Understanding Counterfactual Generation using Maximum Mean Discrepancy

Wei Zhang
Columbia University
New York, NY, USA

Brian Barr
Capital One
New York, NY, USA

John Paisley
Columbia University
New York, NY, USA

ABSTRACT

With the dramatic development of deep learning in the past decade, interpretability has been one of the most important challenges that often prevents neural networks from being applied to fields such as finance. Among many existing explainable analyses, counterfactual generation has become widely used for understanding neural networks and making tailored recommendations. However, few studies are devoted to providing quantitative measures for evaluating counterfactuals. In this paper, we propose a quantitative approach based on maximum mean discrepancy (MMD). We employ several existing counterfactual methods to demonstrate this proposed method on the MNIST image data set and two tabular financial data sets, Lending Club (LCD) and Give Me Some Credit (GMC). The results demonstrate the potential usefulness as well as the simplicity of the proposed method.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

KEYWORDS

datasets, neural networks, gaze detection, text tagging

ACM Reference Format:
Wei Zhang, Brian Barr, and John Paisley. 2022. Understanding Counterfactual Generation using Maximum Mean Discrepancy. In *3rd ACM International Conference on AI in Finance (ICAIF '22), November 2–4, 2022, New York, NY, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3533271.3561759

1 INTRODUCTION

In the past decade, deep learning algorithms have revolutionized data science approaches in many fields. Nevertheless, the lack of interpretability of most off-the-shelf deep learning models has been a bottleneck for their further adoption and application to fields that require explainability, such as finance. A deep classifier may be capable of making very accurate predictions, but neural networks (NN) usually fall short on explaining those decisions. Indeed, Su et al. [27] shows that often only small and meaningless random perturbations are required to easily deceive a well-trained NN classifier.

Counterfactual methods ultimately aim to understand a deep classifier by answering the question: "what changes could be made to an input so that the prediction outcome will be reversed." This can help data scientists understand what data properties led to a certain prediction and simultaneously propose personalized changes for aiming at a target class. This is especially useful in the financial field. For example, if a loan application is rejected by a deep classifier, counterfactual explanations can not only explain what properties the trained deep classifier looks for, but also propose tailored changes to reverse the outcome.

In this paper, we focus on evaluating the quality of counterfactual samples using maximum mean discrepancy (MMD) [14] in order to understand the quality of counterfactual proposals. MMD is a kernel-based statistical test used to determine how similar two distributions are. The MMD maps a distribution into a reproducing kernel Hilbert space (RKHS) and measures the distance between two distributions in that RKHS. MMD can be adopted as a loss function in different machine learning scenarios such as density estimation, generative models [10, 20], and also in invertible neural networks[3]. In contrast to generative adversarial networks (GANs) which require a solution to a complex min-max optimization problem, MMD criteria can be used as a simpler discriminator.

Previously, Ibrahim et al. introduced a global attribution method (GAM) to qualitatively explain the decision boundary of a classifier, with particular interest in neural networks. Their framework selects K subsets of features by finding medoids where K is predefined. This approach can be used to evaluate counterfactual proposals as well. For example, given a counterfactual generator, we can first calculate the absolute difference between a factual data profile and its counterfactual appearance and then run GAM to find K medoids, which will provide clusters of typical modifications in the counterfactual. However, such a method suffers from the limitation that the generated clusters only provide a qualitative evaluation of the counterfactual approach, not quantitative. An example of GAM is visualized in Figure 2.

We propose a new framework that can evaluate counterfactual samples quantitatively based on MMD. Our counterfactual quality measurement is visualized in Figure 1. There start from a handwritten digit of one class and generate a sequence of counterfactual samples along the path to the counter class. In this illustration, we expect that the MMD score will decrease along the path as we see that the generated samples become more and more similar to the target class. Although researchers commonly use L1/L2 norm to evaluate how minimal changes a counterfactual sample is[23], we

ICAIF '22, November 2–4, 2022, New York, NY, USA

ACM ISBN 978-1-4503-9376-8/22/11…$15.00
https://doi.org/10.1145/3533271.3561759

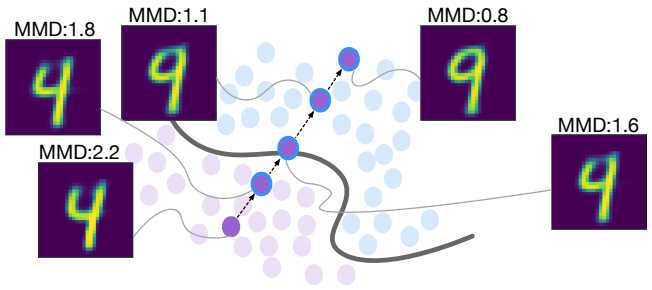

Figure 1: An illustration of MMD for progressive counterfactual generation. As we generate counterfactual samples along the path to the other class, we observe that the counterfactual samples more resemble the other class. It is expected that the MMD should decrease in the transition.

argue that our framework considers both minimality and meaningfulness of a counterfactual profile. In Figure 1, although the counterfactual sample on the decision boundary produces minimal changes to reverse the label, the final counterfactual sample has a better in-distribution of the target class and thus the lowest MMD score. We also demonstrate our idea using tabular data sets, discussed in Section 5.

The paper is organized as follows. In Section 2 we briefly review related works. In Section 3 we discuss the background of MMD, which is a key component in our framework. In Section 4, we discuss our proposed framework using MMD for measuring the quality of counterfactual algorithms. Section 5 shows experiments on image data set, two financial data sets. We conclude in Section 6.

2 RELATED WORK

In addition to the discussion of [16] above, our work is broadly related to the fields of explainable machine learning and counterfactual explanations.

2.1 Counterfactual explanation

Counterfactual explanations have a long history in many areas. Wachter et al. [30] first cast the counterfactual explanation into an optimization problem. The general optimization objective aims to solve $\arg\min_{x'} d(x, x')$ subject to $f(x') = y'$ where $f(\cdot)$ is a trained classifier; x is input; x' and y' are the counterfactual sample and its label, respectively. Many works construct their own optimization objective with different distance functions and iteratively searches for the counterfactual [8, 9, 11, 12, 15, 22, 29].

2.2 Explainable machine learning

Our work is also related to explainable machine learning. Often these fall under three titles: feature importance, rule-based, and prototype.

Feature importance is one of the most popular local explanation methods. It aims to explain the decision of a black-box model for one particular input. Here, a score is assigned to indicate how important each feature is for the decision. Local Interpretable Model-agnostic Explanations (LIME)[24] explains an instance by looking at samples that are randomly selected from local surrounding. Shapley Additive exPlanations (SHAP) [21] produces a unified measure of feature importance based on Shapley values which originates from game theory. Other methods [1, 2, 4] also provide model-agnostic methods to calculate feature importance, explaining the decisions globally or locally.

Rule-based methods explain a model by giving a sequence of decisions rules that are human-understandable. Ribeiro et al. introduces a model-agnostic framework that first forms an anchor point and then use the anchor point to generate decision rules that are human-understandable. Local Rule-based Explainer [15] targets tabular data sets and also generates a batch of rules in the vicinity of the sample of interest. Other methods including [7, 31] also tackle the problem by generating the explanation in the form of rules.

The prototype method generally finds or creates a sample called a "prototype" that represents a set of similar samples. The prototype sample can be (1) a sample close to the query sample, (2) the centroid of a cluster that contains the query sample or (3) a synthetic sample that represents the query sample. Kim et al. also use MMD, proposing an algorithm MMD-critic, but aim for generating prototype samples that are close to the querying data distribution and the samples that are far from the data distribution, called criticisms. Russell introduces a variant of MMD-critic. Blanco-Justicia et al. also produces a prototype and a tree as explanations but it is the first work that include the privacy concept in the explanation. Tan et al. tries to make ensemble methods explainable. Bien and Tibshirani applies the nearest-neighbor-searching style algorithm for generating prototype.

3 BACKGROUND

Maximum mean discrepancy (MMD) is a non-parametric test that calculates distance in the space of probability distributions. In this section, we review the core idea of MMD and define a few important notations.

3.1 Problem setup

The inner product between matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ is given by $\langle \boldsymbol{A}, \boldsymbol{B} \rangle = \sum a_{i,j} b_{i,j}$. The indicator function $\mathbb{1}[\cdot]$ outputs the value of 1 if the argument holds true and the value of 0 otherwise. $\boldsymbol{P}$ and $\boldsymbol{Q}$ denote two different probability distributions. We denote the lower case $\boldsymbol{x}$ as vectors and the capital $\boldsymbol{X}$ as matrix.

3.2 Maximum Mean Discrepancy (MMD)

MMD is a measurement of the difference between distributions $\boldsymbol{P}$ and $\boldsymbol{Q}$. The difference is calculated by the supremum over a function space $\mathcal{F}$ of differences between the expectations with respect to two distributions.This is given by:

$$MMD(\boldsymbol{P}, \boldsymbol{Q}, \mathcal{F}) = \sup_{f \in \mathcal{F}} \left(\mathbb{E}_{\boldsymbol{X} \sim \boldsymbol{P}}[f(\boldsymbol{X})] - \mathbb{E}_{\boldsymbol{Y} \sim \boldsymbol{Q}}[f(\boldsymbol{Y})] \right) \quad (1)$$

If $\mathcal{F}$ is a reproducing kernel Hilbert space (RKHS) with kernel function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, [13] shows the supremum is achieved at

$$f(x) = \mathbb{E}_{\boldsymbol{X} \sim \boldsymbol{P}}[k(x, \boldsymbol{X})] - \mathbb{E}_{\boldsymbol{X}' \sim \boldsymbol{Q}}[k(x, \boldsymbol{X}')] \quad (2)$$

which measures the maximum discrepancy between the two distributions in the RKHS $\mathcal{F}$. It can be observed that the above equation is positive when the density P overfits Q and is negative when the

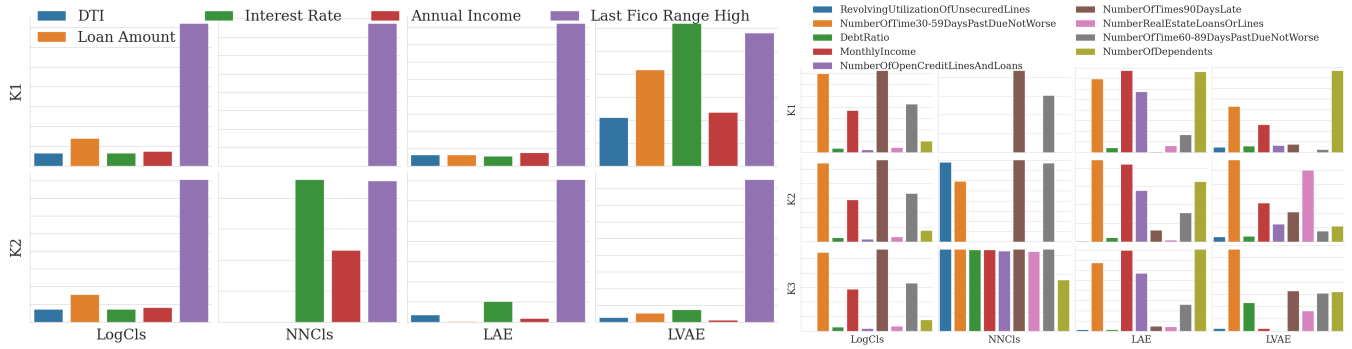

Figure 2: GAM(Ibrahim et al.) run on LCD and GMC data sets. Left: Cluster quality score. The top and the bottom are LCD and GMC data set, respectively. Right: GAM results of LCD and GMC data set. In both cases, K is chosen based on the silhouette score. K=2 for LCD and K=3 for GMC in this case.

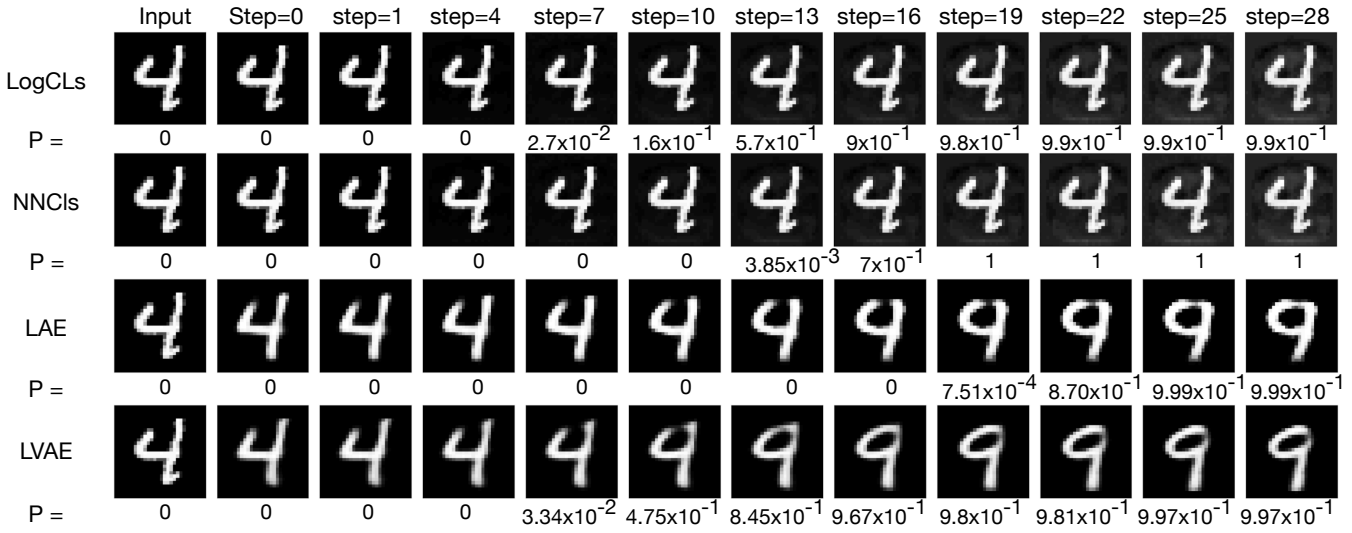

Figure 3: Comparisons of counterfactual samples on MNIST data. We compare LogCls, NNCls, LAE and LVAE on same counterfactual. (See text for model descriptions.) Step=0 is the starting point. We uniformly select 10 counterfactual samples along the path. The number P below each plot is the probability of the sample classified as the counter class ('9').

density P underfits Q. Then, we plug Equation 2 back to Equation 1 and square it, leading to

$$MMD^2(\boldsymbol{P},\boldsymbol{Q},\mathcal{F}) = \mathbb{E}_{\boldsymbol{X},\boldsymbol{X}'\sim\boldsymbol{P}}[k(\boldsymbol{X},\boldsymbol{X}')] - 2\mathbb{E}_{\boldsymbol{X}\sim\boldsymbol{P},\boldsymbol{X}'\sim\boldsymbol{Q}}[k(\boldsymbol{X},\boldsymbol{X}')] + \mathbb{E}_{\boldsymbol{X},\boldsymbol{X}'\sim\boldsymbol{Q}}[k(\boldsymbol{X},\boldsymbol{X}')] \tag{3}$$

It is clear that Equation 3 is always larger than 0 and is equal to 0 if and only if the $\boldsymbol{P}$ is identical to $\boldsymbol{Q}$. This quantity can be approximated using Monte Carlo estimation. Given two data sets $\boldsymbol{X} = \{x_1, x_2, \ldots, x_n\}, \boldsymbol{Y} = \{y_1, y_2, \ldots, y_m\}$, the Equation 3 can be approximated as

$$MMD^2(\boldsymbol{X},\boldsymbol{Y},\mathcal{F}) = \frac{1}{n^2}\sum_{i,j\in[n]} k(x_i,x_j) - \frac{2}{n*m}\sum_{i\in[n],j\in[m]} k(x_i,y_j) + \frac{1}{m^2}\sum_{i,j\in[m]} k(y_i,y_j) \tag{4}$$

with Equation 2 approximated as

$$f(x) = \frac{1}{n}\sum_{i\in[n]} k(x,x_i) - \frac{1}{m}\sum_{j\in[m]} k(x,y_j). \tag{5}$$

4 METHODOLOGY

MMD is typically used to measure if two data sets come from the same distributions. We extend this idea to measure the quality of counterfactual samples.

Denote $\boldsymbol{X} = \{x_1, x_2, \ldots, x_n\}$ and each $x_i \in \mathbb{R}^D$ where D is the dimensions of input space. Then, we define $\boldsymbol{X}(x_i) = \{x_j | y_j \neq y_i\}$. We also denote labels $\boldsymbol{Y} = \{y_1, y_2, \ldots, y_n\}$ with each $y_i \in \{0, 1\}$. For each x_i, denote $x_i^s \in \mathbb{R}^D$ the counterfactual sample at step s where $s \in [1, 2, \ldots, S]$. Each x_i^s is obtained from the counterfactual generation process as we vary the steps.

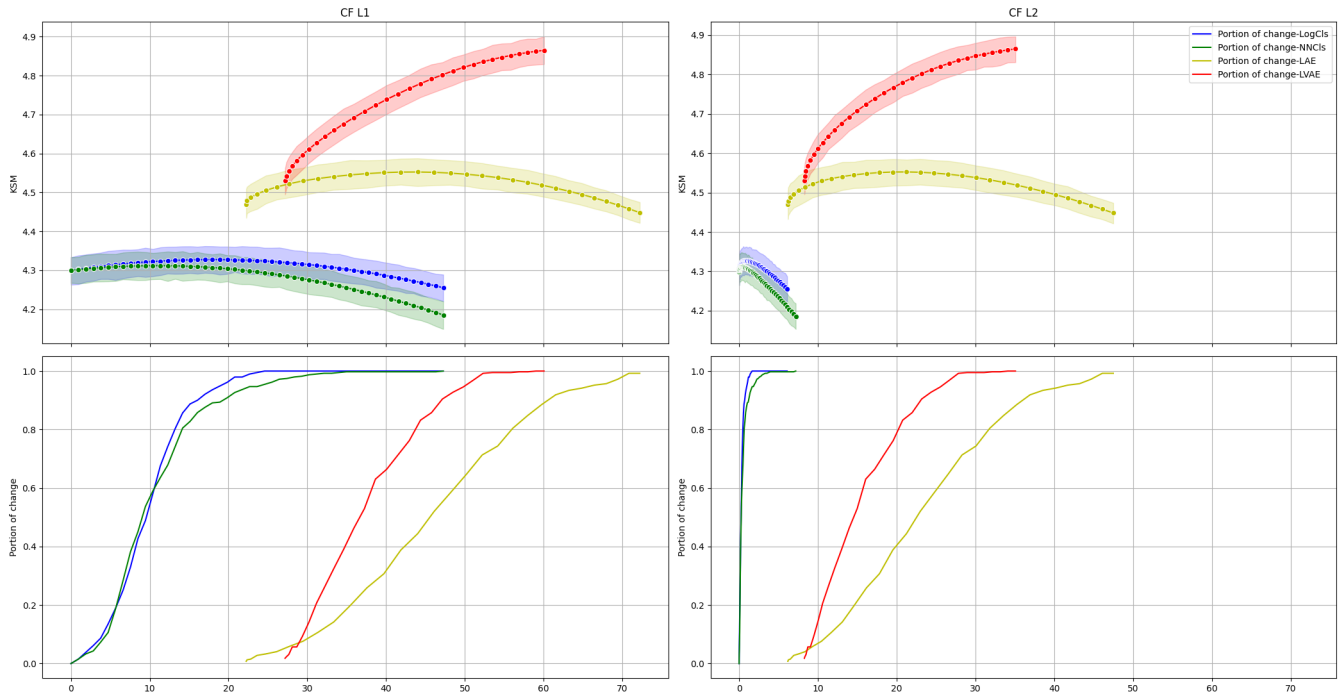

Figure 4: KSM results of MNIST data set. (See text for relation between KSM and MMD.) Top: KSM between counterfactual samples and its target class vs L1 distance and L2 distance from its input, respectively. Bottom: the portion of change of the labels along the counterfactual path vs L1 distance and L2 distance, respectively.

For each x_i and its counterfactual x_i^s at step s, we can calculate the MMD distance between the two. That is, we calculate:

$$MMD_i^s(\boldsymbol{X}(x_i), x_i^s) = \frac{1}{m^2}\sum_{p,q\in[m]} k(x_p, x_q) - \frac{2}{m}\sum_{p\in[m]} k(x_p, x_i^s) + k(x_i^s, x_i^s) \tag{6}$$

We observe that the third term in the above equation stays the same when we focus on one particular sample. We also notice that the first term remains the same when we fix the counter-class samples that we compare with. In other words, all the underlying information between two clusters' samples is represented by the middle term. Thus, we define a Kernel Similarity Measure (KSM)

$$KSM_i^s(\boldsymbol{X}(x_i), x_i^s) = \frac{2}{m}\sum_{p\in[m]} k(x_p, x_i^s) \tag{7}$$

We also study how the L1 and L2 distance change between each counterfactual and its original input in the input space,

$$L1_i^s(x_i, x_i^s) = \sum_{d=1}^{D} |x_i(d) - x_i^s(d)| \tag{8}$$

and

$$L2_i^s(x_i, x_i^s) = \sum_{d=1}^{D} |x_i(d) - x_i^s(d)|^2 \tag{9}$$

Algorithm 1 Quality measurement of counterfactual samples

1: **Input:** $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, Counterfactual steps $[\mathcal{S}]$, Counterfactual model $\mathcal{M}$, Classifier $\mathcal{F}$
2: **Initialize** $\text{KSM}^{\mathcal{S}}, L1^{\mathcal{S}}, L2^{\mathcal{S}}, L^{\mathcal{S}}$ placeholders for all the samples and steps
3: **Calculate prediction based on the classifier** $\bar{Y} = \mathcal{F}(\mathcal{D})$
4: **for** each s in $\mathcal{S}$ **do**
5: **Generate counterfactual data set at step s** $\mathcal{D}^s = \mathcal{M}(\mathcal{D}, s)$
6: **Calculate prediction of counterfactual samples based on the classifier** $\bar{Y}^s = \mathcal{F}(\mathcal{D}^s)$
7: **Calculate kernel similarity measure** KSM^s based on Equation 7
8: **Calculate L1 distance** $L1^s$ based on Equation 8
9: **Calculate L2 distance** $L2^s$ based on Equation 9
10: **Calculate label change indicator** L^s based on Equation 10
11: **end for**
12: **return** $\text{KSM}^{\mathcal{S}}, \text{L2}^{\mathcal{S}}, \text{L1}^{\mathcal{S}}, \text{L}^{\mathcal{S}}$

Lastly, we examine what portion of labels has been changed as we move from x_i to its counterfactual x_i^s,

$$L_i^s = \mathbb{1}[y_i = y_i^s] \tag{10}$$

where y_i^s is the predicted label for x_i^s. The algorithm is summarized in Algorithm 1.

5 EXPERIMENTS

We first illustrate our idea through examples on the MNIST handwritten digits data set [19] and then focus on two large scale tabular data sets. We use a Gaussian kernel with bandwidth=100 for our kernel function $k(\cdot,\cdot)$. In all experiments, we use PyTorch to implement the algorithm. [1]

For all experiments, the training batch size for each update is 128 random samples with nearly proportional number of samples from each class. The initial learning rate is 10^{-4} and decreases by 10 times after the loss fails to significantly decrease for 10 consecutive epochs. The training stops after the loss fails to significantly decrease for 20 consecutive epochs. We use Adam [18] for stochastic gradient method. We run our experiments 5 times for each setting and show error bar in the results.

5.1 Counterfactual generation

We first describe the means we adopt to generating counterfactual samples. We consider the logistic regression model (LogCls), a neural network (NNCls), an auto-encoder with logistic regression model supervising latent space (LAE) and a variational auto-encoder with logistic regression in latent space (LVAE).

LogCls. We train a logistic regression model directly on the input feature space. Once the model $\boldsymbol{f}$ is trained, we fix the model's weights $\boldsymbol{w}$ and bias $\boldsymbol{w_0}$. we generate a counterfactual sample

$$\hat{x}_i = x_i + s * \overline{w} \tag{11}$$

where $\overline{w} = \frac{1}{n}\sum_i \alpha_i w$ and $\alpha_i = -((x_i^K)^T w + w_0)/w^T w$. We vary the step s from 0 to 30.

NNCls. We train a neural network $\boldsymbol{f}$ directly on the input feature space. For each sample x_i we define Δ_i such that the label of $x_i + \Delta_i$ is changed to the counter-class by minimizing

$$CrossEntropy\big(\boldsymbol{f}(x_i + \Delta_i), 0.5\big) + 0.01 * |\Delta_i|_1 \tag{12}$$

where $\boldsymbol{f}$ outputs a predictive probability and $|\cdot|$ denotes the L1 regularization that minimizes the proposed changes. We then move along the direction defined by Δ_i.

LAE. We first construct an auto-encoder neural network with logistic regression supervising the latent space. The loss consists of both the reconstruction loss and classification loss for this model. Once the model is trained, we apply the same projection as Equation 11, except in the latent space, then decode the counterfactual.

LVAE. A variational Bayesian version of LAE, using variational auto-encoders and probit regression.[32]

5.2 Illustrative example: MNIST data set

MNIST is a public benchmark data set of handwritten digits. It contains 60,000 samples in training set and 10,000 in testing set, both distributed across the ten digits. In our scenario, we select 4 and 9 as class 0 and class 1, which contains 10,761 samples in training set and 1,991 samples in testing set.

[1] For our Python implementation, see https://github.com/Wei2624/KSM

Results. We plot our results of MNIST data set in Figure 4. We show comparisons of LogCls, NNCls, LAE and LVAE models. The top two plots show that how KSM is distributed over L1 and L2 distance, respectively. We observe that the maximum KSM that LogCls can achieve is on par with NNCls over both L1 and L2 distance. The LAE can achieve the second best in terms of the closeness between the counter-class distribution and the counterfactual distribution. The LVAE is the best among them. Furthermore, as we make changes to an input to produce counterfactual samples, we show what percent of samples in the test set have reversed its label to the counter-class. LogCls is always the first one that flip all the labels at constant slope, which is expected because of its linearity. NNCls trends in similar way. However, the KSM score suffers due to less meaningful counterfactuals. LAE and LVAE can also reverse all the labels into the counter-class with more obvious yet meaningful changes.

The observations from Figure 4 are also consistent with Figure 3 where we demonstrate the counterfactuals for a particular sample from MNIST data set. We uniformly show 10 counterfactual steps along the path of counterfactual generation. It is clear that both LAE and LVAE significantly outperform NNCls and LogCls, although the label of the sample from LogCls and NNCls have been changed at a faster speed. Comparing LVAE with LAE, we observe that LVAE produces a slightly better counterfactual, which coincides with what is shown in Figure 3 as well. The counterfactual sample of NNCls looks indifferent with LogCls in this example, implying less closeness between the counterfactual distribution and the counter class distribution. Visually, we see that the transition from a 4 to a 9 in LVAE is much smoother and intuitive, which is also reflected through our KSM score.

Based on Figure 4, we can answer the question such as how much L1 distance or L2 distance a query sample has to travel such that its label can be flipped and how closeness that resulted sample stay to the distribution of the target class. In other words, given the same amount of change, we can evaluate how meaningful the generated counterfactual sample is. Looking at the last four columns of Figure 3, even though all the labels have been changed to the counter class, the quality of generated counterfactual downgrades from bottom to up, which corresponds to the order of KSM score shown in Figure 4. This implies LogCls and NNCls might never arrive at the same level of closeness to the counter-class distribution.

5.3 LCD data set

LCD consists of loan application profiles with the label of non-default loan or default loan. We select 10,000 samples for training and 1,000 for testing. The number of positive and negative sample in both the training and testing set is equally distributed. The data set is publicly available at Kaggle.com. The feature dimension for this problem is 7, which contains 5 scalar features and 1 binary feature mapped to one-hot vectors. We standardize real-valued features in LCD.

Results. We plot our results of LCD data set in Figure 5. We again show comparisons of LogCls, NNCls, LAE and LVAE models. The top two plots show that how KSM is distributed over L1 and L2 distance, respectively. We observe the similar performance as we see in MNIST data set. Although NNCls has a much faster rate of

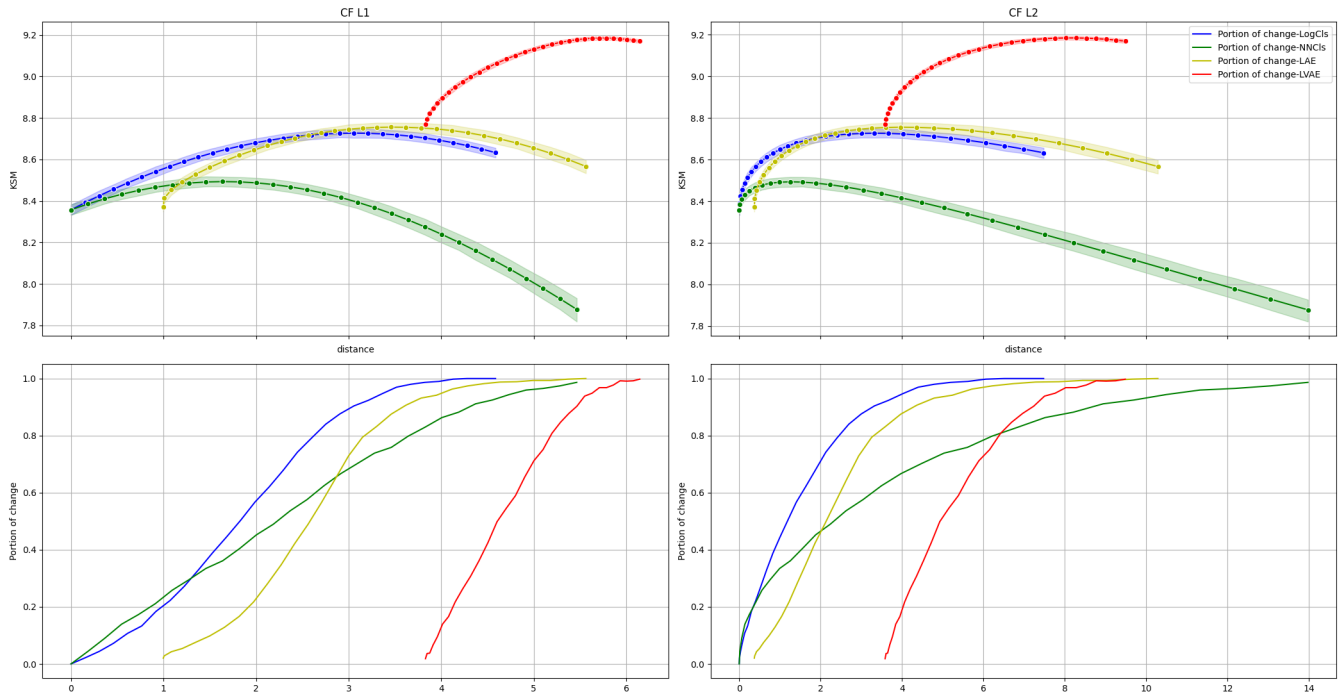

Figure 5: KSM results on LCD. (See text for relation between KSM and MMD.) Top: KSM between counterfactual samples and its target class vs L1 distance and L2 distance from its input, respectively. Bottom: the portion of change of the labels vs L1 distance and L2 distance, respectively.

Figure 6: Comparisons of counterfactual samples on LCD data set. We compare LogCls, NNCls, LAE and LVAE for all the rejected application in the testing data set. We show a transition from a rejected credit application to an accepted one with Step=0 being the starting point. We uniformly select 10 counterfactual profiles along the path. Each feature is normalized to zero mean and unit variance.

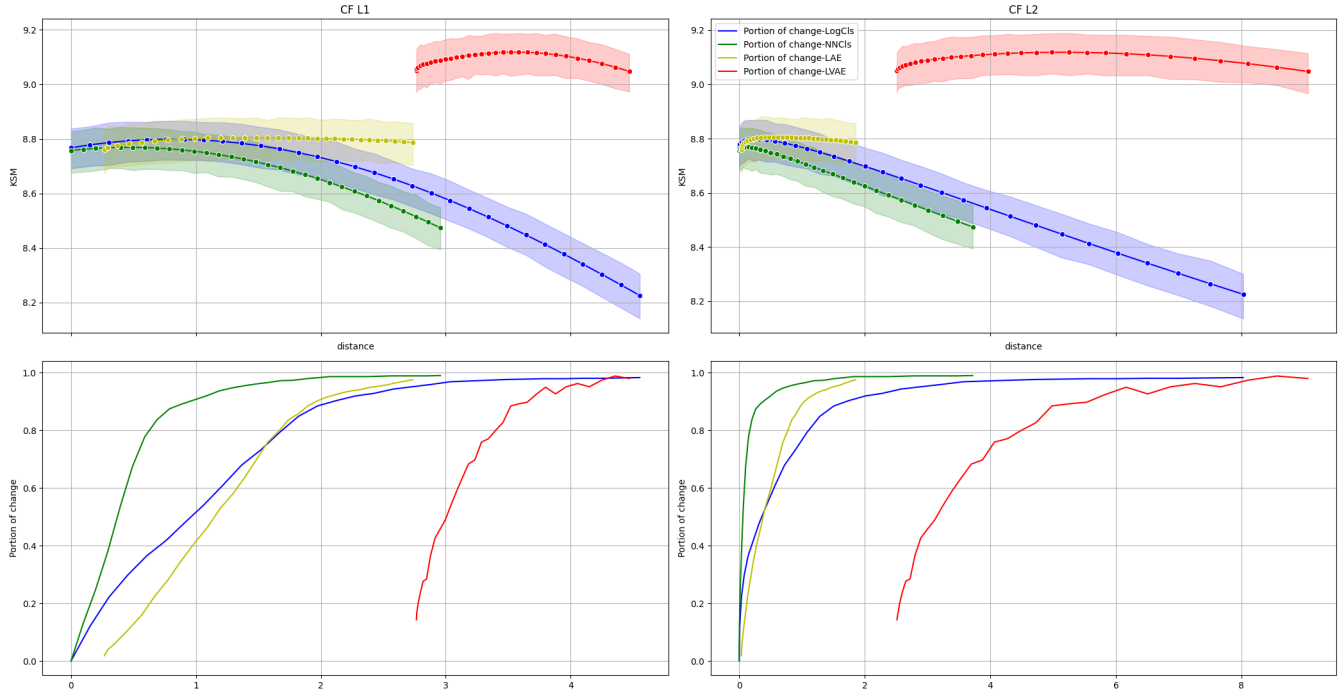

Figure 7: KSM results of GMC. (See text for relation between KSM and MMD.) Top: KSM between counterfactual samples and its target class vs L1 distance and L2 distance from its input, respectively. Bottom: the portion of change of the labels vs L1 distance and L2 distance, respectively.

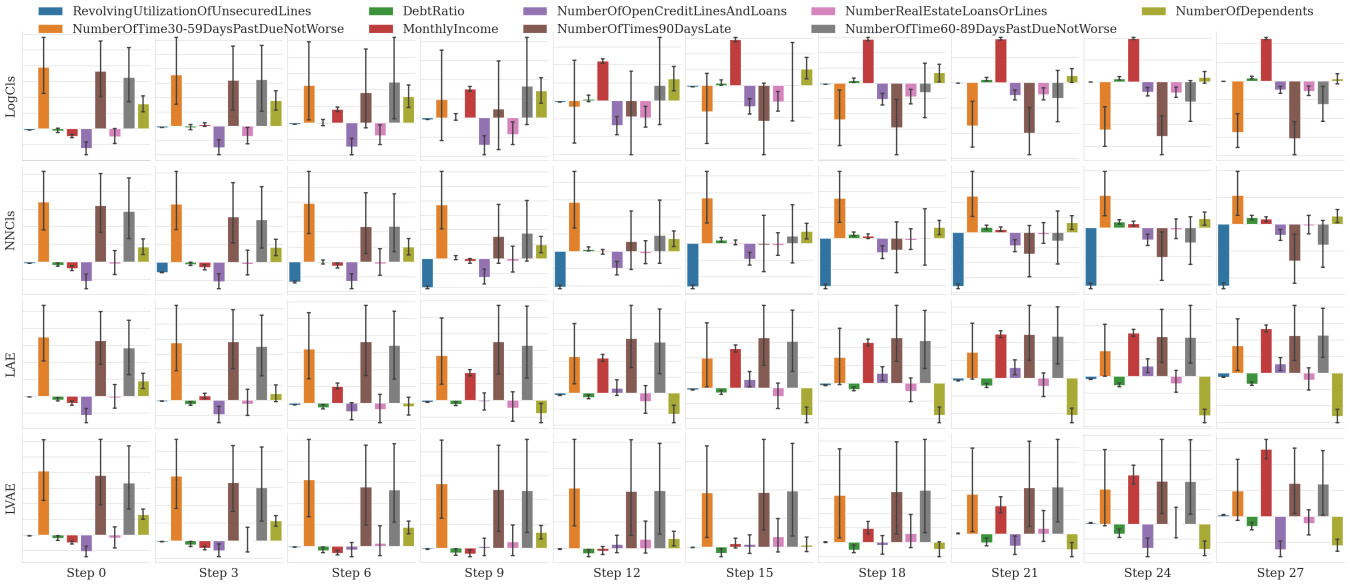

Figure 8: Comparisons of counterfactual samples on GMC data set. We compare LogCls, NNCls, LAE and LVAE for all the rejected application in the testing data set. We show a transition from a rejected credit application to an approved one with Step=0 being the starting point. We uniformly select 10 counterfactual samples along the path. Each feature is normalized to 0 mean and unit variance.

reversing a query sample's label, the KSM between a query sample and its counter-class is the lowest. We notice that LogCls and LAE achieve a comparable level of KSM and the LVAE is again the best.

We also rationalize the above observation through Figure 6. The query is all rejected loan application. We uniformly show 10 counterfactual steps along the path of counterfactual generation. In order to have the application accepted, we expect an applicant to have a better FICO score and the economy has a lower interest rate. This trend is particularly captured by LVAE model which has the highest KSM score. The counterfactual samples from NNCls model, even with flipped label, proposes to increase the interest rate. The LogCls and LAE propose the similar changes. The possible improvement of both models can be driving down more interest rate. Finally, LVAE captures all the key changes, making it the highest KSM score.

In addition, a possible way of explanation on the trained deep classifier using these counterfactual samples would be comparing the first and the last column and inferring that the features with the most significant changes are the key factor that affects the neural network's decision. In this case, we conclude that the FICO score and the interest rate would be the two key factors, which is aligned with the real world. On the other hand, the samples from NNCls and LogCls, though also considered as counterfactual samples, fail to capture those two features.

5.4 Give me some credit (GMC) data set

GMC data set is constructed from credit application profiles with the binary label of having delinquency or no delinquency. We again randomly select 15,000 samples for training and 1,000 samples for testing with equally distributed samples between two classes. The feature dimension for this data set is 9. The data set is publicly available at Kaggle.com. We again standardize real-valued features in GMC.

Results. We again plot our results of GMC data set in Figure **??**. We show comparisons of LogCls, NNCls, LAE and LVAE models. The top two plots show that how KSM is distributed over L1 and L2 distance, respectively. We observe that LogCls, NNCls and LAE can achieve comparable maximum KSM score over both L1 and L2 distance. The LVAE is again the best. Similarly, as we make changes on the input to produce counterfactual samples, we show how much percent of samples have reversed their labels to the counter-class.

We again generate the real examples by generating counterfactual samples for all the rejected profiles and show them in Figure 8. One key factor in GMC is monthly income. As shown in Figure 8, NNCls completely fails to capture the importance of this feature. LogCls can also capture it but it propose to increase debt ratio which is an disfavored feature for accepted profile.

6 CONCLUSION

Counterfactual generation is one of the more trustworthy methods for understanding a black-box model such as neural networks. It can simultaneously explain a deep classifier's decision and provide tailored modifications for each individual. In this paper, we propose a new tool that utilizes MMD to quantitatively measure the quality of counterfactual generation process and thus understand better about counterfactual samples. We also describe how MMD directly relates to the KSM measure shown in the figures. We show that this approach can help us better understand a well-trained black-box model. We demonstrate our idea through an image data set and two large scale tabular data sets and show that our proposed method is simple yet effective for understanding the quality of counterfactual samples.

REFERENCES

[1] Rishabh Agarwal, Levi Melnick, Nicholas Frosst, Xuezhou Zhang, Ben Lengerich, Rich Caruana, and Geoffrey E Hinton. 2021. Neural additive models: Interpretable machine learning with neural nets. *Advances in Neural Information Processing Systems* 34 (2021).

[2] Sule Anjomshoae, Timotheus Kampik, and Kary Främling. 2020. Py-CIU: a python library for explaining machine learning predictions using contextual importance and utility. In *IJCAI-PRICAI 2020 Workshop on Explainable Artificial Intelligence (XAI), january 8, 2020.*

[3] Lynton Ardizzone, Jakob Kruse, Sebastian Wirkert, Daniel Rahner, Eric W Pellegrini, Ralf S Klessen, Lena Maier-Hein, Carsten Rother, and Ullrich Köthe. 2018. Analyzing inverse problems with invertible neural networks. *arXiv preprint arXiv:1808.04730* (2018).

[4] Hubert Baniecki and Przemyslaw Biecek. 2020. The grammar of interactive explanatory model analysis. *arXiv preprint arXiv:2005.00497* (2020).

[5] Jacob Bien and Robert Tibshirani. 2011. Prototype selection for interpretable classification. *The Annals of Applied Statistics* 5, 4 (2011), 2403–2424.

[6] Alberto Blanco-Justicia, Josep Domingo-Ferrer, Sergio Martinez, and David Sanchez. 2020. Machine learning explainability via microaggregation and shallow decision trees. *Knowledge-Based Systems* 194 (2020), 105532.

[7] Mark Craven and Jude Shavlik. 1995. Extracting tree-structured representations of trained networks. *Advances in neural information processing systems* 8 (1995).

[8] Susanne Dandl, Christoph Molnar, Martin Binder, and Bernd Bischl. 2020. Multi-Objective Counterfactual Explanations. *arXiv preprint arXiv:2004.11165* (2020).

[9] Amit Dhurandhar, Tejaswini Pedapati, Avinash Balakrishnan, Pin-Yu Chen, Karthikeyan Shanmugam, and Ruchir Puri. 2019. Model agnostic contrastive explanations for structured data. *arXiv preprint arXiv:1906.00117* (2019).

[10] Gintare Karolina Dziugaite, Daniel M Roy, and Zoubin Ghahramani. 2015. Training generative neural networks via maximum mean discrepancy optimization. *arXiv preprint arXiv:1505.03906* (2015).

[11] Rubén R Fernández, Isaac Martín de Diego, Víctor Aceña, Alberto Fernández-Isabel, and Javier M Moguerza. 2020. Random forest explainability using counterfactual sets. *Information Fusion* 63 (2020), 196–207.

[12] Rory Mc Grath, Luca Costabello, Chan Le Van, Paul Sweeney, Farbod Kamiab, Zhao Shen, and Freddy Lecue. 2018. Interpretable credit application predictions with counterfactual explanations. *arXiv preprint arXiv:1811.05245* (2018).

[13] Arthur Gretton, Karsten Borgwardt, Malte J Rasch, Bernhard Scholkopf, and Alexander J Smola. 2008. A kernel method for the two-sample problem. *arXiv preprint arXiv:0805.2368* (2008).

[14] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. A kernel two-sample test. *The Journal of Machine Learning Research* 13, 1 (2012), 723–773.

[15] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Dino Pedreschi, Franco Turini, and Fosca Giannotti. 2018. Local rule-based explanations of black box decision systems. *arXiv preprint arXiv:1805.10820* (2018).

[16] Mark Ibrahim, Melissa Louie, Ceena Modarres, and John Paisley. 2019. Global explanations of neural networks. In *AAAI/ACM Conference ACM.*

[17] Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. 2016. Examples are not enough, learn to criticize! criticism for interpretability. *Advances in neural information processing systems* 29 (2016).

[18] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[19] Yann LeCun and Corinna Cortes. 2010. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/. (2010). http://yann.lecun.com/exdb/mnist/

[20] Yujia Li, Kevin Swersky, and Rich Zemel. 2015. Generative moment matching networks. In *International Conference on Machine Learning*. PMLR, 1718–1727.

[21] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems* 30 (2017).

[22] Ramaravind K Mothilal, Amit Sharma, and Chenhao Tan. 2020. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. 607–617.

[23] Martin Pawelczyk, Sascha Bielawski, Johannes van den Heuvel, Tobias Richter, and Gjergji Kasneci. 2021. Carla: a python library to benchmark algorithmic recourse and counterfactual explanation algorithms. *arXiv preprint arXiv:2108.00783* (2021).

[24] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 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.*

1135–1144.
[25] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.
[26] Chris Russell. 2019. Efficient search for diverse coherent explanations. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. 20–28.
[27] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. 2019. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation* 23, 5 (2019), 828–841.
[28] Sarah Tan, Matvey Soloviev, Giles Hooker, and Martin T Wells. 2020. Tree space prototypes: Another look at making tree ensembles interpretable. In *Proceedings of the 2020 ACM-IMS on Foundations of Data Science Conference*. 23–34.
[29] Arnaud Van Looveren and Janis Klaise. 2019. Interpretable counterfactual explanations guided by prototypes. *arXiv preprint arXiv:1907.02584* (2019).
[30] Sandra Wachter, Brent Mittelstadt, and Chris Russell. 2017. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harv. JL & Tech.* 31 (2017), 841.
[31] Hongyu Yang, Cynthia Rudin, and Margo Seltzer. 2017. Scalable Bayesian rule lists. In *International conference on machine learning*. PMLR, 3921–3930.
[32] Wei Zhang, Brian Barr, and John Paisley. 2022. An Interpretable Deep Classifier for Counterfactual Generation. In *Proceedings of the Third ACM International Conference on AI in Finance*. 1–8.