Explaining deep neural networks using counterfactuals

A PROJECT REPORT
SUBMITTED IN PARTIAL FULFILMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
Master of Technology
IN
Faculty of Engineering

BY

Mansi Ram Shivani

Computer Science and Automation
Indian Institute of Science
Bangalore – 560 012 (INDIA)

May, 2024

Declaration of Originality

I, **Mansi Ram Shivani**, with SR No. **04-04-00-10-42-19-1-16781** hereby declare that the material presented in the thesis titled

Explaining deep neural networks using counterfactuals

represents original work carried out by me in the **Department of Computer Science and Automation** at **Indian Institute of Science** during the years **Years**.
With my signature, I certify that:

- I have not manipulated any of the data or results.
- I have not committed any plagiarism of intellectual property. I have clearly indicated and referenced the contributions of others.
- I have explicitly acknowledged all collaborative research and discussions.
- I have understood that any false claim will result in severe disciplinary action.
- I have understood that the work may be screened for any form of academic misconduct.

Date: 1/07/2021 Student Signature

In my capacity as supervisor of the above-mentioned work, I certify that the above statements are true to the best of my knowledge, and I have carried out due diligence to ensure the originality of the report.

Advisor Name: Dr. V. Susheela Devi Advisor Signature

DEDICATED TO

My parents

who have always been my strength

Acknowledgements

First of all I want to thank my advisor Dr. V. Susheela Devi for her support in the past one year. Working with her has been a great learning experience for me. I am indebted for everything I learnt from her. I thank her for motivating me to be on the right track always. I am very fortunate to have a guide like her.

My stay at IISc became more memorable because of my friends. I thank Aashish Tolambiya, Parth Gangar, Satya Mishra, Satyendra Yadav, Stanly Samuel and Virti Savla for being my family away from home. I am grateful to my batch mates, learning alongside them was a great experience.

Finally, a heartfelt thank you to my parents for giving me the hope and strength to follow my dreams. None of this would have been possible without their unconditional love for me and their belief in me abilities.

Abstract

The use of black-box models like deep neural networks in regulated areas like finance is problematic given the general and worldwide trends in privacy, data protection, safety, and security. Explanations can improve the autonomy of people subject to automated decisions and increase transparency. Deep-learning models are very output-focused. Areas such as investment decision-making, loan approvals, credit sanctions require high accuracy along with explainability. Research on counterfactual explanations(CE) for justifying a model's decision focuses on providing an alternate worldview closest to the actual scenario that would have given a different result. Along with explaining the decision, counterfactuals also provide an actionable roadmap to the users in case of an adverse decision. Existing algorithms use optimization algorithms to minimize an aggregated loss function to arrive at the counterfactual. We propose a coordinate descent-like algorithm that optimizes on categorical and numeric features separately. The proposed algorithm is model-agnostic and capable of dealing with categorical features.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter we introduce the field of explainable AI, counterfactuals and the motivation for our work.

Machine learning models are increasingly embedded in many aspects of daily life, such as healthcare, finance, and social media. These artificial neural network-based decision systems generalize the transmitted data and learn from it. Therefore, the connection between input and output is hidden. This situation creates difficulties in the application areas of these models in many aspects. Researchers have proposed various techniques for explaining ML models to stakeholders to make them worthy of human trust.

XAI provides the potential to provide stakeholders with insights into model behavior by using various methods such as feature importance scores, counterfactual explanations. Consider scenarios like loan and credit approval or deciding admission to a university. It is not enough to know the cause of an adverse decision. It is important to know what to do to obtain a better outcome in the future.

A counterfactual explanation is a practical and intuitive way to explain a single prediction of a black-box model. An explainability module generating actionable and feasible counterfactuals would propel the application of neural models in the regulated and restrictive domains. The purpose of counterfactuals can be twofold. They can be used to understand the model better, make it more interpretable to stakeholders before deployment. They can also be used as a way to provide actionable recourse to users in case the model hands out an adverse outcome. If the use case is that of increasing interpretability, care should be taken not to explain a black-box by another one. Algorithms used to increase interpretability should be deterministic

and self-explanatory. However, if the only purpose is to provide recourse, models that give the best results can be used without worrying about their own interpretability. The proposed algorithm can be used for both of these applications. The algorithm is a gradient descent-based method that also focuses on the actionability of generated counterfactuals. It does not use complex algorithms like neural networks or autoencoders to solve the problem. We take care of feasibility and actionability by being flexible to user constraints and staying close to the actual data distribution.

The remainder of the report is organized as follows: chapter 2 includes the background and related work. In chapter 3 we present our novel algorithm. Chapter 4 describes and reports the evaluation metrics. Chapter 5 contains conclusions and chapter 5 discusses the future work.

Chapter 2

Background and related work

In this chapter, we give a brief description of the different applications of explainablitity, types of XAI algorithms and existing ways of counterfactual generation.

2.1 Explainability stakeholders

Having an explainability module, along with the ML model, can serve several different purposes. The module should be implemented, keeping in mind the nature of the stakeholder. [7] have delved into how XAI is deployed and used in ML workflows. They have summarized salient directions for future research. The following are some of the uses for explainability modules in practice.

1. **Model debugging:** ML engineers and data scientists are the stakeholders here. Engineers often find it challenging to find out the reasons for poor model performance. XAI can be used to discover the faults in either training or the data used for training. We can also get an idea about the model's fairness and underlying biases and rectify them.

2. **Model transparency:** Automated decision-making is often applied in sensitive and regulated areas that tangibly affect humans. Explanations may be required to communicate predictions to a broader audience or to respond to consumer complaints. For example, when a bank rejects a customer's loan request, the customer has a right to know the reasoning behind the adverse decision.

3. **Model audit** In financial organizations, there is accountability to external regulatory bodies. ML models must go through audits before deployment in order to make sure they follow all regulations. Fairness, correctness and stability are some of the criteria the

model may be checked for. Explainability tools can also be used by regulatory bodies for auditing ML models and checking for any violations.

2.2 Taxonomies of XAI algorithms

XAI algorithms can be classified based on many different features. There are many survey papers like [2] that eloquently summarize research in XAI. We will discuss some of them below.

1. Based on scope: An algorithm can generate either global or local explanations. Local explanations focus on explaining individual data instances. It generates explanation g for data instance x $\in$ X. Global algorithms try to understand the model as a whole and generally take a group of inputs to generate one or more explanations. LIME (Local Interpretable Model-Agnostic Explanations), SHAP(SHapley Additive exPlanations), counterfactuals are some examples of local explanation algorithms. LIME attempts to understand model behavior by perturbing an input sample and finding a local linear approximation of the decision boundary. Concept Activation Vectors (CAV), Global Attribution Mapping, etc., are some of the global ones.

2. Based on methodology: Perturbation-based methods generate explanations by hitting the model with different inputs. They include LIME, RISE (Randomized Input Sampling for Explanations). Backpropagation or gradient-based methods utilize the information flow in the backward pass of a neural network to map the output to feature importance. Saliency maps and Grad-CAM are some examples.

3. Based on usage: Model-specific algorithms deal with the inner workings of the model to interpret the results. Model-agnostic techniques analyze the features and their relationship with the output. InTrees is a model-specific approach, whereas counterfactuals are generally model-agnostic.

2.3 Counterfactual Explanations

A counterfactual explanation describes a situation in the form: "If X had not occurred, Y would not have occurred". Humans have been historically using counterfactuals to communicate their thoughts and reasoning. Counterfactuals establish a causal relationship between the features and the decision of a model. A counterfactual explanation of a prediction describes the slight modification to the input feature that changes the prediction to a predefined output, different from the output of the input feature. The aim of a reasonable explanation is:

1. to help the user comprehend why a particular decision was reached.

2. to provide grounds to contest adverse decisions.

3. to provide an actionable recourse.

A naive approach for generating an explanation would be to perturb x randomly until we find a counterfactual. Instead, we can pose it as an optimization problem. Given an input feature x with output y, our goal is to find a new feature $x^{'}$ which is as close as possible to x but has a different predefined outcome. A simple loss function as given [8] would be:

$L(x, x^{'}, y^{'}, \lambda) = \lambda(f(x^{'}) - y^{'})^2 + d(x, x^{'})$

The loss measures how far the counterfactual's predicted outcome is from the predefined outcome and how far the counterfactual is from the instance x. Different distance functions can be used, but the paper uses Manhattan distance, weighted by the inverse median absolute deviation. An Adam optimizer can be used to solve the problem. However, the counterfactuals generated may not be ideal. It does not penalize unrealistic solutions, does not handle categorical features, and a CE with ten feature changes would be as preferred as that with a single change, i.e., it does not prefer sparse solutions. [1] describe a more well-rounded loss function that takes care of a few more things than the previous one. The loss function consists of four different objectives:

min x o(x) := min x $o_1(f(x), y^{'}), o_2(x, x^{'}), o_3(x, x^{'}), o_4(x, X_{obs})$

1. o_1 quantifies the distance between $f(x^{'})$ and $y^{'}$.

2. o_2 defines how far is the CE X' from the input x.
   $o_2(x, x^{'}) = \frac{1}{p}\Sigma_{j=1}^{p}\delta_G(x_j, x_j^{'})\epsilon[0, 1]$
   where δ_G is the Gower distance,

$$\delta_G(x_j, x_j^{'}) = \begin{cases} \frac{1}{R_j} \mid x_j - x_j^{'} \mid & \text{if } x_j \text{ is numerical} \\ I_{x_j \neq x_j^{'}} & \text{if } x_j \text{ is categorical} \end{cases}$$

   with R_j as the value range of feature j, extracted from the observed dataset. Gower distance is used because it can account for a mixed feature space consisting of both numerical and categorical inputs.

3. o_3 keeps track of how many features changed from x to $x^{'}$. This will give a preference to sparse solutions with less number of features changed.

4. o_4 measures the weighted average Gower distance between x and the k nearest observed data points x [1] , ..., x [k] $\in X_{obs}$ as an empirical approximation of how likely $x^{'}$ originates

from the distribution of x. This ensures that the generated counterfactuals make sense and are feasible.

Apart from treating CF generation as an optimization problem, a few novel approaches have been explored. FACE: Feasible and Actionable Counterfactual Explanations by [6] aims at finding high-density paths of change. It uses f -distance to quantify the trade-off between the path length and the density along this path, which can subsequently be minimized using a shortest path algorithm through a finite graph over the dataset. Model-Agnostic Counterfactual Explanations for Consequential Decisions (MACE) by [4] propose a novel algorithm that solves a sequence of satisfiability problems, where both the distance function (objective) and predictive model (constraints) are represented as logic formulae. MACE maps the nearest counterfactual problem into a sequence of satisfiability (SAT) problems and uses standard SMT (satisfiability modulo theories) solvers as black-box to solve it. [3] Joshi et al. 2019 propose a new framework to provide algorithmic recourse. They use a generative model like GAN or VAE to produce CF. They first obtain the latent encoding of our sample x and then optimize in the encoded space. Notice that, while this is a good approach if the target is just recourse, it does not help make the model interpretable as the approach itself is a black box.

The advantage of counterfactual explanations is that they are very easy to understand and report. They are mostly model-agnostic and do not need access to model weights. Most CE models need the decision boundary and either the dataset or the data distributions. The properties of counterfactual explanations like validity, proximity, diversity, sparsity and the time required for generation are some things that need to be carefully looked at to make it's deployement practical.

Chapter 3

Algorithm

In this chapter, we describe our algorithm in detail and highlight the novelties and advantages of our approach. We look into the various components of our algorithm.

3.1 ALGORITHM

The proposed algorithm draws inspiration from coordinate descent algorithms. The idea behind coordinate descent methods is straightforward. If f is a k-dimensional function, we can minimize f by successively minimizing each of the individual dimensions of f, while holding the values of f in the other dimensions fixed. Instead of minimizing for each dimension, we divide the k input features into two sets:

1. Categorical features
2. Numerical features

The algorithm given below works as follows:

1. First, we find k instances in the dataset D that as closest to our input query with respect to categorical variables only. The instances chosen are subject to some criteria. They should belong to the desired class. In addition, they should follow certain constraints such as immutability or non-actionability of certain features.
2. Once we have our set, we use each instance as a starting point of the gradient descent optimization problem. We run the optimization algorithm for a few iterations (2-5) calculating loss only with respect to the numerical variables this time.
3. The final instances make our counterfactuals. We inverse transform the dataset and display the results.

Algorithm 1: Generate counterfactuals

Result: A set of k counterfactuals CF

x = original input;

k = no. of counterfactuals;

D = dataset used to train the model;

f = model used to make the decision;

CF = {};

Freeze all numeric features;

Find k closest examples in D to input x.;

Put it in CF;

Unfreeze numeric features;

Freeze categorical features;

for $i \leftarrow 1$ **to** k **do**

  cf init = CF[i];

  $x^{'}$ = cf init;

  L = d(x,$x^{'}$) + (1 - f($x^{'}$));

  optimize($x^{'}$ w.r.t L);

  CF[i] = $x^{'}$;

end

Many existing works solve counterfactual generation as an optimization problem. A few of the problems encountered and our approach to solve them are discussed below:

1. The loss functions used for descent methods need to be convex. Hence this limits our choice of the distance function. This is not a problem for numerical features as some scaled version of L2 distance works well. On the other hand, handling of categorical distance is not straightforward. The proposed algorithm breaks down the problem into two parts. The first part finds k closest examples to the given query with respect to categorical features only. We are just iterating over the dataset to find such examples, so the distance functions can be as complex as possible. We can make use of distance metrics that are statistically sound for categorical features.

2. Encoding the constraints that the counterfactuals must follow in the loss function is complicated. Adding general contraints (For example: age of a person can only increase) to an optimization problem can be difficult. Any constraints that need to be considered

can be handled in this part of the algorithm. For example, race and gender are some of the immutable variables. So, we can consider examples that have matching attribute values for these features while finding our set. In order to account for logical constraints, we define certain attributes for our features:

(a) Mutability- This attribute can take values 'True' and 'false'. Sensitive features like gender and race should be kept immutable.

(b) Actionability- Actionabilty differs from mutability in the sense that actionability is subjective and user-specific. For some users, it may be easier to increase their hours_per_week, while others may find it convenient to increase their education. Actionability can take values 'none', 'any', 'same-or-increase', 'same-or-decrease'.

After finding a set of initial points, we run gradient descent for a few iterations on each of them to get our final CFs. During this process, we only update the numerical features. The categorical attributes remain constant. Convergence is faster because we choose the initial point closer to the final requirement. The loss function used is

$$L(x, x^{'}, c, \lambda) = \lambda(f(x^{'}) - c)^2 + d(x, x^{'})$$

where c is confidence that we are aiming for. And c need not always be 1. Having a higher value of confidence would require moving further away from the query instance.

3.2 Main contributions

We highlight the main novelties of our proposed algorithm below.

1. Most existing algorithms do not handle categorical features while others use distance measures meant for numerical features on label encoded dataset. Our algorithm is the first one that enables use of appropriate and statistically sound distance metrics for categorical features.

2. Handling numerical and categorical distances differently frees the categorical distance measure from the constraint of being differentiable.

3. We enforce constraints by having two attributes of 'Actionability' and 'Mutability' for each feature.

4. Finding initial points from the dataset instead of random assignment leads to faster convergence helps us stay true to the data distribution. Starting from points that are true

to the distribution and follow the required constraints makes the resulting counterfactuals feasible and actionable.

3.3 Distance measures for categorical features

Determining the similarity or distance among data objects is an essential part of many research fields such as statistics, data mining and machine learning. There are many measures available in the literature to define the distance between two numerical data objects. It is not easy to define such a metric to measure the similarity between categorical data objects since categorical data objects are not ordered. Only a few distance measures are available in the literature to find the similarities among categorical data objects.

There exist three types of distance measures for categorical variables:

1. those that fill the diagonal entries only.
2. those that fill the off-diagonal entries only.
3. those that fill both diagonal and off-diagonal entries

Some information about the dataset used in the formulae mentioned below:

1. N is the size of the dataset.
2. f(a) is the frequency of value a.
3. n is the number of values taken by the attribute

3.3.1 Overlap

Overlap dissimilarity measure is one of the most common distance measures used in the literature due to its simplicity. In overlap distance measure, a distance value of 0 is assigned for matches and 1 is assigned for mismatches. The overlap distance between two values X_k and Y_k of attribute A_k is defined as

$$D(X_k, Y_k) = \begin{cases} 1 & \text{if } X_k \neq Y_k \\ 0 & \text{otherwise} \end{cases}$$

3.3.2 Eskin

Eskin et al. proposed a normalization kernel for record-based network intrusion detection data. It assigns a weight of $\frac{2}{nk^2}$

$$D(X_k, Y_k) = \begin{cases} \frac{2}{nk^2} & \text{if } X_k \neq Y_k \\ 0 & \text{otherwise} \end{cases}$$

This measure gives more weight to mismatches that occur on attributes that take many values.

3.3.3 Inverse Occurrence Frequency (IOF)

The inverse occurrence frequency(IOF) measure also assigns the maximum value 0 to all matches. Dissimilarity on mismatches is assigned based on the frequency of values $f_k(X_k)$ and $f_k(Y_k)$. This measure assigns lower value to mismatches on more frequent values. The highest value for mismatches is attained when the values occur only once in the data set and the minimum value is attained when they are the only two values for the attribute and each occur equal number of times. The mathematical formulae for computing IOF similarity measure is given as,

$$D(X_k, Y_k) = \begin{cases} \log f_k(X_k) * \log f_k(Y_k) & \text{if } X_k \neq Y_k \\ 0 & \text{otherwise} \end{cases}$$

3.3.4 Occurrence Frequency (OF)

The occurrence frequency measure assigns the same value for matches as given by inverse occurrence frequency, but assigns reverse weighting of the IOF measure for mismatches. The OF measure assigns higher dissimilarity to less frequent values and lower dissimilarity to more frequent values.

$$D(X_k, Y_k) = \begin{cases} \log \frac{N}{f_k(X_k)} * \log \frac{N}{f_k(Y_k)} & \text{if } X_k \neq Y_k \\ 0 & \text{otherwise} \end{cases}$$

3.3.5 Goodall measure

This distance measure always gives one for mismatches but takes occurrence values into account for matches. $p(x_i)$ is the fraction of records in which the i^{th} feature takes on the value of x_i

$$D(X_k, Y_k) = \begin{cases} 1 & \text{if } X_k \neq Y_k \\ p(x_i)^2 & \text{otherwise} \end{cases}$$

It is a distance measure that evaluates diagonal entries only.

Chapter 4

Experiments

In this chapter, we look at the dataset and model used and the evaluation of the generated counterfactuals for different parameter combinations.

4.1 Dataset

Adult income dataset: The Adult income dataset is a binary classification dataset based on the 1994 US census data. The target is to find whether a person earns more than 50K USD per year or not. It contains both categorical and numeric variables as shown in Table 1. It has fourteen attributes describing demographics and other information.

4.2 Model

The model used for demonstrating the algorithm is a sequential model trained on adult income dataset for 10 epochs. Accuracy of the model plays an important role in the quality of generated counterfactuals. If accuracy is slow then chances are that most of the counterfactual initialization set (part 1 of the algorithm) would have a prediction far away from desired outcome.

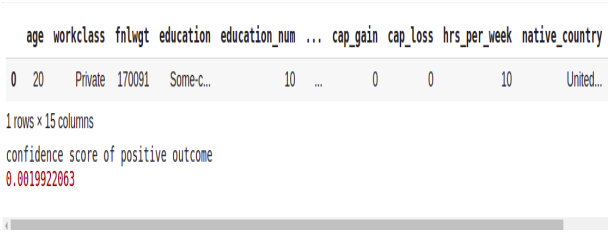
4.3 Counterfactuals

The input to the algorithm is an example on which the model gives an undesired, negative outcome. The factual example shown in Figure 1 has the probability of positive class as 0.009. This will be our input query.

After applying the algorithm, we generate a set of counterfactuals of user-defined length. Confidence indicates the model's probability of giving the desired outcome. In Figure 2 ,we can see the huge increase in the model probability for the desired class. It gives the desired

Attribute	Type
age	continuous
workclass	categorical
fnlwgt	continuous
education	categorical
education-num	continuous
marital-status	categorical
occupation	categorical
relationship	categorical
race	categorical
sex	categorical
capital-gain	continuous
capital-loss	continuous
hours-per-week	continuous
native-country	categorical

Table 4.1: Attributes of the Adult Income dataset



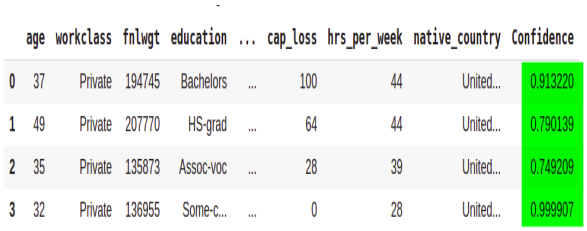
	age	workclass	fnlwgt	education	education_num	...	cap_gain	cap_loss	hrs_per_week	native_country
0	20	Private	170091	Some-c...	10	...	0	0	10	United...

1 rows × 15 columns

confidence score of positive outcome
0.8019922063

Figure 4.1: Input to the algorithm

outcome for all the generated counterfactuals.



	age	workclass	fnlwgt	education	...	cap_loss	hrs_per_week	native_country	Confidence
0	37	Private	194745	Bachelors	...	100	44	United...	0.913220
1	49	Private	207770	HS-grad	...	64	44	United...	0.790139
2	35	Private	135873	Assoc-voc	...	28	39	United...	0.749209
3	32	Private	136955	Some-c...	...	0	28	United...	0.999907

Figure 4.2: Generated counterfactuals using eskin measure

4.4 Evaluation

The fitness of a generated counterfactual can be subjective and user-dependent. Consequently, most evaluation practices used are qualitative in nature. While qualitative studies are useful, we need quantitative measures to be able to compare better newer techniques added into the literature. Counterfactuals can be evaluated quantitatively based on their desired properties like validity, proximity, sparsity, and diversity. We must evaluate our counterfactuals on a different measure than the one used for generating them to get a fair idea.

1. **Validity:** Validity is simply the percentage of counterfactual examples that fall in the desired class. A counterfactual must be of the opposite class than the factual. A good algorithm must generate valid counterfactuals.

$$Validity\% = \frac{|\{c \text{ in C s.t} f(c) > 0.5|\}}{k}$$

2. **Proximity:** It measures how close the counterfactuals are to the input. We must use a different distance measure than the one used during optimization. Evaluation can also be done on original values rather than the standardized or normalized version used during optimization.

$$Proximity = -\frac{1}{k}\sum_{i=1}^{k} d(c_i, x)$$

3. **Sparsity:** Sparsity measures the number of features in which a change is suggested by the counterfactual. In reality, it may be challenging to make a vast number of changes to get the desired outcome. Counterfactuals with fewer and more minor changes are preferable. It is a good criterion for analysis, even when we do not specifically optimize for it.

$$Sparsity = 1 - \frac{1}{kd}\sum_{i=1}^{k}\sum_{l=1}^{d} I_{c_i \neq x_i}$$

4. **Diversity:** The proposed algorithm generates a set of counterfactuals. For the output to be meaningful, the set should be diverse. Different things may work for different users, and hence diversity ensures that we have something for everyone. The metric can be

Results			
Metric	**k=2**	**k=6**	**k=10**
Validity	94.6%	97.2%	98.3%
Proximity	-15.6	-14.8	-13.4
Sparsity	0.52	0.50	0.48
Diversity	20.52	17.43	14.27
Count Diversity	0.535	0.534	0.539

Table 4.2: Metrics for different values of k

defined as the mean of summed distance between each pair or the mean of sparsity.

$$Diversity = \frac{1}{k(k-1)} \sum_{i=1}^{k-1} \sum_{i=j+1}^{k} d(c_i, c_j)$$

$$Count - diversity = \frac{1}{k(k-1)d} \sum_{i=1}^{k-1} \sum_{i=j+1}^{k} \sum_{l=1}^{d} I_{ci \neq xi}$$

We report the above metrics for our algorithm in Table 2 for different values of k. For all the report metrics higher values are better. Validity% improves slightly as k increases. [5] report a validity% of almost 100. Our validity% ranges between 94.6 to 98.3. They report similar scores for sparsity and count-diversity. Even though we don't explicitly minimize for sparsity, we get a good score on it. We can observe that sparsity decreases as k increases and diversity increases with k. These trends are in line with those reported by [5].

4.5 Gradient descent

We apply gradient descent using tensorflow's gradienttape(). The technique of finding the counterfactuals from the dataset itself using some distance measure is called Minimum Observable CF. We show the importance of finetuning the initial samples using descent algorithm. There is a huge bump in classification confidence. The confidence for positive class before and after gradient descent optimization is shown in Figure 3. The reason for this is that no model has a 100% accuracy. A sample's true label and model prediction may not always match. A good counterfactual has to balance between the data distribution and model prediction. Our two step approach tries to do that by choosing starting point close to the query but then also optimizing with respect to prediction score.

	Positive class probability of init CF	Positive class probability of final CF
0	0.0575	0.910
1	0.0637	0.790
2	0.6582	0.749
3	0.9900	0.990

Figure 4.3: Model prediction before and after gradient descent optimization

Results			
Metric	**RandomCF**	**MO**	**Our algorithm**
Validity	67.5%	65.5%	99.3%
Proximity	-22.4	-24.8	-13.9
Sparsity	0.25	0.43	0.52
Diversity	20.9	17.4	16.2
Count Diversity	0.63	0.58	0.53

Table 4.3: Comparison with some basic algorithms

4.6 Comparison to some basic approaches

To understand the importance of initial point, we compare results between our algorithm and if we just used random points from dataset as our initial points.
The technique of finding the counterfactuals from the dataset itself using some distance measure is called Minimum Observable CF.
Parameters used are k=4 and 4 iterations of gradient descent.

4.7 Distance measures

Using a different distance measure leads to different sets of initial and final CFs. As described in the above section, the various distance metrics are based on different underlying concepts. We compare their performance with respect to the confidence scores of final answer.

Figure 4 reports the confidence score of a positive outcome before and after applying gradient descent for three different metrics.

Overlap is a naive distance measure often used for categorical features. Eskin measures

	goodall_init	goodall_final	eskin_init	eskin_final	overlap_init	overlap_final
0	0.316215	0.854216	0.049582	0.913252	0.100329	0.983754
1	0.919205	0.898111	0.047308	0.924317	0.957216	0.902015
2	0.405087	0.969196	0.961309	0.929360	0.040000	0.924317
3	0.975587	0.854154	0.965219	1.000000	1.000000	1.000000

Figure 4.4: Model prediction before and after gradient descent optimization for different distance measures

Results			
Metric	**Overlap**	**Eskin**	**Goodall**
Validity	95.6%	96.8%	100%
Proximity	-13.6	-13.0	-57.7
Sparsity	0.51	0.50	0.22
Diversity	15.7	16.0	28.6
Count Diversity	0.53	0.50	0.71

Table 4.4: Metrics for different categorical distance measures

distance for off-diagonal entries and goodall measures distance for diagonal entries only. From the Table 3 we can infer that , diagonal measuring distance measures give good validity and diversity but not good proximity.

Chapter 5

Conclusions and Future Work

Counterfactuals are actionable feedback to individuals who have received undesirable outcomes from an automated decision system. They fulfill most of the criteria of a good explanation and have many practical applications. The idea is simple and based on how humans have communicated and explained themselves. However, a few things are to be considered while generating counterfactuals to make them useful for a user. Future work concerns deeply analyzing the different features that will generate sensible and diverse counterfactual explanations, making them robust and scalable, present them in a way best suitable for every individual user taking into account the feasibility in practice. The optimization problem of generating counterfactuals could be improved by developing different loss functions that closely resemble the real world. The proposed algorithm is capable of encoding simple constraints that the data must follow. However, there are inherent causal relationships among the attributes. It is essential to ensure that those also hold. Finding a way to incorporate causal constraints without using another black box (like Variational Autoencoders) is crucial. A good algorithm for generating counterfactuals will help users show trust in the models that may be used to take crucial decisions affecting their lives.

Bibliography

[1] Susanne Dandl, Christoph Molnar, Martin Binder, and Bernd Bischl. Multi-objective counterfactual explanations. In *International Conference on Parallel Problem Solving from Nature*, pages 448–469. Springer, 2020. 5

[2] Arun Das and Paul Rad. Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv preprint arXiv:2006.11371*, 2020. 4

[3] Shalmali Joshi, Oluwasanmi Koyejo, Warut Vijitbenjaronk, Been Kim, and Joydeep Ghosh. Towards realistic individual recourse and actionable explanations in black-box decision making systems. *arXiv preprint arXiv:1907.09615*, 2019. 6

[4] Amir-Hossein Karimi, Gilles Barthe, Borja Balle, and Isabel Valera. Model-agnostic counterfactual explanations for consequential decisions. In *International Conference on Artificial Intelligence and Statistics*, pages 895–905. PMLR, 2020. 6

[5] 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. 16

[6] Rafael Poyiadzi, Kacper Sokol, Raul Santos-Rodriguez, Tijl De Bie, and Peter Flach. Face: feasible and actionable counterfactual explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 344–350, 2020. 6

[7] Sahil Verma, John Dickerson, and Keegan Hines. Counterfactual explanations for machine learning: A review. *arXiv preprint arXiv:2010.10596*, 2020. 3

[8] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31:841, 2017. 5