Improving the Quality of Explanations with Local Embedding Perturbations

Yunzhe Jia
University of Melbourne
Melbourne, Australia
yunzhej@student.unimelb.edu.au

James Bailey
University of Melbourne
Melbourne, Australia
baileyj@unimelb.edu.au

Kotagiri Ramamohanarao
University of Melbourne
Melbourne, Australia
kotagiri@unimelb.edu.au

Christopher Leckie
University of Melbourne
Melbourne, Australia
caleckie@unimelb.edu.au

Michael E. Houle
National Institute of Informatics
Tokyo, Japan
meh@nii.ac.jp

ABSTRACT

Classifier explanations have been identified as a crucial component of knowledge discovery. Local explanations evaluate the behavior of a classifier in the vicinity of a given instance. A key step in this approach is to generate synthetic neighbors of the given instance. This neighbor generation process is challenging and it has considerable impact on the quality of explanations. To assess quality of generated neighborhoods, we propose a local intrinsic dimensionality (LID) based locality constraint. Based on this, we then propose a new neighborhood generation method. Our method first fits a local embedding/subspace around a given instance using the LID of the test instance as the target dimensionality, then generates neighbors in the local embedding and projects them back to the original space. Experimental results show that our method generates more realistic neighborhoods and consequently better explanations. It can be used in combination with existing local explanation algorithms.

KEYWORDS

local explanations, black-box explanations, local intrinsic dimensionality, interpretability, explainability.

ACM Reference Format:
Yunzhe Jia, James Bailey, Kotagiri Ramamohanarao, Christopher Leckie, and Michael E. Houle. 2019. Improving the Quality of Explanations with Local Embedding Perturbations. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3292500.3330930

1 INTRODUCTION

There has been growing interest in interpretability of data mining algorithms. One important aspect of interpretability is to provide explanations for the predictions of a model. Explanations reveal the logical reasons why a model makes such predictions. There are four scenarios where the explanations are useful: (1) they are used as complementary information to the corresponding predictions and give users greater confidence to accept/reject the predictions; (2) they reveal logical errors and help to improve a model; (3) they provide scientific insights/hypotheses; (4) they may be required by law [11].

For white-box models that are inherently interpretable (e.g., decision trees, logistic regression and linear models), explanations for predictions can be inferred from the parameters or model structures. However, due to the simplicity of these models, they typically are not able to achieve high prediction accuracy. On the other hand, complex models (black-box models) like random forests and neural networks are known for their state-of-the-art accuracy, but are difficult to interpret. Recently, many techniques have been proposed to provide explanations for the predictions of black-box models. In particular, we focus on **local explanations** that are explanations from the instance-level given a (black-box) model. The usefulness of local explanations has been shown in many applications [22].

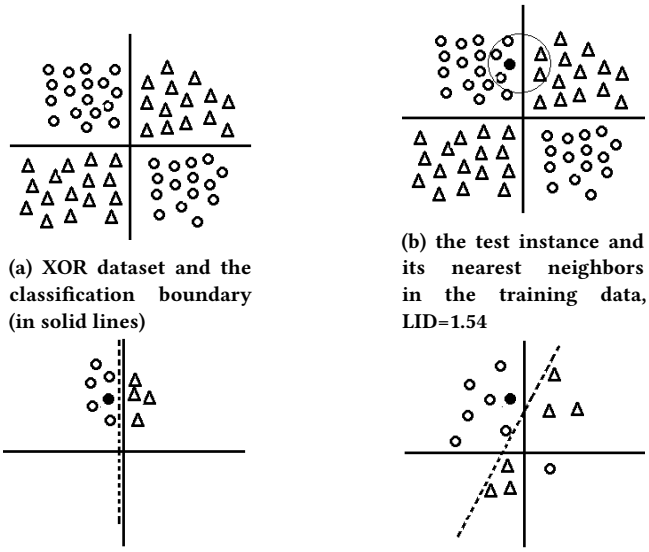
When evaluating an explanation for the prediction of a black-box model, the quality of the explanation refers to the degree to which the explanation reflects how a model truly 'thinks'. If an instance is difficult to explain, an extracted explanation with poor quality is not useful, as it fails to provide insights about the logical reasoning of the model. An important question arises: *why is a particular instance difficult to explain?* We aim to improve the quality of explanations by addressing this question.

A general framework to obtain local explanations from a black-box model is to first generate a set of synthetic neighbors of a given instance, and then extract an explanation from these neighbors after they are labelled by the model. As observed in [17], *the quality of an explanation for a prediction depends heavily on the quality of the generated neighbor instances of the instance.* Despite the importance of the quality of the neighborhood generation, few studies have been conducted to improve the quality of explanations by generating neighbor instances of good quality [8]. Existing local explanation methods generally employ random perturbation and use Euclidean distance to threshold/weight the newly generated instances.

From another perspective, extracting the local explanation from a given test instance is similar to identifying the local classification boundary of a given model. Suppose that the local classification boundary lies within a local subspace around the test instance, we

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330930



(a) XOR dataset and the classification boundary (in solid lines)

(b) the test instance and its nearest neighbors in the training data, LID=1.54

(c) 'good' neighbors with LID=1.73

(d) 'bad' neighbors with LID=3.61

Figure 1: Illustration of a 'good' neighborhood and 'bad' neighborhood (triangles and circles are different labels). Given a test instance (filled circle) and the underlying classification boundary (solid line), generating 'good' neighbors (left) helps to mimic a local model (dashed line) that is similar to the true local model, while a local model (right) from 'bad' neighbors is far away from the true model. LID (local intrinsic dimensionality) is a possible metric for the quality of the neighbors.

can categorize synthetic neighbor instances into 'good' neighbors and 'bad' ones. We could build a local model that is similar to the underlying true model from 'good' synthetic neighbors, whereas a local model trained on the 'bad' synthetic neighbors is far away from the true classification boundary (as shown in Figure 1). One reason that an instance is difficult to explain is that there are too many 'bad' synthetic neighbors and thus explanation extraction methods fail to detect the true local classification boundary. Therefore, a synthetic neighborhood of poor quality can result in an explanation of poor quality.

Our goal in this paper is to generate 'good' synthetic neighbors and avoid 'bad' ones to find a neighborhood of high quality, thus improving the quality of the explanation for a prediction. To facilitate this, we employ a measure of the local intrinsic dimensionality of the data, LID [14], which can be regarded as capturing the number of latent variables required to represent the distribution of data within the vicinity of a reference location. As the LID rises, the number of parameters required in modeling the local distribution rises, which in turn indicates that the complexity of explaining the local model is also rising.

The LID measure has been successfully applied in similarity search and anomaly detection [14, 18] to characterize the complexity of the sub-manifold surrounding a test instance. In this paper, we analyze the average LID of the synthetic neighbors of a test instance from its neighborhood in the training data, to study the difficulty of its explanation. We find that when the average LID of the synthetic neighbors is largely different from the LID of the test instance, the neighborhood is unlikely to be 'good', hence the instance is difficult to explain, (as shown in Figure 1). Based on this observation, we propose a new approach for generating neighbor instances of high quality. The contributions of our paper include:

- We investigate the importance of the generation of an appropriate neighborhood for local explanation methods.
- We analyze the relationship between local intrinsic dimensionality (LID) and the quality of explanations.
- We propose an effective neighbor generation method based on LID, that facilitates local explanations of high quality.

2 RELATED WORK

Global explanations. Many works that focus on explanations aim to provide explanations at the model level [20]. They typically train an interpretable model (e.g., pattern/rule based models) to mimic the behavior of a black-box classifier. Craven and Shavlik [6] train decision trees and Kurfess [16] extracts classification patterns to approximate neural networks, where the training data is relabelled by the (black-box) neural network. Decision trees or classification patterns are then learned from the new training data. Barakat and Diederich [5], Fung et al. [9] and Martens et al. [19] apply the same technique to provide explanations for SVMs. Instead of interpreting the behavior of the whole classifier, SCaPE [7] uses patterns to explain why a classifier is likely to make incorrect predictions on a certain subgroup of the data. GoldenEye++ [13] explains a classifier from the view of interacting attributes, by grouping those attributes that are exploited by the classifier. However, as pointed out in [22], global fidelity does not imply local fidelity — features that are important at the model level are not necessarily important at the instance level.

Local explanations. Recently, many methods have been proposed to provide instance-level local explanations for individual predictions. Robnik-Šikonja and Kononenko [24], Henelius et al. [12] and Adler et al. [1] calculate a weight vector where each entry represents the predictive power of the corresponding attribute. Parzen window techniques [4] can be used to estimate the gradient of a given instance with respect to the prediction probability function, which can then be used as the explanation for its prediction. Wang et al. [27] similarly apply such techniques on images where the gradient is an image mask. LIME [22] and its variant aLIME [21, 23] build a locally interpretable model (a linear model in LIME, and patterns in aLIME) in the newly generated neighborhood of a test instance; the local model is then used as the local explanation. In both LIME and aLIME, random perturbation is used to generate the neighbors; however, Fong and Vedaldi [8] also propose a meaningful perturbation method for use with image datasets.

Locality of synthetic neighbors. Generation of a synthetic neighborhood for a test instance that needs explanation is a necessary step in most local explanation methods. The realism of the synthetic neighborhood is related to the accuracy of the local model/surrogate that is fitted on the synthetic neighborhood, thus determining the quality of the local explanation extracted from the local model. LIME

[22] employs a random perturbation technique that draws the synthetic neighbors from a normal distribution centered on the test instance. A risk here is that random perturbation may hide features with local influence and benefit those with a global influence, and thus the extracted explanations may reflect global influences rather than local. In [8], image data is perturbed through deletion of pixels; however, this technique cannot be directly extended to general tabular data, as deletion of features is problematic in this setting: tabular data models require a fixed number of features, and setting missing feature values (the basic strategy for deleting a feature) can greatly alter the results of distance-based comparisons. In [17], a nearest neighborhood (NN) based perturbation method is proposed that generates synthetic instances within a hypersphere whose radius is determined by the nearest instance of the opposite class label in the training data. Locality is also important in adversarial learning, as adversarial examples are designed to be close to a test instance. A common way to find an adversarial example is the gradient-based method that is introduced in fast Gradient Sign Method (FGSM) [10] and its variations. In FGSM, an adversarial example x' of a given instance x is computed by $x' = x + \epsilon \cdot sign(\Delta g(x))$, where $\Delta g(x)$ is the gradient of the cost function of the model at x, and ϵ is the magnitude of the perturbation. FGSM differs from local explanation methods in that only one neighbor instance is generated, usually along the direction towards a local optimum.

3 PRELIMINARIES

We next introduce some important concepts that will be needed.

Explanation. An *explanation* for the prediction of a D-dimensional instance by a given model is a D-dimensional vector where each entry represents the contribution/importance of the corresponding attribute/feature.

We allow two representations for explanations, namely **numeric explanations** and **binary explanations**.

Given a D-dimensional dataset, a numeric explanation is an explanation in the form of a weight vector as a D-dimensional vector $e = \{e_1, e_2, \ldots, e_D\}$, where $e_i \in \mathbb{R}$.

In a numeric explanation, every entry indicates the contribution of the corresponding feature. The higher e_i is, the higher the contribution of the ith feature.

Given a D-dimensional dataset, a binary explanation is an explanation in the form of a bit vector as a D-dimensional vector $e = \{e_1, e_2, \ldots, e_D\}$, where $e_i \in \{0, 1\}$.

In a binary explanation, the values of the entries simply indicate whether the corresponding feature is relevant or not. For example, for a prediction made by a decision tree, only the features that occur on the prediction path are relevant.

Local explanation. A *local* explanation is an instance-level explanation that explains the behavior of a classifier in the vicinity of a given instance, and it directly discriminates the given instance from its neighbors. It differs from a *global* explanation, which explains the behavior of the classifier for all instances. A globally important feature is not necessarily also locally important; for example, a decision tree may be constructed using all features, but a prediction (decision tree branch) may use only a subset of the features. All features occurring within the tree may be important in a global sense, but only those features that occur on the prediction path are locally important for the prediction of an instance.

Local intrinsic dimensionality (LID). Let a function F be positive and continuously differentiable over some open interval containing $r > 0$. The *LID* of F at r is defined as:

$$\mathrm{LID}_F(r) := r\frac{F'(r)}{F(r)} = \lim_{\epsilon \to 0^+} \frac{F((1+\epsilon)r) - F(r)}{\epsilon\, F(r)}, \tag{1}$$

whenever the limit exists. The (asymptotic) LID of F is then:

$$\mathrm{LID}_F^* = \lim_{r \to 0^+} \mathrm{LID}_F(r). \tag{2}$$

For our scenario, LID is computed with respect to the location of a given instance q, and r represents a radial distance from q. The LID model characterizes the order of magnitude of the growth of cumulative probability measure F with respect to a neighborhood of increasing radius. As it assesses the intrinsic dimensionality of the local data submanifold around the instance q, it serves to characterize the complexity (number of parameters) required by the local data model.

Several estimators exist for calculating the local intrinsic dimensionality [2, 3]. We use the maximum likelihood estimation (MLE) in our experiments as it is known to have good convergence properties and is easy to implement. Given an instance x, and its k nearest neighbors $Nei(x)$ sorted by the distance to x in ascending order, the MLE estimator of $\mathrm{LID}(x, Nei(x))$ is:

$$\mathrm{LID}(x, Nei(x)) = -\Big(\frac{1}{k-1}\sum_{j=1}^{k-1} \log \frac{r_j(x)}{r_k(x)}\Big)^{-1} \tag{3}$$

where $r_j(x)$ is the distance (the Euclidean distance is used in our paper) of the jth neighbor to x.

Note that no explicit knowledge of the underlying function F is needed by the MLE estimator - this information is implicit in the distribution of neighbor distances themselves.

Information dimension: Given a set of instances $\{x_1, \ldots, x_N\}$, we may also compute the quantity $\frac{1}{N}\sum_i \mathrm{LID}(x_i, Nei(x_i))$. This characterizes the overall intrinsic dimensionality of the manifold occupied by $\{x_1, \ldots, x_n\}$. In [25], it was shown that this type of average is in fact an estimator of the well known *information dimension* over the sample domain (or manifold).

4 PROPOSED METHOD

4.1 General framework of local explanations

Given a model/predictor f and an instance x, a typical framework for extracting local explanations for $f(x)$ consists of three steps: (1) generating neighbor instances of x; (2) labelling the neighbors using f; (3) extracting local explanations from the generated neighbors.

Existing methods mainly differ at the third step - how to extract local explanations. In contrast, this paper investigates the impact of the generation of the neighbors in the first step.

4.2 Local perturbation operators

A perturbation operator is used to generate neighbors and is a map $P : \mathbb{R}^D \mapsto \mathbb{R}^D$, given a D-dimensional instance x. It is formally defined as:

$$x' = P(x, \eta) = x + \eta \tag{4}$$

where η is the perturbation term or the change that is applied to the given instance. Next, we will describe some common perturbation operators that differ in how η is drawn.

A simple but useful operator is the random Gaussian perturbation that is used in LIME [22]:

$$P(x, \eta) = x + \eta,\ \eta \sim \mathcal{N}(0, 1) \tag{5}$$

where $\mathcal{N}$ denotes the normal distribution from which η is drawn. If the data is not normalized to the range of $[0, 1]$, η is usually multiplied by a scale term representing the feature range. One major issue with this operator is that it is likely to generate 'bad' neighbors located off the sub-manifold of x.

Another operator is a gradient-based sampling technique that has been widely used in the field of adversarial learning, whereby nearby instances with the opposite class label to a given instance are generated. The strategy used in FGSM [10] is:

$$P(x, \eta) = x + \eta,\ \eta = \epsilon \cdot sign(\Delta g(x)) \tag{6}$$

where $\Delta g(x)$ is the gradient of the cost function at the test instance x and ϵ is the magnitude of perturbation.

Recently, a nearest neighbor (NN) based method was proposed in [17]. It first finds the nearest instance x_c of the given test instance x in the training data such that $f(x) \neq f(x_c)$, where f is the prediction model. The perturbation is then

$$\begin{aligned} &P(x, \eta) = x + \eta,\ \eta \sim Unif(range(x)) \\ &\text{s.t. } dist(x, P(x, \eta)) \leq dist(x, x_c) \end{aligned} \tag{7}$$

where $dist(x, z)$ measures the distance of two instances x and z, $range(x)$ denotes the possible value ranges of the features, and $Unif$ denotes the uniform distribution.

4.3 Learning local explanations

Given a black-box model f and an instance x, the target of learning the local explanation is to find a local model g from a hypothesis set $\mathcal{G}$ in the neighborhood of x. The process can be formulated as:

$$\begin{aligned} \underset{g \in \mathcal{G}, x_i = P(x, \eta_i)}{\text{minimize}} \quad & \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(g(x_i), f(x_i)) + \alpha \mathcal{R}(g) \\ \text{subject to} \quad & Locality\_Constraint(x_i),\ i = 1, \ldots, m. \end{aligned} \tag{8}$$

where $\mathcal{L}$ is the loss function, $\mathcal{R}(g)$ is an interpretability regularization term to ensure that the chosen g is interpretable, α is the regularization rate, $P(x, \eta)$ is the local perturbation operator that generates the synthetic neighborhood of x, and $Locality\_Constraint(x_i)$ is a constraint on the generated neighborhood to ensure locality fidelity of the local explanations.

The main challenges for solving this problem are: (1) interpretability constraints are subjective and difficult to formulate; (2) the underlying local classification boundary is unknown and thus it is difficult to formally define the locality constraints.

In existing methods [8, 22, 23], the first challenge has been addressed by restricting $\mathcal{G}$ to a set of interpretable models $\mathcal{G}^*$ (such as decision trees, rules or linear models). We therefore focus on the second issue, and approach it from the perspective of local instrinsic dimensionality, by defining an LID based locality constraint.

LID can be used to characterize the complexity of the neighborhood (a set of instances) surrounding a test instance (single instance). Given a model f and a test instance x, assume there exists a true neighborhood $Nei^*(x)$ of x, such that we can train a local model g that is a perfect local surrogate for f, without incurring an increase in local intrinsic dimensionality. When using local perturbation operators to generate a synthetic neighborhood $Nei_s(x)$, in order to extract a faithful explanation from $Nei_s(x)$, we want the LID of $Nei_s(x)$ to be as close as possible to that of $Nei^*(x)$. Here, we use the average LID of the neighborhood (an estimator of information dimension [25], or correlation dimension [15]) as an estimator for the divergence (dissimilarity) of the explanability of $Nei_s(x)$ and $Nei^*(x)$:

$$\begin{aligned} divg(Nei_s(x), Nei^*(x)) = &\Big| \frac{1}{|Nei_s(x)|} \sum_{x_i \in Nei_s(x)} \text{LID}(x_i, Nei^*(x_i)) \\ &- \frac{1}{|Nei^*(x)|} \sum_{x_j \in Nei^*(x)} \text{LID}(x_j, Nei^*(x_j)) \Big| \end{aligned} \tag{9}$$

As $Nei^*(x)$ is the true neighborhood of x, we assume that any instance in $Nei^*(x)$ should have a similar LID to x (this natural assumption has been recently used in the development of tight estimators of LID [3]). Under this assumption, we can approximate $\frac{1}{|Nei^*(x)|} \sum_{x_j \in Nei^*(x)} \text{LID}(x_j, Nei^*(x_j))$ by $\text{LID}(x, Nei^*(x))$. The divergence becomes:

$$\begin{aligned} divg(Nei_s(x), Nei^*(x)) \approx &\Big| \frac{1}{|Nei_s(x)|} \sum_{x_i \in Nei_s(x)} \text{LID}(x_i, Nei^*(x_i)) \\ &- \text{LID}(x, Nei^*(x)) \Big| \end{aligned} \tag{10}$$

As $Nei^*(x)$ is unknown, we approximate it using the k nearest neighbors of x in the training data, which is denoted as $Nei_k(x)$. Then the empirical divergence is:

$$\begin{aligned} \overline{divg}(Nei_s(x), Nei^*(x)) \approx &\Big| \frac{1}{|Nei_s(x)|} \sum_{x_i \in Nei_s(x)} \text{LID}(x_i, Nei_k(x_i)) \\ &- \text{LID}(x, Nei_k(x)) \Big| \end{aligned} \tag{11}$$

In the ideal case $divg(Nei_s(x), Nei^*(x))$ is close to 0 and we have:

$$\frac{1}{|Nei_s(x)|} \sum_{x_i \in Nei_s(x)} \text{LID}(x_i, Nei_k(x_i)) \approx \text{LID}(x, Nei_k(x)) \tag{12}$$

Now, we can formally refine the optimization problem as:

$$\begin{aligned} \underset{g,\ x_i = P(x|\eta_n, f)}{\text{minimize}} \quad & \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(g(x_i), f(x_i)) \\ \text{subject to} \quad & \left| \frac{1}{m} \sum_{i=1}^{m} \text{LID}(x_i, Nei_k(x_i)) - \text{LID}(x, Nei_k(x)) \right| \leq \epsilon \end{aligned} \tag{13}$$

where g is chosen from the interpretable model set $\mathcal{G}^*$, the term $\text{LID}(x_i, Nei_k(x_i))$ measures the LID of a generated instance x_i with respect to its k nearest neighbors in the training data, and ϵ is a sufficiently small value. Intuitively, the LID constraint checks whether the LID of the test instance x is sufficiently close to the information dimension of its generated neighborhood. For a 'good' neighborhood, we expect these terms to be very close, as demonstrated in Figure 1.

There isn't, however, an available closed form solution for the above optimization problem. Thus we propose a heuristic method to find an approximate solution by learning a local subspace of appropriate local intrinsic dimensionality around x, and applying random perturbations within this local subspace.

4.4 LEAP - Local embedding aided perturbation

Random perturbations can introduce noise that push an instance far from its underlying local sub-manifold, thereby effectively increasing the number of parameters required for an adequate local model. The LID distribution of instances in the synthetic neighborhood may then be very different to the LID of the test instance. Our proposed method mitigates the effect of noisy perturbations that do not conform to the local manifold, by first approximating a local embedding/subspace around the test instance and then generating the neighbors within the local embedding. So as not to increase the complexity of the local data model, the LID of the test instance is used as the target dimensionality for the embedding. Since we uniformly generate synthetic neighbors within the local embedding, the newly generated instances can be expected to preserve the LID of the test instance. Our proposed method (LEAP - Local embedding aided perturbation) is shown in Algorithm 1. The details are described as follows:

Inputs: x is the test instance whose prediction needs explanation. M is the required number of instances to be generated in the vicinity of x. T is the training data on which the given model is trained. k is the number of nearest neighbors of x used for LID estimation.
Output: Z is the set of M generated instances.
Process:

Step 1 The initial step is to fit a local embedding in the vicinity of x. We first select a set of neighbor instances of x from the training data using k-nearest-neighbor search. Then a local embedding is extracted from these instances using a subspace learning technique (we use PCA in the experiments, but kernel PCA or LDA are alternative possibilities) using $\text{LID}(x, Nei_k(x))$ as the target dimensionality for the embedding.

Step 2 After the local embedding has been found, the given instance x is then projected to $\hat{x}$ on the local embedding.

Step 3 The last step is to generate the required number of instances to form the local neighborhood of x. At each generation step, the random perturbation operator is applied to $\hat{x}$ to get $\hat{z}$ and then the newly generated instance is mapped back into the original feature space, and its weight is (optionally) set according to the distance between $\hat{z}$ and $\hat{x}$ as $\exp(-dist(\hat{z}, \hat{x}))$. These weights can be optionally used by explanation extraction methods. The pseudo code can be found in Appendix A.

4.5 Local explanation extraction with LEAP

For any local explanation extraction method (e.g., LIME) that needs to generate local neighbors, LEAP can be directly used to replace the generation process. We take the given model f and the parameters that are required by LEAP as inputs, and return an explanation for the prediction of the given x by f. First, the neighbor instances of x are generated using LEAP. Then, all new instances are labeled by the given model f. Finally, the local explanation is generated using a technique like LIME (e.g. with a linear model or a decision tree).

5 EXPERIMENTS

In this section, we define the metrics that will measure the quality of the explanations, and experimentally evaluate the performance of our proposed method. The experimental settings are described, and then we demonstrate how the true explanations, which are used as ground truth to evaluate the generated explanations, are found. Comparisons of the performance of the proposed method and that of the baselines on several datasets are reported. Finally, a case study is explained.

5.1 Explanation metrics

Given a model f and a D-dimensional instance x, assume $e' = \{e'_1, \ldots, e'_D\}$ is the true model explanation, and $e = \{e_1, \ldots, e_D\}$ is the explanation generated by an explanation system (e.g., LIME). The quality of e is measured by its closeness to e'. Given two representations of explanations, we define the following metrics.

When e, e' are numeric explanations, where $e_i, e'_i \in \mathbb{R}$, the quality of e, denoted by $Q(e|e')$, is measured using the cosine similarity:

$$Q(e|e') = |cosine\_similarity(e|e')| = \left| \frac{e \cdot e'}{|e|\,|e'|} \right| \tag{14}$$

where $e \cdot e'$ is the dot product of two explanations, and $|e|$ calculates the l^2-norm of e.

The closer $Q(e|e')$ is to 1, the higher the quality of e, and if it is 1 then e and e' are parallel.

When e, e' are binary explanations, where $e_i, e'_i \in \{0, 1\}$, $Q(e|e')$ is measured by the F_1 score:

$$Q(e|e') = \frac{2 * recall(e|e') * precision(e|e')}{recall(e|e') + precision(e|e')} \tag{15}$$

where $recall(e|e') = \frac{\sum_{i=1}^{D} e_i * e'_i}{\sum_{i=1}^{D} e'_i}$, which indicates how many chosen features (with non-zero explanation values in e) are truly important (with non-zero explanation values in e'), and $precision(e|e') = \frac{\sum_{i=1}^{D} e_i * e'_i}{\sum_{i=1}^{D} e_i}$, which indicates how many truly important features are chosen in e. $Q(e|e')$ equals 1 if and only if e and e' are identical, and is less than one otherwise.

5.2 Settings

We conduct our experiments on six synthetic datasets and nine UCI datasets. The proposed neighborhood generation method (LEAP) is plugged into LIME, and compared with three baselines for the numeric explanation case and binary explanation case. The local neighbor generating methods we have compared are described as follows:

LEAP(N) - Numeric explanation method with the proposed local embedding perturbations and local linear models.

Random(N) - Numeric explanation method with random perturbations (Eq. (5)) and linear models, which is the default method used in LIME.

NN(N) - Numeric explanation method with nearest neighbor perturbations (Eq. (7)) and local linear models.

Gradient(N) - Numeric explanation method with gradient-based perturbations (Eq. (6)) and local linear models.

LEAP(B) - Binary explanation method with the proposed local embedding perturbations and decision trees.

Random(B) - Binary explanation method with random perturbations (Eq. (5)) and decision trees.

NN(B) - Binary explanation method with nearest neighbor perturbations (Eq. (7)) and decision trees.

Gradient(B) - Binary explanation method with gradient-based perturbations (Eq. (6)) and decision trees. Details about the parameter setting can be found in Appendix B.

5.3 Synthetic data and true explanations

The baselines are first compared on the synthetic datasets. We generate six synthetic datasets using four synthetic functions. The explanations are directly derived from the synthetic functions such that the explanation for an instance is the gradient at its closest point on the classification boundary. Details about the generation of the synthetic datasets can be found at Appendix C.

The results (Table 1) show that the local explanation method with LEAP achieves the highest performance across these datasets.

5.4 UCI data and true explanations

We also conduct experiments on nine UCI datasets. For the purpose of evaluating the numeric explanations, all nominal attributes are converted to continuous and all attributes are normalized into the range [0,1]. Two interpretable classifiers (from which we can get the true model local explanations) are used as the black-box for local explanation extraction methods. We use the logistic regression and the decision tree as used in [22]. For the logistic regression, the true explanations are numeric explanations and are obtained directly from the model parameters (the coefficients). For a D-dimensional instance x, $e_{true}(x) = (\omega_1, \omega_2, \ldots, \omega_D)$, which is the vector of the coefficients and ω_i is the coefficient for the ith attribute. For the decision tree, the true explanations are binary explanations and are obtained from the structure of the tree. For a D-dimensional instance x, $e_{true}(x) = (e_1, e_2, \ldots, e_D)$, e_i is 1 if the ith attribute occurs in the prediction path from the root to the leaf, and 0 otherwise.

For each dataset, an interpretable model is trained from the training data, and the true explanations for the test data are obtained from the model, though the explanation systems treat the interpretable model as a black box.

Then the local explanation method (LIME framework) with different neighborhood generation methods is used to extract explanations for the instances in the test datasets, while treating these interpretable classifiers as black-box classifiers. Specifically, LIME based on local linear models is used to evaluate the numeric explanations for logistic regression, and LIME based on local decision trees is used to evaluate binary explanations.

The results in Table 2 show that LEAP has excellent performance in comparison to the three baselines.

5.5 Visualization of the quality of explanations for different methods

We conduct an experiment on a 2D synthetic dataset (we use 2D dataset for visulization purposes), generated using the function $f(x[1], x[2]) = sign(x[1] - x[2]\sin(x[2])^2)$, where the true explanation for a point is the gradient (represented as a weight vector) of the closest point on the classification boundary ($x[1] = x[2]\sin(x[2])^2$) to the test point being explained. The quality of explanations for different neighborhood generating methods is shown in Figure 2. For Random(N) (used in LIME), an instance that is close to the classification boundary is likely to be difficult to explain. This observation also suggests that if an instance is far away from the classification boundary, finding an explanation may be more obvious. LEAP and NN are particularly effective for instances close to the classification boundary, with LEAP having an edge over NN in average explanation quality. For the gradient-based method, the quality of the explanations is similar to Random, and worst when the instance is far away from the boundary.

In terms of the LIDs, while the mean of the LID of the test instance in its vicinity is 1.37, the average LIDs of the synthetic neighborhoods are: 1.4 for LEAP, 18.4 for Random, 2.4 for NN and 6.9 for Gradient. It shows that LEAP achieves the closest LID as the test instance.

We also demonstrate some synthetic neighborhoods that are generated by different methods in Figure 3.

5.6 Locality analysis

We compare the locality for different baselines. The same data and ground truth explanations as in Section 5.5 are used, and we adopted the locality measurement that was proposed in [17]. Given a test instance and the local model that is used to extract the local explanations, a set of instances is drawn from a normal distribution centered on the test instance with variance σ. σ is treated as a radius, and the average accuracy for a local model applied to the set of test instances is defined as the locality at radius σ. Results for the locality of the baselines are shown in Figure 4. It can be seen that LEAP achieves the highest locality for smaller radius values, which is expected. On the other hand, the locality of Random(N) is nearly uniform for all radii, illustrating that LIME+Random(N) is impacted by global fidelity (a finding which accords with that of [17]). It is desirable for the locality to drop with increasing radius, because our target is to achieve a 'local' explanation model. For NN, a similar decreasing trend is evident, but the locality (accuracy of the local models) is lower than LEAP.

5.7 Example: Why does the quality of explanations matter?

We have proposed LEAP to improve the quality of the local explanations, and we next provide an example of how an explanation having higher quality can be beneficial for taking an action. Our study is performed in the context of spam classification, on a dataset containing 1956 comments from 5 different YouTube videos [26]. We split it into two groups with 70% for training and 30% for testing. The class label is whether or not the comment is spam.

We train a Random Forest (with 100 trees) and extract explanations for test comments using different synthetic neighborhood generation methods. Each explanation e for a test instance x is evaluated in the following way: based on the explanation e, x is modified so that the occurrence of the word that most supports the opposite class label is increased 10 times. If the class label of the modified instance is flipped (when the modified instance is passed to the random forest), we say the explanation e is useful, since it facilitated a perturbation to be made on the instance that had an intuitive effect. Conversely, if the class label is not flipped, then the explanation is not useful, since the perturbation didn't operate as expected.

Dataset	Random(N)	Gradient(N)	NN(N)	LEAP(N)	Random(B)	Gradient(B)	NN(B)	LEAP(B)
Synthetic-1	0.88	0.65	0.82	**0.97**	-	-	-	
Synthetic-2	0.82	0.71	0.78	**0.90**	-	-	-	
Synthetic-3	0.73	0.42	0.51	**0.83**	-	-	-	
Synthetic-4	0.65	0.61	0.75	**0.90**	-	-	-	
Synthetic-5	-	-	-	-	0.87	0.78	0.76	**0.92**
Synthetic-6	-	-	-	-	0.81	0.62	0.74	**0.88**

Table 1: Explanation quality on synthetic datasets (Best value in bold; '-' means method does not apply on corresponding dataset, e.g., Synthetic-1 is for numeric explanation and binary explanation method does not apply).

Dataset	Random(N)	Gradient(N)	NN(N)	LEAP(N)	Random(B)	Gradient(B)	NN(B)	LEAP(B)
	logistic regression (numeric explanations)				decision trees (binary explanations)			
Adult	0.84	0.63	0.72	**0.90**	0.64	0.61	0.57	**0.73**
Balloon	0.93	0.82	0.54	**1.00**	0.92	0.81	0.50	**0.94**
Blood	0.89	0.85	0.61	**0.96**	0.94	0.58	0.92	**0.97**
Breast-cancer	0.82	0.78	0.59	**0.87**	0.65	0.57	0.50	**0.70**
Diabetes	0.87	0.74	0.62	**0.99**	**0.80**	0.47	0.71	**0.80**
Hepatitis	0.72	0.68	0.56	**0.80**	0.82	0.38	0.72	**0.87**
Iris	0.91	0.82	0.64	**0.94**	**1.00**	**1.00**	**1.00**	**1.00**
Ionosphere	0.91	0.82	0.74	**0.92**	0.92	0.66	0.89	**1.00**
Labor	0.86	0.75	0.64	**0.89**	0.88	0.64	0.77	**0.95**
Titanic	0.85	0.83	0.59	**0.87**	0.74	0.73	0.66	**0.79**
Vote	0.90	0.77	0.62	**0.93**	0.82	0.66	0.50	**0.89**

Table 2: Explanation quality on UCI datasets (the best value is highlighted in bold).

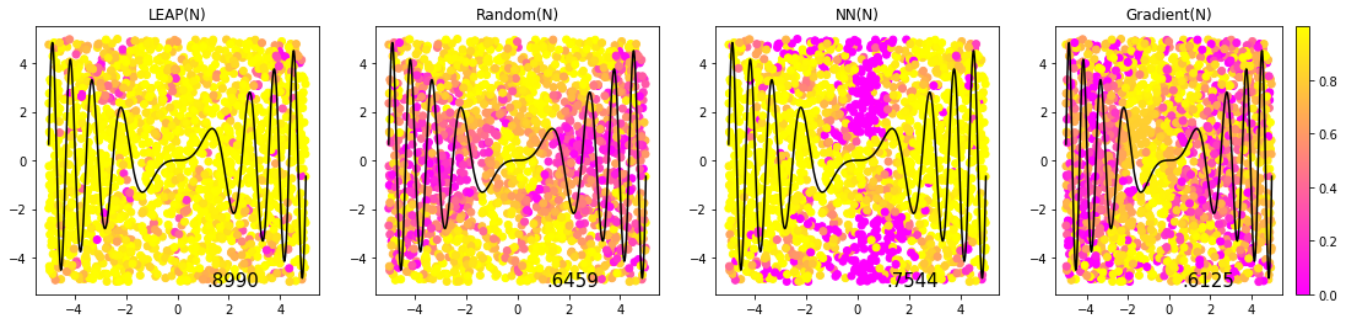


Figure 2: Demonstration of difficulty of explanation for different neighborhood generating methods. The black solid line is the classification boundary, and difficulty (explanation quality) varies from difficult (red) to easy (yellow). The average explanation quality is shown in the bottom-right of each figure.

For this task, Random(N) produces 87.3% useful explanations while LEAP produces 95.4% useful explanations. An example of where Random(N) fails to find a useful explanation but LEAP succeeds, is shown in Table 3. Random(N) finds the word 'the' is the most supportive feature for class label 'Not spam' but the corresponding modification is still classified as 'Spam'. LEAP finds the word 'projects' and the corresponding modification fools the classifier successfully. Although humans may regard both modifications as spam, the explanation of LEAP reveals how the classifier 'thinks'.

6 CONCLUSION

We proposed a new approach for neighborhood generation, which is a critical phase in local explanation systems. Our approach is based on consideration of the local intrinsic dimensionality of the instance whose predicted label is being explained. Our method can be used as a modular procedure in any existing local explanation method that requires neighborhood generation. We compared our proposed technique with the baselines that are used in local explanation methods, and demonstrated it achieves excellent performance when compared with state of the art alternatives.

7 ACKNOWLEDGMENTS

M. E. Houle supported by JSPS Kakenhi Kiban (B) Research Grant 18H03296.

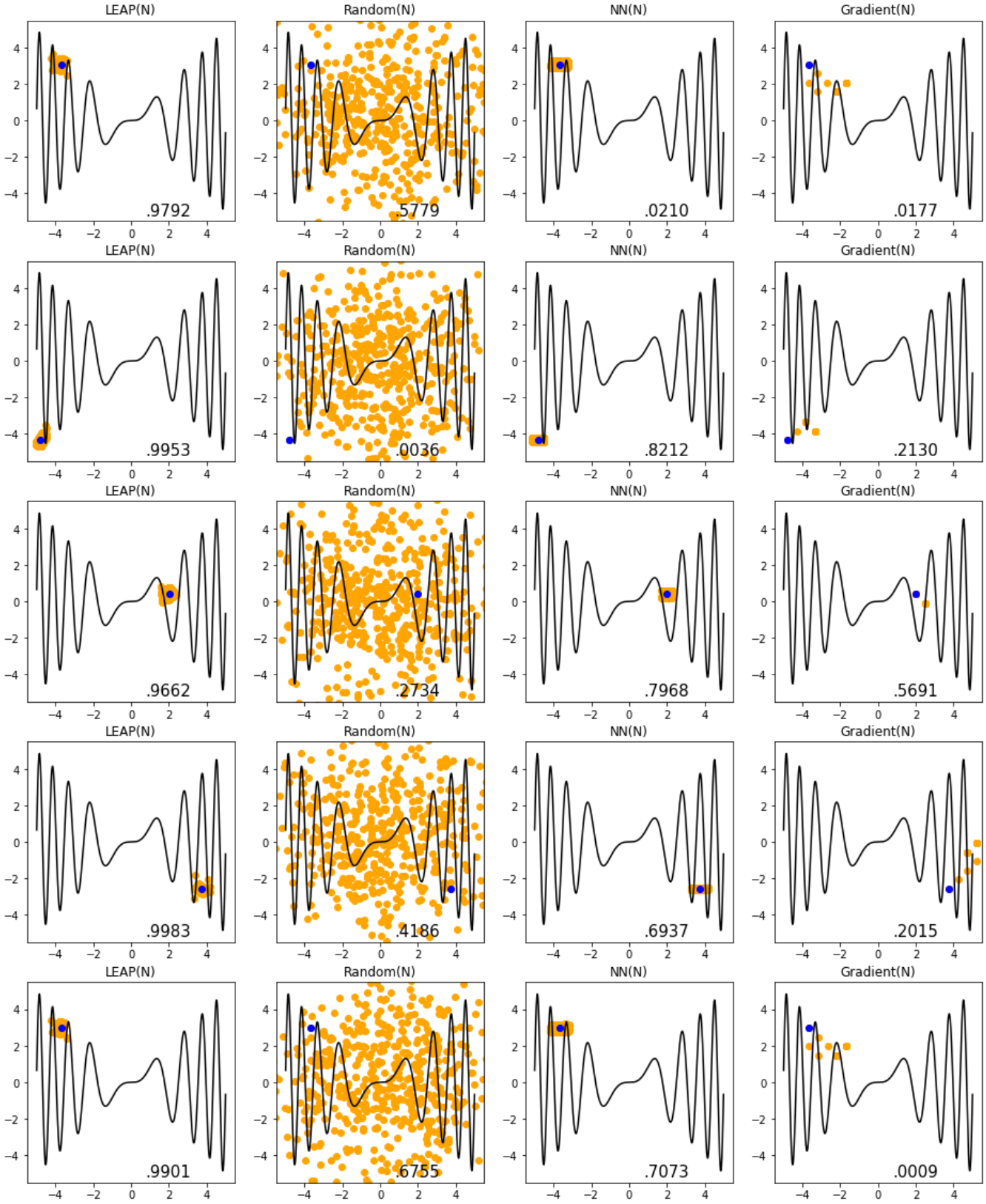


Figure 3: Demonstration of the synthetic neighborhoods that are generated by different methods. The test instances are shown in blue color, and the synthetic neighborhoods are shown in orange color. The quality of explanations extracted from the synthetic neighbors is shown in the bottom-right.

Method	Explanations	Modified Comment using explanation	Classification
Random(N):	Spam: {http, and, ref, you}; Not spam: {The, here, Effects}	The The The The The The The The The The projects After Effects, Music, Foto, Web sites and another you can find and buy here http...	Spam (label of Modified comment not flipped as expected)
LEAP:	Spam: {http, ref, Music, buy}; Not spam: {projects, Effects, find}	projects projects projects projects projects projects projects projects projects projects The projects After Effects, Music, Foto, Web sites and another you can find and buy here http...	Not spam (label of Modified comment is flipped as expected)

Table 3: Explanations extracted with Random(N) and LEAP for the comment *"The projects After Effects, Music, Foto, Web sites and another you can find and buy here http..."* which is classified as Spam

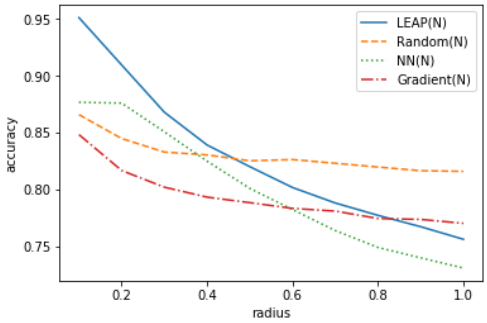


Figure 4: Locality analysis (plot of the accuracy of the local models for when increasing the radius

REFERENCES

[1] Philip Adler, Casey Falk, Sorelle A Friedler, Gabriel Rybeck, Carlos Scheidegger, Brandon Smith, and Suresh Venkatasubramanian. 2016. Auditing black-box models for indirect influence. In *Proc. 16th IEEE International Conference on Data Mining (ICDM)*. IEEE, 1–10.

[2] Laurent Amsaleg, Oussama Chelly, Teddy Furon, Stéphane Girard, Michael E. Houle, Ken-ichi Kawarabayashi, and Michael Nett. 2018. Extreme-value-theoretic estimation of local intrinsic dimensionality. *Data Mining and Knowledge Discovery* 32, 6 (2018), 1768–1805.

[3] Laurent Amsaleg, Oussama Chelly, Michael E. Houle, Ken-ichi Kawarabayashi, Miloš Radovanović, and Weeris Treeratanajaru. 2019. Intrinsic Dimensionality Estimation within Tight Localities. In *Proc. 19th SIAM International Conference on Data Mining (SDM)*. SIAM, 181–189.

[4] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. 2010. How to explain individual classification decisions. *Journal of Machine Learning Research* 11, Jun (2010), 1803–1831.

[5] Nahla Barakat and Joachim Diederich. 2005. Eclectic rule-extraction from support vector machines. *Int. Journal of Comp. Intelligence* 2, 1 (2005), 59–62.

[6] Mark W. Craven and Jude W. Shavlik. 1996. Extracting tree-structured representations of trained networks. *Advances in Neural Information Processing Systems (NIPS)* (1996), 24–30.

[7] Wouter Duivesteijn and Julia Thaele. 2014. Understanding where your classifier does (not) work–the SCaPE model class for EMM. In *Proc. 14th IEEE International Conference on Data Mining (ICDM)*. IEEE, 809–814.

[8] Ruth C Fong and Andrea Vedaldi. 2017. Interpretable Explanations of Black Boxes by Meaningful Perturbation. In *Proc. International Conference on Computer Vision (ICCV)*. IEEE, 3449–3457.

[9] Glenn Fung, Sathyakama Sandilya, and R. Bharat Rao. 2005. Rule extraction from linear support vector machines. In *Proc. 11th International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 32–40.

[10] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and Harnessing Adversarial Examples. *CoRR* (2014).

[11] Bryce Goodman and Seth Flaxman. 2016. EU regulations on algorithmic decision-making and a "right to explanation". In *ICML Workshop on Human Interpretability in Machine Learning (WHI)*.

[12] Andreas Henelius, Kai Puolamäki, Henrik Boström, Lars Asker, and Panagiotis Papapetrou. 2014. A peek into the black box: exploring classifiers by randomization. *Data Mining and Knowledge Discovery* 28, 5-6 (2014), 1503–1529.

[13] Andreas Henelius, Kai Puolamäki, Isak Karlsson, Jing Zhao, Lars Asker, Henrik Boström, and Panagiotis Papapetrou. 2015. Goldeneye++: A closer look into the black box. In *Int. Sym. on Stat. Learning and Data Sciences*. Springer, 96–105.

[14] Michael E. Houle. 2017. Local intrinsic dimensionality I: an extreme-value-theoretic foundation for similarity applications. In *Proc. 10th International Conference on Similarity Search and Applications (SISAP)*. Springer, 64–79.

[15] Michael E. Houle. 2017. Local intrinsic dimensionality II: multivariate analysis and distributional support. In *Proc. 10th International Conference on Similarity Search and Applications (SISAP)*. Springer, 80–95.

[16] Franz J. Kurfess. 2000. Neural Networks and Structured Knowledge: Rule Extraction and Applications. *Applied Intelligence* 12, 1 (2000), 7–13.

[17] Thibault Laugel, Xavier Renard, Marie-Jeanne Lesot, Christophe Marsala, and Marcin Detyniecki. 2018. Defining Locality for Surrogates in Post-hoc Interpretablity. *arXiv preprint arXiv:1806.07498* (2018).

[18] Xingjun Ma, Bo Li, Yisen Wang, Sarah M. Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Michael E. Houle, Dawn Song, and James Bailey. 2018. Characterizing Adversarial Subspaces Using Local Intrinsic Dimensionality. In *Proc. 6th International Conference on Learning Representations (ICLR)*.

[19] David Martens, Bart Baesens, Tony Van Gestel, and Jan Vanthienen. 2007. Comprehensible credit scoring models using rule extraction from support vector machines. *European Journal of Operational Research* 183, 3 (2007), 1466–1476.

[20] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Model-Agnostic Interpretability of Machine Learning. In *ICML Workshop on Human Interpretability in Machine Learning (WHI 2016)*.

[21] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Nothing Else Matters: Model-Agnostic Explanations By Identifying Prediction Invariance. In *NIPS Workshop on Interpretable Machine Learning in Complex Systems*.

[22] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why Should I Trust You?: Explaining the Predictions of Any Classifier. In *Proc. 22nd International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 1135–1144.

[23] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-precision model-agnostic explanations. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[24] Marko Robnik-Šikonja and Igor Kononenko. 2008. Explaining classifications for individual instances. *IEEE Transactions on Knowledge and Data Engineering* 20, 5 (2008), 589–600.

[25] Simone Romano, Oussama Chelly, Xuan Vinh Nguyen, James Bailey, and Michael E. Houle. 2016. Measuring Dependency via Intrinsic Dimensionality. In *Proc. 23rd International Conference on Pattern Recognition (ICPR)*. 1207–1212.

[26] Alper Kürşat Uysal. 2018. Feature Selection for Comment Spam Filtering on YouTube. *Data Science and Applications* 1, 1 (2018), 4–8.

[27] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. 2016. Dueling Network Architectures for Deep Reinforcement Learning. In *Proc. 33rd International Conference on Machine Learning (ICML)*. JMLR.org, 1995–2003.

APPENDIX A (CODE REPRODUCIBILITY)

In this part, we describe the pseudo code for LEAP in Algorithm 1.

Algorithm 1: LocalEmbeddingPerturbation

Input: Instance x, number of generated neighbors M, training data T, KNN threshold k
Output: $Z = \{z_1, z_2, \ldots, z_M\}$

```
nei = getkNearestNeighbors(x, T, k);
lid = LocalIntrinsiDim(x, nei) ;
le = new LocalEmbedding(nei, lid) ;
x̂ = le.project(x);
Z = {};
while |Z| < M do
    uniformly generate an instance ẑ within the local embedding;
    z = le.projectToOriginal(ẑ);
    z.setWeight(exp(−dist(ẑ, x̂)));
    Z = Z ∪ {z};
end
return Z;
```

APPENDIX B (PARAMETER REPRODUCIBILITY)

We describe the parameter settings used. Parameter optimization via a nested validation process during the training process is a common technique for classification tasks; however, it presents difficulties for explanation systems, due to the fact that true model explanations are always unavailable when deploying such systems in real-life applications. One possible solution is to generate parameter settings using validation datasets whose true model explanations are available, and then to apply the same setting for any new dataset. To simulate such a process, we apply LEAP with different parameter settings for two major parameters K and M on four separate UCI datasets (*crx*, *ILPD*, *sick* and *sonar*, which are different from those used for evaluation) as validation datasets, then the recommended parameter setting ($K = 5D$ where D is the dimensionality of data, and M=500) (that achieves the best score on average over the validation datasets) is used for all the datasets whose results are reported.

APPENDIX C (DATASET REPRODUCIBILITY)

In this part, we give details about how the synthetic datasets and the true explanations are generated. The synthetic datasets are generated using synthetic functions described below as the classifiers, and true explanations are generated from these functions:

- $f_1(x) = \begin{cases} x[1] - 4x[2] + 2x[3] + 3, & x[1] \leq 10 \\ -2x[1] - 3x[2] + x[2] - 2, & 10 < x[1] \leq 20 \\ 3x[1] + x[2] - 2x[3] + 2, & x[1] > 20 \end{cases}$
  where $x[i] \in [0, 30]$.

- $f_2(x) = (x[1])^3 - 2(x[2])^2 + 3x[3]$, where $x[i] \in [-100, 100]$.

- $f_3(x) = x[1] - x[2]\sin(x[2])^2$ where $x[i] \in [-10, 10]$.

- $f_4(x) = ((x[1] == 1) \wedge x[2] \wedge x[3]) \vee ((x[1] == 2) \wedge x[4] \wedge x[5]) \vee ((x[1] == 3) \wedge x[6] \wedge x[7]) \vee ((x[1] == 4) \wedge x[8] \wedge x[9])$, where $x[1] \in [1, 4]$ and $x[2], \ldots, x[9]$ are binary attributes.

For each synthetic dataset, the attribute values are drawn from a uniform distribution, and for $i = 1, 2, 3$, the class label is 1 if $f_i(x) > 0$ else 0 otherwise. For $i = 4$, the class label is 1 if $f_4(x) = true$ else 0 otherwise. Six synthetic datasets (two for numeric explanation scenarios and the other two for binary explanation scenarios) are generated as follows: for the numeric functions $f_i (i = 1, 2, 3)$, the true explanation of a given x is the gradient of the curve $f_i = 0$ at the closest point on the curve from x. This is because we expect the true explanation to be like a locally linear model. For the binary explanation function f_4, the true explanation is defined below.

- **Synthetic-1** consists of three attributes ($x[1], x[2], x[3]$) and a class label. The function f_1 is used to label the instances. Given an instance x, the true local (numeric) explanation is:
$$e_{true}(x) = \begin{cases} (1, -4, 2), & x[1] \leq 10 \\ (-2, -3, 1), & 10 < x[1] \leq 20 \\ (3, 1, -2), & x[1] > 20 \end{cases}$$
- **Synthetic-2** consists of ten attributes ($x[1], x[2], \ldots, x[10]$) and a class label. The function f_1 used in Synthetic-1 is also used to label the instances such that the other seven attributes are noise and not used in the prediction process. The true explanations of the instances are defined similarly to the Synthetic-1 dataset.
- **Synthetic-3** consists of three attributes ($x[1], x[2], x[3]$) and a class label. The function f_2 is used to label the instances. Given an instance x, the true local (numeric) explanation is:
$$e_{true}(x) = (3(x^*[1])^2, -4x^*[2], 3)$$
where x^* is the closest point of x to the function $f_2 = 0$.
- **Synthetic-4** consists of three attributes ($x[1], x[2]$) and a class label. The function f_3 is used to label the instances. Given an instance x, the true local (numeric) explanation is:
$$e_{true}(x) = (1, -\sin(x[2])^2 - 2(x[2])^2\cos(x[2])^2)$$
where x^* is the closest point of x to the function $f_3 = 0$.
- **Synthetic-5** consists of nine attributes ($x[1], x[2], \ldots, x[9]$) and a class label. The function f_4 is used to label the instances. Given an instance x, the true local (binary) explanation is:
$$e_{true}(x) = \begin{cases} (1,1,1,0,0,0,0,0,0), & x[1] = 1 \\ (1,0,0,1,1,0,0,0,0), & x[1] = 2 \\ (1,0,0,0,0,1,1,0,0), & x[1] = 3 \\ (1,0,0,0,0,0,0,1,1), & x[1] = 4 \end{cases}$$
- **Synthetic-6** consists of twenty attributes ($x[1], \ldots, x[20]$) and a class label. The function f_4 used in Synthetic-3 is also used to label the instances such that the other eleven attributes are noise and not used in the prediction process. The true explanations of the instances are defined the same way as in the Synthetic-5 dataset.