Advanced Topics in AI: Counterfactual Explanations

Stefan Radic Webster
University of Bristol
hn19405@bristol.ac.uk

***Abstract*—Counterfactual explanations, a machine learning interpretation technique, identify the feature value changes required to flip a classification or achieve a desired prediction output. A range of methods to generate counterfactuals are explored and implemented firstly on a synthetic data set to demonstrate the distinct behaviours of each method, and then applied to a building energy efficiency data set. A method to identify the nearest neighbour, according to a distance metric, that satisfy predefined conditions is implemented on the data sets. The method proposed by Wachter et al [1] is explored that involves a trade-off between achieving the desired state and minimising the distance between the example point to be explained and generated counterfactual. Feasible and actionable counterfactuals (Poyiadzi et al [2]) are generated which is an alternative method based on the shortest feasible path via density-weighted steps. The code that implements these counterfactual generator methods for the two data sets is available from GitHub[1].**

I. Introduction

Counterfactuals fall under the subset of interpretable machine learning techniques that are example-based. Such example-based reasoning selects particular instances of the data set to provide explanations to machine learning model outputs or to help understand the underlying data distribution [3]. Example-based explanations are generally model-agnostic where the interpretation is decoupled from the machine learning providing model, explanation and representation flexibility [4].

Counterfactual explanations output a casual dependence statement of a hypothetical reality that contradicts the observation, "If X had not occurred, Y would not have occurred". In context of machine learning, a counterfactual explanation is provided for a prediction that describes the smallest, or most feasible, feature value manipulations that changes the prediction class or to some predefined desired output. A common application is for financial institutions that provide credit. An example counterfactual explanation for a loan application being denied being "if you had £1000 less debt, you would have qualified for the loan" [5].

As shown counterfactuals provide contrastive, human-friendly explanations and ideally focus on a small number of feasible input changes. However, counterfactuals can suffer from the 'Rashomon effect' where multiple, contradictory explanations can be produced for a single example [6].

II. Generating Counterfactuals

A simple approach to generate counterfactual explanations involves searching the feature space by trial and error until a prediction with the desired output is produced. This crude method is unlikely to produce an optimal explanation in terms of the smallest possible or most feasible change. Alternatively a counterfactual can found by a nearest neighbour search which returns an instance from the data set with the desired output that has the smallest distance from the example to be explained. The distance metric could be the Euclidean distance (l_2 norm) or as an improvement the Manhattan distance (l_1 norm) which promotes sparse solutions with only a small number of features changes which makes the explanations more human-understandable.

The nearest neighbour method can be improved by use of a loss function, which not only measures how far the counterfactual is from the example to be explained, but also how far the prediction outcome of the counterfactual is from the desired outcome. The loss function can be used when searching around the instance of interest, as suggested in the "Growing Spheres" method [7]. Alternatively, the loss function can be directly minimised with an optimisation algorithm as done by the approach suggested by Wachter et al [1] which is explored in this paper.

Issues with counterfactual explanations generated by use of a loss function arise when applied to real world examples. The closest possible world counterfactuals may be unachievable and/or the action to reach the desired state unfeasible based on the path to get there. As a result, Poyiadzi et al suggested a Feasible and Actionable Counterfactual Explanations (FACE [2]) algorithm which produces explanations based on the shortest distance path via density-weighted steps.

A. Minimising Loss Function

To generate counterfactuals for a trained machine learning model, the following loss function is minimised [1].

$$\arg \min_{x'} \max_{\lambda} \lambda(f_w(x') - y')^2 + d(x_i, x') \tag{1}$$

The term on the left quantifies the difference in the model prediction for the generated counterfactual x' and the predefined desired outcome y', which is normally set to 1. The second term is a distance metric measuring how far the counterfactual is from the instance x_i to be explained. The parameter λ balances the trade-off between meeting the prediction objective and not moving too far from the original point.

To generate a counterfactual, the authors suggest defining a tolerance ϵ and then iteratively solving for x' by minimising the loss function, using any suitable optimiser, and then increasing λ until the tolerance is satisfied (2). After setting the

[1]https://github.com/sradicwebster/counterfactuals

initial condition for the first optimisation, the subsequent initial condition is the counterfactual generated during the previous iteration. To prevent returning a poor counterfactual due to a local minima then the optimisation is ran multiple times with different initial conditions for x' and the counterfactual with the lowest loss selected.

$$|f_w(x') - y'| \leq \epsilon \tag{2}$$

The distance function used in (1) is of particular importance to generate suitable counterfactuals. The authors suggest as an initial choice to use the Manhattan distance weighted by the inverse median absolute deviation (MAD) of k features, over the set of points P.

$$MAD_k = median_{j \in P}(|X_{j,k} - median_{l \in P}(X_{l,k})|) \tag{3}$$

The distance function is the sum of all k feature-wise distances scaled by the inverse MAD_k.

$$d(x_i, x') = \sum_{k \in F} \frac{|x_{i,k} - x'_k|}{MAD_k} \tag{4}$$

This distance metric utilises the Manhattan distance which has the advantage over the Euclidean distance that it induces sparsity, meaning the counterfactual has fewer feature changes compared with the original. In addition it is robust to outliers due to its use of the median average.

B. Feasible and Actionable Counterfactual Explanations

The authors of FACE propose an algorithm to generate counterfactual examples situated in dense regions and are connected from the instance to be explained via a path of high-density. Utilising graph theory, the route from the example instance to counterfactual follows the shortest path whilst remaining in regions of high density. The vertices of the graph correspond to the training data instances with edges that connect the vertices that are within a certain user specified distance threshold ϵ. The edge weights w_{ij} encode a density weighted distance measure.

$$w_{ij} = w\Big(\hat{p}\Big(\frac{x_i + x_j}{2}\Big)\Big) \cdot d(x_i, x_j) \tag{5}$$
$$\text{when } d(x_i, x_j) \leq \epsilon$$

The density is estimated using a *Kernel Density Estimator* (KDE) $\hat{p}$, the distance function used was the Euclidean distance and the weight function $w(z) = -log(z)$.

The edge weights quantify a trade-off between the path length and the density along this path. With a graph constructed, a counterfactual can be generated that satisfies a desired prediction threshold by utilising a shortest path algorithm. In addition to specifying a distance and prediction threshold, the generated counterfactual can be constrained by setting of a density threshold to ensure the path is feasible. The Shortest Path First Algorithm (Dijkstra's algorithm) [8] is used on the resulting nodes and vertices of the graph that meet the various thresholds. The FACE algorithm [2] is available in Appendix A.

III. Experiments

The methods of generating counterfactuals: nearest neighbour, minimising loss function and FACE were implemented on a 2D synthetic data set and a building energy efficiency data set [9] from UCI Machine Learning Repository[2].

A. Synthetic Data Set

A support vector machine classifier was trained on the synthetic data set achieving a classification accuracy of 98%. Firstly, counterfactuals were produced by finding the nearest neighbour based on Euclidean distance that switched the predicted label. A counterfactual was then generated minimising the loss function (1) using a Nelder-Mead optimiser with a prediction threshold of 0.9 ($\epsilon = 0.1$). λ was initialised at 0.1 and incrementally increased by 0.1 every iteration until the prediction threshold was satisfied. As shown in figure 1. The counterfactual generated from minimising the loss function only changes one of feature values due to the use of the l_1 norm in the loss function which promotes sparsity. The point is further from the example point compared with the nearest neighbour counterfactual in order to achieve the desired 90% prediction confidence requirement.

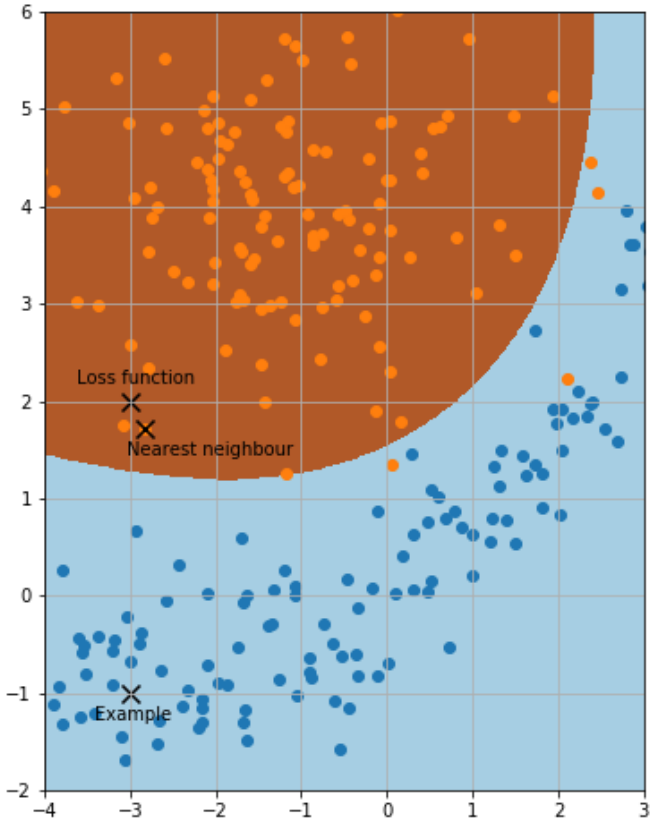

Fig. 1. Counterfactuals generated to switch the prediction class of the example using the nearest neighbour and loss function minimisation methods.

[2]https://archive.ics.uci.edu/ml/datasets/Energy+efficiency

The same synthetic data set was then used to generate counterfactuals for the same example instance and prediction threshold using the FACE algorithm. Figure 2 shows three different paths to generate counterfactuals by changing the distance and density thresholds. With a lower distance threshold the path taken to CF 2 follows more data points with smaller changes in feature values compared with CF 1. Increasing the density threshold, the path followed for CF 3 has a higher density of data set instances and generates a counterfactual with higher prediction confidence. With the prediction, density and distance thresholds as tuning parameters the transition from an example data point to an achievable counterfactual can be found via a feasible path.

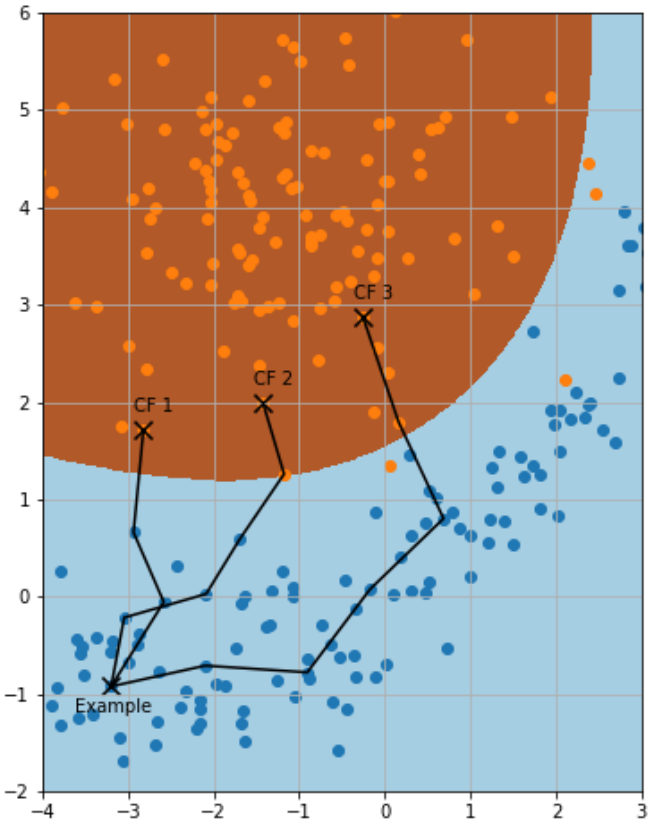

Fig. 2. Counterfactuals generated using the FACE algorithm showing the shortest path for different distance ϵ thresholds and density t_d thresholds. CF 1: $\epsilon = 1.2$ and $t_d = 0.008$. CF 2: $\epsilon = 1$ and $t_d = 0.008$. CF 3: $\epsilon = 1.2$ and $t_d = 0.015$. Prediction threshold $t_p = 0.9$.

B. Energy Efficiency Data Set

The data set was collected for a study to predict the heating and cooling load requirements of buildings (which determine energy efficiency) from building parameters such as wall, roof and glazing surface areas. The eight features in the data set were used predict the heating load. Although the variables in data set consisted of a limited number of possible values, as shown in table I, the variables are monotonic which permit numerical regression.

TABLE I
NUMBER OF POSSIBLE VALUES FOR INPUT (X) AND OUTPUT (Y) VARIABLES [9]

Representation	Variable	Number of possible values
X1	Relative compactness	12
X2	Surface area	12
X3	Wall area	7
X4	Roof area	4
X5	Overall height	2
X6	Orientation	4
X7	Glazing area	4
X8	Glazing distribution	6
y1	Heating load	586

The data was scaled to a zero to one range before linear regression, support vector machine regression and random forest regression models were constructed using scikit-learn. After evaluation using cross validation, the random forest regressor was chosen due to its superior prediction accuracy although none of the models were optimised.

Counterfactuals were generated to provide examples of how to reduce the predicted building heating load (by a user defined percentage) by the smallest or most feasible feature value changes. The three methods presented in this paper were used to generate counterfactuals that achieve a 10% reduction in predicted desired heating load for an example point. The nearest neighbour method used the Manhattan distance which is preferable over the Euclidean distance metric for high dimensional sparse data [10].

In order to implement the counterfactual generator proposed by Wachter et al, the loss function was changed because the model prediction for the generated counterfactual and predefined desired outcome in (1) are indistinguishable. The loss function used was:

$$\arg \min_{x'} \max_{\lambda} \lambda(f_w(x')) + d(x_i, x') \tag{6}$$

In addition, the optimiser used to minimise the loss function was changed to Sequential Least SQuares Programming (SLSQP) which supported bounded optimisation to constrain the scaled feature values to between zero and one.

Table II presents the generated counterfactuals by the three methods that reduce the heating load of an example data point by 10%.

TABLE II
COUNTERFACTUAL POINTS TO REDUCE EXAMPLE HEATING LOAD (Y1) BY 10%

Feature	Example	Nearest neighbour	Loss function	FACE
X1	0.98	0.76	0.98	0.82
X2	711	662	625	613
X3	368	417	306	319
X4	110	123	110	147
X5	3.5	7	3.5	7
X6	2	2	2	2
X7	0.4	0	0.4	0.25
X8	5	0	5	5
Output	**Example**	**Nearest neighbour**	**Loss function**	**FACE**
y1	28.4	24.8	25.1	25.5

The nearest neighbour method output a counterfactual that meets the desired 10% reduction to the predicted heating load. However, the point returned involves changing 7 out of the 8 features which may be undesirable. Alternatively, the counterfactual generated by the method proposed by Wachter et al only changed two feature values as the loss function involves a term to minimise the Manhattan distance between the example and the counterfactual. The FACE algorithm produced a different counterfactual alongside a path to reach the desired state via two other data points which would desirably represent a feasible path to reduce the heating load by 10%. This exemplifies the Rashomon effect as three different counterfactuals have been produced to explain the same example. As a result, context specific knowledge is required to evaluate the usefulness of the explanations.

IV. Conclusion

Counterfactual explanations provide a clear interpretation of machine learning models by showing the feature value changes required to flip the classification or achieve a predefined prediction. Such interpretations are human-friendly because they are contrastive to the current state and ideally focus on a small number of feature changes that are feasible to reach the desired state [6]. A crude method of generating a counterfactual from a data set is to simply identify the nearest neighbour that satisfies the predefined conditions. Alternative methods involve considering the distance between the example point to be explained and generated counterfactual, the density of the path travelled and the step changes in feature values to reach a counterfactual. The choice of which method to use is context dependant with the incorporation of domain knowledge essential to produce useful counterfactual explanations.

References

[1] S. Wachter, B. Mittelstadt, and C. Russell, "Counterfactual Explanations Without Opening the Black Box: Automated Decisions and the GDPR," *SSRN Electronic Journal*, 2017.

[2] R. Poyiadzi, K. Sokol, R. Santos-Rodriguez, T. De Bie, and P. Flach, "FACE: Feasible and actionable counterfactual explanations," in *AIES 2020 - Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 2020.

[3] C. Fernández-Loría, F. Provost, and X. Han, "Explaining Data-Driven Decisions made by AI Systems: The Counterfactual Approach," 1 2020. [Online]. Available: http://arxiv.org/abs/2001.07417

[4] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why should i trust you?" Explaining the predictions of any classifier," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[5] C. Rudin, "Please Stop Explaining Black Box Models for High Stakes Decisions," *NIPS 2018*, 2018.

[6] C. Molnar, "Interpretable Machine Learning. A Guide for Making Black Box Models Explainable." *Book*, 2019.

[7] T. Laugel, M. J. Lesot, C. Marsala, X. Renard, and M. Detyniecki, "Comparison-based inverse classification for interpretability in machine learning," in *Communications in Computer and Information Science*, 2018.

[8] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms , Second Edition*, 2001.

[9] A. Tsanas and A. Xifara, "Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools," *Energy and Buildings*, 2012.

[10] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional space," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2001.

Appendix A
FACE Counterfactual Generator Algorithm

Algorithm 1: FACE Counterfactual Generator

input : Data ($\boldsymbol{X} \in \mathbb{R}^d$), density estimator ($\hat{p} : \mathcal{X} \to [0,1]$), probabilistic predictor ($\boldsymbol{clf} : \mathcal{X} \to [0,1]$), distance function ($d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$), distance threshold ($\epsilon > 0$), weight function ($w : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{>=0}$), and conditions function ($c : \mathcal{X} \times \mathcal{X} \to \{True, False\}$).

output: Graph (V, E, W) and candidate targets ($\boldsymbol{I}_{CT}$).

```
/* Construct a graph.                                                    */
for every pair (x_i, x_j) in X do
    if d(x_i, x_j) > ε and c(x_i, x_j) is True then
        i ≁ j
        w_ij = 0
    else
        i ∼ j
        /* In this case we use Equation 2 (KDE). This should
           be adjusted for k-NN and ε-graph constructions by
           using Equation 3 and 4 respectively.                          */
        w_ij = w(p̂((x_i + x_j)/2)) · d(x_i, x_j)
/* Get a set of candidate targets.                                       */
I_CT = {}
for x_i in X do
    if clf(x_i) ≥ t_p and p̂(x_i) ≥ t_d then
        I_CT = I_CT ∪ i
```