Effect of Sampling on Robustness of LIME

Sara Moin
PhD21035
IIITD
Delhi, India
saram@iiitd.ac.in

Sumedha Chugh
PhD21123
IIITD
Delhi, India
sumedhac@iiitd.ac.in

***Abstract*—The popular LIME(Local Interpretable Model-Agnostic Explanations) framework suffers from lack of robustness, making it an uncertain choice for safety-critical domains like healthcare, law, etc. This work presents a modification of LIME based on CTGAN sampling and analyses it on a popular adversarial attack. Better robustness of the algorithm by comparing its explanations to vanilla LIME's and calculating the percentage of time it could recognize adversaries is demonstrated.**

***Index Terms*—Explainable AI, Robustness, LIME, CTGAN**

I. Introduction

As AI technology advances, it becomes increasingly challenging for humans to understand how the algorithm generates a particular outcome. The complex models have transformed into a black box that even the engineers who designed the algorithm cannot explain. Explainable AI (XAI) techniques enable humans to understand the outcomes produced by machine learning algorithms. One of the first and most commonly used Local Posthoc explanation methods is LIME (Local Interpretable Model Agnostic Explanations). It approximates the machine learning model locally with a simpler surrogate model, whose weights give human-understandable feature importance. The local surrogate model is trained on a subset of the training data close to the explained instance. If the training data is unavailable, it is done by perturbing the given sample for various features and randomly sampling around it. LIME can be useful for generating local explanations [1]. Sampling bias in LIME due to random sampling can lead to issues like a lack of consistency and make it more prone to adversarial attacks [2].

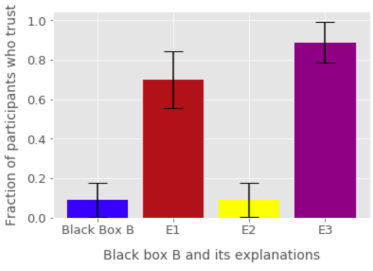

Fig. 1. Trustworthiness and Robustness [3].

A user study done by [3] is shown in Fig 1. E1 explanations exclude prohibited and desired features, E2 includes prohibited and desired features, and E3 excludes prohibited and includes desired features. It can be seen that more than 80 % of experts believed a model when it gave explanations that consist of desired features and not prohibited ones. If the model is not robust enough and can be made to behave like E3, it will be trusted even if it is inherently extremely biased, so we need XAI methods to be more robust. To reduce sampling bias we use CT-GAN to generate perturbations in LIME instead of random sampling and use LIME as a baseline to compare the robustness by exposing these to an adversarial attack designed by Slack et al. [4]. Details of this method are discussed in section 2. Experiments performed and Results are mentioned in section 3. Section 4 talks about the conclusion and future work to extend this project.

II. Approach

The project approach is divided into three major subparts: 1. Replace the random sampler with CTGAN sampler to generate synthetic data 2. Build LIME explainer around this modified sampler 3. Apply modified and Original LIME approaches on adversarial classifiers to check the robustness.

Sub-part 2, i.e. LIME explainer has been discussed in the introduction section, and details of sub-parts 1 and 3 are discussed in the following subsections.

A. Synthetic data generation using CTGAN

Since tabular data varies from text and image data in multiple ways, like range for its features can be determined and consists of a mix of discrete and continuous columns. Continuous columns can be of multiple types like numerical, string, date-time etc., whereas discrete columns may have unrepresented features leading to difficult modeling using GANs. In 2019 Lei Xu et al. proposed 'Modeling Tabular data using Conditional GAN' which models the probability distribution of rows in tabular data to generate realistic synthetic data. Fig. 2. shows the working of CTGAN; it utilizes mode-specific normalization to address columns with non-Gaussian and multimodal distributions and a conditional generator and training by sampling techniques to tackle class imbalance. The conditional generator creates synthetic rows based on a specific discrete column while training by sampling samples training

data according to the frequency of each category, enabling CTGAN to explore all possible discrete values.

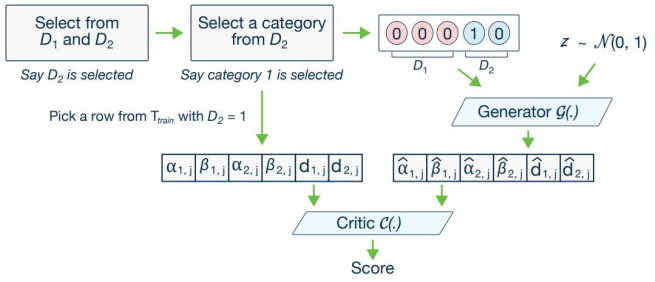

Fig. 2. Working of CTGAN [5]

B. Adversarial Attack

This attack was introduced in [4], where authors aim to create an adversarial classifier that behaves like the original classifier (perhaps extremely discriminatory) on the input data points but looks unbiased and fair on the perturbed instances, thus effectively fooling LIME. Since real-world data follows a distribution X_{dist}, an adversarial classifier that exhibits biased behavior on instances sampled from X_{dist} but remains unbiased on instances that do not come from X_{dist} can fool LIME. As feature importance output by LIME relies highly on perturbed instances which have a different distribution from X_{dist} (are OOD, shown in experiments section), the resulting explanations will make the classifier designed by the adversary look innocuous.

Mathematically,

$$e(x) = f(x), if x \in X_{dist},$$
$$\chi(x), otherwise$$

where, e(x) is adversarial classifier, f(x) is biased model, $\chi(x)$ is unbiased model. Implementation of this attack can be seen in fig.3

```
In [3]: class racist_model_f:
            # Decision rule: classify negatively if race is black
            def predict(self,X):
                return np.array([params.negative_outcome if x[race_indc] > 0 else params.positive_outcome for x in X])

            def predict_proba(self, X):
                return one_hot_encode(self.predict(X))

            def score(self, X,y):
                return np.sum(self.predict(X)==y) / len(X)

        class innocuous_model_psi:
            # Decision rule: classify according to randomly drawn column 'unrelated column'
            def predict(self,X):
                return np.array([params.negative_outcome if x[unrelated_indcs] > 0 else params.positive_outcome for x in X])

            def predict_proba(self, X):
                return one_hot_encode(self.predict(X))

            def score(self, X,y):
                return np.sum(self.predict(X)==y) / len(X)
```

Fig. 3. Adversarial Classifier

To train OOD classifier dataset is preprocessed. If $dataset \in X$ it is labelled as False, indicating no OOD samples; if $dataset \in X_p$ we label it as True, indicating OOD samples unless they are already in X. Then a classifier is trained in it. We use random forest with 100 tree estimators on the $X \bigcup X_p$ and their corresponding class labels

III. Experiments and Results

To perform the experiments, we utilize the COMPAS dataset. It consists of features like criminal history, demographics, risk score, etc., with race being the sensitive feature. On running principal component analysis (PCA) on the dataset containing original data instances and the perturbed instances obtained using random sampling, reducing the dimensionality to 2 we can see in fig.4 that perturbed instances (blue) have a different distribution than that of original instances (orange). Process flow for experiments looks as follow:

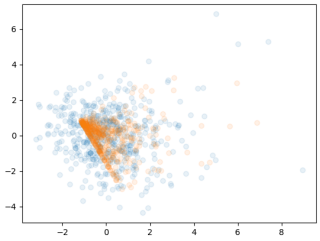

Fig. 4. PCA on original dataset and perturbed samples [3].

- Design discriminatory classifier f that makes predictions based on sensitive attribute
- Generate x_p by adding random noise sampled from N (0, 1) to each feature value
- Train the OOD classifier
- Build the unbiased classifier χ by constructing synthetic uncorrelated features that have zero correlation with sensitive attributes
- Use LIME and LIME with CTGAN sampling to generate explanations
- Evaluate robustness by checking where race appears in the top 3 important features. If higher importance, then model is robust to attack; otherwise not

To feed data to OOD classifier, we pre-process the dataset by assigning African Americans as a protected class (label = 1), removing features related to race and doing normalization, calculating the length of stay, etc. Fig. 5 shows how preprocessed data looks.

In [8]: X

Out[8]:

id	age	two_year_recid	priors_count	length_of_stay	c_charge_degree_F	c_charge_degree_M	sex_Female	sex_Male	race
1	69	0	0	0	1	0	0	1	0
3	34	1	0	10	1	0	0	1	1
4	24	1	4	1	1	0	0	1	1
7	44	0	0	1	0	1	0	1	0
8	41	1	14	6	1	0	0	1	0
...	...	...	...	...	...	...	...	...	...
10996	23	0	0	1	1	0	0	1	1
10997	23	0	0	1	1	0	0	1	1
10999	57	0	0	1	1	0	0	1	0
11000	33	0	3	1	0	1	1	0	1
11001	23	1	2	1	1	0	1	0	0

6172 rows × 9 columns

Fig. 5. Dataset after Pre-processing

We then use the CTGANSynthesizer and train the GAN for 100 epochs to generate synthetic data, as shown in Fig. 6; and generate explainer around it.

```
In [13]: ctgan.sample(1000)
```

Out[13]:

	age	two_year_recid	priors_count	length_of_stay	c_charge_degree_F	c_charge_degree_M	sex_Female	sex_Male	race	unrelated_column
0	19	1	1	-11	0	1	0	1	0	1
1	40	1	0	0	1	1	0	0	1	1
2	18	1	2	2	1	0	1	1	0	0
3	19	0	2	3	1	1	0	1	1	0
4	49	0	0	0	1	1	1	1	1	1
...	...	...	...	...	...	...	...	...	...	...
995	40	1	8	5	1	0	0	1	0	1
996	60	0	15	0	1	1	0	0	0	1
997	39	1	10	-3	1	0	0	1	0	1
998	25	1	0	4	1	1	0	1	0	1
999	35	1	3	3	1	1	0	0	0	0

1000 rows × 10 columns

Fig. 6. Synthetic data generated by CTGAN

After this, we apply LIME and CTGAN-LIME on the adversarial classifier to generate explanations for the first test instance.

```
In [95]: numpy_explainer.explain_instance(xtest[ex_indc], adv_lime.predict_proba, label=1, num_samples=5000, num_features=3)
Out[95]: [('race', 0.5016209270590931), ('length_of_stay', 0.016096179525062176), ('unrelated_column', 0.012876938185426817)]
```

Fig. 7. CTGAN-LIME output on adversarial attack

```
Explanation on adversarial model:
 [('unrelated_column=0', 0.9982571500602809), ('sex_Male=1', 0.0007581075797357132), ('length_of_stay', -0.0003172374
916087674)]
```

Fig. 8. Vanilla LIME output on adversarial attack

From fig.7 and fig.8 we can see that vanilla lime doesn't show race in the first three important features, but CTGAN lime shows race as the most important feature, showing it was able to recognize the adversarial attack. To compare CTGAN-LIME on all the test samples, we ran the above experiment for all the 1500 data points in the test set and calculated the percentage of times CTGAN-LIME detected race in the top 3 features (recognizing the underlying bias of classifier f). Fig.9 shows that for around 43% test samples, CT-GAN lime was able to detect race in the top three features, proving CT-GAN LIME to be more robust than vanilla LIME

```
In [26]: print(len([a for a in list_exp if a == 'race']) / len(list_exp))
         0.4387556707712249
```

Fig. 9. Race in top-3 features for CTGAN-LIME

IV. Conclusion and Future Work

This work generated perturbations in LIME using CT-GAN based approach instead of random sampling for tabular data, which proves to be more robust than vanilla LIME when analyzed on a popular adversarial attack. The idea can be extended by using other data generators like VAE for sampling and to examine the robustness of various perturbation-based methods like SHAP, UnRaVEL, DLIME etc.

References

[1] M. T. Ribeiro, S. Singh, and C. Guestrin, ""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*, KDD '16, (New York, NY, USA), p. 1135–1144, Association for Computing Machinery, 2016.

[2] A. Saini and R. Prasad, "Select wisely and explain: Active learning and probabilistic local post-hoc explainability," in *Proceedings of the 2022 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '22, (New York, NY, USA), p. 599–608, Association for Computing Machinery, 2022.

[3] H. Lakkaraju and O. Bastani, ""how do i fool you?": Manipulating user trust via misleading black box explanations," in *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, AIES '20, (New York, NY, USA), p. 79–85, Association for Computing Machinery, 2020.

[4] D. Slack, S. Hilgard, E. Jia, S. Singh, and H. Lakkaraju, "Fooling lime and shap: Adversarial attacks on post hoc explanation methods," 2020.

[5] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional gan," in *Advances in Neural Information Processing Systems*, 2019.