A Study of Algorithmic Recourse

Dawei Xie

Abstract

The increasing reliance on data-driven automated decisions in various domains has raised concerns about fairness, transparency, and accountability. Algorithmic recourse, which refers to the ability of individuals affected by algorithmic systems to alter their input to reverse the outcome, has emerged as a set of critical approaches in addressing these concerns. This project aims to provide a comprehensive evaluation and comparison of three representative algorithmic recourse methods in terms of their primary motivations, technical novelties and practical concerns through experiments, to determine their effectiveness and applicability in real-world scenarios. The experimental results show that those methods are performing quite distinctly regarding evaluation metrics like recourse validity and recourse cost. Moreover, none of these methods is stable in their performance across different datasets. We claim that existing methods are serving diverse purposes, which is also implied by a thorough elaboration on involved techniques.

1. Introduction

As our society increasingly relies on data-driven automated decisions, algorithmic decision making systems have become ubiquitous in many domains, including but not limited to finance, healthcare and criminal justice. With the promise of greater efficiency and efficacy, however, these systems also pose critical challenges in terms of fairness, transparency, and accountability. One of the aspects that appeal for attention is the understanding of algorithmic recourse, which refers to the ability of individuals affected by algorithmic decisions to alter their input so that they can obtain a more favorable outcome.

Consider that (Figure 1) a loan-granting institution (e.g., a bank) denies a user the loan based on his attributes. Some of his attributes are shown in table (this list is incomplete). The user may want to understand why and furthermore, not simply which attributes are important, but which can be altered to change the outcome. In this setting, unless the specific decision rules (the binary classifier) of the bank is released, the user must put a tremendous effort to change their situation to be favorably treated by the classifier.

A broadly discussed algorithmic recourse method can be traced back to Wachter et al. [13]. It was primarily rooted in a context of a legal framework, i.e., the European Union's General Data Protection Regulation (GDPR). In 2018, recourse was formally proposed in [11] and studied in the context of machine learning. The definition has been used since then as *the ability of a person to change the decision of a model by altering actionable input variables*. However, there is an assumption in [11] that the model the decision subjects are faced with is a linear classifier, which might not always hold in reality.

Along with the development of algorithmic recourse, counterfactual explanations have been widely studied and have actually become an important part of algorithmic recourse. Diverse Counterfactual Explanations (DiCE) [7] was proposed in 2019, motivated by the concerns of feasibility and diversity. DiCE tries to generate a diverse set of counterfactual explanations based on determinantal point processes. Immediately also in 2019, researchers proposed counterfactual explanations guided by prototypes (ProtoCF) [12], which made use of class prototypes.

We are not differentiating counterfactual explanations and algorithmic recourse in this article, and the terminology as well as the concepts related to *counterfactual explanations* and *algorithmic recourse* will be clarified in Section 2.

Motivation We have seen numerous algorithmic recourse methods in recent years, yet there is little effort put in comparing and evaluating those methods in an informative way that can provide useful guidelines for machine learning practitioners when they need to choose proper methods in concrete application scenarios.

On the other hand, some methods are variants of approaches proposed earlier that may suffer from significant pitfalls, but it is not clear how certain methods can make a difference. This may result in ineffective use of those methods without carefully examining the limitations. Thus, we attempt to explore in detail three algorithmic recourse methods that are representative enough from which we are able to acknowledge recent efforts in the community of machine learning and algorithmic fairness.

Scope and Contribution This project attempts to provide a comprehensive evaluation and comparison of three representative algorithmic recourse methods in terms of their primary motivations, technical novelties and practical concerns through experiments, aiming to identify practical limitations, uncover the underlying connections and highlight the crucial differences in an informative way to promote effective use of a broad array of algorithmic recourse methods in real life.

Relevance to the Class All algorithmic recourse methods covered in this article entail optimization over nonlinear terms in a variety of loss functions. A generic set of algorithms used in those methods are typically considered as gradient descent and their extensions or variants. In addition to optimization algorithm that plays a critical role in solving for algorithmic recourse itself, the general evaluation pipeline also entails training neural network models serving as classifiers on tabular data.

2. Related theory and practice

Details of the three aforementioned methods will be covered in Section 3, including their optimization procedures. This section summarizes the closest connection to other machine learning subfields.

Counterfactual Explanations

Counterfactual explanations are a set of approaches used to understand and explain machine decisions in terms of causal relationships and the impact of certain events on outcomes. These explanations focus on demonstrating alternative scenarios that would have occurred under other conditions. In other words, counterfactual explanations attempt to answer questions like "What if ...?" to enhance the interpretability of automated decisions.

In the context of machine learning, counterfactual explanations typically involve generating an alternative instance that should result in a different decision from the model. The generated explanations make machine learning models more interpretable to humans in terms of informing features that contribute most significantly to the classifier and are most likely to be altered to change the outcome. Therefore, algorithmic recourse and counterfactual explanations are used interchangeably by convention, particularly in literature of explainable artificial intelligence (XAI).

More concretely, we may view this procedure from the perspective of decision boundaries. Decision boundaries are the surfaces and curves in the feature space that separate different classes. In Figure 2, it is suggested for the user to increase his income by $5000 and have one more year of credit history in order to change the decision from "loan denied" to "loan approved".

Adversarial Examples

Adversarial examples are specifically designed inputs to deceive machine learning models by causing them to produce incorrect classifications [2]. Adversarial examples can be generated by solving an optimization problem, which aims to find the smallest possible perturbation that will result in an incorrect answer. Please refer to Figure 3 for an example in which a panda with perturbations is misclassified as a gibbon.

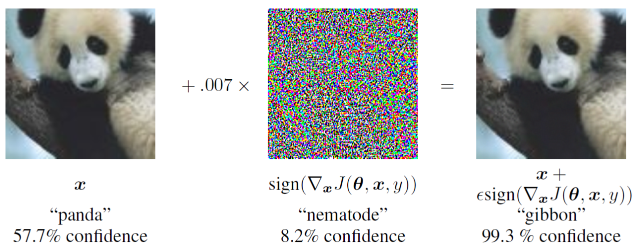

$+ .007 \times$
$=$
$\boldsymbol{x}$
"panda"
57.7% confidence
$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence
$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

Figure 3. Adversarial examples in images. Credits: [2]

Algorithmic recourse is essentially on the opposite side of adversarial training which attempts to defend against adversarial examples. Instead, algorithmic recourse encourages adversarial examples to exist. In some sense, algorithmic recourse is trying to find such adversarial examples by gradient-based optimization approaches.

Formally, given a binary classifier, $f_w : \mathcal{X} \rightarrow \mathcal{Y}$, where $\mathcal{Y} = \{0, 1\}$, an adversarial example is typically a perturbation $\delta \in \Delta$ such that $f_w(x + \delta) \neq f_w(x)$. Adversarial training aims to guarantee that within a "small ball" Δ, adversarial examples do not exist, which is $\forall \delta \in \Delta$, $f_w(x + \delta) = f_w(x)$. In contrast, for x that leads to an undesired outcome $f_w(x) = 0$, algorithmic recourse attempts to find some $\delta \in \Delta$ such that $f_w(x + \delta) = 1$. This has also been studied to design effective learning models [9].

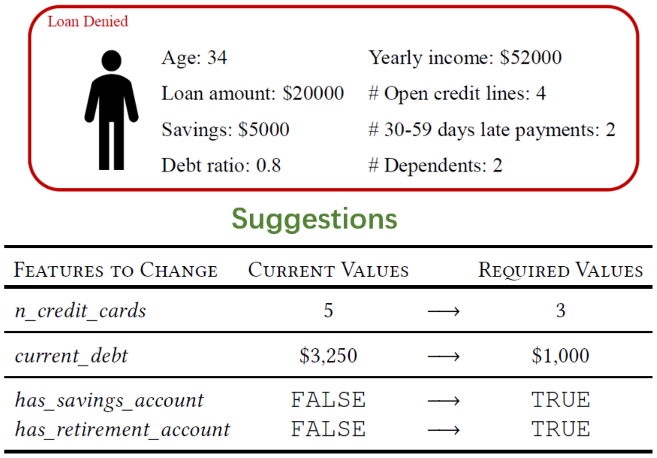
3. Technical details

3.1. Recourse Methods

3.1.1 Wachter's [13]

We refer to this kind of method as the Wachter method, as one of the earliest methods proposed to tackle algorithmic recourse. It has been considered as a generic baseline method that inspired a line of follow-up work in counterfactual explanations and algorithmic recourse.

Consider a binary classifier, $f_w : \mathcal{X} \rightarrow \mathcal{Y}$, where $\mathcal{Y} = \{0, 1\}$. We refer x such that $f_w(x) = 0$ as the input with an undesired outcome. We aim to find a counterfactual x', as close as possible to the original input x, such that $f_w(x') = y'$ where y' is the desired target. Formally, x' will solved for through the following procedure:



Loan Denied

Age: 34	Yearly income: $52000
Loan amount: $20000	# Open credit lines: 4
Savings: $5000	# 30-59 days late payments: 2
Debt ratio: 0.8	# Dependents: 2

Suggestions

Features to Change	Current Values		Required Values
n_credit_cards	5	$\longrightarrow$	3
current_debt	\$3,250	$\longrightarrow$	\$1,000
has_savings_account	`FALSE`	$\longrightarrow$	`TRUE`
has_retirement_account	`FALSE`	$\longrightarrow$	`TRUE`

Figure 1. Algorithmic recourse examples in loan approval. Credits: [8, 11]

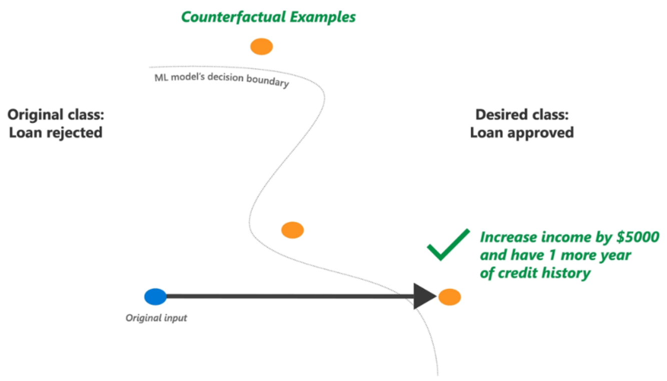

Counterfactual Examples
ML model's decision boundary
Original class: Loan rejected
Desired class: Loan approved
Increase income by $5000 and have 1 more year of credit history
Original input

Figure 2. An example of a decision boundary. Credits: [7]

$$x' = \arg\min_{x'} \max_{\lambda} \lambda \left(f_w\left(x'\right) - y'\right)^2 + \operatorname{dist}\left(x, x'\right) \quad (1)$$

For this objective, we typically follow an iterative procedure to solve for

$$x' = \arg\min_{x'} \lambda' \left(f_w\left(x'\right) - y'\right)^2 + \operatorname{dist}\left(x, x'\right)$$

starting from a relatively small λ' and increase λ' until a feasible solution x' is found.

$\operatorname{dist}\left(x_i, x'\right)$ is the distance metric that measures how far the counterfactual x' and the original input x are from each other. It is essentially the Manhattan distance weighted by the inverse median absolute deviation. For all feature $k \in F$, the distance between x and x' is written as

$$\operatorname{dist}\left(x, x'\right) = \sum_{k \in F} \frac{\left|x_k - x'_k\right|}{\mathrm{MAD}_k} \quad (2)$$

with

$$\mathrm{MAD}_k = \operatorname{median}_{j \in P}\left(\left|X_{j,k} - \operatorname{median}_{i \in P}\left(X_{i,k}\right)\right|\right)$$

where P denotes the set of all points in the dataset.

Wachter et al. [13] put a good deal of effort in elaborating the desirable properties of this distance metric. When it comes to examining certain features in a dataset, there is usually a concern of intrinsic volatility for the distribution of some feature. More specifically, for some feature k, if k varies wildly across the dataset, a counterfactual x' should still be considered close to the original input x if it varies at the same scale (without normalization in this setting).

Another concern is the desire of sparsity, as ℓ_1 norm typically induces sparse solutions where most entries are 0. When applying algorithmic recourse in practice, it is usually desirable to have only a small number of features changed and most remain as they were.

Optimization Wachter follows the gradient descent algorithm to generate the recourse. As in a general setup, it is typically considered as an unconstrained optimization problem, which is not always the case in practice. There are usually some features that are considered immutable or at least costly to change, such as race and birthplace. Accordingly, the problem will be turned into a constrained optimization problem.

3.1.2 DiCE [7]

A straightforward motivation of DiCE comes from the difficulty to generate a recourse that is truly actionable for a user's situation. As [7] pointed out, in the loan approval example, the recourse may suggest to "change your house rent", but provide no information about alternative choices, or consider the relative difficulty between different changes the user may need to make.

Therefore, instead of seeking for one feasible recourse, DiCE attempts to generate a set of recourses that can serve diverse situations. To ensure the diversity of the generated recourses, the objective is built on determinantal point processes. Formally,

$$\begin{aligned} C(x) = \underset{x_1',\ldots,x_k'}{\operatorname{argmin}} \frac{1}{k} \sum_{i=1}^{k} \left(f_w\left(x_i'\right) - y'\right)^2 \\ + \frac{\lambda_1}{k} \sum_{i=1}^{k} \operatorname{dist}\left(x, x_i'\right) \\ - \lambda_2 \det(K) \end{aligned} \tag{3}$$

where $K_{i,j} = \frac{1}{1+\operatorname{dist}\left(x_i', x_j'\right)}$ and $\operatorname{dist}\left(x_i', x_j'\right)$ denotes the distance between the two counterfactuals (recourses) x_i' and x_j'. Besides, k is the number of recourses we are about to generate, and λ_1 and λ_2 are hyperparameters to balance the three parts of the loss function.

We will parse the objective term by term from the last to the first.

The last term suggests an important component that is built on determinantal point processes to serve the diversity purpose. Determinantal point process inherently captures negative correlations, while in our setting we aim to push the generated recourses to be more diverse and more different from each other.

The middle term is exactly the distance used in Wachter [13] as we just covered. In particular, DiCE proposed a way of dealing with categorical features which has become a standard approach to be applied in such tasks.

$$\text{dist\_categorical}(x', x) = \frac{1}{N_{cat}} \sum_{i=1}^{N_{cat}} \mathbb{I}\left(x'^i \neq x^i\right) \tag{4}$$

where N_{cat} is the number of categorical features.

There are some discussions about the first term with respect to the ℓ_2 loss. We write it in a way compatible with Wachter, whereas DiCE claims that both ℓ_1 loss and ℓ_2 loss work well in this objective. In practice, however, DiCE prefers hinge loss instead of ℓ_1 loss or ℓ_2 loss in a sense that the later penalize the difference between $f_w(x')$ and y'. This is motivated by the fact that a valid recourse only needs $f_w\left(x'\right)$ to exceed the threshold, not necessarily to be close to $y' = 1$. Besides, the authors of DiCE concluded in [7] that optimizing for $f_w\left(x'\right)$ to be close to 1 encourages large changes to x towards the counterfactual, which in turn make the generated recourse less feasible for the user.

Optimization The composed loss in Eq. 3 is optimized using the gradient descent algorithm. It's worth mentioning here that because of the non-convexity nature of the objective, it is likely that the generated recourse is not feasible.

3.1.3 ProtoCF [12]

To further enhance interpretability and improve the convergence speed of generating recourses, ProtoCF was proposed empowered by modern deep learning components. Since ProtoCF was primarily designed for multi-class counterfactual explanations, it can be easily simplified to binary classifiers as in algorithmic recourse. ProtoCF uses a composed loss function, which will be articulated in detail.

ℓ_1 loss and ℓ_2 loss are the Manhattan distance and the Euclidean distance between the counterfactual x' and the original input x, respectively. In addition, there comes a commonly used loss ℓ_{pred}

$$\ell_{\text{pred}} = \max\left(f_w(x')_{y_0} - f_w(x')_{y'}, -\kappa\right) \tag{5}$$

where $f_w(x')_{y_0}$ denotes the prediction probability of $y_0 = 0$, and $f_w(x')_{y'}$ is the prediction probability of $y' = 1$, with $\kappa \geq 0$ capping the divergence between $f_w(x')_{y_0}$ and $f_w(x')_{y'}$.

A major concern with regard to previous methods is that the generated recourse does not lie on the manifold of real data, which results in out-of-distribution instances. To resolve this, an autoencoder AE is used and thus we can derive the ℓ_{AE} loss:

$$\ell_{\text{AE}} = \gamma \cdot \left(x' - \text{AE}(x')\right)^2 \tag{6}$$

Besides, we need some data drawn from the training dataset, i.e., $X = \{x_1, \ldots, x_n\}$. The point of interest is denoted by x_0 such that $f_w(x_0) = 0$. Then the classifier is called to label the data points with the output classes. For each class i (here we only care about class $y' = 1$), data points belonging to that class are encoded and ordered by increasing their ℓ_2 distances to $\text{AE}(x_0)$. The class prototype is defined as the average encoding over the K nearest points in the latent space within the same class y':

$$\text{proto}_{y'} = \frac{1}{K} \sum_{k=1}^{K} \text{AE}\left(x_k^{y'}\right) \tag{7}$$

Now we can obtain the ℓ_{proto} as follows

$$\ell_{\text{proto}} = \theta \cdot \left(\text{AE}\left(x'\right) - \text{proto}_{y'}\right)^2 \tag{8}$$

And then the final objective (the ideal recourse) with respect to above loss functions are defined as

$$x' = \arg\min_{x'} c \cdot \ell_{\text{pred}} + \beta \cdot \ell_1 + \ell_2 + \ell_{\text{AE}} + \ell_{\text{proto}} \quad (9)$$

Optimization The objective is optimized by leveraging the fast iterative shrinkage-thresholding algorithm (FISTA) [1], which preserves the computational simplicity of iterative shrinkage-thresholding algorithms (ISTA) but with a global rate of convergence proven to be significantly better.

3.2. Evaluation Metrics

We almost follow metrics used in DiCE [7], which are fairly reasonable in addressing major concerns when applying those algorithmic recourse methods.

3.2.1 Validity

Validity is simply the fraction of recourses returned by a method that are truly verified as effective, which required that they result in a different outcome than the original input.

3.2.2 Cost

Cost measures how far the generated recourse x' is from the original input x. We are using a slightly different metric from what we applied in optimization, ℓ_1 norm distance between the generated recourse and the original input, normalized by the number of features. In general, we refer recourses with lower proximity value as those more likely to be applied or easier for users to act upon.

3.3. Datasets Review

3.3.1 Adult

As officially known as the Census Income Dataset, the *Adult* dataset is extracted by Barry Becker from the 1994 Census database [5]. The binary feature as the target is whether or not the annual income has exceeded 50K. Thus the user may want to know how algorithmic recourse can help idenfy features that can be adjusted to help increase the income to above 50K.

An individual's annual income is affected by various factors, such as education level, age, gender, occupation, etc. Some features used in the experiments are

- `age`
- `workclass` (Government, Other/Unknown, Private, Self-Employed)
- `education` (Assoc, Bachelors, Doctorate, HS-grad, Masters, Prof-school, School, Some-college)
- `marital_status` (Divorced, Married, Separated, Single, Widowed)
- `occupation` (Blue-collar, Other/Unknown, Professional, Sales, Service, White-collar)
- `hours_per_week` (continuous)

3.3.2 Oulad

The Open University Learning Analytics dataset (Oulad) [6] contains data about courses, students and their interactions with Virtual Learning Environment for selected courses. The binary feature as the target is students' final grade (P or F). Some features are shown below.

- `gender:` ('M', 'F')
- `highest_education` ('HE Qualification', 'A Level or Equivalent', 'Lower Than A Level')
- `age_band` ('0-35', '35-55', '55$\leq$')
- `num_of_prev_attempts` (continuous)
- `studied_credits` (continuous)

3.3.3 Heloc

Home Equity Line of Credit (HELOC) recorded information about credit evaluation. The binary feature is customers' risk performance (good or bad). Some of the features are listed as follows (all adjustable features are continuous in this dataset)

- `ExternalRiskEstimate`
- `NumSatisfactoryTrades`
- `NumTotalTrades`
- `NetFractionRevolvingBurden`
- `NumRevolvingTradesWBalance`

3.4. Experimental Setup

We use a standard Python environment on Colab to run all experiments. In particular, two external packages are used: `dice_ml` (accompanied with [7]) and `ReLax` [3]. `ReLax` leverages `jax` to accelerate generating recourses.

We first showcase the use of `dice_ml` to generate a set of recourses on the *Adult* dataset, as well as demonstrating the desirable property of customization in practice.

Then we use `ReLax` to evaluate the three algorithmic recourse methods on the *Adult* dataset, the *Oulad* dataset, and the *Heloc* dataset, reporting the corresponding classifier's accuracy, the validity as well as the associated cost of the three algorithmic recourse methods. The evaluation follows a general procedure:

Dataset	Method	Accuracy	Validity	Cost
Adult	Wachter	0.824346	0.933178	8.139759
	DiCE	0.824346	0.508537	2.905395
	ProtoCF	0.824346	0.824592	6.932194

Table 1. Algorithmic recourse methods on the *Adult* dataset

Dataset	Method	Accuracy	Validity	Cost
Oulad	Wachter	0.928826	0.997423	12.241540
	DiCE	0.928826	0.981838	1.067762
	ProtoCF	0.928826	0.774328	4.209029

Table 2. Algorithmic recourse methods on the *Oulad* dataset

1. Specify the dataset to evaluate algorithmic recourse methods on. Datasets: *Adult*, *Oulad*, *Heloc*.

2. Train a simple neural network model with 3 hidden layers as the classifier (one for each dataset).

3. Set configurations for three algorithmic recourse methods and use them to generate counterfactuals (recourses).

4. Evaluate the algorithmic recourse methods based on metrics as validity and associated cost.

4. Experimental results

We showcased the use of `dice_ml` on the *Adult* dataset with regard to an affected individual (the user) classified as the group with the annual income less than 50K. DiCE generates five recourses, as shown in Figure 4. Examining the generated recourse, we can observe that not all recourses are feasible. Some features are usually considered as immutable or costly to change, while some are not realistic to adjust. Perhaps the most practical recourse is to obtain a master degree.

In addition to the use case above, `dice_ml` also supports specifying certain features that are able to change, while keeping other features as they were. For example, if we specify features `education` and `hours_per_week` to be adjustable, as in Figure 5, then we will obtain recourses with only possible changes to `education` and `hours_per_week`. In this case, the generated recourses are more informative in terms of practical instructions. The simplest recommendation would thus probably be to get a master degree and increase hours per week to above 61 hours.

Next, we use `ReLax` to evaluate algorithmic recourse methods on three datasets.

The statistics of concerned algorithmic recourse methods on the *Adult* dataset is reported in Table 1.

The statistics of concerned algorithmic recourse methods on the *Oulad* dataset is reported in Table 2.

The statistics of concerned algorithmic recourse methods on the *Heloc* dataset is reported in Table **??**. It's a bit surprising that Wachter and DiCE have exactly the same validity performance for the particular trial.

5. Discussions, observations, and comparisons

Through the evaluation of those algorithmic recourse methods, it is seen that the simplest method Wachter can typically has higher validity rate, but also suffers from extremely high cost at the same time. DiCE may not always generate valid recourse, but the generated recourses have relatively lower cost in most times. ProtoCF effectively balances the trade-off between the validity and the cost.

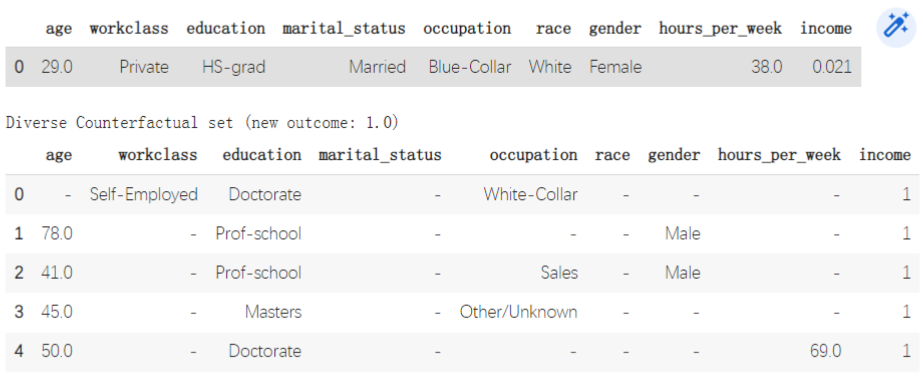
However, no safe conclusion can be drawn regarding an absolute winner in a sense that these three representative methods serve quite different purposes and application scenarios.

Wachter is fairly straightforward in its motivation as it aims at presenting the possibility of changing a model's outcome by intentionally altering input features in the context with social / legal concerns. As clarified in Section 2, it is essentially derived from the idea of adversarial examples. Adversarial examples are harmful as they are deceiving a machine learning model leveraging neural networks' inherent vulnerabilities. Wachter was instead proposed to illustrate that we might be able to make use of adversarial examples socially beneficially. Wachter is not suggested for use in practice, as it typically yields costly recourses which are kind of useless.

DiCE was proposed as a sign that the community of algorithmic recourse (or XAI) had started devoting efforts to practical use of counterfactual explanations. Diversity of the provided recourses has become a major concern when designing the loss function, to encourage recourses to be ”more different with each other” so that users have effectively more options. The evaluation in this article might not be fair when reporting the validity of DiCE in a sense that it is by nature more likely to have invalid recourse when we are generating a set of recourses at the same time.

ProtoCF was not designed directly to serve algorithmic recourse, but has a broader use in model interpretability. It is also observed that the major effort has been put in composing a variety of loss functions to serve different needs. Another desired feature of ProtoCF is the search process has been significantly accelerated benefited from the fast convergence of FISTA [1].

In the end, we would like to highlight an assumption of differentiability of the underlying classifier users are faced with, which motivates the use of neural network models in this kind of literature to approximate the classifier, as well

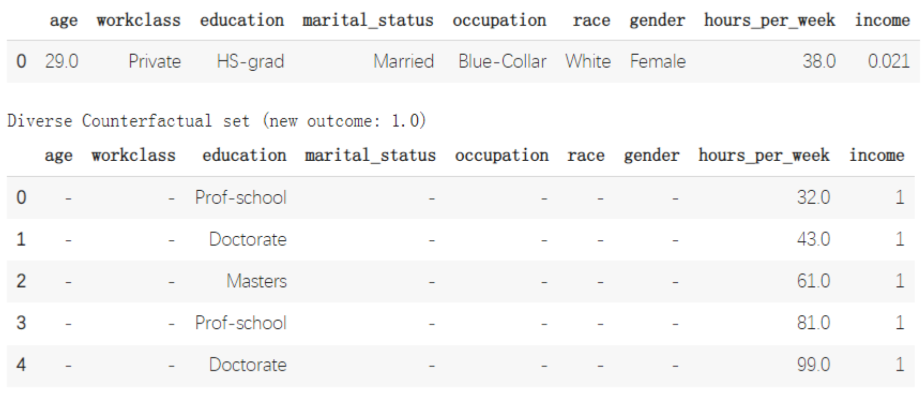


	age	workclass	education	marital_status	occupation	race	gender	hours_per_week	income
0	29.0	Private	HS-grad	Married	Blue-Collar	White	Female	38.0	0.021

```
Diverse Counterfactual set (new outcome: 1.0)
```

	age	workclass	education	marital_status	occupation	race	gender	hours_per_week	income
0	-	Self-Employed	Doctorate	-	White-Collar	-	-	-	1
1	78.0	-	Prof-school	-	-	-	Male	-	1
2	41.0	-	Prof-school	-	Sales	-	Male	-	1
3	45.0	-	Masters	-	Other/Unknown	-	-	-	1
4	50.0	-	Doctorate	-	-	-	-	69.0	1

Figure 4. Recourses generated by DiCE using dice_ml.



	age	workclass	education	marital_status	occupation	race	gender	hours_per_week	income
0	29.0	Private	HS-grad	Married	Blue-Collar	White	Female	38.0	0.021

```
Diverse Counterfactual set (new outcome: 1.0)
```

	age	workclass	education	marital_status	occupation	race	gender	hours_per_week	income
0	-	-	Prof-school	-	-	-	-	32.0	1
1	-	-	Doctorate	-	-	-	-	43.0	1
2	-	-	Masters	-	-	-	-	61.0	1
3	-	-	Prof-school	-	-	-	-	81.0	1
4	-	-	Doctorate	-	-	-	-	99.0	1

Figure 5. Recourses generated by DiCE with specified features to adjust.

as provides the possibility of those gradient-based algorithmic recourse methods. Please refer to [11] for a recourse method that applies in linear classifiers using mixed-integer programming for optimization.

6. Recommendations

Algorithmic recourse has been widely studied in recent three years. Researchers started to tackle different aspects of algorithmic recourse from quite diverse angles. Karimi et al. [4] attempted to address cases involving causal relationships; Ross et al. [9] proposed that instead of computing recourses for some black-box model, we can design such classifiers that guarantee the existence of recourse in the first place; Upadhyay et al. [10] examined algorithmic recourse under various distribution shifts, aiming to promote more robust recourses.

It seems that we could study algorithmic recourse by taking uncertainty into consideration, or we may also explore the problem in a dynamic setting where recourses that have been carried out would significantly influence the classifier as well as the recourse model. The lack of up-to-date data, however, hinders the development in some sense as people might question that if there is any known deployed system that we are able to apply algorithmic recourse methods on.

7. Contributions of team members

Dawei Xie: survey, coding, evaluation, write-up, review.

References

[1] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009. 5, 6

[2] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 2

[3] Hangzhi Guo, Xinchang Xiong, and Amulya Yadav. ReLax: Recourse explanation library in jax. http://github.com/birkhoffg/ReLax, 2023. 5

[4] Amir-Hossein Karimi, Julius Von Kügelgen, Bernhard Schölkopf, and Isabel Valera. Algorithmic recourse under imperfect causal knowledge: a probabilistic approach. *Advances in neural information processing systems*, 33:265–277, 2020. 7

[5] Ronny Kohavi and Barry Becker. UCI machine learning repository. https://archive.ics.uci.edu/ml/datasets/adult, 2017. 5

[6] Jakub Kuzilek, Martin Hlosta, and Zdenek Zdrahal. Open university learning analytics dataset. *Scientific data*, 4(1):1–8, 2017. 5

[7] Ramaravind K Mothilal, Amit Sharma, and Chenhao Tan. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pages 607–617, 2020. 1, 3, 4, 5

[8] Martin Pawelczyk, Teresa Datta, Johan Van den Heuvel, Gjergji Kasneci, and Himabindu Lakkaraju. Probabilistically robust recourse: Navigating the trade-offs between costs and robustness in algorithmic recourse. In *The Eleventh International Conference on Learning Representations*, 2023. 3

[9] Alexis Ross, Himabindu Lakkaraju, and Osbert Bastani. Learning models for actionable recourse. *Advances in Neural Information Processing Systems*, 34:18734–18746, 2021. 2, 7

[10] Sohini Upadhyay, Shalmali Joshi, and Himabindu Lakkaraju. Towards robust and reliable algorithmic recourse. *Advances in Neural Information Processing Systems*, 34:16926–16937, 2021. 7

[11] Berk Ustun, Alexander Spangher, and Yang Liu. Actionable recourse in linear classification. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 10–19, 2019. 1, 3, 7

[12] Arnaud Van Looveren and Janis Klaise. Interpretable counterfactual explanations guided by prototypes. 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*, pages 650–665. Springer, 2021. 1, 4

[13] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harvard Journal of Law & Technology*, 31:841, 2017-2018. 1, 2, 3, 4