GALE: Globally Assessing Local Explanations

Peter Xenopoulos [1] **Gromit Chan** [2] **Harish Doraiswamy** [3] **Luis Gustavo Nonato** [4] **Brian Barr** [5] **Claudio Silva** [1]

Abstract

Local explainability methods – those which seek to generate an explanation for each prediction – are increasingly prevalent. However, results from different local explainability methods are difficult to compare since they may be parameter-dependant, unstable due to sampling variability, or in various scales and dimensions. We propose GALE, a topology-based framework to extract a simplified representation from a set of local explanations. GALE models the relationship between the explanation space and model predictions to generate a topological skeleton, which we use to compare local explanation outputs. We demonstrate that GALE can not only reliably identify differences between explainability techniques but also provides stable representations. Then, we show how our framework can be used to identify appropriate parameters for local explainability methods. Our framework is simple, does not require complex optimizations, and can be broadly applied to most local explanation methods.

1. Introduction

Increasingly complex machine learning (ML) models are being deployed in industries such as healthcare, cybersecurity, and banking. While industries welcome the performance boost from these ML models, organizations are also starting to require their models to provide clear explanations for their predictions, as necessitated by regulations, such as GDPR. Thus, explainable artificial intelligence (XAI) techniques are especially relevant, particularly those which provide explanations for individual samples.

There are many techniques which provide *local* explanations for predictive models (Ribeiro et al., 2016). These techniques typically generate a real-valued vector which represents the attributions of the input features to the prediction. However, even using a single explanation technique, it is common to observe different outcomes based on different parameters, which can be hard to tune. Furthermore, since some local explainability methods rely on random sampling, there may also exist sampling variability, which can further impact the stability of the explanation output. Since there are many local explanation methods, each with their own set of parameters, it raises an important question – *how do we assess and compare explanations from different local explainability frameworks*? By doing so, for a given prediction problem, we can develop a quantitative sense of "consensus" among various local explanation frameworks for different prediction tasks, which can be useful for benchmarking.

We propose GALE (globally assessing local explanations), a simple approach to assess the difference between sets local explanations. We accomplish this through an analysis of the topological properties of a given explanation space for binary classification problems. Specifically, we model an explanation method as a scalar function that captures the relationship between the explanation space and the class prediction. Using this function, we compute its topological skeleton. This skeleton is used to generate a topological signature which is then used to compare explanation methods. We can calculate distances between topological skeletons produced by different local explanation methods to compare their similarity. GALE is easy to implement and lacks complex optimizations or parameter tuning.

To the best of our knowledge, our approach is the first to use computational topology for comparing XAI methods. We view topology as a promising direction for understanding and comparing explanation methods and GALE, a powerful yet accessible framework, is the first step in that direction. The contributions of this work can be summarized as follows:

1. GALE (Global Assessing Local Explanations) a topology-based approach to generate a global signature for a given local explanation method's output. By providing a domain-agnostic signature for explanation techniques, our approach allows comparison across heterogeneous explanation approaches.

2. We demonstrate that GALE is both stable and can elucidate differences between explanations through experi-

[1]New York University [2]Adobe Research [3]Microsoft Research India [4]University of São Paulo [5]Capital One. Correspondence to: Peter Xenopoulos <xenopoulos@nyu.edu>.

Presented at the ICML 2022 Workshop on Topology, Algebra, and Geometry in Machine Learning.

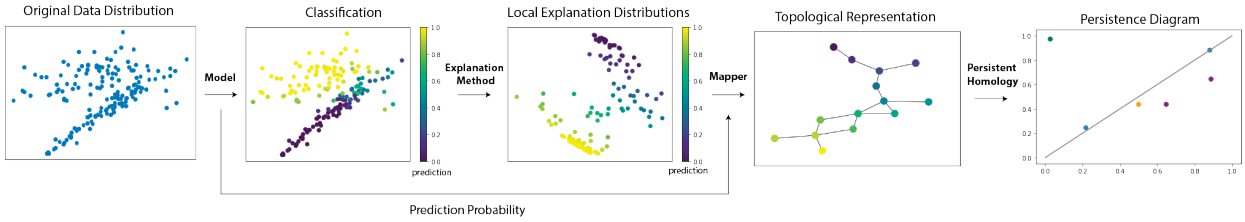

Figure 1. GALE transforms local explanations into a topological representation used to compare local explanation methods.

ments on gradient-based and surrogate model techniques with real-world and synthetic data.

3. We show that GALE can be used to find appropriate parameters for local explanation methods by comparing topological signatures for different sets of parameters.

2. Related Work

To balance the desire for machine learning model performance improvements along with growing calls for explainable model decisions, practitioners are increasingly turning to posthoc local explainability methods (Weld & Bansal, 2019). These local explanation methods typically produce a vector representing the attributions of input features, and are generally well-received by users. For example, LIME (Ribeiro et al., 2016) and SHAP (Lundberg & Lee, 2017) are two common posthoc, model-agnostic methods that produce local feature attributions. However, it is hard to compare results across different explainability methods. Jeyakumar et al. (2020) find, through surveys with hundreds of non-technical users, that explanation-by-example and LIME were the preferred explanation styles. Barr et al. (2020) propose synthetic data generation with known ground truth to explore explainability. Taxonomies, such as *Explainability Fact Sheets*, also provide some guidance on a unified language around explainability and raise important questions for explainability researchers to address (Doshi-Velez & Kim, 2017; Sokol & Flach, 2020).

Some local explanation methods, like integrated gradients, rely on baselines to generate feature attributions. Haug et al. (2021) show that popular local explanation methods are sensitive to the choice of baseline. Sundararajan and Najmi (2020) propose Baseline Shapley and prove its desirable properties. Sturmfels et al. (2020) visually investigate choices of baselines and note that while one can compare baselines through visual inspection, this approach is difficult to scale to enable a large-scale study of interpretability. Furthermore, Mohseni et al. (2021) identify user biases in the subjective rating of model saliency explanations.

To assess and compare the attributions from local explanation methods, a straightforward way is to ablate the top K features ranked by the attributions and observe the decrease of the predicted output score (Sturmfels et al., 2020). To avoid simply removing the top K features without considering the correlations among features, we can ablate the center of mass of the input instead (Ghorbani et al., 2019). Also, to avoid issues of model extrapolation on ablated inputs, we can retrain the model on the ablated data and measure the performance degradation (Hooker et al., 2018). Furthermore, local explanation methods can also be assessed by comparing their behavior between a randomly parameterized model and a trained model (Adebayo et al., 2018). Besides ablation, we can measure the quality of explanations with metrics such as (in)fidelity and sensitivity under perturbations (Yeh et al., 2019), or impact score that measures the feature importance on decision marking process (Lin et al., 2019). If the apriori of feature importance of the dataset is known, the feature importance of input across different models can also be assessed (Yang & Kim, 2019).

Although topological data analysis (TDA) is beginning to be considered for explainability purposes, the field is still nascent. One example of TDA for explainability purposes is shown by Elhamdadi et al. (2021) use TDA to study face poses used in affective computing and find that their topology-based approach captures known patterns. Van Veen (2020) proposed visual constraints on TDA output to aid interpretability, specifically for viewing global, cluster and local explanations. However, there is little work on applying TDA to the outputs of explanation methods in order to compare heterogeneous explanations or quantitatively measure consensus among explanations.

3. Methods

3.1. Topological Background

Reeb Graph and Mapper. Consider a scalar function $f : \mathbb{M} \to \mathbb{R}$, that maps from a manifold $\mathbb{M}$ to $\mathbb{R}$. The *level set* $f^{-1}(a)$ at a given scalar value a is the set of all points that have the function value a. The *Reeb graph* (Reeb, 1946) of f is computed by contracting each of the connected components of the level sets of f to a single point, resulting in a skeleton-like representation of the input.

However, many real-world data sets are available as functions defined on a set of discrete high-dimensional points

rather than as continuous functions. The *Mapper* algorithm (Singh et al., 2007) computes an approximation of Reeb graph of some user-defined function (often called *lens* or *filter* function) of such data. Essentially, the Mapper algorithm divides the function range into a set of overlapping intervals and approximates the level sets to be the set of points that fall within each of these intervals. The connected components of these approximate level sets are then computed by clustering the points that are part of a given interval. Each cluster then forms a node of the approximate Reeb graph and an edge is present between two nodes if they share one or more input points.

Persistence. Given a scalar value a, the sublevel set $f^{-1}((-\infty, a])$ is defined as the set of all points on the domain that have a function value less than or equal to a. Consider a filtration of the input that sweeps the input scalar function f with increasing function values. As the function value increases, the topology of the sublevel sets changes at the *critical points* of the function (where its gradient is zero), and remains constant at other points.

In particular, at a critical point, either a new topology is created, or some topology is destroyed. Here, topology is quantified by a class of k-dimensional cycles (or k-cycles). For example, a 0-cycle represents a connected component, a 1-cycle is a loop that represents a tunnel, and a 2-cycle bounds a void. A critical point is a creator if a new topology appears and a destroyer otherwise. Given a set of critical points $c_1, c_2, \ldots, c_m$, one can pair up each creator c_i uniquely with a destroyer c_j which destroys the topology created at c_i. We say that a topological feature is born at c_i and it dies at c_j. The *topological persistence* (Edelsbrunner et al., 2002) of this topological feature that is created at c_i is defined as $f(c_j) - f(c_i)$, which intuitively indicates the lifetime of this feature in this sweep.

Since we are working with functions defined over discrete points, we use the graph computed by the Mapper algorithm to compute the topological persistence (Edelsbrunner et al., 2002) of the features of the input. Here, the filtration is defined on the nodes and edges of the graph as follows. Each node is assigned a function value equal to the mean of the function values of the clustered points represented by that node. The order of the nodes added during the filtration (or sweep) is defined by the function value of the nodes. An edge is added during the step of the filtration as soon as both its endpoint nodes are added.

Persistence Diagrams. A persistence diagram plots the topological features as a 2-dimensional scatter plot (Edelsbrunner et al., 2002). Each point in the plot corresponds to a single feature and has x and y coordinates equal to its birth and death values respectively obtained from the extended filtration (Figure 2, lower left). Persistence diagrams provide a useful mechanism to assess the structure of scalar functions.

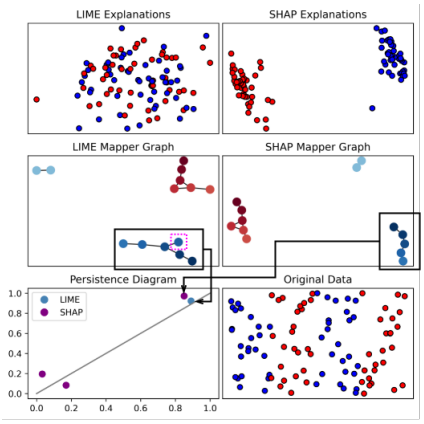

Figure 2. Comparing two sets of explanations using GALE. First, the explanation spaces are summarized using Mapper. From the Mapper output, we create persistence diagrams. Topological features correspond to points in the persistence diagram. In this example, we see that while the two Mapper outputs are similar, the Mapper output from the SHAP explanations is different from that of the LIME explanations (nodes in upper right differ, nodes in lower left overlap).

Moreover, it has also been shown that persistence diagrams are robust to noise (Cohen-Steiner et al., 2007). We can also calculate the distance between two persistence diagrams. In this work, we use the *bottleneck distance* (Cohen-Steiner et al., 2007). Bottleneck distance measures the similarity between two persistence diagrams by finding the shortest distance b for which there exists a perfect matching between the two persistence diagrams, along with all points on the diagonal, such that any pair of matched points have at most a distance of b between them. Formally,

$$d_b(P_A, P_B) = \inf_{\gamma} \sup_{x} \|x - \gamma(x)\|_\infty \tag{1}$$

where P_A and P_B are persistence diagrams, which are multisets, $x \in P_A$, $y \in P_B$ and γ ranges over all bijections from P_A to P_B.

3.2. GALE: Globally Assessing Local Explanations

In this section, we introduce our approach to **g**lobally **a**ssessing **l**ocal **e**xplanations (GALE). Let $\mathcal{X} \subset \mathbb{R}^n$ be a data set and $P : \mathcal{X} \to [0, 1]$ a binary classification model, where $P(x) \in [0, 1]$ is the probability of $x \in \mathcal{X}$ belonging to the "1" class. Given $\mathcal{X}$ and the model P, a local explanation method can be seen as a mapping from the data set to the explanation space $E : \mathcal{X} \to \mathbb{R}^d$, where $E(x)$ provides the "importance" of the different attributes for the classification of $x \in \mathcal{X}$. While $d = n$ for most approaches, d can be greater than n if the explanation method outputs more than a scalar value for each attribute of the input. The mapping $\mathcal{X}' = E(\mathcal{X})$ gives rise to a point

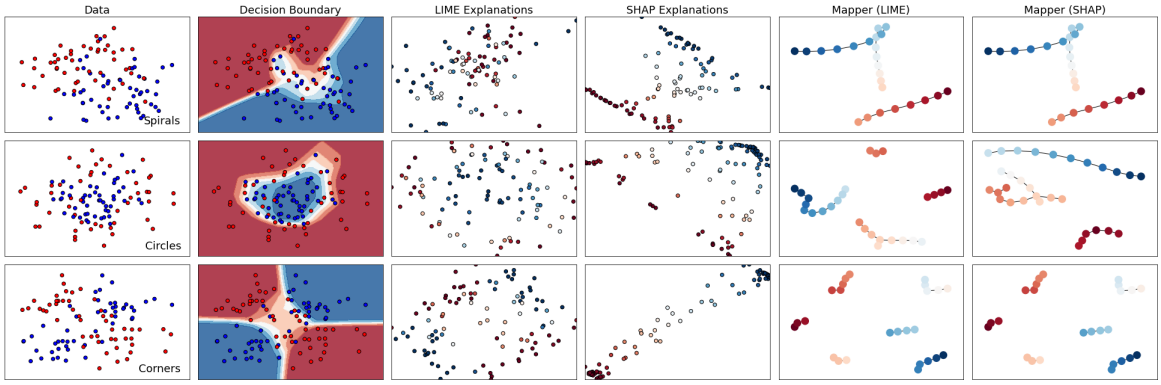

Figure 3. End-to-end pipeline showing the initial data, learned decision boundary, LIME and SHAP explanations, and the topological representations for three different 2-dimensional synthetic classification problems.

set in $\mathbb{R}^d$, which is the explanation manifold. We can define a function $f : \mathcal{X}' \to [0, 1]$ as $f(x') = P(x)$, where $x' = E(x), x \in \mathcal{X}$. In GALE, we use the function f as the lens function from which Mapper builds a summary representation G. Intuitively, f captures the relationship between the explanations and the classification probabilities. From G, we can produce its persistence diagram D.

Since the explanation manifold can drastically vary across different methods and parameters, a direct comparison of the geometry of these manifolds is not possible. However, studying the topology of such functions allows us to analyze how explanation methods differ and thus, we can make geometry agnostic comparisons. To do so, we can take the distance between two persistence diagrams arising from two different explanation methods. If the distance is low, then the topologies of the explanation spaces are similar.

Furthermore, each topological feature (a point in the persistence diagram) can be easily mapped back to a set of input data points in $\mathcal{X}$, thus allowing us to also compare how the explanation space is spread across the input data. We show an example of GALE on three synthetic data sets in Figure 3.

3.3. Tuning Mapper Parameters

Mapper requires three parameters: 1) the *resolution* r of the lens function that defines the number of intervals into which the scalar function range is divided; 2) the *gain* g, which defines the percentage overlap between successive intervals; and 3) the *clustering* algorithm used (which may carry its own parameters). The value of these parameters determines the structure of the resultant graph, and hence the persistence diagram.

In an ideal scenario, increasing the resolution and decreasing the overlap would result in the graph computed using the Mapper algorithm converging to the Reeb graph. However, in real-world data, too high a resolution or too low a gain can result in the graph being a set of disconnected nodes. In our implementation, we use agglomerative clustering. Due to the varying spaces arising from different explanation methods, we set the agglomerative clustering distance threshold parameter to a fraction of the range of the explanation space.

Depending on the data, small changes in the Mapper parameters can drastically change the resulting graph, and hence the persistence diagram. Poor selections of Mapper parameters can produce graphs which are disjointed or unstable under small perturbations to the input. Blaser and Aupetit (2020) suggest that direct measures for measuring the quality of Mapper output could be a combination of cluster quality and their consistency. We are interested in identifying a set of Mapper parameters that not only produces a "stable" graph computation but also a graph with clear structure (i.e., a small number of connected components.)

To measure the aforementioned stability, we use bootstrapping (Chazal et al., 2017) to compute the confidence intervals for the bottleneck distances between persistence diagrams. Consider a set of input explanation values $E = \{e_1, e_2, ..., e_n\}$ and the persistence diagram D generated by its Mapper graph G. G is generated from some parameter set containing the resolution, gain and clustering distance threshold. Then, for each iteration in the bootstrap, we sample with replacement from E to construct $E^* = \{e_1^*, e_2^*, ..., e_n^*\}$, compute Mapper graph G^* and persistence diagram D^* from G^*, and calculate $d_b(D, D^*)$. Using the distribution of distances created by these iterations, we can then find the value $\hat{b}_\alpha$ such that

$$P(d_b(D, D^*) \geq \hat{b}_\alpha) = \alpha \tag{2}$$

A parameter set which is stable will produce a low value for $\hat{b}_\alpha$. Additionally, for each iteration, we count the number

of connected components in G^*. We can find the value $\hat{c}_\alpha$ such that

$$P(\text{connected components of } G^* \geq \hat{c}_\alpha) = \alpha \qquad (3)$$

To tune the Mapper parameters, we perform a grid search over the resolution, gain and distance threshold parameters. For each parameter combination, we use 100 bootstrap iterations to estimate $\hat{b}_\alpha$ and $\hat{c}_\alpha$ for $\alpha = 0.05$. Using the estimated stability and connected components for every combination of parameters, we then iterate through the parameter set and greedily select the best combination. While this approach does not guarantee an optimal solution (e.g., we could arrive at a parameter set which has a low $\hat{b}_\alpha$ but a high $\hat{c}_\alpha$), we find that a greedy strategy works well on a wide array of synthetic and real world data.

4. Evaluation

4.1. Comparing Baselines in Gradient-Based Methods

Here, we illustrate how GALE can be used to understand and compare different explanation methods. In particular, we aim to address a common challenge in using local explanation techniques—what should be the baseline for gradient-based explanation methods?

Baselines act as references to compare the relative importance of features for an input so that attributions can be calculated. We apply three explanation methods—Integrated Gradients (Sundararajan et al., 2017), DeepLIFT (Shrikumar et al., 2017), and SHAP (Lundberg & Lee, 2017) with five different baselines: (1) zero baseline (an input with all values being zeros), (2) maximum distance baseline (Sturmfels et al., 2020), (3) Gaussian baseline (Smilkov et al., 2017; Sturmfels et al., 2020), (4) uniform baseline (Sturmfels et al., 2020), and (5) a trained baseline (Izzo et al., 2020), resulting in 15 explanations for each experiment.

Our goal is to illustrate how GALE can be used to identify explanation methods that behave differently. To do so, we generate a synthetic data set with 5 features and determine the labels with an extremely simple logic—the input will be labeled as "1" if any of the columns contain a "zero" as value. Otherwise, they are labeled as "0". When working with real data sets, it is normal for them to contain zero values that may provide important information. Under our construction, only zero values are important to the classification. Thus we expect the zero baseline is the only reference that is not a neutral input to the classifier. We created 20 different synthetic data sets under the aforementioned conditions, where each set varied in its rate of "0" labeled instances.

We first calculate a pairwise distance matrix for each of the 20 data sets. Then, we average the distance matrices, and show the result in Figure 4. We can see that there are three rows and columns which have much greater average distances (yellow color) than others, meaning that the persistence diagrams generated by these explanations differ greatly as compared to the other explanations. Such observation is consistent with the fact that zero baselines produce different feature attributions by treating the important values as neutral references. Furthermore, when we explore the corresponding topological skeletons, we observe that these explanations typically have a more scattered graph compared with the others. Such visual evidence provide a sense that the explanation values differ greatly among the inputs in the data set. This evidence can be found statistically as well. When computing the variances of the explanation values among different outcomes, the variances are generally high among the outputs with scattered topological graphs.

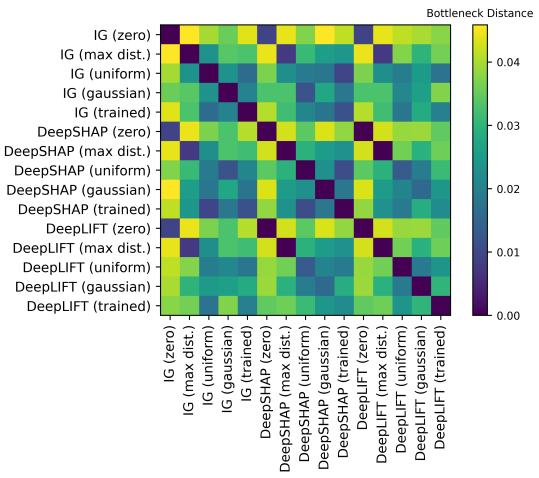

Figure 4. Pairwise distance matrix showing the average bottleneck differences among gradient-based explanation methods across 20 synthetic datasets. We see that the methods which use the zero-baseline (rows/columns denoted with "zero") have higher bottleneck distances to the explanations using other baselines.

4.2. Evaluating Topological Stability

Many methods, like LIME, rely on stochastic behavior, which can produce different explanations for different runs, even when given the same parameters. It is important that GALE is resistant to sampling variability incurred by local explanation methods. In this section, we benchmark GALE's stability with regards to the variation induced by local explainability methods. Additionally, we show how our greedy approach to Mapper parameter selection aids in finding stable topological representations.

To determine the robustness of GALE against local explanation method variability, for each synthetic data set and the diabetes data set, we run LIME 30 times. We restricted our experiments to these data sets for computational purposes.

Data	Row-Wise Distance		Connected Components	
	Greedy	*Fixed*	*Greedy*	*Fixed*
Spirals	0.00	0.83	2.0	4.1
Circles	1.88	2.56	6.1	10.5
Corners	1.22	1.61	9.0	14.6
Lin. Sep.	0.00	0.00	4.0	4.0
Toy	0.03	0.00	2.0	3.0
Toy-Flip	0.00	0.00	2.0	2.2
Toy-Int.	0.16	0.00	2.0	5.0
Diabetes	0.38	2.70	4.3	36.4

Table 1. Average row-wise sum of distance matrix and average connected component counts after using parameters found by our greedy parameter search. We find that our parameter tuning procedure produces more connected and stable Mapper output.

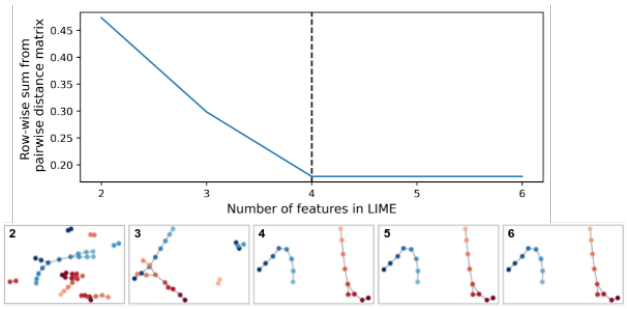

Figure 5. Row-wise sum of distance matrix for Mapper graphs computed on LIME explanations of different feature counts. Since the average row-wise distance stays constant when using 4 or more features for LIME, we know that the produced topologies are similar for these parameter values.

For each run, we generate two Mapper outputs: one which uses the Mapper parameters found via a greedy search, as described in section 3.2, and one which uses fixed parameters: a resolution of 15, a gain of 0.3 and an agglomerative distance threshold of 0.3 times the range of the explanation space. We then generate two bottleneck distance matrices – one for distances between the optimized Mappers and another for distances between the fixed parameter Mappers. In Table 1, we report the average row-sum for each of these matrices for each synthetic data set, as well as the average number of connected components from the generated Mapper graphs. The higher the row-sum, the less consensus there is among the explanation topologies produced by Mapper. We see that by using our greedy Mapper parameter search, our output is resistant to variation in LIME output.

4.3. Using GALE to Tune Explainability Method Parameters

Many local explainability methods use a variety of parameters which guide the output. For example, in SHAP, one can specify parameters such as the number of times to re-evaluate the model when explaining each prediction in Kernel SHAP, or the limit on the number of trees used in Tree SHAP. Choosing an appropriate set of parameters is important, as there can be significant variability in explanations coming from even just slightly different parameters (Visani et al., 2020). Here, we show how to use GALE to tune the parameters of a local explainability method – LIME. We consider a toy synthetic dataset with six independent features, where only four are used for the prediction. We train a random forest classifier on this data using the default sklearn parameters. Our goal is to determine the number of features to use for our LIME explanation, which we know to be four. We set the size of the neighborhood to 50 observations.

Using our Mapper parameter tuning technique to find appropriate parameters, we compute the corresponding graph for explanations using k features, where $k \in \{2, ..., 6\}$. Then, we calculate the bottleneck distance between each of the persistence diagrams. Next, we calculate the row-wise sum of the distance matrix. Persistence diagrams which differ strongly from others will have a high row-wise sum. We plot the row-wise sum in Figure 5. We see that when the number of features is four or higher, the returned topological signatures are identical. Thus, one may conclude that the addition of a fifth or sixth feature in the LIME explanations has no effect on the overall topology of the explanations. Likewise, by only including two or three features in LIME, one may not be estimating appropriate explanations.

5. Conclusion

We present GALE, a topology-based framework to globally summarize the outputs of local explanation methods. To do so, we compute a topological skeleton that captures the relationship between the explanations and model predictions. Then, using this skeleton, we find its persistence diagram, which represents the topological features in the explanation space. Finally, we calculate the distance between the different explanations' corresponding persistence diagrams to derive a measure of similarity between explanations. GALE (1) allows for easy comparison of heterogeneous local explanations, (2) is resistant to outliers and variation, and (3) can be used to optimize explanation method parameters. While GALE sheds no light on what are the *correct* explanations for a given prediction problem, it is a powerful and simple tool to quantitatively compare heterogeneous local explanations, compared to visual inspection or user studies.

Limitations. Although topological data analysis shows promise in tackling challenges encountered in XAI applications, there are still several shortcomings which we plan to address in the future. While we focus on binary classifica-

tion with tabular data, which is a canonical machine learning task, extending GALE to consider multiclass problems, as well as text or image models, will be a key line of future work. Additionally, although we did not evaluate GALE on regression problems, the approach would be similar, except for changing the lens function from the class predictions to the regression output.

Acknowledgements

We would like to thank Bruno Coelho and Richen Du for their support in running preliminary experiments. This collaboration has been supported by a grant from Capital One. Silva's research has also been supported by NASA; NSF awards CNS-1229185, CCF1533564, CNS-1544753, CNS-1730396, CNS-1828576, CNS-1626098;and DARPA. Nonato's research is supported by Fapesp-2013/07375-0 and CNPq-307184/2021-8. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA, NSF, NASA, Fapesp, CNPq, or Capital One.

References

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., and Kim, B. Sanity checks for saliency maps. *arXiv preprint arXiv:1810.03292*, 2018.

Barr, B., Xu, K., Silva, C., Bertini, E., Reilly, R., Bruss, C. B., and Wittenbach, J. D. Towards ground truth explainability on tabular data. *arXiv preprint arXiv:2007.10532*, 2020.

Blaser, N. and Aupetit, M. Research directions to validate topological models of multi-dimensional data. In *NeurIPS 2020 Workshop on Topological Data Analysis and Beyond*, 2020.

Chazal, F., Fasy, B., Lecci, F., Michel, B., Rinaldo, A., Rinaldo, A., and Wasserman, L. Robust topological inference: Distance to a measure and kernel distance. *The Journal of Machine Learning Research*, 18(1):5845–5884, 2017.

Cohen-Steiner, D., Edelsbrunner, H., and Harer, J. Stability of persistence diagrams. *Discrete Comput. Geom.*, 37: 103–120, 2007.

Doshi-Velez, F. and Kim, B. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

Edelsbrunner, H., Letscher, D., and Zomorodian, A. Topological Persistence and Simplification. *Discrete Comput. Geom.*, 28(4):511–533, 2002.

Elhamdadi, H., Canavan, S., and Rosen, P. Affectivetda: Using topological data analysis to improve analysis and explainability in affective computing. *IEEE Transactions on Visualization and Computer Graphics*, 2021.

Fruchterman, T. M. J. and Reingold, E. M. Graph drawing by force-directed placement. *Softw. Pract. Exp.*, 21(11): 1129–1164, 1991.

Ghorbani, A., Abid, A., and Zou, J. Interpretation of neural networks is fragile. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3681–3688, 2019.

Haug, J., Zürn, S., El-Jiz, P., and Kasneci, G. On baselines for local feature attributions. *arXiv preprint arXiv:2101.00905*, 2021.

Hooker, S., Erhan, D., Kindermans, P.-J., and Kim, B. A benchmark for interpretability methods in deep neural networks. *arXiv preprint arXiv:1806.10758*, 2018.

Izzo, C., Lipani, A., Okhrati, R., and Medda, F. A baseline for Shapely values in MLPs: from missingness to neutrality. *arXiv preprint arXiv:2006.04896*, 2020.

Jeyakumar, J. V., Noor, J., Cheng, Y.-H., Garcia, L., and Srivastava, M. How can I explain this to you? an empirical study of deep neural network explanation methods. *Advances in Neural Information Processing Systems*, 33: 4211–4222, 2020.

Lin, Z. Q., Shafiee, M. J., Bochkarev, S., Jules, M. S., Wang, X. Y., and Wong, A. Do explanations reflect decisions? A machine-centric strategy to quantify the performance of explainability algorithms. *arXiv preprint arXiv:1910.07387*, 2019.

Lundberg, S. M. and Lee, S.-I. A unified approach to interpreting model predictions. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 4765–4774. Curran Associates, Inc., 2017.

Mohseni, S., Block, J. E., and Ragan, E. D. Quantitative evaluation of machine learning explanations: A human-grounded benchmark. In Hammond, T., Verbert, K., Parra, D., Knijnenburg, B. P., O'Donovan, J., and Teale, P. (eds.), *IUI '21: 26th International Conference on Intelligent User Interfaces, College Station, TX, USA, April 13-17, 2021*, pp. 22–31. ACM, 2021. doi: 10.1145/3397481.3450689.

Nori, H., Jenkins, S., Koch, P., and Caruana, R. Interpretml: A unified framework for machine learning interpretability. *arXiv preprint arXiv:1909.09223*, 2019.

Reeb, G. Sur les points singuliers dune forme de Pfaff complètement intégrable ou d'une fonction numérique. *Comptes Rendus des séances de l'Académie des Sciences*, 222(847-849):76, 1946.

Ribeiro, M. T., Singh, S., and Guestrin, C. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pp. 1135–1144, 2016.

Shrikumar, A., Greenside, P., and Kundaje, A. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3145–3153. JMLR. org, 2017.

Singh, G., Mémoli, F., and Carlsson, G. E. Topological methods for the analysis of high dimensional data sets and 3d object recognition. In *Eurographics Symp. on Point-Based Graphics*, volume 91, pp. 100, 2007.

Smilkov, D., Thorat, N., Kim, B., Viégas, F., and Wattenberg, M. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

Smith, J. W., Everhart, J. E., Dickson, W., Knowler, W. C., and Johannes, R. S. Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the annual symposium on computer application in medical care*, pp. 261. American Medical Informatics Association, 1988.

Sokol, K. and Flach, P. Explainability fact sheets: a framework for systematic assessment of explainable approaches. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pp. 56–67, 2020.

Sturmfels, P., Lundberg, S., and Lee, S.-I. Visualizing the impact of feature attribution baselines. *Distill*, 5(1):e22, 2020.

Sundararajan, M. and Najmi, A. The many shapley values for model explanation. In *International Conference on Machine Learning*, pp. 9269–9278. PMLR, 2020.

Sundararajan, M., Taly, A., and Yan, Q. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pp. 3319–3328. PMLR, 2017.

van Veen, H. J. Novel topological shapes of model interpretability. In *NeurIPS 2020 Workshop on Topological Data Analysis and Beyond*, 2020.

Visani, G., Bagli, E., Chesani, F., Poluzzi, A., and Capuzzo, D. Statistical stability indices for lime: obtaining reliable explanations for machine learning models. *Journal of the Operational Research Society*, pp. 1–11, 2020.

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

Yang, M. and Kim, B. Benchmarking attribution methods with relative feature importance. *arXiv preprint arXiv:1907.09701*, 2019.

Yeh, C.-K., Hsieh, C.-Y., Suggala, A. S., Inouye, D. I., and Ravikumar, P. On the (in) fidelity and sensitivity for explanations. *arXiv preprint arXiv:1901.09392*, 2019.

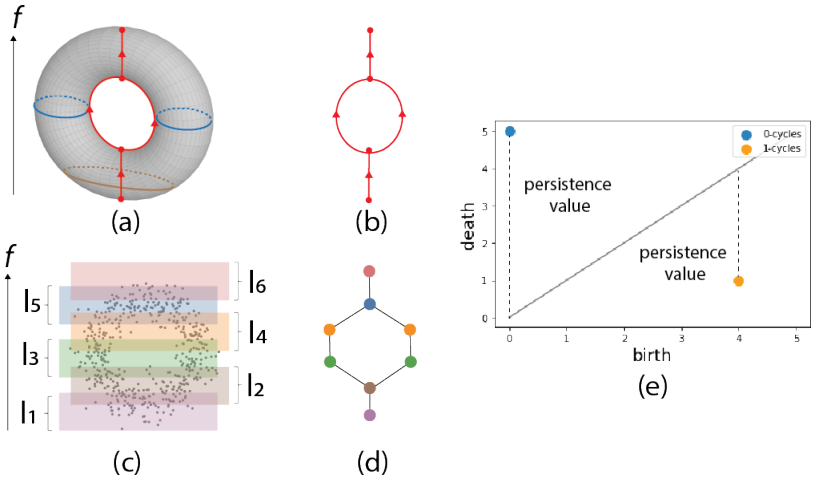

Figure 6. Approximate Reeb graph of a point cloud. (a) Height function defined on a torus. Level sets at two different heights are illustrated. (b) Reeb graph computed on the height function defined on the torus. (c) A point cloud sampled from the torus. (d) The approximate Reeb graph computed using the Mapper algorithm when the height function is divided into 6 intervals as shown in (c). (e) The persistence diagram computed using the Reeb graph.

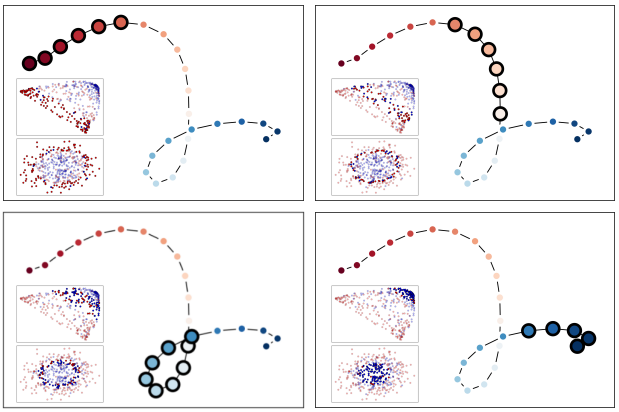

Figure 7. Mapper output for SHAP explanations on a synthetic circles data set with 400 observations. Each quadrant highlights one unique region of the graph, with the lower left of each quadrant showing the selected original data space (bottom) and explanation space (top).

A. Appendix

A.1. Reeb Graphs, Mapper Algorithm, and Persistence

Figure 6 shows an example of a Reeb graph and its corresponding Mapper output and persistence diagram. Figure 6(b) shows the Reeb graph of the height function defined on the torus. In (c), we show points sampled from the torus. Assuming the height function, this is divided into a set of 6 intervals, which is defined by the resolution parameter in Mapper. The gain dictates the amount of overlap between adjacent intervals. Then, the connected components of the points that fall into these intervals are used to generate the graph as shown in (d). In this work, we use Mapper to compute the graph in (d). From the Mapper graph we can compute the persistence diagram (e). When plotting the Mapper graphs, we use the Fruchterman-Reingold force-directed algorithm (Fruchterman & Reingold, 1991). Furthermore, we color nodes by the average prediction of the observations contained in each node. In Figure 7, we can see how the nodes in the Mapper graph correspond to points in both the explanation and original data space.

A.2. Comparing LIME, SHAP and Explainable Boosting Machines

Many local explainability methods are run post-hoc, meaning that first a model is trained and subsequently an explainability method is applied to the model's output to produce explanations. However, understanding the computed explanation space

Data	Type	Obervations	Features	SHAP / LIME	EBM / LIME	EBM / SHAP
Spirals	Synthetic	100	2	0.00	0.11	0.11
Circles	Synthetic	100	2	0.12	0.34	0.35
Corners	Synthetic	100	2	0.08	0.15	0.15
Linearly Separable	Synthetic	100	2	0.00	0.02	0.02
Toy	Synthetic	300	6	0.00	0.03	0.03
Toy-Flip	Synthetic	300	6	0.00	0.08	0.08
Toy-Interaction	Synthetic	300	6	0.01	0.11	0.11
Breast Cancer	Real	569	30	0.00	0.02	0.02
Diabetes	Real	768	8	0.00	0.45	0.45
Sonar	Real	208	60	0.00	0.00	0.00
Banknote	Real	1,372	4	0.00	0.00	0.00
Ionosphere	Real	351	34	0.00	0.01	0.01
Bank Marketing	Real	4,119	19	0.01	0.44	0.44
German Credit	Real	1,000	24	0.00	0.45	0.45

Table 2. Bottleneck distances between LIME, SHAP and EBM explanation topologies for synthetic and real world data. Intuitively, LIME and SHAP generally have low-bottleneck distances, indicating that their explanation topologies are similar.

is oftentimes unintuitive. Generalized additive models, or GAMs, are a modeling approach whereby each feature contributes additively to a model's prediction. One desirable trait of this approach is that the feature contributions are easily interpretable, and thus present a alternative to LIME and SHAP. Explainable Boosting Machine (EBM) are a popular interpretable tree-based GAM from the `interpret` Python library (Nori et al., 2019). While we expect there to be consensus between LIME and Kernel SHAP explanations, since the two methods are similar (Lundberg & Lee, 2017), it is unknown how the explanations from these methods compare to those from EBMs.

We consider both synthetic and real world data sets for our experiments. For our synthetic data sets, we consider the spirals, circles and corners data from Figure 3, one linearly-separable 2-dimensional data set, three "toy" data sets and the Pima Indians diabetes (Smith et al., 1988) data set. In the toy data sets, we have six features $\{x_1, ..., x_6\}$, and the target for observation i is determined by $y_i = x_{i,1} + x_{i,2} + x_{i,3} + x_{i,4}$ and each feature is independent from one another. In "independent", each feature is weighted equally and is independent from one another. In "flip", $y_i = x_{i,1} - x_{i,2} + x_{i,3} - x_{i,4}$. Finally, in "interaction", we add equal-weighted interactions in the form of $y_i = \sum_{k=1}^{4} x_{i,k} + \sum_{j=2}^{4} 10 x_{i,1} \cdot x_{i,j}$. For each data set, we train a 2-layer feedforward neural network with 64 hidden units per layer, as well as an Explainable Boosting Machine (EBM). We use the max number of features and a 50-sample neighborhood for our LIME explainer.

We show the bottleneck distances between the explanations from LIME, SHAP and EBM on each data set in Table 2. As expected, we see strong consensus between LIME and SHAP explanations, particularly in our real world data sets. However, we see varied consensus between EBM and LIME/SHAP explanations. Noticeably, there was more consensus among the three explanation methods when the prediction problem was "easy", such as in the linearly separable and toy synthetic data sets. One possible explanation for the lack of consensus on some data sets is that EBMs, unlike LIME and SHAP, for a k-dimensional data set, return explanations of size j, where $j \geq k$. This is because EBMs include an interaction component, where the interactions are detected by the EBM algorithm. We can enforce zero interactions when training, so that $j = k$ in the explanation output. However, by enforcing no feature interactions, the effect on explanation consensus was ambiguous, which suggests that the topologies generated by GAMs and LIME/SHAP can differ significantly.

A.3. Stability of Explanations under Different Parameters

In Section 1 we mentioned that explanations may vary due to parameter choices or sampling variability. Figure 8 illustrates a scenario where an input's explanation is different after simply calculating LIME explanations multiple times with the same parameters. While it is easy to observe the differences between multiple local explanations through the bar charts, such analysis is only useful for analyzing the method for a single input. Such a visual approach becomes cumbersome when applied to the entire data set. Moreover, it will be almost impossible to understand how stable the explanation method is for that data. Visani et al. (2020) present two indices to measure the stability of LIME outputs.

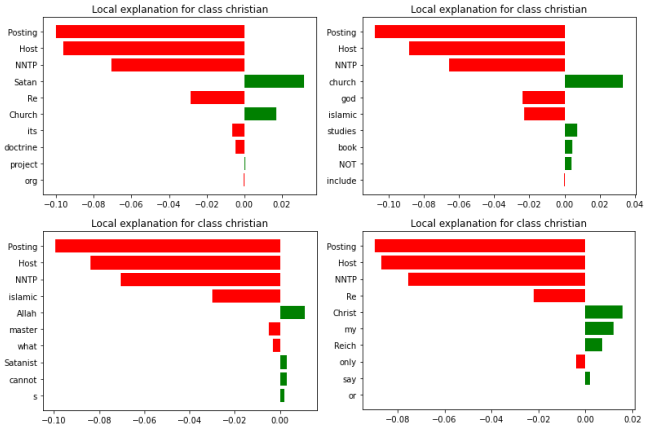

Figure 8. Five trials of LIME-generated explanations with the same parameters and inputs on sklearn's 20 newsgroups dataset. The red bars indicate the words that contribute to the prediction of "Christian" news, and green bars indicate the words that contribute to the prediction of "theism" news. The lengths of the bars represent the feature importances. We show the top-10 features by feature attribution magnitude.

A.4. Implementation

We use the implementations for Mapper, LIME and SHAP from the `sklearn_tda`, `lime` and `shap` Python libraries, respectively. For the Mapper clustering algorithm, we used the agglomerative clustering implementation from the `sklearn` Python library. All predictive models were trained using their respective implementations from `sklearn` and we use the EBM implementation from the `interpret` library (Nori et al., 2019). We use the DeepLIFT and Integrated Gradients implementation from the `captum` Python library. We generated our synthetic data for Spirals, Circles, Corners, Linearly Separable using `sklearn` and for Toy, Toy-Flip and Toy-Interaction, we used the methodology provided by Barr et al. (2020). The real world data we used is available via the UCI Machine Learning Repository (Breast Cancer, Diabetes, Sonar, Banknote, Ionosphere, Bank Marketing, German Credit). We plan on making a Python implementation of GALE available to the public via PyPI.