Bachelor's Seminar

LIME

Department of Statistics
Ludwig-Maximilians-Universität München

Haoran Ju

Munich, Feb 25th, 2025

Submitted in partial fulfillment of the requirements for the degree of B. Sc.
Supervised by Dr. Ludwig Bothmann

Abstract

Understanding the decisions made by machine learning models, often perceived as black boxes, is a critical step towards building trust and ensuring fairness in AI systems. Local interpretable model-agnostic explanation(LIME) is one of the most widely used methods contributing to explainable AI(XAI). This work explores the fundamental concepts of LIME and demonstrates this method across different data types. Following a discussion of LIME's advantages and limitations, we introduce its notable extensions— DLIME, and BayLIME—and lastly evaluate their performance relative to LIME through comparative analysis.

Contents

1 Introduction

The rapid development of artificial intelligence(AI) has brought us powerful and precise machine learning models. However, most of these models are perceived as "black boxes",lacking transparency and leaving their internal workings largely undisclosed.[Weld and Bansal (2019)] This has led to a growing concern about explainable AI(XAI), a research field aiming to make AI systems results more understandable to humans. The importance of XAI can be summarized into four key objectives: explaining to justify, explaining to control, explaining to improve and explaining to discover.[Adadi and Berrada (2018)] The European Union's General Data Protection Regulation (GDPR) even regulates the transparency of machine learning models from a legal perspective.[Tesfay et al. (2018)]

Among XAI methods, Local Interpretable Model-agnostic Explanations (LIME) stands out as one of the most representative approaches which generate local surrogate models to approximate the behavior of a black-box model within the neighborhood of a specific prediction of interest. Researchers have systematically evaluated the performance of LIME in real-world applications across different fields. In the medical domain, [Kumarakulasinghe et al. (2020)] assessed the utility of LIME for tabular data using a sepsis classification model as a case study, demonstrating its potential for clinical decision support. Similarly, [Lakshmi and Das (2023)] combined LIME with convolutional neural networks (CNNs) to classify skin images, improving interpretability in image-based diagnosis. In the financial field, [Aljadani et al. (2023)] explored LIME's application in credit scoring models, where it effectively revealed the factors influencing model predictions. These studies collectively highlight LIME's effectiveness in improving the interpretability of machine learning models in high-stakes applications.

Although LIME has proven effective, its limitations have motivated researchers to develop various extensions. LIMEK, GraphLIME, DLIME, BayLIME... many methods emerged to overcome its limitations.

This paper begins with a review of the related work on LIME, followed by a detailed explanation of its fundamental concept. We also demonstrate LIME's application on tabular data and images. After discussing its advantages and limitations, we focus on two notable extensions of LIME—DLIME, and BayLIME—and finally evaluate these extensions in comparision to the original LIME, highlighting their improvements and also their potential shortcomings.

2 Related work

Reliance on validation set accuracy as the primary measure of trust has been well-studied prior to the proposal of LIME. However, this approach often leads to issues such as overestimation of a model's performance and the failure to detect Out-Of-Distribution (OOD) generalization problems. Moreover, existing methods at the time either did not address the need for explaining individual predictions or were limited to model-specific approaches.

In this context, [Ribeiro et al. (2016)] proposed LIME as both a complement to existing tools and a significant breakthrough in explainable AI (XAI). This novel approach intro-

duced new possibilities for enhancing trust and transparency in machine learning systems, inspiring subsequent studies to evaluate its potential in various domains.
Before we delve deep into the explanation system of LIME, we'd like to introduce three typical classifications of XAI as background knowledge.

- **model-agnostic v.s specific:** So as the terms suggest, model-specific means the method is only specific for the particular model. While model-agnostic methods does not depend on the internal structure or type of a machine learning model. These techniques are universally applicable to any model, including neural networks, random forests, and support vector machines. Model-agnostic methods are typically used as *post-hoc* techniques and can be either local or global in nature. A common approach to model-agnostic interpretability is the use of **surrogate models**—simple, interpretable models such as linear regression or decision trees—to approximate and explain the predictions of complex black-box models. In the LIME framework, surrogate models play a critical role in providing interpretable explanations for individual predictions.

- **local v.s global:** Local Interpretability refers to methods that aim to explain a model's prediction for a specific instance, in contrast to global interpretability, which seeks to explain the overall logic of a model. Local interpretability focuses on providing individualized explanations for specific inputs, addressing the question: *Why did the model make this particular prediction?* These explanations are particularly valuable for validating decisions, diagnosing errors, and gaining insights into specific predictions. LIME is one of the most representative local interpretable models. While building on the concept of LIME, its proposer[Ribeiro et al. (2018)] later introduced Anchors, a rule-based local explanation method that generates decision rules serving as "anchors" for predictions.

  Representation of global interpretability should be SHAP, proposed by [Lundberg (2017)], which leverages Shapley values from cooperative game theory to provide consistent and globally interpretable feature attributions.

  Actually, local and global methods should not be regarded as contrast to each other, but rather, researchers would like to contribute to both ways when proposing new ideas. Proposer of LIME suggests that by generating explanations for a representative set of instances, LIME can also help assess a model's overall behavior. Although this approach is not broadly applicable—particularly for non-numerical data—the idea provides an intriguing perspective: selecting representative local explanations rather than relying on randomly chosen instances. For further reflection, we discuss this idea in more detail in 3.8 SPLIME.

- **perturbation-based v.s gradient-based**: perturbation-based methods observes changes in model predictions by making slight perturbations to the input data, such as modifying input feature values or removing specific features in order to predict the importance of each feature. LIME is one of the perturbation-based methods.

  Gradient-based methods leverage the internal gradient information of the model to calculate the sensitivity or rate of change of input features with respect to the

output prediction in order to explain the model's decisions. GradCAM[Selvaraju et al. (2017)] used for CNN, also Gradient-weighted Class Activation Mapping, is one of the representation. It generates a heatmap that intuitively highlights the regions in the input most relevant to the prediction of that class by combining the gradient information of the target class with feature maps.

3 Theoretical concept

With the prior knowledge in hand, we begin to explore the theory behind LIME. One important distinction to keep in mind is the difference between the features of the original model and interpretable data representations. Interpretable data representations can be regarded as *post-hoc* techniques for interpreting already-trained models or as visualization tools that bridge the gap between model complexity and human interpretability. These representations simplify data by reducing its dimensionality, making it easier to understand. As an example, we can denote $x \in \mathbb{R}^d$ as the original representation of an instance being explained, and $x' \in \{0,1\}^{d'}$ as a binary vector representing its interpretable form. In later sections of our applications, we will explicitly illustrate how this process works.

3.1 Algorithm of LIME

The primary goal of LIME is to explain the predictions of complex machine learning models by approximating their behavior within the local neighborhood of a specific instance. And the process of LIME can be presented by following algorithm:

- Train a complex model f on the entire dataset X, treating it as a 'black box' whose internal workings are not interpretable.
- Select an instance ξ that we want to explain.
- Generate a perturbed dataset Z by creating N perturbations of ξ.
- Use f to predict the corresponding outputs $\hat{y}_Z$ for the perturbed samples in Z.
- Assign weights to the perturbed samples $z \in Z$, where the weights are calculated based on their distance to the original instance ξ.
- Train a weighted, interpretable surrogate model g (e.g., Ridge regression) on the perturbed dataset Z and their predictions $\hat{y}_Z$.
- Use the surrogate model g to approximate and explain the behavior of the complex model f in the local neighborhood of ξ.

To better understand the LIME algorithm, we use a two-dimensional classification problem on simulated tabular data for visualization. The data is defined with features x_1 and x_2, and the true label y, where points inside the circle $x^2 + (y + 0.5)^2 = 1$ belong to class 0, and points outside the circle belong to class 1. Additionally, random noise is added to the data and we name it $\tilde{y}$. A random forest classifier,acting as a black-box

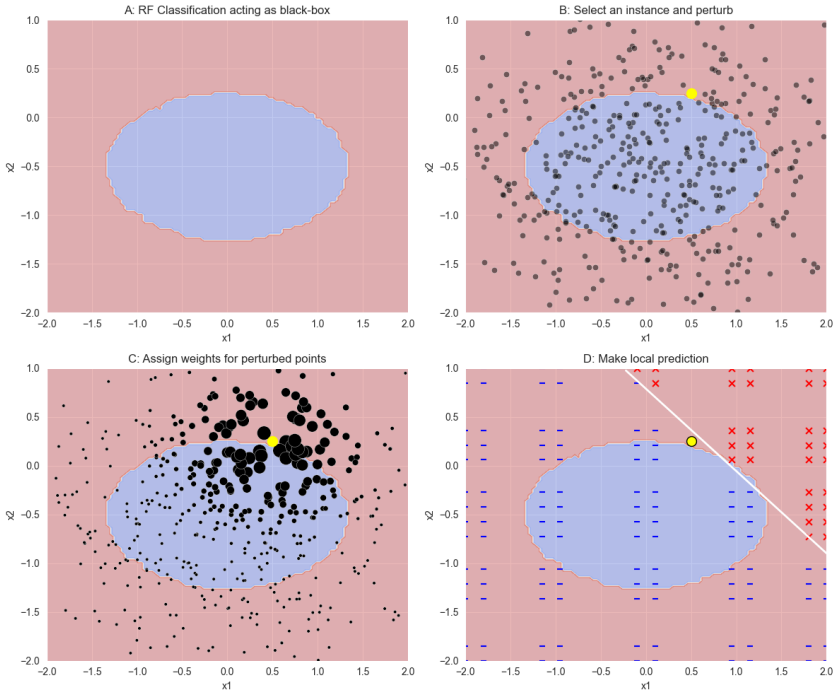

Figure 1: simplified representation of LIME algorithm

model, is then trained on x_1,x_2 and $\tilde{y}$ and its predictions are shown in Figure (A): the blue ellipse-shaped area is predicted as class 0, while the red area is predicted as class 1. Next, we randomly select an instance x near the decision boundary for explanation. To simulate its local neighborhood, perturbations are applied to the instance by sampling from a normal distribution, where the mean and standard deviation are derived from the training dataset. The selected instance is highlighted in yellow, and the perturbed samples are shown as black dots in Figure (B). Weights are then assigned to these perturbed samples based on their Euclidean distance to the instance x. Samples closer to x are given higher weights, represented by larger black dots in Figure (C). Finally, a logistic regression model is trained on this weighted perturbed dataset to provide a local interpretation. The resulting local decision boundary is visualized in white in Figure (D), with the grid symbols indicating classifications predicted by the original random forest model.
This visualization demonstrates that even in a highly non-linear classification problem, using an explainable model like LASSO or Ridge can provide a locally faithful approximation of the black-box model. However, it is also evident that LIME's effectiveness may diminish when the decision boundary is even more complex or when the instance point is located far from the decision boundary. These potential limitations will be discussed further.
Beside describing and visualizing, we also provide the formal mathematical definition of LIME, which minimizes the following objective function:

$$g^* = \arg\min_{g \in G} \mathcal{L}(f, g, \pi_\xi) + \Omega(g)$$

where we assume f as our original model and define $g \in G$ as an explanation model. G represents a family of explanation functions (a series of possible interpretable models

such as linear models, decision tree models, etc.). The term $\mathcal{L}(f, g, \pi_{\tilde{x}})$ is a weighted loss function that evaluates the quality of the explanation g with respect to the original model f, under the condition of a local kernel $\pi_{\tilde{x}}$. The regularization term $\Omega(g)$ is introduced to penalize the complexity of the explanation g.

3.2 Sampling strategy

In LIME, Sampling refers to the process of generating perturbed data points and assigning weights to these points. Thus, each sample generated through this process can be understood as consisting of two components: the perturbed data point and its corresponding weight. For further exploration, we can extend the weighted loss function $\mathcal{L}(f, g, \pi_x)$ and formulate LIME's task as follows :

$$g^* = \arg\min_{g \in G} \sum_{i=1}^{n'} \pi_\xi(x^{(i)}) L(f(x^{(i)}), g(z^{(i)})) + \Omega(g),$$

where $\tilde{x}$ refers to the instance of interest, $x^{(i)}$ represents the i-th perturbed data point sampled from the neighborhood of ξ, and n' is the total number of perturbed samples. The binary vector $z^{(i)}$ is the discretized and binarized representation of $x^{(i)}$, while $\pi_\xi(x^{(i)})$ is a weight function that quantifies the similarity between $x^{(i)}$ and ξ.
For the theoretical explorations in the subsequent sections, we focus primarily on tabular data. Sampling strategies and results for text and image data are covered in Section 4 with practical applications. This focus is motivated by two key reasons: the absence of a robust theoretical foundation for text and image data, and the shared underlying LIME concepts for explanations across all three data types. For the following exploration, we define $x \in \mathbb{R}^d$ and local surrogate model $g(z) = \hat{\beta}_0 + \sum_{j=1}^{d} \hat{\beta}_j z_j$.

3.2.1 Perturbation

The perturbation process in LIME involves both discretization and undiscretization steps [Garreau and Luxburg (2020)]. Given a fixed number of bins p, the empirical quantiles $\hat{q}_{j,0}, \ldots, \hat{q}_{j,p}$ are first computed for each feature j. A mapping $\hat{\phi}_j : \mathbb{R} \to \{1, \ldots, p\}$ is then introduced to assign each real number to the index of the quantile bin it belongs to. Based on this mapping, we define an indicator function c_j as: $c_j = \mathbf{1}_{\hat{\phi}_j(x) = \hat{\phi}_j(\xi)}$for all $1 \leq j \leq d$, which indicates whether the perturbed sample $x^{(i)}$ and the instance of interest ξ belong to the same quantile bin. During this step, the instance ξ is discretized into $\hat{\phi}(\xi)$.
Next, perturbed samples $\hat{\phi}(x^{(i)})$ are randomly drawn from the discretized feature space. These samples are then mapped back to the continuous feature space via an undiscretization procedure $\psi : \{1, \ldots, p\} \to \mathbb{R}$, which involves sampling at each feature dimension within each bin using a truncated Gaussian distribution $N(\mu, \sigma^2)$. One resulting perturbed sample in the continuous feature space is denoted as $x^{(i)} = (x_1^{(i)}, \ldots, x_n^{(i)}) \in \mathbb{R}^d$.
In the discretized feature space, we apply the indicator function c_j to compare $\hat{\phi}(x^{(i)})$ with $\hat{\phi}(\xi)$. The binary representations of one perturbed sample, used for the local surrogate model, is denoted as $z^{(i)} = z_1^{(i)}, \ldots, z_n^{(i)} \in \{0, 1\}^d$. Iterating this perturbation process, we obtain $x^{(i)}$ and $z^{(i)}$ for further uses.

3.2.2 Weight function: distance and kernel width

In this section, we further discuss π_ξ, which is also referred to as a kernel function based on a chosen distance function D. In LIME, $\pi_{\tilde{x}}$ is defined as:

$$\pi_\xi(x^{(i)}, \xi) = \exp\left(-\frac{D(x^{(i)}, \xi)^2}{\nu^2}\right),$$

The exponential function is chosen as the kernel in LIME because it smoothly maps distances to weights, ensuring that weights decay rapidly as the distance increases.
Distance functions are different across datatypes. For **numerical features**, distance functions such as the **Manhattan distance (L1 norm)** and the **Euclidean distance (L2 norm)** are commonly used. L1 is robust to outliers and emphasizes linear relationships while L2 is sensitive to outliers but suitable for measuring spatial relationships in continuous spaces. For **categorical features**, the **Hamming distance (L0 norm)** is frequently employed. This distance is suitable for measuring dissimilarities in discrete, unordered data. When dealing with **mixed data types**, **Gower's distance** is a recommended choice, as it normalizes each feature's contribution to the overall distance. For **text data**, the **cosine distance** is particularly effective. It measures the angular difference between vector representations, capturing the similarity in word distributions regardless of vector magnitude. For **image data**, the **Euclidean distance** is again recommended due to its ability to measure pixel-wise differences effectively.
The parameter ν, known as the kernel width, determines the rate at which weights decrease. Therefore, the kernel width directly affects the neighborhood of the explainer and, consequently, the interpretability results. To illustrate the importance of choosing an appropriate kernel width, we simulate data with one target variable and two features: one is pure noise, and the other exhibits a non-linear cosinoidal effect on the target. Two models are trained on this simulated data:

- A **random forest Regressor**, acting as a black-box model, which provides predictions $\hat{y}$.
- A **piecewise linear model**, serving as a reference to evaluate LIME's accuracy.

We then select an instance (marked in red) and train a linear model as the explainer for LIME using different kernel widths. Since x_2 is defined as pure noise, we simplify the visualization to focus on x_1 and $\hat{y}$.
The first plot presents the true cosinoidal function (red curve) and predictions made by the random forest (black dots). It shows that the random forest model accurately fits the true function. The second plot displays the piecewise linear function, which serves as the ground truth for evaluating LIME's local interpretability. To analyze the kernel-width problem, we examine two instance points: one with a positive true local slope and another with a negative one, supported by the piecewise linear function.
From plots (C) and (D), we observe that when the kernel width is too small (depicted as the blue constant fit), LIME loses its local faithfulness, failing to provide any explanation. Mathematically, this is because only examples in the immediate neighborhood receive positive weights, leading to missing all relevant information. As the kernel width increases,

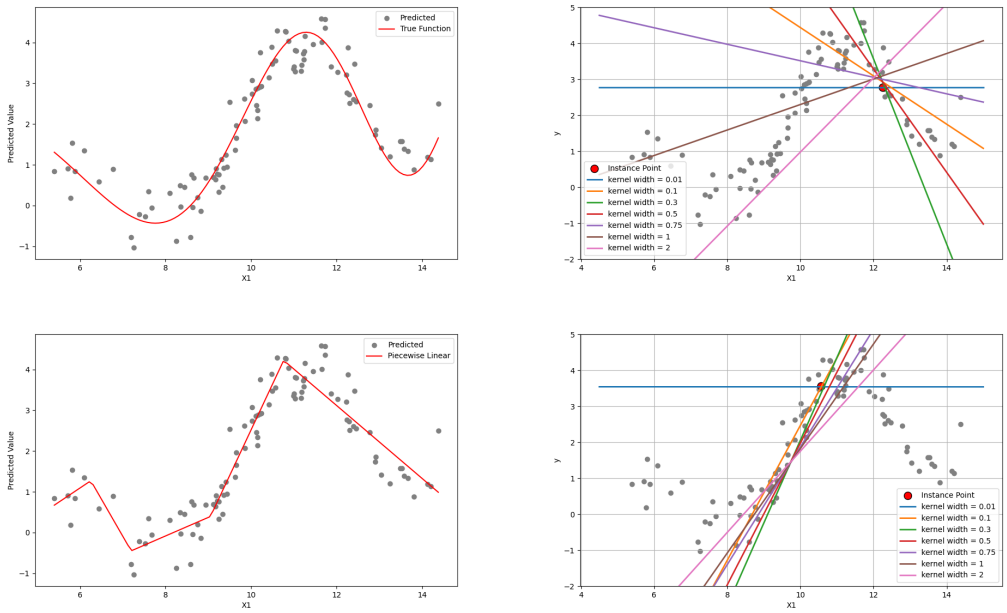

Figure 2: LIME under different kernel width

more data points begin contributing to the interpretation and LIME starts to perform a local fit.
However, when the kernel width becomes very large, the explainer starts to captures the complexity of the black-box model at a global scale. In such cases, all perturbed data points receive positive weights, leading the explainer to approximate a global model rather than capturing local behavior.

3.3 Coefficients of the local explainer

In this section, we set $\Omega(g)$ to zero for simplified derivation. After the weighted perturbation process, we minimize the task function and then analyze the coefficients of the local surrogate model, denoted as $\hat{\beta}$. And we define β as the average value of it and denote the gradient of f that represents the true feature importance by $a_1, a_2, \ldots, a_d$. To simplify the derivation, we assume f is of the form $x \mapsto a^T x + b$, the weighted mean and weighted variance of perturbed samples x as :

$$\tilde{\mu} = \frac{\nu^2\mu + \sigma^2\xi}{\nu^2 + \sigma^2}, \quad \tilde{\sigma}^2 = \frac{\nu^2\sigma^2}{\nu^2 + \sigma^2}$$

then the coefficients β can be expressed as:

$$\beta := \begin{pmatrix} f(\tilde{\mu}) + \sum_{j=1}^{d} \frac{a_j\theta_j}{1-\alpha_j} \\ -\frac{a_1\theta_1}{\alpha_1(1-\alpha_1)} \\ \vdots \\ -\frac{a_d\theta_d}{\alpha_d(1-\alpha_d)} \end{pmatrix} \in \mathbb{R}^{d+1},$$

where, for any $1 \leq j \leq d$, we define

$$\alpha_j := \left[\frac{1}{2}\text{erf}\left(\frac{x-\tilde{\mu}_j}{\tilde{\sigma}\sqrt{2}}\right)\right]_{q_{j-}}^{q_{j+}},$$

and

$$\theta_j := \left[\frac{\tilde{\sigma}}{\sqrt{2\pi}}\exp\left(-\frac{(x-\tilde{\mu}_j)^2}{2\tilde{\sigma}^2}\right)\right]_{q_{j-}}^{q_{j+}}.$$

α_j, based on the error function (erf), represents the probability weight of the perturbed samples falling within the specific interval $[q_{j-}, q_{j+}]$ for feature j. θ_j represents the distribution intensity of the perturbed samples within the interval $[q_{j-}, q_{j+}]$, indicating the peak value or density concentration of the distribution. β_0 represents the intercept of target model on the weighted mean $\tilde{\mu}$, while β_j that represents the predicted feature importance, is given by $-\frac{a_j\theta_j}{\alpha_j(1-\alpha_j)}$.
We can further define the local error of the surrogate model with probability greater than $1-\eta$ as:

$$\left|\hat{f}(\xi) - f(\xi) + \sum_{j=1}^{d}\frac{a_j\theta_j}{\alpha_j}\right| \leq \max\left(\sigma\|f\|, f(\mu) + \tilde{\sigma}\|f\|\right)\sqrt{\frac{\log(1/\eta)}{n}},$$

where $\eta \in (0,1)$ represents the confidence level used to bound the probability of error in the approximation and $\max\left(\sigma\|f\|, f(\mu) + \tilde{\sigma}\|f\|\right)$ is a scaling constant.

3.4 Regularization of the local Explainer

A key property of a local explainer is its ability to balance interpretability and prediction accuracy. This balance is achieved through the regularization term $\Omega(g)$, which penalizes model complexity to ensure simplicity in the explanation. If the regularization is too strong, important features may be excluded; if it is too weak, the explanation may become overly complex and less interpretable. Mathematically for a linear model, $\Omega(g)$ is often defined as $\Omega(g) = \lambda\|w_e\|_2$, where λ controls the strength of regularization, and $\|w_e\|_2$ is the L_2-norm of the model's weights.
To further control the complexity of the local surrogate model g, a parameter K is introduced as a direct constraint on the number of features included in the explanation. For example, in decision trees, K can correspond to hyperparameters like tree depth, while in Ridge or LASSO, K specifies the maximum number of dominant features to retain. In practice, K ensures that the explanation remains interpretable by limiting the number of features under consideration. Take LIME's Python implementation for example, this complexity control is realized through the parameter `num_feature`, which directly specifies the maximum number of features K to include in the explanation. By combining $\Omega(g)$ for penalizing complexity and K for limiting feature selection, LIME effectively balances interpretability with prediction accuracy.

3.5 SPLIME: an approach to global

As mentioned in the related work, this section seeks to enhance the understanding of how local interpretability for a specific instance can be extended to global interpretability of

the overall model behavior. Submodular Pick LIME, also SPLIME introduced by Ribeiro et al. (2016), aims to bridge this gap by extending LIME from local to global interpretability. While it proves effective for tabular and text data, it encounters significant challenges when applied to high-dimensional or unstructured data, such as images, which still lack solutions. Given these limitations, we just provide a brief overview of SPLIME using a simplified toy example, with the detailed algorithm deferred to the Appendix.
By selecting a representative subset of instances through submodular optimization based on feature weights across different instances and using a greedy algorithm, SPLIME approximates the overall model behavior using explanations by picking these instances. The following toy example illustrates the basic idea:

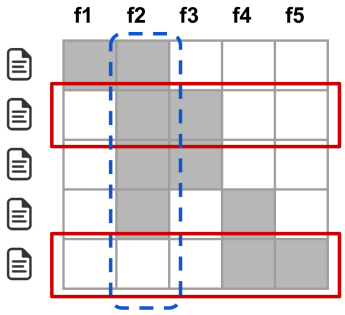

Figure 3: Toy example for SPLIME

In this simplified example, each row represents an instance, and each column represents a feature $(f_1, f_2, \ldots, f_5)$. The weight of each feature is binary: grey blocks indicate that the feature is selected for the instance, while white blocks mean it is not selected. Based on the contributions of the features, the **importance scores** for the five features are ranked as follows: $f_2 > f_3 = f_4 > f_1 = f_5$.
Following this ranking, SPLIME first selects the second row, as it has the highest scores overall because of $(f_2 + f_3)$. Once the second row is selected, the contributions of f_2 and f_3 are regarded as "covered" (i.e., their importance scores are set to zero). Among the remaining rows, the fifth row is chosen next, as it provides the highest remaining importance score by contributing f_4 and f_5. This iterative process ensures that the selected subset of instances maximally represents the overall feature importance across the dataset.

4 Applications

After establishing the theoretical foundations of LIME, we now shift our attention to its practical applications. This section examines how LIME can be applied across various data types and tasks, with a primary focus on classification problems. In particular, we demonstrate how LIME generates and perturbs samples for text and image data (process for tabular data presented in Figure 1) in this subsection. Furthermore, we extend the discussion to multiclass classification as a natural progression from binary classification.

For regression tasks, we provide an example of an explanation using tabular data to show the adaptability of LIME to handle continuous prediction problems.

4.1 LIME for Classification

4.1.1 Binary Classification on Tabular Data

We begin by exploring a binary classification task using the classic wine quality dataset from UCI Repository. This dataset contains numeric features and binary labels, denoted as *good* and *bad*. For this task, we employ a Random Forest (RF) classifier as the black-box model, where the outputs represent the probabilities for each class for individual samples. For LIME, we use the default Ridge model as the local surrogate. And for all the tabular and text tasks below, we use these two models. It is important to clarify why Ridge, a regression model, can be used in this classification context. This is because LIME explains the predicted probabilities rather than the discrete class labels. These probabilities are treated as regression targets, which Ridge approximates using a sparse linear model. We select a random instance from the test dataset and present its explanation below.

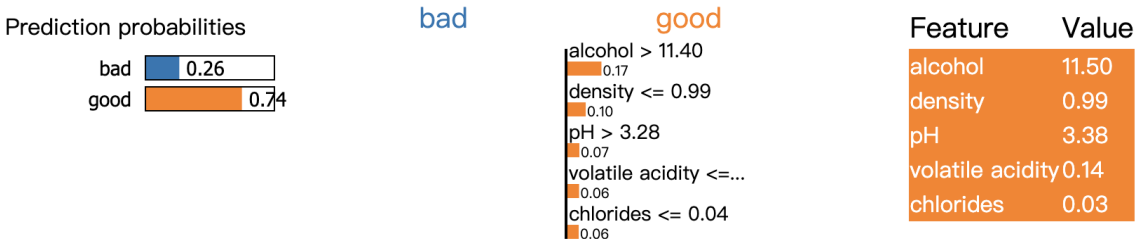

Figure 4: Binary Classification on Numerical Features

To interpret the results, let us focus on each section of the explanation. The **left panel** shows the predicted probabilities for the binary labels, as provided by the RF classifier. It is crucial to note that LIME does not alter these probabilities—it simply serves as a method to explain the model's predictions. The **right panel** lists the features and their corresponding values for this sample. Orange-colored features contribute positively to the prediction of *good*, while blue-colored features indicate negative contributions (i.e., they push the prediction toward *bad*).
Finally, the **middle panel** presents the detailed LIME explanation. Here, we observe that LIME discretized the continuous numerical features into pseudo classes. This behavior is controlled by LIME's internal settings, which can be adjusted to disable discretization if needed. The purpose of discretization is to enhance interpretability, as explaining continuous numerical values directly can be less intuitive. With discretization, the explanation becomes clearer and easier to interpret. Take the feature *alcohol* as an example. The value $alcohol = 11.50$ is categorized into the interval $alcohol > 11.40$. This means that, holding other features constant (*ceteris paribus*), being in the interval $alcohol > 11.40$ increases the predicted probability of this sample being classified as *good* by 0.18. Alternatively, this can be interpreted in reverse: if $alcohol <= 11.40$, the predicted probability of this sample being classified as *good* decreases *ceteris paribus* in average by approximately 0.18.

It is also worth mentioning that the weight 0.18 will remain consistent for another sample that also falls into the interval *alcohol* $>$ *11.40*, as long as the explanation is generated using the same random seed. It is easy to understand, as the local surrogate model is not changed under the same seed.
Now, we turn our attention to classification tasks that involve mixed features, i.e., both categorical and numerical data. To process categorical features for explanation, we first apply one-hot encoding to transform them into binary format before passing them through the weighted perturbation mechanism. For visualization, we use the Titanic dataset to classify whether or not a passenger survived.

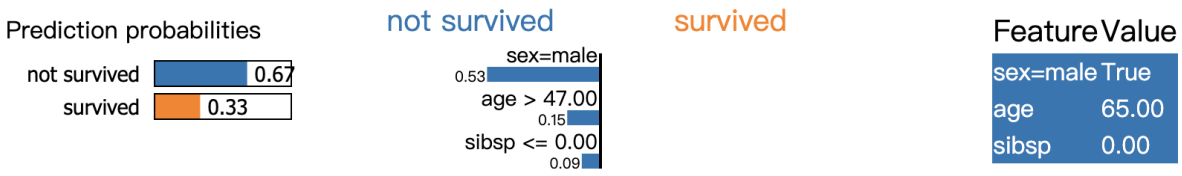

Figure 5: Binary Classification on Mixed Features

The methodology for explaining is consistent with the previous example, as explaining numerical features after discretization is conceptually similar to explaining categorical features. For the categorical feature *sex* in this case, we can conclude that being male increases *ceteris paribus* the predicted probability of *not survived* by 0.53.

4.1.2 Binary Classification on text data

The perturbation process in text classification tasks is illustrated in the table below. Each row corresponds to a perturbed sample, generated by randomly masking specific words from the original text. For each perturbed sample, the black-box model predicts a probability, and a weight is calculated based on its similarity to the original text using cosine similarity. These perturbed samples, along with their corresponding weights and predictions, are then used to fit a local surrogate model for interpretability.

It	is	quite	rare	that	a	movie	comes	along	that	is	so	useless	prob	weight
0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.013264961159866528	0.23076923076923073
0.0	1.0	0.0	1.0	1.0	1.0	0.0	1.0	0.0	1.0	1.0	1.0	1.0	0.9422017556848528	0.6923076923076923
1.0	1.0	1.0	1.0	0.0	0.0	1.0	1.0	1.0	0.0	1.0	0.0	0.0	0.5632882178455393	0.6153846153846154
0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	0.0	1.0	0.3854165025399161	0.6153846153846154
0.0	1.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.015966252220214194	0.23076923076923073

Figure 6: Simulated Perturbation in Text Classification Tasks

Although this process may appear similar to tasks based on categorical tabular features after encoding, the weighting function represents a fundamental difference.
The following example illustrates an explanation produced by LIME on the IMDB dataset, where movie reviews are classified as either *negative* or *positive*. The explanation methodology remains consistent with the process used for tabular data. From the output, we observe that the word *bad* is the most significant feature contributing to predicting the

review as negative. LIME also highlights important words directly in the original text, with deeper or lighter highlights indicating the relative contribution of each word to the prediction.

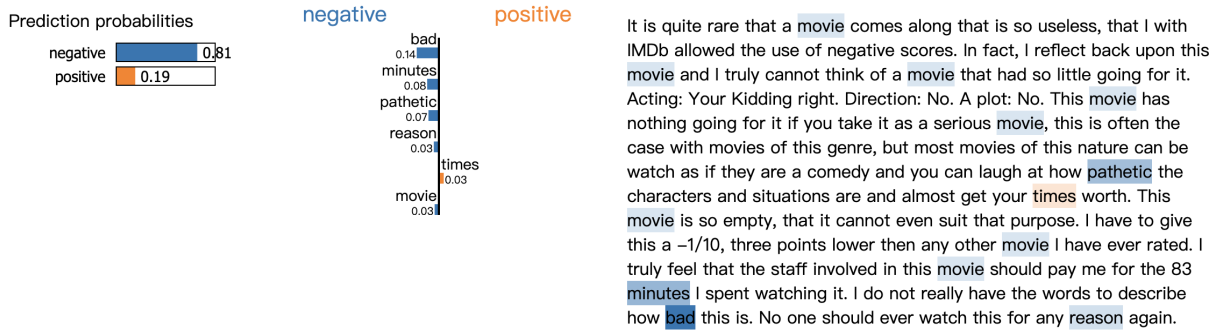

Figure 7: Binary Classification on Text

4.1.3 Multi-class Explanations

Next, we provide an example of multi-class explanations using the Iris dataset. In multi-class scenarios, instead of displaying explanations for all labels in a single plot, LIME displays whether a feature contributes positively or negatively to each label individually. In practice, we can customize the label selection by either modifying the parameter *top labels* or manually specifying the labels to explain.

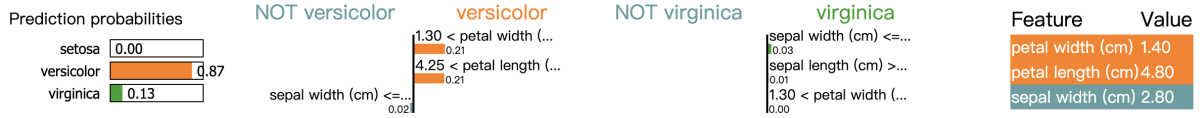

Figure 8: Multi-class Explanations Using the Iris Dataset

As illustrated in the figure, the **right panel** highlights the explanation for the selected class. The colors indicate the class for which each feature contributes most positively to the prediction. For this instance, we can explain that sepal width contributes negatively to classifying as versicolor, while positively to classifying as virginica.

4.1.4 Classification on images

After analyzing LIME on tabular data and text, we now explore its application to image classification tasks. Unlike tabular or text data, image classification models are typically designed for multi-class classification rather than binary classification. Modern models such as Convolutional Neural Networks (CNNs) inherently handle multi-class problems, even if they occasionally output one or two top label predictions. For this analysis, we use Google Inception V3 as the black-box model and a Ridge model as the local interpreter. We start by explaining how sampling and perturbation work in image data. Each pixel in an image can be treated as a feature. However, using individual pixels for prediction is often impractical since many pixels collectively contribute to a class prediction. To address

this, LIME employs superpixels, which are clusters of pixels with similar characteristics, such as color or texture, for perturbation and sampling. A set of perturbed images is then generated by randomly masking superpixels. The weight of each perturbed image is computed as the probability of the target class predicted by the black-box model. These perturbed samples, along with their corresponding weights, are then used to train the local Ridge model for explanation. The following figure illustrates how perturbation works in image classification based on a image with tabby and collie from Imagenet. The probabilities predicted by the Inception V3 model for the tabby and collie classes are 0.45 and 0.12.

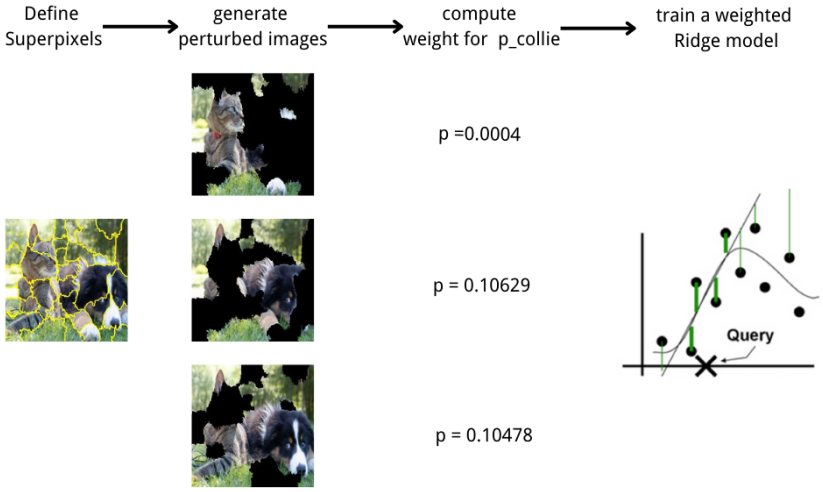

Figure 9: simulated perturbation in Image Classification tasks

And the following figure shows the result of LIME's explanation with sample size = 5000 on two labels, collie and tabby. In this visualization, we choose to display both positive (green) and negative (red) superpixels while regularizing the number of features to 20.

Figure 10: Classify as Collie (left) and Tabby (right).

When comparing the explanations for the Collie and Tabby labels, we observe distinct patterns in the superpixel contributions. For the Tabby class, green regions highlight textures and colors representative of a cat, such as its fur and body shape, whereas red regions identify features more characteristic of the Collie, such as the dog's face or background features.

4.2 LIME for Regression

Since we explore classification tasks explictly, we provide only one example of its application to a regression task to allign with LIME's core theme of model-agnosticism—it can be used to explain all types of predictive tasks. However, we do not provide a detailed discussion of regression examples, as the explanation methodology for regression is conceptually similar to that for classification. For classification tasks, LIME provides explanations for predicted probabilities, whereas for regression tasks, it explains continuous numerical predictions. Based on the diabetes dataset in Python, which predicts the progression of diabetes using numerical values ranging from 25 to 346 (with higher values indicating more severe progression), we explain the predictions using LIME as follows:

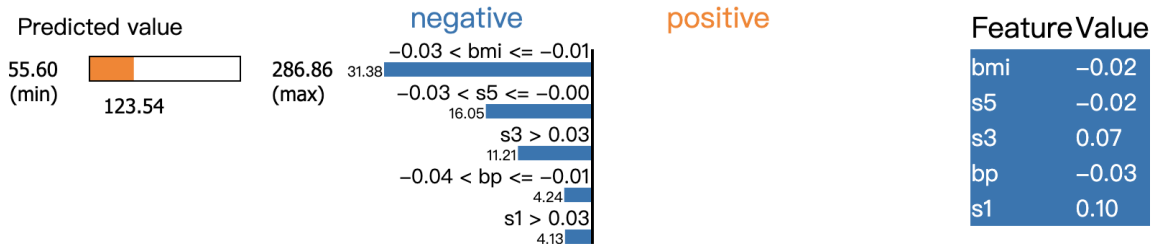

Figure 11: LIME on Regression

For regression tasks, LIME provides additional outputs in Python to help explain the predictions. These outputs include three key values: **Intercept**, which represents the baseline prediction when all feature contributions are zero; **Prediction_local**, the prediction made by the local Ridge model; and **Right**, the prediction generated by the original model. In the example shown, the values are as follows: Intercept = 176.41, Prediction_local = 109.39, and Right = 123.54. Taking the feature *bmi* as an example, we can interpret that a *bmi* value between -0.03 and -0.01 has the most significant negative effect on the predicted value. Specifically, it decreases the target value *ceteris paribus* by an average of 31.38 for this specific instance.

5 Discussion

LIME offers several advantages. The algorithm is both theoretically intuitive and practically straightforward to implement. As a model-agnostic approach, LIME provides a consistent interpretation framework that can be applied regardless of the underlying black-box model. It supports a variety of tasks, including classification and regression, and is adaptable to different data modalities such as tabular, text, and image data. Moreover, the explanations generated by LIME are concise and easy to interpret, making them particularly user-friendly. However, it is important to acknowledge its limitations. One notable drawback lies in its sampling strategy. One issue arises from LIME's use of univariate distributions for sampling. This approach ignores the covariance structure among features, potentially generating unrealistic samples that deviate from the real data distribution. For instance, in the case of categorical features, LIME may produce combinations

that are impossible in real-world scenarios. Though the impact is assumed not to be affective, this problem should be explored with further experiments.
Also, for the sake of simplicity, LIME samples from a global distribution, which can result in inconsistent and unreliable explanations, even under identical scenarios. This issue is illustrated in the figure below, where different random seeds produce significantly different outcomes.

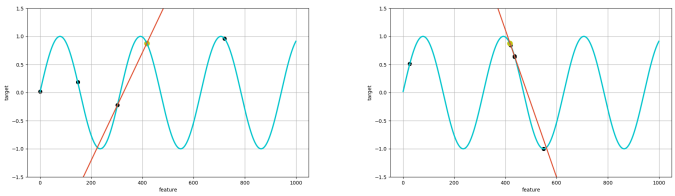

Figure 12: LIME using different random seeds

Although increasing the sample size leads to more stable and reliable explanations, the marginal improvement diminishes as the sample size grows. This is also proven by our theory, that as the number of perturbed samples n increases, the error bound of LIME converges at a rate of $O\left(\frac{1}{\sqrt{n}}\right)$. This highlights the need to strike a balance between sample size and computational efficiency, a task that is inherently challenging.

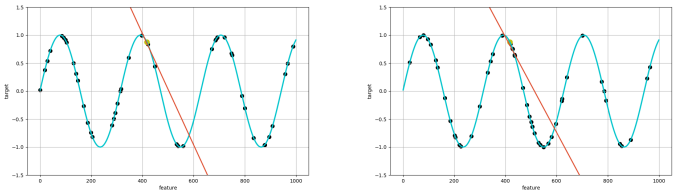

Figure 13: LIME with increased sample size

Similar trade-offs are evident in kernel width settings. The optimal configuration varies depending on the task: if the kernel width is too small, the explanations become less informative; if it is too large, the local model begins to approximate global behavior, yielding explanations that are neither locally nor globally accurate.
Furthermore, despite its model-agnostic nature, LIME has limitations in specific tasks. For instance, when dealing with graph-structured data that includes nodes and edges, LIME struggles to capture feature interactions effectively, often producing non-informative results. This limitation is illustrated in Figure 14, which shows explanations generated for the Cora dataset that fail to provide meaningful insights.

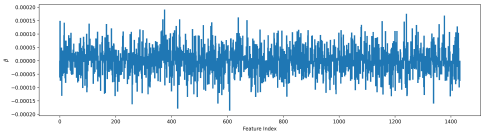

Figure 14: LIME's limitation on graph-structured data

6 Extension of LIME

Following our discussion, we now explore several improved methods that address some of the limitations of LIME. [Laugel et al. (2018)] proposed a LIMEK using the *Growing-Spheres* algorithm. This method generates a series of gradually expanding hyperspheres in the feature space of x until it identifies a point x_{border} that crosses the decision boundary of the black-box model. Centered on x_{border}, a hypersphere S with a radius of r_{sx} is defined, from which n sample points are uniformly drawn. While [Huang et al. (2022)] introduced GraphLIME, an extension of LIME tailored for graph-structured data. It samples the N-hop neighbors of a target node (where N-hop denotes the shortest path distance through the graph's edges) to construct a local subgraph. The features of the target node and its neighbors are combined into a feature matrix X_n, and the corresponding prediction results y are generated. HSIC-Lasso is then applied to perform nonlinear modeling of the input features.
However, despite their advancements, both LIME-K and GraphLIME retain LIME's essential random perturbation mechanism. In contrast, two other variations, DLIME (Deterministic LIME) and BayLIME (Bayesian LIME), introduce fundamental changes to LIME's core methodology. DLIME replaces the random perturbations with deterministic sampling techniques, while BayLIME leverages Bayesian inference to account for uncertainty in the explanation process. These distinct approaches make DLIME and BayLIME particularly valuable for comparative analysis. In the following sections, we evaluate and compare their performance with the original LIME framework.

6.1 DLIME

DLIME employs Agglomerative Hierarchical Clustering (AHC), a bottom-up clustering method that iteratively merges the closest points or clusters to construct a hierarchical structure. This process ultimately forms a tree-like representation, which is used to partition the training dataset into different clusters. After this, the test instance is assigned to the nearest cluster, and KNN is applied to identify its closest neighbors within that cluster. DLIME is termed deterministic because it establishes an efficient and meaningful selection rule for choosing samples from the training set, in contrast to the more random sample selection process used in LIME. For subsequent steps, DLIME shares the same weighting and local prediction methodology as LIME.
Mathematically, the process can be defined as follows: assign ξ to:

$$\mathcal{C}_{\text{nearest}} = \arg\min_{\mathcal{C}_i} \min_{\mathbf{x}_j \in \mathcal{C}_i} d(\xi, \mathbf{x}_j),$$

and then apply KNN as:

$$\mathcal{N}_{\text{knn}} = \{\mathbf{x}_{j_1}, \mathbf{x}_{j_2}, \ldots, \mathbf{x}_{j_k}\}, \quad j_1, j_2, \ldots, j_k = \arg\min_{j} d(\xi, \mathbf{x}_j), \quad \mathbf{x}_j \in \mathcal{C}_{\text{nearest}}.$$

Using the wine quality dataset again, we compare DLIME with LIME on a specific instance, with different random seeds applied to the explainer in LIME.
As expected, DLIME produces consistent results across multiple runs, as its sample selection is deterministic. In contrast, LIME's results fluctuate due to its randomized perturbation process. For instance, the feature marked in green retains its significant importance

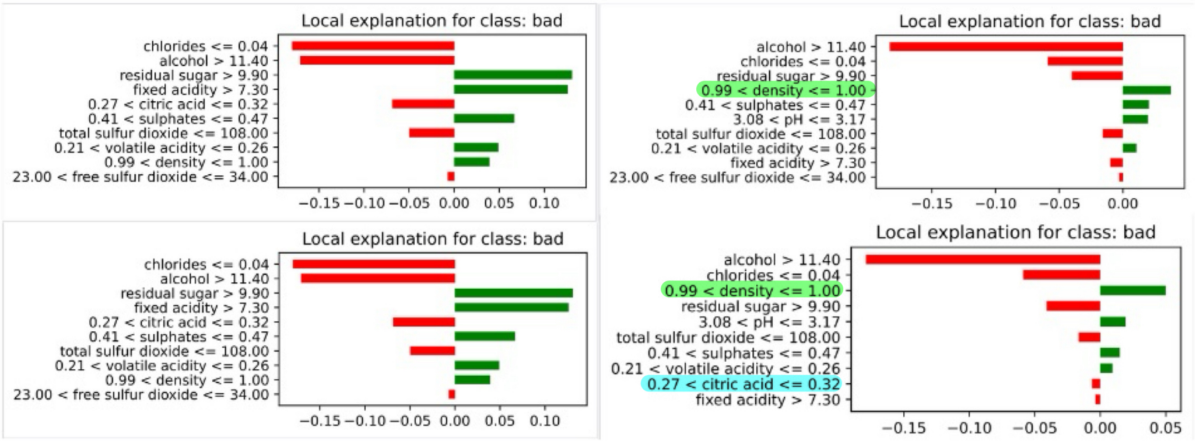

Figure 15: Comparing DLIME and LIME

across all experiments in DLIME, while LIME shows variations in its weight. On the other hand, the feature marked in blue is not selected in both LIME runs, highlighting a potential limitation in its sample selection strategy.
These results demonstrate that DLIME provides stable and faithful explanations, as it draws samples exclusively from the true training set rather than relying on perturbed data points. However, this approach also introduces potential drawbacks and new challenges. The fidelity may worsen if the training dataset is small or the AHC-based selection area is overly restricted due to the AHC settings, causing DLIME to fail in capturing broader patterns and relationships. While one of LIME's origin problem is also not solved: the trade-off between computational efficiency and sample size, as we still need to rely on KNN-selections.

6.2 BayLIME

BayLIME inherits LIME's random perturbation method. However, it introduces prior knowledge that can significantly reduce the impact of random perturbations. The concept of BayLIME is fundamentally aligned with the Bayesian linear regression model, leveraging a conjugate normal-normal prior and likelihood distribution.
Given the perturbation samples (X, y), where:

$$X = \{x_1, \ldots, x_n\}, \quad \boldsymbol{y} = [y_1, \ldots, y_n]^T,$$

the posterior estimate of β is defined as:

$$\mu_n = S_n(\lambda I_m \mu_0 + \alpha X^T W y), \quad S_n^{-1} = \lambda I_m + \alpha X^T W X,$$

where: λ is the precision parameter governing the Gaussian prior, I_m is an $m \times m$ identity matrix, and $W = \text{diag}(w_1, \ldots, w_n)$ is a diagonal matrix of weights determined by the kernel function.
The first equation can be rewritten as:

$$\mu_n = (\lambda I_m + \alpha X^T W X)^{-1} \lambda I_m \mu_0 + (\lambda I_m + \alpha X^T W X)^{-1} \alpha X^T W X \beta_{\text{MLE}},$$

where:

$$\beta_{\text{MLE}} = (X^T W X)^{-1} X^T W y$$

is the Maximum Likelihood Estimate (MLE) for the linear regression model.
To construct priors, we can leverage the explanations of a set of similar instances or other XAI techniques based on fundamentally different principles, such as SHAP or Grad-CAM. In this example, we use Grad-CAM as prior knowledge and train a LIME explanation with a sample size of 200, based on one of the most classic images from ImageNet, to classify the image as a dog.

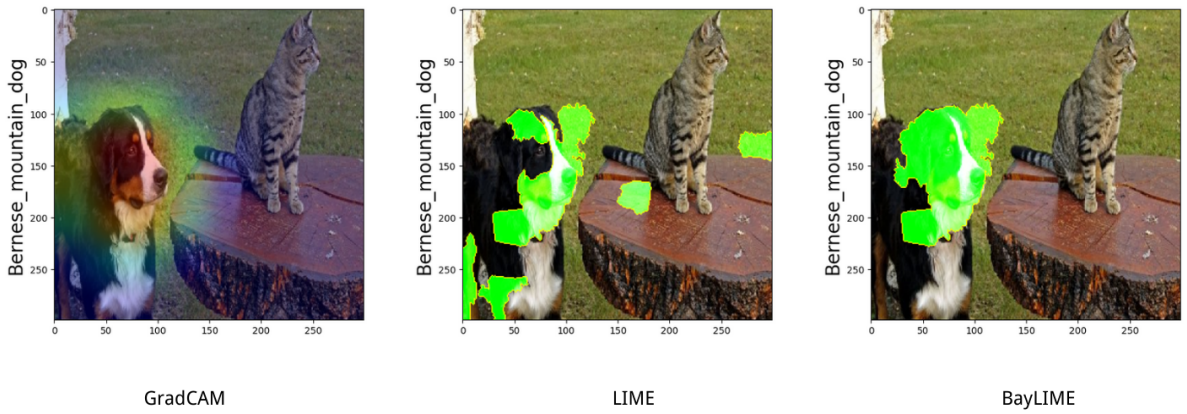

Figure 16: Comparison of BayLIME and LIME

From the plot, we observe that BayLIME significantly improves explanation fidelity. Under explanation of LIME, several non-relevant superpixels are selected as the top features for explaining, while the features selected by LIME are concentrated mostly on the feature of a dog.
However, it is plausible that LIME could achieve similar results by increasing the sample size to a sufficiently large number, as $\mu_n \to \beta_{\text{MLE}}$, if $n \to \infty$. What's more, BayLIME relies on the fidelity of prior knowledge, which introduces both opportunities and risks. While prior knowledge can enhance explanation stability and accuracy, it comes at the cost of simplicity, which is one of LIME's major advantages. Additionally, there is a potential risk of incorporating incorrect prior knowledge, particularly when the priors are derived from explanations of similar instances. Such priors may not always be rigorous, and in these cases, BayLIME could generate stable but misleading explanations.

7 Conclusion

In this paper, we conduct a comprehensive exploration of LIME, encompassing its theoretical foundations, practical applications, and various extensions. We highlight its model-agnostic nature and interpretability through applications across diverse tasks and data types. However, through simulated experiments on random perturbation and kernel width, we also identify inherent limitations, particularly regarding instability. By examining LIME's variations, we find that while these adaptations can mitigate instability to

some extent and enhance stability, they also introduce new challenges, and the original issues remain only partially resolved.
Despite the stagnation of the LIME package, which has not been updated for over three years, and the possibility that this method may eventually phase out, it is important to acknowledge LIME's milestone contribution to the field of explainable AI. It pioneered a novel and impactful approach to model interpretation, setting the stage for subsequent advancements in the field. Our future directions do not solely focus on LIME itself, but rather on its underlying concepts and their broader implications. Building on the foundational concepts of LIME, future research can explore more advanced techniques to overcome LIME's limitations while extending its applicability to new domains and challenges:

- Exploration on adaptive sampling strategies or deterministic perturbation methods that could address the instability caused by random sampling.
- Extensions on time-series data or generative models like GANs and transformers to broaden the utility of LIME-inspired frameworks in modern AI applications.
- Integration of LIME's core concepts with current debate. Incorporating causal inference techniques could enable XAI to move beyond correlation-based explanations and uncover causal relationships between features. Similarly, uncertainty quantification, as exemplified by BayLIME, could improve the reliability of explanations, particularly in high-stakes domains such as healthcare and finance.

A Appendix

$$\pi_x(z, x) = \exp\left(-\frac{D(z, x)^2}{\sigma^2}\right).$$

$$D_{L2}(z, x) = \sqrt{\sum_{i=1}^{n}(z_i - x_i)^2},$$

$$D_{L0}(z, x) = \sum_{i=1}^{n} \mathbb{1}(z_i \neq x_i),$$

$$D_{\text{Gower}}(z, x) = \frac{1}{n}\sum_{i=1}^{n}\frac{d_i(z_i, x_i)}{R_i},$$

$$D_{\text{cosine}}(z, x) = 1 - \frac{\sum_{i=1}^{n} z_i x_i}{\sqrt{\sum_{i=1}^{n} z_i^2} \cdot \sqrt{\sum_{i=1}^{n} x_i^2}},$$

References

Adadi, A. and Berrada, M. (2018). Peeking inside the black-box: a survey on explainable artificial intelligence (xai), *IEEE access* **6**: 52138–52160.

Aljadani, A., Alharthi, B., Farsi, M. A., Balaha, H. M., Badawy, M. and Elhosseini, M. A. (2023). Mathematical modeling and analysis of credit scoring using the lime explainer: a comprehensive approach, *Mathematics* **11**(19): 4055.

Garreau, D. and Luxburg, U. (2020). Explaining the explainer: A first theoretical analysis of lime, *International conference on artificial intelligence and statistics*, PMLR, pp. 1287–1296.

Huang, Q., Yamada, M., Tian, Y., Singh, D. and Chang, Y. (2022). Graphlime: Local interpretable model explanations for graph neural networks, *IEEE Transactions on Knowledge and Data Engineering* **35**(7): 6968–6972.

Kumarakulasinghe, N. B., Blomberg, T., Liu, J., Leao, A. S. and Papapetrou, P. (2020). Evaluating local interpretable model-agnostic explanations on clinical machine learning classification models, *2020 IEEE 33rd international symposium on computer-based medical systems (CBMS)*, IEEE, pp. 7–12.

Lakshmi, M. and Das, R. (2023). Classification of monkeypox images using lime-enabled investigation of deep convolutional neural network, *Diagnostics* **13**(9): 1639.

Laugel, T., Renard, X., Lesot, M.-J., Marsala, C. and Detyniecki, M. (2018). Defining locality for surrogates in post-hoc interpretablity, *arXiv preprint arXiv:1806.07498* .

Lundberg, S. (2017). A unified approach to interpreting model predictions, *arXiv preprint arXiv:1705.07874* .

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

Ribeiro, M. T., Singh, S. and Guestrin, C. (2018). Anchors: High-precision model-agnostic explanations, *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D. and Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization, *Proceedings of the IEEE international conference on computer vision*, pp. 618–626.

Tesfay, W. B., Hofmann, P., Nakamura, T., Kiyomoto, S. and Serna, J. (2018). I read but don't agree: Privacy policy benchmarking using machine learning and the eu gdpr, *Companion Proceedings of the The Web Conference 2018*, pp. 163–166.

Weld, D. S. and Bansal, G. (2019). The challenge of crafting intelligible intelligence, *Communications of the ACM* **62**(6): 70–79.

Declaration of authorship

I hereby declare that the report submitted is my own unaided work. All direct or indirect sources used are acknowledged as references. I am aware that the Thesis in digital form can be examined for the use of unauthorized aid and in order to determine whether the report as a whole or parts incorporated in it may be deemed as plagiarism. For the comparison of my work with existing sources I agree that it shall be entered in a database where it shall also remain after examination, to enable comparison with future Theses submitted. Further rights of reproduction and usage, however, are not granted here. This paper was not previously presented to another examination board and has not been published.

Location, date

Name