THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF COMPUTER SCIENCE

ReLax: A Recourse Explanation Library in Jax

Xinchang Xiong
SPRING 2023

A thesis
submitted in partial fulfillment
of the requirements
for baccalaureate degrees
in COMPUTER SCIENCE
with honors in COMPUTER SCIENCE

Reviewed and approved* by the following:

Amulya Yadav
PNC Technologies Career Development Assistant Professor
Thesis Supervisor

C. Lee Giles
David Reese Professor at the College of Information Sciences and Technology
Thesis adviser

Danfeng Zhang
Associate Professor of Computer Science
Honors Adviser

*Signatures are on file in the Schreyer Honors College.

Abstract

Counterfactual explanation has been proposed by several works as a means to explain the decisions of machine learning models. By providing actionable suggestions to achieve the desired outcome, counterfactual explanation enables affected end users to better understand the decision-making process. As the number of counterfactual explanation methods increases, there is a pressing need for a standardized benchmarking platform that allows researchers to compare and assess prior works using standardized evaluation metrics. However, there are currently limited options for benchmarking, with existing solutions relying on solving optimization problems for each input data point iteratively, making them not scalable. Addressing these limitations, we present ReLax (**Re**course Explanation **L**ibrary in J**ax**), a jax-based Python library designed for benchmarking counterfactual explanation methods on different datasets and developing new counterfactual explanation methods. In summary, our work provides the following contributions: (i) fast and scalable benchmarking on 4 state-of-art counterfactual explanation methods, (ii) ReLax provides a comprehensible and easy-to-use API for users to customize and develop new counterfactual explanation methods, (iii) a standardized evaluation metrics and datasets for transparent and extensive comparisons of these methods. Our extensive experiments on multiple real-world datasets show that ReLax runs 134.37 times faster on average compared to CARLA a benchmark library for CF explanations.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Dr. Amulya Yadav, for his guidance and support throughout this research project. would also like to express my sincere appreciation to Hangzhi Guo for his exceptional guidance and dedicated work on this project.

I would like to thank Dr. Lee Giles for supporting me as my thesis honors co-advisor and reviewing my thesis.

Lastly, I would like to thank Dr. Danfeng Zhang for supporting me as my academic honors advisor.

Chapter 1

Introduction

Machine learning (ML) is being increasingly adopted as an effective way to assist consequential decision-making in everyday applications. Decisions made by the ML models on some sensitive domains can impact people's lives and well-being, as the final decision might be disadvantageous for an end user. For example, credit risk assessment in loan approval process [14], risk-assessment in courts and corrections departments [30], and disease diagnostic [5]. Therefore, it becomes important to provide explanations for the decisions made by the ML models. To this end, several works have proposed to explain ML models' decisions through counterfactual explanation, which helps the affected end user understand the decision with recourse examples that lead to different predicted outcomes for the black-box ML models and offers the end user actionable suggestions to achieve the desired outcome [10, 33].

As counterfactual explanation methods accumulate, there is a growing need for a standardized benchmarking platform, on which researchers can draw comparisons between the prior works in recourse algorithms using widely used metrics, and assess the prior works by reproducing the findings. A standardized benchmarking platform is important as it aids peer reviewers in reproducing the experiment results and contributes to transparency and open science. In addition, a benchmarking platform provides researchers a means to assess their recourse algorithms against the prior works in CFEs, which helps the field to grow.

Limited numbers of prior works at CFE libraries, e.g., CARLA[21], which offers a competitive baseline for benchmarking and a common framework that includes more than 10 counterfactual explanation methods. One drawback in existing solutions is that prior works rely on solving a separate optimization problem for each input data point iteratively, which makes it impractical to generate CFEs for a large data set. Additionally, the existing CFE libraries also failed to include some of the novel CFE approaches that avoid post-hoc methods, such as CounterNet [7].

To address these limitations, we present ReLax (**Re**course Explanation **L**ibrary in J**ax**), a python library built on top of jax (a high-performance auto-differentiation library) to generate counterfactual and recourse explanations. More specifically, our work makes the following key contributions. First, ReLax provides quantitative evaluation metrics for evaluating counterfactual methods for standardized and transparent comparisons of different data sets. Second, ReLax includes 4 counterfactual explanation methods implemented using jax with comprehensible instructions and examples to help users to reproduce the results. ReLax also supports customized data loading, such that users can specify immutable features and hyperparameters before training the ML models. Third, ReLax provides a comprehensible and easy-to-use API for users to customize and develop new recourse algorithms. ReLax leverages the features of jax and offers the following merits: First, counterfactual explanation methods implemented in jax offers parallel CFEs generation via jax.vmap/jax.pmap, thus it's more efficient to generate multiple CF examples. Jax parallelism also enables the CFEs in our library to process a larger magnitude of data sets than the prior works. Second, our implementation of CFEs in jax can be accelerated on CPU/GPU/TPU via jax.jit, thus generating each CF example is faster than prior works.

We validate the effectiveness of our new library ReLax by benchmarking all CFEs available to our library on real-world data sets such as Adult[18] and Credit[35]. And we compare the runtime of our CFEs implementation in jax with the runtime of the same set of CFEs implementation in the existing CF library CARLA. Our results suggest that the implementation of CFEs in jax on average 134.37 times faster than CFEs in CARLA.

Chapter 2

Related Works

2.1 Prior Works in Explainable ML

A lot of prior work has been done in explainable Machine Learning, thus we provide a brief overview of work inside this space. First, we discuss related work in explainable ML that relies on attribution importance. Next, we discuss related work on case-based explanations. In addition, we discuss related work on CF explanations, which can be categorized into non-parametric methods and parametric methods [7]. Finally, we talk about prior efforts and benchmarking CF explanation algorithms.

2.1.1 Attribution Importance Based Explanation Methods

Attribution importance-based methods find the most influential attribution for each data instance that contributes to the model's overall accuracy or for a particular decision, e.g., LIME [28], SHAP [19], and QII[3]. Ribeiro et al. [28] proposed LIME, which generates local explanations by sampling new data points near the input data point, then applying a linear model to fit the sampled data. Similarly, Lundberg and Lee [19] introduced SHAP, which uses a unified framework for interpreting predictions using Shapley values. Datta et al. [3] introduced Quantitative Input Influence (QII) measures that capture the degree of influence of inputs on outputs of systems to provide a foundation for interpreting system decisions.

2.1.2 Case Based Explanation Methods

Case-based explanation methods seek to provide explanation to the end user by finding the datapoints that are similar to the explainee datapoint. For example, Chen et al. [2] introduced ProtoPNet, which combines evidence from prototypical parts of the image to explain the image classification. Koh and Liang [15] adopted influence functions to identify training points that are most responsible for given predictions. However, neither attribution importance- nor case-based

provides relatable information for the end users. In recent years, there has been a lot of literature accumulated on algorithmic recourse, which provides "an actionable set of changes a person can undertake in order to improve their outcome" [8].

2.2 Prior Work on Counterfactual Explanations

There's a number of literature on Counter Factual explanation (CFE) algorithms, which focuses on generating recourse examples that lead to different predicted outcomes for the black-box machine learning models [10, 33]. Counterfactual Explanations are preferable to end users [7] because it provides contrastive examples that are both easily digestible and practically useful for the end users [33]. Almost all prior work in this area generates recourse examples with post-hoc paradigm.

2.2.1 Non-parametric Methods

Non-parametric methods find counterfactual explanations without the use of parametrized models. Wachter et al. [33] propose VanillaCF, which generates CF explanations by minimizing the distance between the input data point and the counterfactual example and encouraging the classifier prediction of the counterfactual towards the desired class. Other algorithms focused on other CF attributes, such as sparsity [13, 25], recourse cost[29], diversity[24, 23], data manifold closeness[31, 32], causality[11], etc. Sparsity measures the number of feature changes to obtain the counterfactual. Since shorter explanations are easier to understand for humans [21], some recourse methods focused on constraining sparsity. For example, Keane and Smyth [13] limits counterfactual to at most two feature changes. Similarly, [29] using solvers explicitly constrain sparsity. Recourse cost measures the cost of action to achieve counterfactual. Recourse cost constraint encourages the algorithm to choose relatively low cost actions to achieve counterfactual. Spangher et al. [29] propose a cost function based on percentile shift, which accounts for the relative diffi-

culty of achieving counterfactual. Diversity measures the diversity among generated counterfactual explanations, such that the algorithm could generate multiple different counterfactual results based on limited number of input data. For example, Mothilal et al. [24] proposed a framework for generating a diverse set of counterfactual explanations by maximizing the determinant of the kernel matrix of the generated counterfactuals. Data manifold closeness measures the distance from the counterfactual to k nearest training data points. Data manifold closeness constraint forces the generated counterfactual to be near the input data points and adheres to observed correlations among the features. Kanamori et al. [9] proposed a framework of counterfactual explanations that evaluates counterfactuals and extracts realistic actions for users by using k-nearest neighbors. Causality constraint enforces the causal relations between features, such that the generated counterfactuals are achievable by the users. For example, Karimi et al. [12] proposed a causal graph-based counterfactual explanation algorithm.

However, this line of work relies on post-hoc procedures, which need to solve optimization problems to find CF explanations for each input data point. The post-hoc method in the existing CFEs negatively impacts the runtime, making them impractical on time-sensitive applications.

2.2.2 Parametric Methods

Parametric methods use parametric models (e.g., variational autoencoder model) to generate CF explanations to achieve optimization amortization. Mahajan et al. [20] proposed VAE-CF, which learns the mapping of the input data to CFs using a variational auto-encoder (VAE) [4]. Such that once VAE is trained, it can generate multiple CF explanations for multiple input data points, which mitigates the limitation of post-hoc procedure. Nemirovsky et al. [26] introduced CounteRGAN, which uses Generative Adversarial Nets (GANs) model to achieve optimization amortization. Similarly, Yang et al. [34] proposed a model based counterfactual framework that based on conditional generative adversarial net (CGAN) to synthesize CFEs with umbrella sam-

pling. However, these methods are still post-hoc in nature, and thus they suffer from poorly balanced cost-invalidity trade-offs [7].

2.3 Prior Attempts at Benchmarking CF Explanation Algorithms

To our knowledge, there exist limited numbers of prior works at CFE libraries, e.g., CARLA [27] and DiCE [24]. Mothilal et al. [24] propose a python library for benchmarking counterfactual explanation methods, which includes 11 popular counterfactual explanation methods and a standardized set of integrated evaluations for comparison of the methods. Similarly, Mothilal et al. [24] propose a framework for generating and evaluating a diverse set of counterfactual explanations based on determinantal point processes and provide metrics that enable comparison of counterfactual-based methods to other local explanation methods. However, none of the existing CFE libraries have a good solution to the slow runtime limitation in existing CFEs. The existing CFE libraries also failed to include some of the novel CFE approaches that avoid post-hoc methods, such as CounterNet [7].

Chapter 3

Background

This section provides an overview of the general algorithmic recourse problem and presents the problem formulation and objective function for each of the recourse algorithm included in ReLax.

3.1 Counterfactual Explanations

Given a fixed predictive model, commonly assumed to be a binary classifier, $f : x \to \{0, 1\}$, which needs to be explained. And we denote the input data of size N as $D = (x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$. And we denote the counterfactual explanation x^{cf} for a input data point x^f, where $f(x^{cf}) \neq f(x^f))$. Assume $d(x^f, x^{cf})$ denotes the choice of metric (e.g. L1 Norm/L2 Norm, cost) we want to optimize based on the counterfactual explanation algorithm and $y = 1$ indicates the desirable outcome, we can define the counterfactual explanations as follows:

$$x^{cf} = argmin_{x^{cf} \in \mathcal{A}}\ d(x^f, x^{cf})$$
$$\text{s.t.} \quad f(x^{cf}) = 1 - f(x^f)$$

where $\mathcal{A}$ is the set of admissible shift from x^f. For example, we can define attributes in the dataset to be immutable, such as race and sex, to force the x^{cf} only make changes to actionable features in x^f. But in practice, we modify the objective function and solve it by iteratively solving for x^{cf} and increasing λ until a sufficiently close solution is found. The definition of this training objective is similar to Wachter et al. [33].

$$x^{cf} = argmin_{x^{cf} \in \mathcal{A}}\ d(x^f, x^{cf}) + \lambda \cdot \mathcal{L}(f(x^{cf}), 1 - f(x^f))$$

Building on [33], other literature introduce some additional terms to improve the quality of counterfactuals. Mothilal et al. [24] introduce a determinantal point processes (DPP) to capture the

diversity of x^{cf}.

$$dpp\_diversity = det\left(\frac{1}{1 + d(x_i^{cf}, x_j^{cf})}\right)$$

Then combining the validity loss term and diversity term, the loss function becomes:

$$x^{cf} = argmin_{x^{cf} \in \mathcal{A}}\ d(x^f, x^{cf}) + \lambda \cdot \mathcal{L}(f(x^{cf}), 1 - f(x^f)) - \lambda \cdot dpp\_diversity(x_1^{cf}, ..., x_k^{cf})$$

Where k is the number of counterfactuals generated.

Van Looveren and Klaise [31] adopt autoencoder (AE) to evaluate L2 reconstruction error of x^{cf}, thus they introduce autoencoder loss term that penalizes out-of-distribution counterfactual instances to enforce the data manifold closeness. Combining the proximity term, validity term, and autoencoder loss term, the objective function becomes:

$$x^{cf} = argmin_{x^{cf} \in \mathcal{A}}\ \lambda_1 \cdot d(x^f, x^{cf}) + \lambda_2 \cdot \mathcal{L}(f(x^f), 1 - f(x^{cf})) + \lambda_3 \cdot \mathcal{L}(AE(x^f), AE(x^{cf}))$$

Guo et al. [7] propose a novel method of generating CFEs during training the ML model. Therefore, CounterNet includes predictive loss in the objective function for training the classifier.

$$x^{cf} = argmin_{x^{cf} \in \mathcal{A}}\ \mathcal{L}_{pred}(y - f(x^f)) + d(x^f, x^{cf}) + \lambda \cdot \mathcal{L}(f(x^{cf}), 1 - f(x^f))$$

CounterNet uses two-stage gradient updates during optimization, which use the gradient of $\mathcal{L}_{pred}$ is used to improve predictive accuracy, then use the gradient of d and $\mathcal{L}$ to generate CF explanation.

Chapter 4

ReLax

In this section, we begin by introducing the JAX library, which was utilized to construct ReLax. Next, we present an overview of ReLax's architecture. Following that, we briefly discuss the counterfactual explanation methods in ReLax and introduce a range of measures that are utilized to assess the quality of the generated counterfactual explanations. Finally, we demonstrate the usability of ReLax's API by showcasing how to customize data loading and implement new counterfactual explanation methods.

4.1 JAX Library

JAX[1] is an open-source library developed by Google for high-performance numerical computing and machine learning research. It is designed to provide efficient and scalable tools for building and training machine learning models, especially those involving large-scale, data-parallel computations. A fundamental feature of JAX is that it allows users to transform Python functions, including jax.grad, jax.vmap, and jax.jit.

- jax.grad is a function that provides automatic differentiation for any Python function. Given a function that computes a scalar output from an input array, jax.grad returns a new function that computes the gradient of the scalar output with respect to the input array. This allows for easy computation of gradients for use in optimization algorithms such as gradient descent.

- jax.jit (Just-In-Time) compilation is one of the key features of JAX that enables it to achieve fast computation speeds. JIT compilation compiles Python code on the fly, optimizing it for efficient execution on CPU/GPU/TPU. With JIT compilation, JAX is able to achieve performance improvements of up to several orders of magnitude over pure Python or NumPy code.

- JAX.vmap is a function that applies a function to a batch of inputs in a vectorized manner. This is useful for applications where the same function needs to be applied to a large number

of inputs, such as generating counterfactuals for each of the datapoint in the input.

ReLax leverages these features of jax to provide counterfactual explanation methods that are highly efficient and scalable. Specifically, ReLax utilizes jax.vmap/jax.pmap to enable parallel generation of counterfactual examples, which makes it faster and more efficient to generate multiple examples. Additionally, ReLax's implementation of counterfactual explanation methods in jax can be accelerated on CPU, GPU, or TPU via jax.jit, making it even faster than prior works.

4.2 ReLax's Architecture

The purpose of ReLax is to provide a platform for people to evaluate state-of-art counterfactual explanation methods proposed in prior works using standardized metrics. Figure 4.1 demonstrates the simplified software architecture of ReLax, where blue boxes represent the main module involved in ML model training and counterfactual examples generation, the orange box represents input data, and the green boxes represent the output.

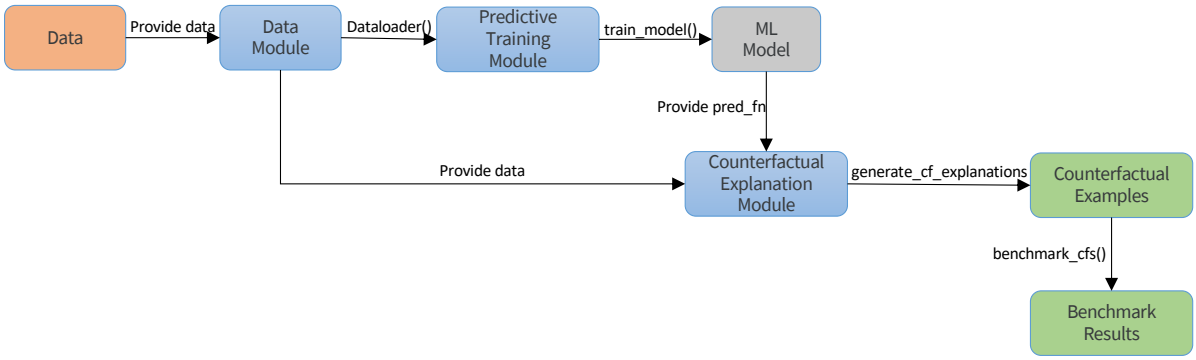

Figure 4.1: ReLax's Architecture

Data Module loads the data file, and prepares the data for ML model training. Users can set some features as immutable, such as race and sex, such that the counterfactual explanation will avoid changing them during counterfactual generation. To improve performance, users can normalize continuous features and encode the categorical features. In the Predictive Training Module,

users can define the model structure and the optimization procedure. With the number of epochs and batch size defined, users can train the ML model by simply calling `train_model()`. In the Counterfactual Explanation module, users can choose implemented recourse algorithms and define the hyperparameters for the counterfactual explanation. With the predictive function and data as input, users can generate a counterfactual for each data instance by calling `generate_cf_explanations()`. Finally, users can use `benchmark_cfs()` to evaluate the quality of the counterfactuals with the standardized metrics.

4.3 Counterfactual Explanation Methods

VanillaCF: Wachter et al. [33] propose VanillaCF, which generates counterfactual explanations by minimizing the distance (l1-norm) between the input data point and the counterfactual example and encouraging the classifier prediction of the counterfactual towards the desired class using gradient descent.

DiverseCF: Mothilal et al. [24] suggests DiverseCF, which aims to generate a diverse set of counterfactual explanations by maximizing the determinant of the kernel matrix of the generated counterfactuals. Regarding Solving the optimization problem, DiverseCF finds the solution that trade-off between diversity and proximity via gradient descent.

ProtoCF: Van Looveren and Klaise [31] introduce ProtoCF, which optimizes for consistency by using class prototypes. The use of prototypical examples encourages resulting counterfactual instances be realistic. ProtoCF uses a gradient-free optimization algorithm known as fast iterative shrinkage-thresholding algorithm (FISTA).

CounterNet: Guo et al. [7] propose CounterNet, which makes a novel departure from post-hoc

method of generating CF explanations by optimizing the predictive model and generating the CFE of a datapoint simultaneously during training.

4.4 ReLax's API

Customizable Data Load. To enable users to explore the quality of counterfactual explanations on different datasets, ReLax library provides an easy-to-use API for users to load customized datasets. Figure 4.2 shows an example of loading a customized dataset in ReLax library.

```
from relax.data import TabularDataModuleConfigs, TabularDataModule, load_data

data_config = TabularDataModuleConfigs(
    # The name of the dataset
    data_name="custom",
    # The directory of the data
    data_dir=".../custom.csv",
    # List all continuous variables
    continous_cols=[...],
    # List all categorical (discrete) variables
    discret_cols=[...],
    # List all immutable features that we do not wish to change
    imutable_cols=[...]
)
```

Figure 4.2: Pseudo-implementation of customized data load

Extensibility. To encourage users to design and implement their own counterfactual explanation methods and help the field to grow, ReLax offers an API that facilitates the implementation of counterfactual explanation methods, allowing users to seamlessly develop their own approaches. Once implemented, users can leverage the platform to benchmark their methods against state-of-the-art recourse techniques available in the ReLax library. In Figure 4.3, we demonstrate how an implementation of counterfactual explanation method is structured, and how to use jax.vmap/jax.pmap to vectorize the function to generate counterfactual for all input data instead of one data point.

```python
from relax.import_essentials import *
from relax.methods.base import BaseCFModule

# All recourse implementation need to inherit from BaseCFModule interface
class CustomCF(BaseCFModule):
    name = "CustomCF"

    def __init__(
        self,
        configs: dict | VanillaCFConfig = None
    ):

    # Generate a counterfactual example
    def generate_cf(
        self,
        # `x` shape: (k,), where `k` is the number of features
        x: jnp.ndarray,
        # Predictive function
        pred_fn: Callable[[jnp.DeviceArray], jnp.DeviceArray],
    ) -> jnp.DeviceArray:
        [...]
        return counterfactual example

    # Utilize jax.vmap/jax.pmap to vectorize the generate_cf
    def generate_cfs(
        self,
        # `x` shape: (b, k), where `b` is the batch size, `k` is the number of features
        X: jnp.DeviceArray,
        # Predictive function
        pred_fn: Callable[[jnp.DeviceArray], jnp.DeviceArray],
        is_parallel: bool = False,
    ) -> jnp.DeviceArray:

        def _generate_cf(x: jnp.DeviceArray) -> jnp.ndarray:
            return self.generate_cf(x, pred_fn)

        return (
            # Returns an array of same dimension as 'X' with the counterfactuals
            jax.vmap(_generate_cf)(X) if not is_parallel else jax.pmap(_generate_cf)(X)
        )
```

Figure 4.3: Pseudo-implementation of CustomCF

4.5 Evaluation Metrics for Counterfactual Explanation Methods

Validity. We define validity as the fraction of CF explanation methods output x' are valid counterfactual examples. That is, their predicted outcomes by the classifier f are different compared to the original input data point x. High validity is desirable, as it implies the method's effectiveness at generating valid CF examples.

$$Validity = \frac{|f(x') = 1 - y|}{n}$$

Proximity. We define proximity as the $L1$ norm distance between input data x and generated counterfactual x' divided by the number of features. Low proximity is desirable for CF examples, because having fewer modifications in the features makes CF examples to be easy to act upon.

$$Proximity = \frac{1}{nd} \sum_{i=1}^{n} \sum_{j=1}^{d} ||x_i'^{(j)} - x_i^{(j)}||_1$$

Sparsity. We define sparsity as the $L0$ norm distance between input data x and generated counterfactual x'. It has been argued that shorter explanations are more comprehensible to people in prior works [22, 25], therefore low sparsity and proximity are desirable for CF instances.

$$Sparsity = \frac{1}{nd} \sum_{i=1}^{n} \sum_{j=1}^{d} ||x_i'^{(j)} - x_i^{(j)}||_0$$

Manifold distance. We define data manifold distance as the $L1$ norm distance from the counterfactual to k nearest training data points. Data manifold closeness constraint forces the generated counterfactual to be near the input data points and adheres to observed correlations among the features. We use $k = 1$ to remain consistent with prior work [7, 32]. Low manifold distance is

desirable as closeness to the training data manifold indicates realistic CF explanations [32, 31].

Runtime. By measuring the average runtime for each CFE method to generate CF examples for all input datasets, we evaluate the effectiveness of each CFE method as well as the effectiveness of our CFE library in benchmarking CFE methods compared to prior works.

Chapter 5

Experiment

We conducted extensive benchmarking on four state-of-the-art counterfactual explanation methods available in our ReLax library, using four different large-scale real-world datasets, to validate the effectiveness of our library. The benchmark results are presented in Table 5.2. Additionally, we compared the performance of our library with that of CARLA [27], a benchmark library for CF explanation similar to ReLax.

5.1 Experiment Results and Analysis

5.1.1 Experiment Setup

Datasets. We evaluate methods on eight real-world datasets:

- Adult[16] dataset was extracted from the census bureau database from 1994, consisting of 48,842 instances, where 32,561 of them are selected for training. The classifier aims to determine whether an individual makes over 50K USD a year.

- Credit[35] dataset was obtained from real cardholders' credit risk data in Taiwan, consisting of 30,000 instances. The classifier uses historical payments to predict the default of payment.

- HELOC[6] dataset is an anonymized dataset of Home Equity Line of Credit (HELOC) applications made by real homeowners, with 10,459 instances. A HELOC is a line of credit typically offered by a bank as a percentage of home equity. The classifier uses information of the applicants to determine whether they will repay their HELOC account within 2 years.

- OULAD[17] dataset comprises 32,593 instances and is a subset of the 2013 and 2014 OU student data. It includes both demographic data and interaction data of the students. The classifier determines whether MOOC students drop out or not, based on their online learning logs.

Dataset	Task	Size	# Continuous	# Categorical
Adult	Predicts income ($50K or not)	32,561	2	6
Credit	Predicts payment default (yes or no)	30,000	20	3
HELOC	Predicts credit qualification (yes or no)	10,459	21	2
OULAD	Predicts MOOC student dropout (yes or no)	32,593	23	8

Table 5.1: Summary of datasets used for evaluation

Software and Hardware Specification. Our benchmark results are obtained on a Debian-10 Linux-based Deep Learning Image with CUDA 11.0 on the Google Cloud Platform. We use Python(v3.7), jax(v0.4.8), numpy (v1.19.3), pandas (v1.1.1) and scikit-learn (v0.23.2) for the implementations. The CounterNet network was trained using an NVIDIA Tesla V100 GPU and an 8-core Intel machine. Meanwhile, the counterfactual generation of the four baselines was performed on a 16-core Intel machine equipped with 64 GB of RAM. Evaluation of the results was also conducted on the same 16-core machine.

5.1.2 Results

Validity. We evaluate the validity of all four counterfactual explanation methods in ReLax, the results are displayed in Table5.2. We observe that the CounterNet method consistently outperforms other methods, with 100% validity across all four datasets. VanillaCF and ProtoCF also perform well in terms of validity. The result suggests CounterNet, VanillaCF, and ProtoCF are effective in generating valid counterfactual examples.

Proximity & Sparsity. The comparison of proximity and sparsity achieved by all CF methods is also presented in Table 5.2, with CounterNet showing superior performance in proximity, except for the Credit dataset, and VanillaCF and DiverseCf performing well in terms of sparsity. This shows that CounterNet generates counterfactuals that are easy to act upon, while VanillaCF and DiverseCf are better at generating counterfactuals that are more comprehensible.

Manifold Distance. Additionally, Table 5.2 reports on the data manifold distance achieved by the methods. The results show that CounterNet consistently achieves low data manifold distance on Credit, HELOC, and OULAD datasets. The result shows that CounterNet generates realistic counterfactual that adheres to observed correlations among the features in the input data.

Table 5.2 summarizes the performance of four counterfactual explanation methods, namely VanillaCF, DiverseCF, ProtoCF, and CounterNet, on four different datasets, namely Adult, Credit, HELOC, and OULAD. The performance is evaluated based on three metrics, namely validity, proximity/sparsity, and manifold distance.

Table 5.2: Summary of benchmark results.

Datasets	Metrics	VanillaCF	DiverseCF	ProtoCF	CounterNet
Adult	Validity	0.76	0.54	0.59	**1.00**
	Proximity	.202	.276	.250	**.196**
	Sparsity	**.556**	.662	.648	.644
	Manifold Distance	**0.57**	1.16	0.62	0.64
Credit	Validity	0.92	**1.00**	0.92	**1.00**
	Proximity	**.123**	.264	.197	.132
	Sparsity	**.841**	.918	.855	.912
	Manifold Distance	0.59	1.68	0.82	**0.56**
HELOC	Validity	**1.00**	0.90	**1.00**	**1.00**
	Proximity	.154	.149	.168	**.125**
	Sparsity	.883	**.434**	.805	.740
	Manifold Distance	0.71	1.34	**0.56**	**0.56**
OULAD	Validity	**1.00**	0.68	**1.00**	**1.00**
	Proximity	.101	.117	.107	**.075**
	Sparsity	.762	**.565**	.754	.725
	Manifold Distance	1.30	2.51	1.46	**0.87**

5.2 Performance Comparison

We compare the runtime with Carla, a benchmark library for CF explanations. Figure 5.1 compares the runtime performance of ReLax and Carla using the VanillaCF counterfactual explanation

method on three different datasets: Adult, HELOC, and OULAD. The results show that VanillaCF on ReLax significantly outperforms VanillaCF on Carla in terms of runtime on all three datasets, with an average speedup of 134.37 times. Notably, on the Adult dataset, VanillaCF on ReLax runs 258.09 times faster than VanillaCF on Carla. This speedup factor varies across datasets but is significant in all cases.

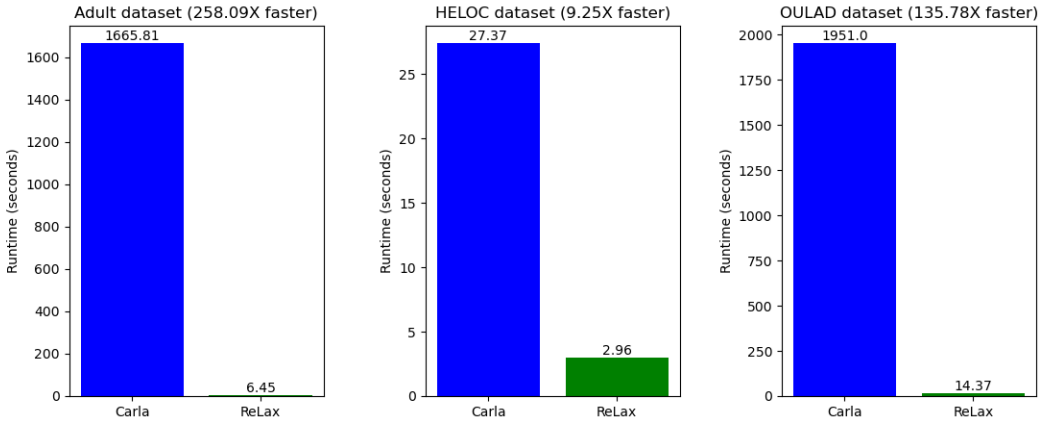

Figure 5.1: Comparison of Carla's and ReLax's VanillaCF performance

Chapter 6

Conclusion

In conclusion, ReLax provides a comprehensive and efficient python library for generating counterfactual and recourse explanations in machine learning. Our library includes 4 counterfactual explanation methods implemented using jax, with quantifiable evaluation metrics for standardized and transparent comparisons of different data sets. Our work leverages the features of jax to provide faster and more efficient generation of counterfactual examples, which can process larger data sets than prior works.

In future work, we plan to extend the benchmarking to larger datasets with more diverse domains, including data sets containing more than 1 million instances, to evaluate the scalability of our library and the robustness of counterfactual explanation methods. We also plan to incorporate additional counterfactual explanation methods into ReLax as the baseline for users to compare their own counterfactual explanation methods. Finally, we will continue to improve and refine the ReLax library to provide a user-friendly and accessible tool for benchmarking and developing counterfactual explanation methods.

Bibliography

[1] Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. (2018). JAX: composable transformations of Python+NumPy programs.

[2] Chen, C., Li, O., Tao, D., Barnett, A., Rudin, C., and Su, J. K. (2019). This looks like that: deep learning for interpretable image recognition. *Advances in neural information processing systems*, 32.

[3] Datta, A., Sen, S., and Zick, Y. (2016). Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In *2016 IEEE symposium on security and privacy (SP)*, pages 598–617. IEEE.

[4] Doersch, C. (2016). Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*.

[5] Fatima, M., Pasha, M., et al. (2017). Survey of machine learning algorithms for disease diagnostic. *Journal of Intelligent Learning Systems and Applications*, 9(01):1.

[6] FICO (2021). Home Equity Line of Credit (HELOC) Dataset. https://community.fico.com/s/explainable-machine-learning-challenge?tabset-158d9=3.

[7] Guo, H., Nguyen, T. H., and Yadav, A. (2021). Counternet: End-to-end training of counterfactual aware predictions. *arXiv preprint arXiv:2109.07557*.

[8] Joshi, S., Koyejo, O., Vijitbenjaronk, W., Kim, B., and Ghosh, J. (2019). Towards realistic individual recourse and actionable explanations in black-box decision making systems. *arXiv preprint arXiv:1907.09615*.

[9] Kanamori, K., Takagi, T., Kobayashi, K., and Arimura, H. (2020). Dace: Distribution-aware counterfactual explanation by mixed-integer linear optimization. In *IJCAI*, pages 2855–2862.

[10] Karimi, A.-H., Barthe, G., Schölkopf, B., and Valera, I. (2020a). A survey of algorithmic recourse: definitions, formulations, solutions, and prospects. *arXiv preprint arXiv:2010.04050*.

[11] Karimi, A.-H., Schölkopf, B., and Valera, I. (2021). Algorithmic recourse: from counterfactual explanations to interventions. In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 353–362.

[12] Karimi, A.-H., Von Kügelgen, J., Schölkopf, B., and Valera, I. (2020b). Algorithmic recourse under imperfect causal knowledge: a probabilistic approach. *Advances in neural information processing systems*, 33:265–277.

[13] Keane, M. T. and Smyth, B. (2020). Good counterfactuals and where to find them: A case-based technique for generating counterfactuals for explainable ai (xai). In *Case-Based Reasoning Research and Development: 28th International Conference, ICCBR 2020, Salamanca, Spain, June 8–12, 2020, Proceedings 28*, pages 163–178. Springer.

[14] Khandani, A. E., Kim, A. J., and Lo, A. W. (2010). Consumer credit-risk models via machine-learning algorithms. *Journal of Banking & Finance*, 34(11):2767–2787.

[15] Koh, P. W. and Liang, P. (2017). Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR.

[16] Kohavi, R. and Becker, B. (1996). Adult dataset. uci machine learning repository.

[17] Kuzilek, J., Hlosta, M., and Zdrahal, Z. (2017). Open university learning analytics dataset. *Scientific data*, 4(1):1–8.

[18] Luna, A., Bello, M., Hernández, A., and Bonilla, E. (2020). Prediction of missing values in adult data set of uci machine learning: A case of study. *Zeszyty Naukowe Warszawskiej Wyższej Szkoły Informatyki*, 14(22):7–21.

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

[20] Mahajan, D., Tan, C., and Sharma, A. (2019). Preserving causal constraints in counterfactual explanations for machine learning classifiers. *arXiv preprint arXiv:1912.03277*.

[21] Miller, T. (2017). Explanation in artificial intelligence: Insights from the social sciences. arxiv 2017. *arXiv preprint arXiv:1706.07269*.

[22] Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38.

[23] Mohammadi, K., Karimi, A.-H., Barthe, G., and Valera, I. (2021). Scaling guarantees for nearest counterfactual explanations. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pages 177–187.

[24] Mothilal, R. K., Sharma, A., and Tan, C. (2020). Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pages 607–617.

[25] Naumann, P. and Ntoutsi, E. (2021). 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*, pages 682–698. Springer.

[26] Nemirovsky, D., Thiebaut, N., Xu, Y., and Gupta, A. (2022). Countergan: Generating counterfactuals for real-time recourse and interpretability using residual gans. In *Uncertainty in Artificial Intelligence*, pages 1488–1497. PMLR.

[27] Pawelczyk, M., Bielawski, S., Heuvel, J. v. d., Richter, T., and Kasneci, G. (2021). Carla: a python library to benchmark algorithmic recourse and counterfactual explanation algorithms. *arXiv preprint arXiv:2108.00783*.

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

[29] Spangher, A., Ustun, B., and Liu, Y. (2018). Actionable recourse in linear classification. In *Proceedings of the 5th workshop on fairness, accountability and transparency in machine learning*.

[30] Tashea, J. (2017). Courts are using ai to sentence criminals. that must stop now. *Wired Magazine, Digital Article*.

[31] Van Looveren, A. and Klaise, J. (2021). 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.

[32] Verma, S., Hines, K., and Dickerson, J. P. (2022). Amortized generation of sequential algorithmic recourses for black-box models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8512–8519.

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

[34] Yang, F., Alva, S. S., Chen, J., and Hu, X. (2021). Model-based counterfactual synthesizer for interpretation. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 1964–1974.

[35] Yeh, I.-C. and Lien, C.-h. (2009). The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert systems with applications*, 36(2):2473–2480.

Xinchang XIONG

xqx5103@psu.edu

The Pennsylvania State University, UNIVERSITY PARK **CLASS OF 2023**

Computer Science （Schreyer Honors College） **Dean's list** for all semesters

Core Courses: Electronic Circuit Design, Operating System Design, Object Oriented programming, Data Structure and Algorithm, Cryptography, Linear Programming, Applied Statistics in Science, Artificial Intelligence, Game Theory, Machine Learning

ACADEMIC PROJECTS

Machine Learning Project: Counterfactual Explanation Library in JAX

Researcher **Aug 2022 - Present**

- Instructed by PSU Assistant Prof. Amulya Yadav, create an open-source counterfactual explanation methods library with JAX, which trains machine learning models, creates counterfactual examples, and evaluates the accuracy of the examples.
- Apply python skills and knowledge of machine learning to implement machine learning models and counterfactual methods, and complete the documentation for the project.
- Study implementing machine learning with JAX, and the importance of counterfactual explanations in explaining the decision of AI to the public.

The Dynamic Memory Allocator Project

Team Leader **Feb 2021 - Mar 2021**

- Instructed by PSU Prof. Bhuvan Urgaonkar, led a team of 3, implemented and optimized the throughput and memory utilization of the memory allocator in a virtual environment in Linux.
- Programmed in C, debugged segmentation faults with GDB, and studied the throughput and memory utilization difference using different data structures for free lists, to build a better memory allocation structure.
- Achieved a better understanding of the underlying memory structure, and constructed an efficient memory allocator in a Linux environment.

OCT Based Scanning System to enable 3D Viewing, Sharing & Printing of Artworks

Researcher **May 2020 - June 2020**

- Joined Dr. Yi Yang, PSU Associate Professor's lab, engineered and presented a hybrid OCT scanning platform combined with SIFT image stitching algorithm to achieve macroscopic OCT (macro-OCT) imaging, the system enables large FOV, HD examination, and 3D reconstruction of the surface topography of paintings.
- Demonstrated the potential applications of the OCT by rendering OCT-captured data into 3D volumetric data that is compatible with 3D printing, and packaged them into AR/VR applications for Android, IOS, or PC platforms.
- Completed the thesis: *A Note on Macroscopic Optical Coherence Tomography Imaging Enabled 3D Scanning for Museum and Cultural Heritage Applications*, which was published in the *Journal of the American Institute for Conservation,* 27 Oct 2022; also released on 2 conferences: a) 2021 OSA Imaging and Applied Optics Congress; b) 2021 American Institute for Conservation (AIC)/Society for the Preservation of Natural History Collections (SPNHC) Joint Virtual Annual Meeting.

PSU Undergraduate Research Project: LIDAR and Sonar Enabled Electrical Wheelchair

Lab Assistant **Jan 2020 – May 2020**

- Worked for the laboratory led by Dr. Yi Yang, assisted in the project aiming at developing a smart wheelchair system which can inform visually impaired users about the spatial information around the wheelchair and warn of potential collisions with stereo sound.
- Worked in a team of 3, applied Arduino programming skills to code the sonar sensors system, then installed and tested the system, to insure it can get spatial information from LIDAR sensor and process the information into stereo sound information, and use sonar sensors to detect distance, then inform the user.
- The Prototype is completed, and ready for volunteer experiments to verify its reliability and user experience.

INTERNSHIP

Guangzhou Okay Information Technology Co., Ltd.

Assistant Graphic Engineer **May 2021 –Aug 2021**

- Worked in the R&D department, mainly using digital means to help collect information and build a virtual reality resource database to promote the protection of historical & cultural heritage in Guangdong-Hong Kong-Macao Greater Bay Area, and strengthen the activation and utilization of Lingnan cultural heritage.
- Completed 6 VR models of intangible cultural heritage and related virtual scenes according to the acquired 3D data, using the 3D drawing technology based on graphics and images, to help to inherit and preserve the culture.

PROGRAMMING SKILLS

Mastered: Python, Java, C, Arduino Programming, R Studio, MatLab, Lambda Calculus, JAX, Machine Learning

Intermediate: Multisim Circuit Simulation, Intel Quartus Prime, SQL