals	14.007	896.582	1.14	21.54	6.34
Validity Percentage	100	100	80	100	50
Average Time to Execute (s)	1.222	1.331	0.771	9.002	116.037

Table 4.2: Common Results on Synthetic Dataset

A thing to note here is the high value for average number of features changed - this is entirely due to the fact that in our automated script the number of features generated as "user choice subsets" are of length 3 or more (we saw that our method caught the entire subset or a superset as the preference correctly).

We also calculate some addition parameters as shown in Table 4.3. We compare our method with Custom Method 2, which also uses comparisons to detect preference.

Paramater	**Custom 2**	**Our Method**
Average number of comparisons	40.94	45.78
Average percentage of cases it failed to detect user preference/changed fixed features	100	4

Table 4.3: Other Results on Synthetic Dataset

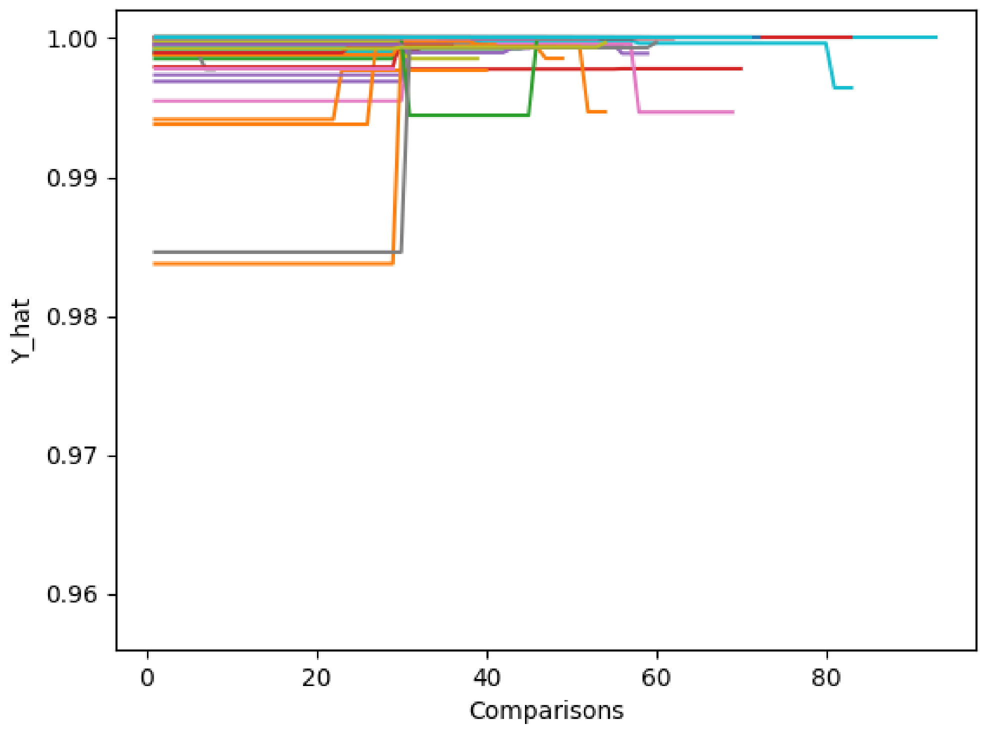
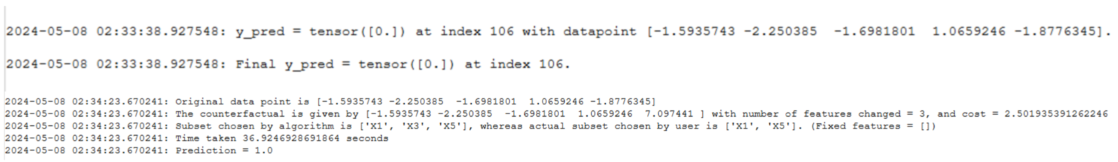


FIGURE 4.3: Prediction label v/s Comparisons for Custom Dataset for Each User

We show in Figure 4.2 how our method effectively reduces the cost of the best counterfactual (till that point) with each comparison as well as how the prediction label improves in Figure 4.3 for all users who recieve recourse (in both figures each color represents a single user). This shows that our method manages to choose a better arm with time, and improves on the overall metric. The reason the arms show a prediction of 0.98 or more right at the start is because it records the prediction value of the best counterfactual - it shows that for users achieveing recourse, our method gives a counterfactual which manages to change classes significantly almost everytime. In reality, some of these datapoints could have had the prediction value initially as low as y = 0 (this is without even applying sigmoid). For example, for user number 100 of this run, y = 0, but we end up getting a valid counterfactual regardless as shown in Figure 4.4.



```
2024-05-08 02:33:38.927548: y_pred = tensor([0.]) at index 106 with datapoint [-1.5935743 -2.250385  -1.6981801  1.0659246 -1.8776345].

2024-05-08 02:33:38.927548: Final y_pred = tensor([0.]) at index 106.

2024-05-08 02:34:23.670241: Original data point is [-1.5935743 -2.250385  -1.6981801  1.0659246 -1.8776345]
2024-05-08 02:34:23.670241: The counterfactual is given by [-1.5935743 -2.250385  -1.6981801  1.0659246  7.097441 ] with number of features changed = 3, and cost = 2.501935391262246
2024-05-08 02:34:23.670241: Subset chosen by algorithm is ['X1', 'X3', 'X5'], whereas actual subset chosen by user is ['X1', 'X5']. (Fixed features = [])
2024-05-08 02:34:23.670241: Time taken 36.9246928691864 seconds
2024-05-08 02:34:23.670241: Prediction = 1.0
```

FIGURE 4.4: Example of successful recourse

4.2.2 Experiment on Give Me Some Credit Dataset

Paramater	**PEAR**	**CSCF**	**FACE**	**My Method**
Average Number of Features Changed	2.79 ± 0.42	2.51 ± 1.12	5.97 ± 0.62	5.16
Average L_2 Cost of Counterfactuals	96.04 ± 31.96	100.69 ± 120.22	327.18 ± 78.85	125.922
Validity Percentage	89	0.57 ± 0.42	0.24 ± 0.38	100

TABLE 4.4: Common Results on Give Me Some Credit Dataset

For comparison with the baselines such as PEAR [2], CSCF [6] and FACE [40], we apply our method on the Give Me Some Credit dataset. We sample 5000 random users and then try to find the metrics for 50 users who reach a negative classification. We utilise the results for all methods based on the results mentioned in de Toni's paper [2].

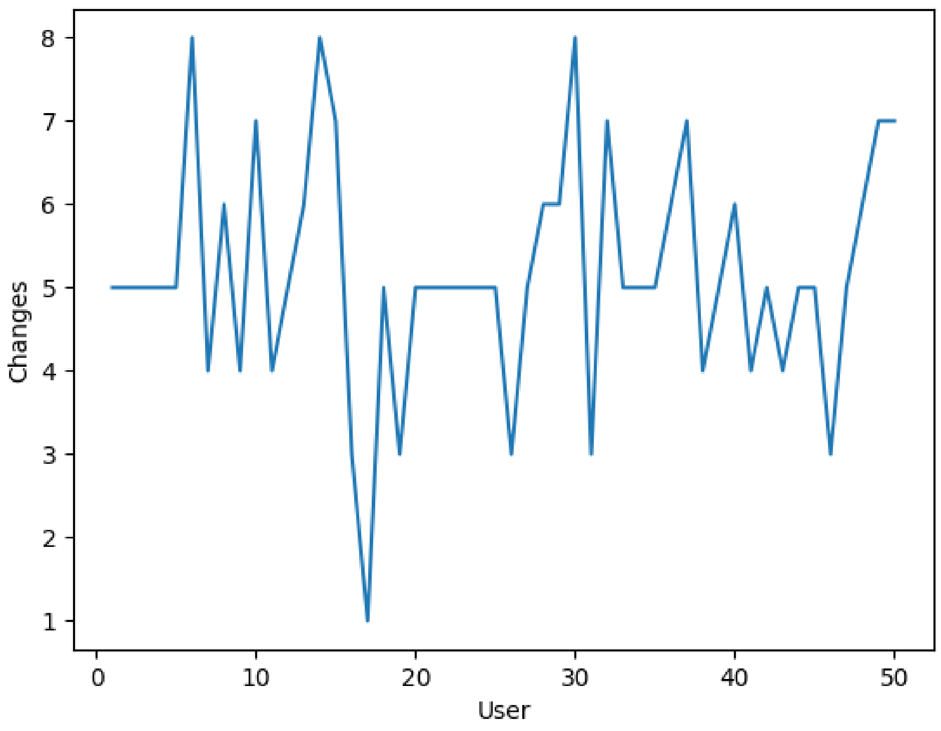


FIGURE 4.5: Counterfactual feature changes over 50 users

We showcase our results in Table 4.4.

We see that even though our method may not be the most effective in generating counterfactuals, it is somewhat comparable to the baselines. We see it gives remarkable validity as compared to the other methods.

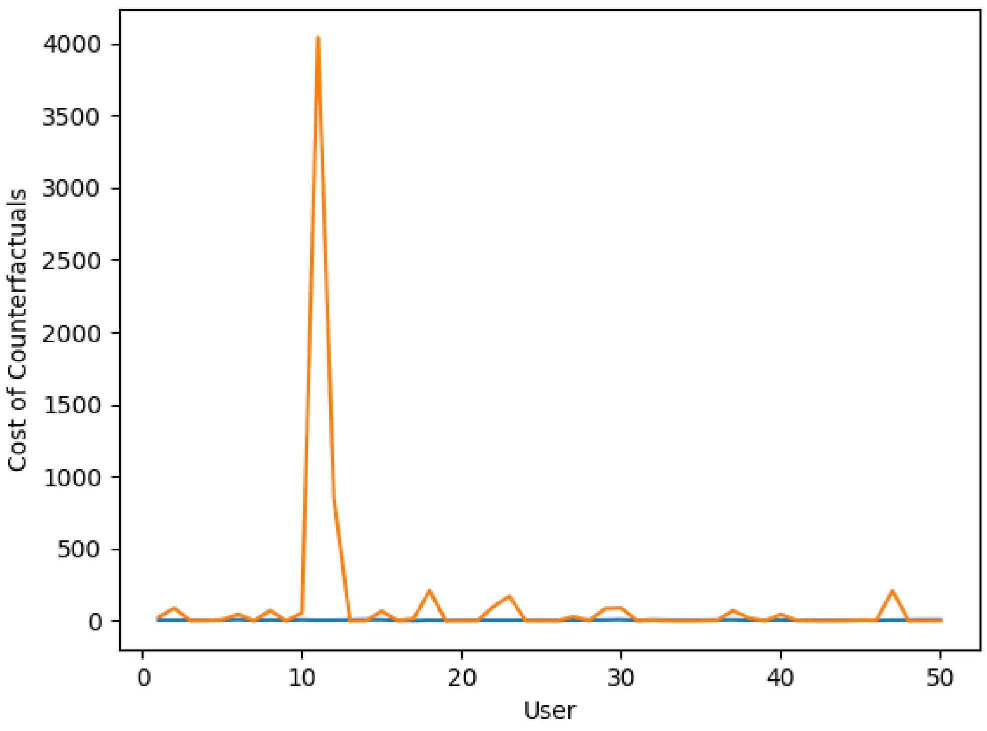


FIGURE 4.6: Counterfactual cost over 50 users

As was the case with the custom dataset, our method's high value for number of features changed is a reflection of the automated script's tendency to select large-sized preference sets. In general, our method managed to capture such preference sets. As an example, we show the user preferred subset as well as the one chosen by our algorithm for some user in Figure 4.7.

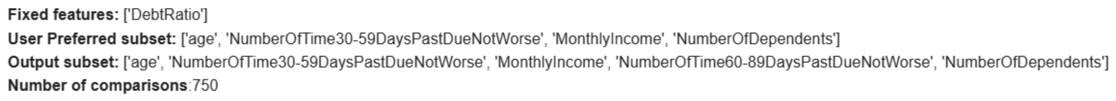

Fixed features: ['DebtRatio']
User Preferred subset: ['age', 'NumberOfTime30-59DaysPastDueNotWorse', 'MonthlyIncome', 'NumberOfDependents']
Output subset: ['age', 'NumberOfTime30-59DaysPastDueNotWorse', 'MonthlyIncome', 'NumberOfTime60-89DaysPastDueNotWorse', 'NumberOfDependents']
Number of comparisons:750

FIGURE 4.7: User Choice Subset Learning Example

We also showcase some other metrics for our run on this dataset in Table 4.5. We find that it always picked up user preference - indicating that our algorithm managed to learn the user's preference successfully.

Paramater	**Our Method**
Average number of comparisons	477.5
Average percentage of cases it failed to detect user preference/changed fixed features	0
Average time to execute in seconds	420.583

Table 4.5: Other Metrics for Give Me Some Credit

We also show how the counterfactual costs and number of features changed vary for each user in Figures 4.6 and 4.5 respectively. We see that in general cost of counterfactuals generated are, in fact, very low.

A last thing to observe about these experiments is the number of comparisons for both datasets. While there is no doubt that it is still very high and this limits its usability for a regular user - 477 questions to find the perfect recourse for a user using the "give me some credit" dataset (which has 10 features) for example - there is still some optimization which has been achieved as compared to the entirely brute method of comparing all subsets with each other. For 10 features that would have been $\binom{1024}{2}$= 523776 comparisons.

Chapter 5

Conclusion

As we move towards a future which involves critical systems with more and more automation, especially using AI, it is imperative that we are more aware of how these systems operate. Through this work, we provide the first end-to-end system which generates a recourse of minimal cost in keeping with the preferences of the user. We develop a method which has strong roots in game theory and causality and implement the same in pytorch and python. We show that our method is comparable to, if not always better than, the baselines and that there is potential to lead to lowest cost recourse with some more research on potential loss functions and improvements in methodology.

Bibliography

[1] M. Singh, S. S. Kancheti, S. Gupta, G. Ghalme, S. Jain, and N. C. Krishnan, "Algorithmic recourse based on user's feature-order preference," ser. CODS-COMAD '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 293–294. [Online]. Available: https://doi.org/10.1145/3570991.3571039

[2] G. De Toni, P. Viappiani, S. Teso, B. Lepri, and A. Passerini, "Personalized algorithmic recourse with preference elicitation," *arXiv preprint arXiv:2205.13743*, 2022.

[3] A.-H. Karimi, B. Schölkopf, and I. Valera, "Algorithmic recourse: from counterfactual explanations to interventions," in *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, 2021, pp. 353–362.

[4] K. Rawal and H. Lakkaraju, "Beyond individualized recourse: Interpretable and interactive summaries of actionable recourses," *Advances in Neural Information Processing Systems*, vol. 33, pp. 12 187–12 198, 2020.

[5] P. Yadav, P. Hase, and M. Bansal, "Low-cost algorithmic recourse for users with uncertain cost functions," *arXiv preprint arXiv:2111.01235*, 2021.

[6] P. Naumann and E. Ntoutsi, "Consequence-aware sequential counterfactual generation," in *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part II 21.* Springer, 2021, pp. 682–698.

[7] Y. Yue, J. Broder, R. Kleinberg, and T. Joachims, "The k-armed dueling bandits problem," *Journal of Computer and System Sciences*, vol. 78, no. 5, pp. 1538–1556, 2012.

[8] D. S. KV, "Neural network-based liquidity risk prediction in indian private banks," *Intelligent Systems with Applications*, vol. 21, p. 200322, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667305323001473

[9] S. Kujawa and G. Niedbała, "Artificial neural networks in agriculture," p. 497, 2021.

[10] T. Pook, J. Freudenthal, A. Korte, and H. Simianer, "Using local convolutional neural networks for genomic prediction," *Frontiers in genetics*, vol. 11, p. 561497, 2020.

[11] J. Duan, "Financial system modeling using deep neural networks (dnns) for effective risk assessment and prediction," *Journal of the Franklin Institute*, vol. 356, no. 8, pp. 4716–4731, 2019.

[12] R. Bartlett, A. Morse, R. Stanton, and N. Wallace, "Consumer-lending discrimination in the fintech era," *Journal of Financial Economics*, vol. 143, no. 1, pp. 30–56, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0304405X21002403

[13] J. Dastin, "www.reuters.com," 2018. [Online]. Available: https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK08G/

[14] M. Ali, P. Sapiezynski, M. Bogen, A. Korolova, A. Mislove, and A. Rieke, "Discrimination through optimization: How facebook's ad delivery can lead to biased outcomes," *Proceedings of the ACM on Human-Computer Interaction*, vol. 3, no. CSCW, p. 1–30, Nov. 2019. [Online]. Available: http://dx.doi.org/10.1145/3359301

[15] A. Z. Huq, "Racial equity in algorithmic criminal justice," *Duke LJ*, vol. 68, p. 1043, 2018.

[16] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[17] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[18] J. Collins, N. Hall, and L. A. Paul, "Counterfactuals and causation: History, problems, and prospects," 2004.

[19] S. L. Engerman, "Iii. counterfactuals and the new economic history," *Inquiry*, vol. 23, no. 2, pp. 157–172, 1980. [Online]. Available: https://doi.org/10.1080/00201748008601899

[20] T. Horgan, "Counterfactuals and newcomb's problem," *The Journal of Philosophy*, vol. 78, no. 6, pp. 331–356, 1981. [Online]. Available: http://www.jstor.org/stable/2026128

[21] R. N. Strassfeld, "If...: Counterfactuals in the law," *Geo. Wash. L. Rev.*, vol. 60, p. 339, 1991.

[22] S. Wachter, B. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the gdpr," *Harv. JL & Tech.*, vol. 31, p. 841, 2017.

[23] J. Pearl, "The art and science of cause and effect," 1996. [Online]. Available: http://singapore.cs.ucla.edu/LECTURE/lecture_sec1.htm

[24] B. Schölkopf, "Causality for machine learning," in *Probabilistic and causal inference: The works of Judea Pearl*, 2022, pp. 765–804.

[25] T. Monleón Getino and J. Canela i Soler, "Causality in medicine and its relationship with the role of statistics." *Biomedical Statistics and Informatics, 2017, vol. 2, num. 2, p. 61-68*, 2017.

[26] B. Atems, "Identifying the dynamic effects of income inequality on crime," *Oxford Bulletin of Economics and Statistics*, vol. 82, no. 4, pp. 751–782, 2020.

[27] C. Granger, “Causality in economics,” *Thinking About Causes: From Greek Philosophy to Modern Physics*, pp. 284–296, 2007.

[28] G. Xu, T. D. Duong, Q. Li, S. Liu, and X. Wang, “Causality learning: A new perspective for interpretable machine learning,” *arXiv preprint arXiv:2006.16789*, 2020.

[29] J. K. Tarus, Z. Niu, and G. Mustafa, “Knowledge-based recommendation: a review of ontology-based recommender systems for e-learning,” *Artificial intelligence review*, vol. 50, pp. 21–48, 2018.

[30] B. Lika, K. Kolomvatsos, and S. Hadjiefthymiades, “Facing the cold start problem in recommender systems,” *Expert systems with applications*, vol. 41, no. 4, pp. 2065–2073, 2014.

[31] H. Jiang, X. Qi, and H. Sun, “Choice-based recommender systems: a unified approach to achieving relevancy and diversity,” *Operations Research*, vol. 62, no. 5, pp. 973–993, 2014.

[32] R. A. Bradley and M. E. Terry, “Rank analysis of incomplete block designs: I. the method of paired comparisons,” *Biometrika*, vol. 39, no. 3/4, pp. 324–345, 1952.

[33] G. Pigozzi, A. Tsoukias, and P. Viappiani, “Preferences in artificial intelligence,” *Annals of Mathematics and Artificial Intelligence*, vol. 77, pp. 361–401, 2016.

[34] B. Ustun, A. Spangher, and Y. Liu, “Actionable recourse in linear classification,” in *Proceedings of the conference on fairness, accountability, and transparency*, 2019, pp. 10–19.

[35] R. K. Mothilal, A. Sharma, and C. Tan, “Explaining machine learning classifiers through diverse counterfactual explanations,” in *Proceedings of the 2020 conference on fairness, accountability, and transparency*, 2020, pp. 607–617.

[36] J. Pearl, M. Glymour, and N. P. Jewell, *Causal Inference in Statistics: A Primer*, 2016th ed. John Wiley and Sons, 2016.

[37] S. Shimizu, P. O. Hoyer, A. Hyvärinen, A. Kerminen, and M. Jordan, “A linear non-gaussian acyclic model for causal discovery.” *Journal of Machine Learning Research*, vol. 7, no. 10, 2006.

[38] Y. Zheng, B. Huang, W. Chen, J. Ramsey, M. Gong, R. Cai, S. Shimizu, P. Spirtes, and K. Zhang, “Causal-learn: Causal discovery in python,” *Journal of Machine Learning Research*, vol. 25, no. 60, pp. 1–8, 2024.

[39] A. Sharma and E. Kiciman, “Dowhy: An end-to-end library for causal inference,” *arXiv preprint arXiv:2011.04216*, 2020.

[40] R. Poyiadzi, K. Sokol, R. Santos-Rodriguez, T. De Bie, and P. Flach, “Face: feasible and actionable counterfactual explanations,” in *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 2020, pp. 344–350.

List of Publications

National and International Conferences

- Soumya Sarkar, Nitin Singhal, Shweta Jain, Shashi Shekhar Jha, "**CGFLasso: Combating Multicollinearity using Domain Knowledge**," CODS-COMAD 2024: The 7th Joint International Conference on Data Science & Management of Data (11th ACM IKDD CODS and 29th COMAD).
- Kumar Mangalam, Soumya Sarkar, Yakul Dogra, Mukesh Saini, Neeraj Goel, "**What Did the Chicken Say?: A Multi-class Classification Method on Chicken Vocalisations**," ICSTA 2023: International Conference On Systems And Technologies For Smart Agriculture.

Awards/Honor/Talks

- Webinar on Artificial Intelligence at Fakir Chand College, West Bengal (to be held on June 8, 2024)