GLEAMS: Bridging the Gap Between Local and Global Explanations

Giorgio Visani[1,*], Vincenzo Stanzione[2] and Damien Garreau[3]

[1]Leithà S.r.l. - Unipol Group, via Stalingrado 53, Bologna, Italy
[2]Computer Science Department, Bologna University
[3]Julius-Maximilians-Universität Würzburg, Germany

Abstract
The explainability of machine learning algorithms is crucial, and numerous methods have emerged recently. Local, post-hoc methods assign an attribution score to each feature, indicating its importance for the prediction. However, these methods require recalculating explanations for each example. On the other side, while there exist global approaches they often produce explanations that are either overly simplistic and unreliable or excessively complex. To bridge this gap, we propose GLEAMS, a novel method that partitions the input space and learns an interpretable model within each sub-region, thereby providing both faithful local and global surrogates. We demonstrate GLEAMS' effectiveness on both synthetic and real-world data, highlighting its desirable properties and human-understandable insights.

Keywords
Explainable AI, Global / local, Partition-based machine learning

1. Introduction

In numerous applications, the data present themselves as large arrays—tabular data, where each line corresponds to a data point and each column to a feature. In this setting, deep learning is becoming as popular as it already is for more structured data types, and deep neural networks achieve state-of-the-art in many scenarios [1]. A caveat of this approach is the difficulty to obtain insights on a specific prediction: the intricate architectures and the sheer number of parameters prevent the user of getting a clear picture of what is actually important to the model.

Since getting such *explanations* is desirable in many applications, and could even become mandated by law in certain regions [2], many interpretability methods have been proposed. Without getting into details (we refer to the recent surveys [3] and [4] for this purpose), we want to make two distinctions clear. First, it is challenging for interpretable models to have the same accuracy as non-interpretable ones. As a consequence, many explainability methods rely on a *post hoc* approach, *i.e.*, producing explanations in a second step, exploiting already trained black-box models. This is the road we take, since *post-hoc* methods allow us to deal with any given model and not being dependent on a specific architecture. Second, the explanations can either be *local* (related to a specific example), or *global* (for all the input space). A problem

HI-AI@KDD, Human-Interpretable AI Workshop at the KDD 2024, 26th of August 2024, Barcelona, Spain
*Corresponding author.
giorgio.visani@leitha.eu (G. Visani); v.m.stanzione@gmail.com (V. Stanzione); damien.garreau@uni-wuerzburg.de (D. Garreau)
© 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

with local explanations is that they have to be computed individually for each new example, which can be time-consuming. For instance, LIME [5] generates $5,000$ new datapoints for each example to explain. The black-box model then has to be queried on each of these points.

In this paper, we propose **GLEAMS** (Global & Local ExplainAbility of black-box Models through Space partitioning), a *post hoc* interpretability method providing both global and local explanations. Our goal is to build a *global surrogate* $\hat{f}$ of f that shares the global shape of the black-box model f but, at the same time, has a simple local structure. Our main motivation in doing so is to extract simultaneously from $\hat{f}$: i) local explanations for any instance; ii) overall importance of the features. Secondary objectives are the production of "what-if"-type explanations, and visual summaries. In more details, the core idea of GLEAMS is to recursively split the input space in rectangular cells on which the black-box model can be well-approximated by linear models. We describe the method in Section 2. The local explanations are then simply given by the feature coefficients of these local surrogate models, while global indicators can be readily computed from those, without querying f model further. We describe how to obtain the other indicators in Section 2.4, and show experimentally that GLEAMS compares favorably with existing methods in Section 3. GLEAMS code as well as the experiments for this paper are publicly available.[1] Related work can be found in Appendix A.

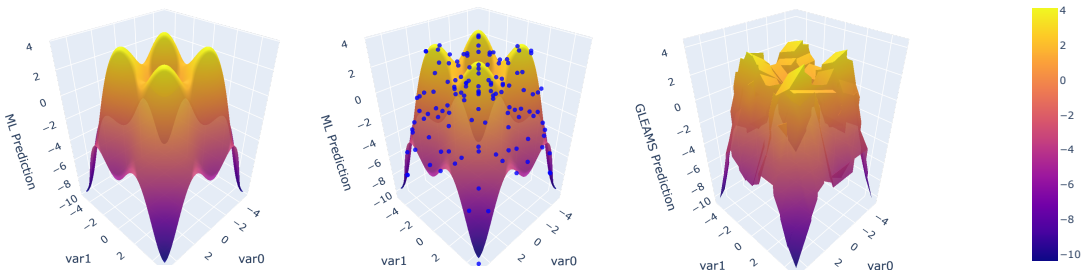

Figure 1: Overview of the global surrogate model construction. *Left panel:* the black-box model maps the input space (here $\mathcal{X} = [0,1]^2$) to $\mathbb{R}$, which we can visualize as a surface. *Middle panel:* we generate N Sobol points on $\mathcal{X}$, giving rise to N measurement points on the surface (in blue). *Right panel:* we fit a piecewise-linear global surrogate model $\hat{f}$ on the measurement points by recursively splitting $\mathcal{X}$.

2. Description of the method

We consider the following general setting: f is a mapping from the input space $\mathcal{X}$ to the output space $\mathcal{Y}$. This corresponds to the usual situation in machine learning, where $y = f(x)$ is a good prediction of the true label of x. In this paper, we assume that $\mathcal{Y} \subseteq \mathbb{R}$. We want to stress that this can correspond both to the *classification* and *regression* setting. In the regression setting, for any input $x \in \mathcal{X}$, $f(x)$ is simply a real-valued quantity of interest, whereas in the classification case $f(x)$ corresponds to the pseudo-probability to fall into a given class.

The construction of the global surrogate on which GLEAMS relies to provide explanations is a two-step process. The first step is to sample N points $x_1, \ldots, x_N$, evenly spread in $\mathcal{X}$. For

[1] https://github.com/giorgiovisani/GLEAMS

each of these points, we query f to obtain $y_i = f(x_i)$. Thus we effectively create a dataset (x_i, y_i), which we call *measurement points*.

Subsequently, the idea is to recursively split the input space $\mathcal{X}$ and to approximate the model f by a *linear model* on each cell of the partition until the linear model in each rectangular leaf fits the measurement points with a sufficiently good accuracy. This process is summarized in Figure 1. Once this global surrogate constructed, GLEAMS can provide both local and global explanations for any new example to explain. Intuitively, given a partition with not too many cells, it is convenient for the user to look at individual features and to visualize globally the model as a piecewise-linear function. Concurrently, any new example to explain will fall into a cell, and the user can get a local explanation solely by looking at the coefficients of the local linear model. We now give more details on each of these steps.

2.1. Measurement points

Our main assumption, going into the description of the method, is the following:

Assumption 1. The input space $\mathcal{X}$ is a hyper-rectangle of $\mathbb{R}^d$. That is, there exist reals numbers $a_j, b_j \in \mathbb{R}$ for any $j \in [d]$ such that $\mathcal{X} = \prod_{j=1}^{d} [a_j, b_j]$.

Intuitively, this corresponds to a situation where each variable has a prescribed range (for instance, age takes values in $[0, 120]$). In particular, we assume that further examples to explain all fall within this range. If they do not, we attribute to these points the explanations corresponding to the projection of these points on $\mathcal{X}$. Note that we do not assume normalized data: a_j and b_j can be arbitrary. In practice, the boundaries are either inferred from a training set, or directly provided by the user.

A consequence of Assumption 1 is that we can easily find a set of points covering $\mathcal{X}$ efficiently. To achieve this, we use **Sobol sequences** [6]. In a nutshell, a Sobol sequence is a quasi-random sequence of points $x_1, \ldots, x_N$ filling the unit cube $[0, 1]^d$ as N grows. We use Sobol sequences because (a) we want to be able to provide explanations for the whole input space $\mathcal{X}$, thus the measurement points should cover $\mathcal{X}$ as N grows; and (b) the decision surface of the black-box model can be quite irregular, thus we want low discrepancy.

2.2. Splitting criterion

A very convenient way to partition the input space is to recursively split $\mathcal{X}$, cutting a cell if some numerical criterion is satisfied. Additionally, we consider splits that are parallel to the axes. The main reason for doing so is the *tree structure* that emerges, allowing later on for fast query time of new inputs (linear in the depth of the final tree). In this section, we describe the criterion used by GLEAMS.

A popular approach is to fit a constant model on each cell [7, CART]. We see two problems with constant fits. First, this tends to produce large and thus hard to interpret trees (see, *e.g.*, Chan and Loh [8]). Second, the local interpretability would be lost, since the local models would not depend on the features in that case. Thus we prefer to fit linear models on each leaf, and our numerical criterion should indicate, on any given leaf, if such simple model is a good fit for f. GLEAMS relies on a variant of model-based recursive partitioning [9, MOB] based on score computations, described in detail in Appendix B.

2.3. Global surrogate model

To build the global surrogate model $\hat{f}$, starting from the hyper-rectangle $\mathcal{X}$, we **iteratively split along the best dimension**. This process is stopped once one of two criteria is met: i) leaves have R^2 higher than 0.95 (chosen empirically) or ii) number of points in the leaf is less than n_{min} threshold, set to $n_{\text{min}} = \min(20, d+1)$. This recursive procedure is described in Algorithm 2, in Appendix B. $\hat{f}$ is then achieved by sampling Sobol points on $\mathcal{X}$, computing the value of the black-box model at these points, and then using Algorithm 2, Appendix B.

2.4. GLEAMS

Let us now assume that we computed the global surrogate model $\hat{f}$, that is, a piecewise-linear model based on a partition of $\mathcal{X}$. At query time, each new example to explain $\xi \in \mathcal{X}$ is rooted to the corresponding hyper-rectangle R in the tree structure, and thus associated to a linear model $\hat{\beta}^{\xi}$. From this local linear model, we can directly get different sorts of explanations.

Local explanations. Simply looking at the coefficient $\hat{\beta}^{\xi}_j$ tells us how important feature j is to predict $f(\xi)$, locally. The intuition here is that f is well-approximated by $x \mapsto (\hat{\beta}^{\xi})^\top x$ on the cell R, thus local variations of the model along specific axes are well-described by the coefficients of $\hat{\beta}^{\xi}$. These local explanations can be obtained in *constant time*, without re-sampling and querying the model. Moreover, we have precise insights regarding the *scale* of these explanations, which is given by the size of R. We see this as an advantage of using GLEAMS: the portion of space on which the black-box model is locally approximated is well-identified and accessible to the user. This is not the case with other methods such as LIME.

Global explanations. We define the global importance of feature j as the aggregation of the local importance of feature j on all cells of the partition $\mathcal{P}$ of $\mathcal{X}$. There are several ways to aggregate those, GLEAMS outputs the weighted average of the absolute value of the local importance. More precisely, let us define $\hat{\beta}^{R}$ the local model on hyper-rectangle $R \in \mathcal{P}$, then the global importance of feature j is given by

$$I_j := \frac{1}{\mathcal{V}(\mathcal{X})} \sum_{R \in \mathcal{P}} \mathcal{V}(R) \left| \hat{\beta}^{R}_j \right| , \tag{1}$$

where $\mathcal{V}(E)$ denotes the d-dimensional volume of E. The motivation for weighting by the cells' volume in Eq. (1) is to take into account the fact that even though feature j is very important in a tiny region of the space, if it is not on all other cells then it is not *globally* important. We take the absolute value to take into account the possible sign variations: if feature j is positively important on many cells but negatively important on a similar number of cells, then simply taking the average would tell us that the feature is not globally important. Such *global attributions* provide a faithful variables ranking, based on the impact they show on f predictions.

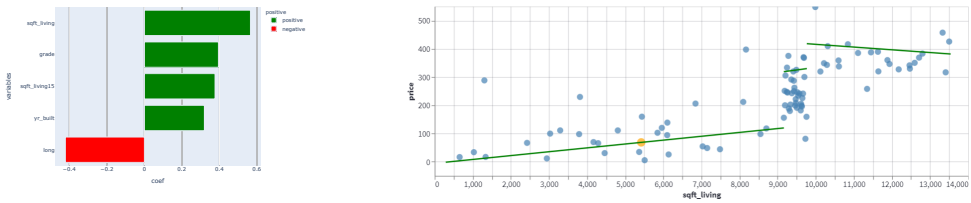

Figure 2: Gleams explanations consist of both: i) Local importance (feature attribution) and ii) counterfactuals, *i.e.*, answering what-if questions. On the left feature importance for a regressor trained on the house sell dataset (Section 3), while figure on the right answers to "What if my house was bigger?" Moving along feature `sqft_living`, for a given example (yellow dot), we can visualize specific values of f (Sobol points in blue) as well as the GLEAMS linear approximations (in green).

Counterfactual explanations. Querying this new model $\hat{f}_j^{\xi}$ at different values of x_j, we can answer to the question "what would be the decision if we changed feature j by this amount?" More precisely, for a given example ξ and a feature j, we can look at the evolution of $\hat{f}$ when all coordinates of the input are fixed to ξ, except the jth: it suffices to travel in the cells intersecting the line of equation $x_j = \xi_j$ and to read the coefficients of the associated linear models. To be precise, we compute

$$W_j(x) := \hat{f}(\xi_1, \ldots, \xi_{j-1}, x, \xi_{j+1}, \ldots, \xi_d)\,. \tag{2}$$

This can be computed quickly, since, again, there is no sampling and further calls to f. This allows us to present the user with a cursor which can be moved to set the feature value and present explanations in real time (see Figure 2).

3. Experiments on real data

Our goal here is to show that GLEAMS provides explanations whose quality is comparable to that of other attribution approaches, while providing these attributions in constant time and additional insights. We benchmark the various methods on the *Wine* dataset [10], *House sell* dataset[2] and *Parkinson telemonitoring* dataset [11]. We refer to Appendix D for more experiments.

Models. We considered two different models, trained similarly across all datasets. The first is an *XGBoost* model [12]. XGBoost iteratively aggregates base regressors greedily minimizing a proxy loss. Following Friedman [13] prescriptions, we employ simple CART trees with maximum depth 2 as base regressors, learning rate of 0.05, and early stopping rounds set to 100. The second is a *multi-layer perceptron* (MLP), composed by two hidden dense layers of 264 neurons each, trained by adaptive stochastic gradient descent [14, ADAM] and early stopping on a hold-out validation set. Both these models achieve state-of-the-art accuracy in many settings and are not inherently interpretable.

[2]available at https://www.openml.org/search?type=data&status=active&id=42092

Table 1
Experimental results (higher is better), 5th and 95th confidence intervals are to be found in brackets. GLEAMS is on par with state-of-the-art explanation methods in terms of reliability of explanations.

dataset	evaluation	LIME		SHAP		PDP		**GLEAMS**	
		XGB	MLP	XGB	MLP	XGB	MLP	XGB	MLP
wine	local	0.38	0.51	0.40	0.53	0.56	0.84	0.51	0.77
		[-0.09, 0.77]	[0.19, 0.84]	[-0.08, 0.76]	[0.22, 0.78]	[0.23, 0.83]	[0.69, 0.95]	[0.15, 0.78]	[0.7, 0.86]
wine	global	0.24	0.74	0.27	0.78	0.70	0.89	0.36	0.85
		[-0.27, 0.66]	[0.53, 0.9]	[-0.19, 0.65]	[0.53, 0.93]	[0.44, 0.9]	[0.78, 0.97]	[-0.19, 0.79]	[0.69, 0.95]
wine	recall (6 true features)	0.89	0.80	1.0	0.87	0.50	0.50	0.77	0.75
		[0.67, 1.0]	[0.67, 1.0]	[1.0, 1.0]	[0.83, 1.0]	[0.5, 0.5]	[0.5, 0.5]	[0.5, 0.83]	[0.5, 1.0]
house	local	0.05	0.63	0.52	0.82	0.51	0.83	0.37	-0.19
		[-0.25, 0.32]	[0.37, 0.81]	[0.22, 0.79]	[0.73, 0.89]	[0.73, 0.9]	[0.74, 0.9]	[0.26, 0.47]	[-0.3, -0.05]
house	global	0.23	0.63	0.37	0.81	0.38	0.82	0.47	0.24
		[-0.09, 0.55]	[0.36, 0.1]	[0.07, 0.64]	[0.69, 0.87]	[0.26, 0.54]	[0.71, 0.9]	[0.07, 0.72]	[-0.02, 0.57]
house	recall (10 true features)	0.85	0.74	0.99	0.69	0.40	0.40	0.61	0.58
		[0.7, 1.0]	[0.6, 0.8]	[0.9, 1.0]	[0.5, 0.8]	[0.4, 0.4]	[0.4,0.4]	[0.4, 0.7]	[0.4,0.7]
Parkinson	local	0.10	0.26	0.47	0.55	0.28	0.49	0.50	0.01
		[-0.11, 0.41]	[-0.04, 0.59]	[0.3, 0.66]	[0.28, 0.76]	[0.16, 0.34]	[0.26, 0.76]	[0.36, 0.6]	[-0.2, 0.23]
Parkinson	global	0.31	0.46	0.41	0.71	0.52	0.66	0.30	0.36
		[0.02, 0.59]	[0.16, 0.71]	[0.08, 0.71]	[0.47, 0.88]	[0.29, 0.71]	[0.43, 0.83]	[-0.06, 0.63]	[0.01, 0.68]
Parkinson	recall (10 true features)	0.86	0.77	0.97	0.80	0.40	0.40	0.50	0.60
		[0.7, 1.0]	[0.5, 0.8]	[0.9, 1.0]	[0.4, 0.7]	[0.4, 0.4]	[0.4, 0.4]	[0.3, 0.7]	[0.43, 0.6]

Methods. We compared GLEAMS to three methods, namely LIME for tabular data, SHAP, and PDP. We used default parameters, except for the number of feature combinations tested in SHAP for the recall metric, which were shrinked to 500 to limit computing time. GLEAMS results have been obtained using $N = 15$ (number of Sobol points), which guaranteed a running time in the order of 5 minutes for each of the three datasets at hand.

Evaluation. We compare GLEAMS to existing methodology using monotonicity and recall. For each interpretability method to evaluate, for each point in the test set, we compute the vectors of attributions $\alpha \in \mathbb{R}^d$ and expected loss $e \in \mathbb{R}^d$ restricted to the corresponding feature. Monotonicity is then defined as the Spearman rank correlation between $|\alpha|$ and e [15]. Depending on the region where the e values are computed we obtain **local monotonicity**, *i.e.*, inside each specific GLEAMS leaf, or **global monotonicity** when considering the whole input space. Let us now consider a model trained to use only a subset T of the features. We can actually be certain that some feature attributions should be 0. For any given example ξ, we can

define T^ξ the set of the top $|T|$ features, ranked by $|\alpha_j|$. **Recall of Important Features** [5] is the fraction of T features retrieved in T^ξ, averaged among the test units. More on this in the Appendix C.

Results. We present our results in Table 1, reporting the average metrics taken on the test set. All evaluation metrics are described in Appendix C.

PDP generally achieves better monotonicity metrics, since it is designed precisely to obtain feature *attributions* related to the feature impact on the predictions, but it performs worse on the recall task. We notice that GLEAMS shows better results on *local* and *global monotonicity* compared to LIME and SHAP on the wine dataset, while GLEAMS performances start to degrade with an increased number of features. In particular we notice the poor performance on *local monotonicity* of the MLP model for the Parkinson dataset. The degradation is due to the higher dimensionality of the two datasets, which makes models surface more complex (MLP in particular). Increasing N would likely improve the results, enabling smaller partitions and more accurate local models. Unfortunately, we did not have time to run experiments with higher N. Regarding the recall metric, SHAP stands as the best in class, while GLEAMS is usually slightly worse but still comparable with LIME, and systematically better than PDP.

4. Conclusion

In this paper, we presented GLEAMS, a new interpretability method that approximates a black-box model by recursively partitioning the input space and fitting linear models on the elements of the partition. Both local and global explanations can then be obtained in constant time, for any new example, in sharp contrast with existing *post hoc* methods such as LIME and SHAP.

The main limitation of the method is the assumption that all features are continuous. As future work, we aim to extend GLEAMS to mixed features (both categorical and continuous). Another limitation of the method is the initial number of model queries needed. One possible way to reduce it would be to evaluate the local regularity of f and to sample less when the model is smooth.

References

[1] V. Borisov, T. Leemann, K. Seßler, J. Haug, M. Pawelczyk, G. Kasneci, Deep neural networks and tabular data: A survey, IEEE Transactions on Neural Networks and Learning Systems (2022).

[2] S. Wachter, B. Mittelstadt, L. Floridi, Why a Right to Explanation of Automated Decision-Making Does Not Exist in the General Data Protection Regulation, International Data Privacy Law 7 (2017) 76–99.

[3] A. Adadi, M. Berrada, Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI), IEEE Access 6 (2018) 52138–52160.

[4] P. Linardatos, V. Papastefanopoulos, S. Kotsiantis, Explainable AI: A review of machine learning interpretability methods, Entropy 23 (2020) 18.

[5] M. T. Ribeiro, S. Singh, C. Guestrin, Why Should I Trust You?: Explaining the Predictions of Any Classifier, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016, pp. 1135–1144.
[6] I. M. Sobol, Points which uniformly fill a multidimensional cube, Mathematics, cybernetics series (1985) 32.
[7] L. Breiman, J. H. Friedman, R. A. Olshen, C. J. Stone, Classification and regression trees, Wadsworth & Brooks / Cole Advanced Books & Software, 1984.
[8] K.-Y. Chan, W.-Y. Loh, LOTUS: An algorithm for building accurate and comprehensible logistic regression trees, Journal of Computational and Graphical Statistics 13 (2004) 826–852.
[9] A. Zeileis, T. Hothorn, K. Hornik, Model-based recursive partitioning, Journal of Computational and Graphical Statistics 17 (2008) 492–514.
[10] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, J. Reis, Modeling wine preferences by data mining from physicochemical properties, Decision support systems 47 (2009) 547–553.
[11] A. Tsanas, M. Little, P. McSharry, L. Ramig, Accurate telemonitoring of parkinson's disease progression by non-invasive speech tests, Nature Precedings (2009) 1–1.
[12] T. Chen, C. Guestrin, XGBoost: A Scalable Tree Boosting System, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794.
[13] J. H. Friedman, Greedy function approximation: a gradient boosting machine, The Annals of Statistics (2001) 1189–1232.
[14] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).
[15] A.-P. Nguyen, M. R. Martínez, On quantitative aspects of model interpretability, arXiv preprint arXiv:2007.07584 (2020).
[16] M. Craven, J. W. Shavlik, Extracting tree-structured representations of trained networks, Advances in Neural Information Processing Systems (1996) 24–30.
[17] J. R. Quinlan, C4.5: Programs for Machine Learning, Elsevier, 1993.
[18] R. D. Gibbons, G. Hooker, M. D. Finkelman, D. J. Weiss, P. A. Pilkonis, E. Frank, T. Moore, D. J. Kupfer, The CAD-MDD: A computerized adaptive diagnostic screening tool for depression, The Journal of clinical psychiatry 74 (2013) 669.
[19] Y. Zhou, G. Hooker, Interpreting Models via Single Tree Approximation (2016).
[20] J. Lei, M. G'Sell, A. Rinaldo, R. J. Tibshirani, L. Wasserman, Distribution-free predictive inference for regression, Journal of the American Statistical Association 113 (2018) 1094–1111.
[21] S. M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, Advances in neural information processing systems 30 (2017).
[22] M. T. Ribeiro, S. Singh, C. Guestrin, Anchors: High-precision model-agnostic explanations, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
[23] M. Setzu, R. Guidotti, A. Monreale, F. Turini, D. Pedreschi, F. Giannotti, Glocalx-from local to global explanations of black box AI models, Artificial Intelligence 294 (2021) 103457.
[24] F. Harder, M. Bauer, M. Park, Interpretable and differentially private predictions, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, 2020, pp. 4083–4090.

[25] E. C. Merkle, A. Zeileis, Tests of Measurement Invariance without Subgroups: A Generalization of Classical Methods, Psychometrika 78 (2013) 59–82.
[26] J. Zhou, A. H. Gandomi, F. Chen, A. Holzinger, Evaluating the quality of machine learning explanations: A survey on methods and metrics, Electronics 10 (2021) 593.
[27] F. Doshi-Velez, B. Kim, Towards a rigorous science of interpretable machine learning, arXiv preprint arXiv:1702.08608 (2017).
[28] C. Spearman, The proof and measurement of association between two things, The American journal of psychology 100 (1904) 441–471.
[29] G. F. Montufar, R. Pascanu, K. Cho, Y. Bengio, On the number of linear regions of deep neural networks, Advances in neural information processing systems 27 (2014).

A. Related work

The idea of approximating globally a complex model such as a deep neural network by a simpler *surrogate model* relying on a tree-based partition of the space can be dated back at Craven and Shavlik [16]. The proposed method, TREPAN, approximates the outputs of the network on the training set by a decision tree, choosing the splits using the *gain ratio* criterion [17]. More recently, inspired by Gibbons et al. [18], Zhou and Hooker [19] proposed to use another split criterion, based upon an asymptotic expansion of the Gini criterion. Both these approaches are limited to the classification setting, and also require access to the training data (which is implicitly assumed to have a good covering of the input space).

From the interpretability side, a standard way to explain f, which is closely related to our approach, is to compute *attributions* scores for each feature. Lei et al. [20] exclude a specific feature from f and define its impact as the deterioration of model predictions. On the contrary, Partial Dependence Plots (PDP) developed by Friedman [13] measure the sensitivity of f to changes in the specific feature, isolating its effect from the ones of the other features. Ribeiro et al. [5] propose LIME, which proposes as attribution for a specific example the coefficients of a linear model approximating f locally. Similarly, Lundberg and Lee [21] introduce the idea of decomposing f predictions into single variables *attributions*, using SHAP. The technique samples combinations of features S and average changes in single instance prediction between the restricted $f(S \cup i)$ against $f(S)$, obtaining local *attributions* for the feature i. Global SHAP *attributions* arise by averaging local attributions over the entire dataset.

Attribution methods usually have to be recomputed for each new example, a caveat that we propose to overcome with GLEAMS, computing a global surrogate and relying on it to provide explanations. We are not the first to propose to provide explanations on a global scale: Ribeiro et al. [22] propose Anchors, a method extracting local subsets of the features (anchors) that are sufficient to recover the same prediction, while having a good global coverage. A caveat of the method is that local explanations are not quantitative since all members of a given anchor are equivalent. Additionally, the anchors can have many elements if the example at hand is near the decision boundary. Let us also mention Setzu et al. [23], which proposes to *aggregate* local explanations: starting from local decision rules, GlocalX combines them in a hierarchical manner to form a global explanation of the model. In the *ad hoc* setting, Harder et al. [24] proposes to train an interpretable and differentially private model by using several local linear maps per class: the pseudo-probability of belonging to a given class is the softmax of a mixture of linear models. This approach is limited to the classification setting, as with GlocalX and Anchors.

B. Splitting criterion

Intuitively, any well-behaved function can be locally approximated by a linear component, yielding a statistical model which we now describe. On any given leaf, for any given feature j, we can reorder the measurement points such that the points contained in the leaf are $x_1, \ldots, x_n$, with $x_{1,j} \leq \cdots \leq x_{n,j}$. To these points correspond model values $y_1, \ldots, y_n$. In accordance with the intuition given above, we write $y_i = \beta^\top \tilde{x}_i + \epsilon_i$ for all $i \in [n]$, where $\beta \in \mathbb{R}^{d+1}$ are the

coefficients of our local linear model, $\tilde{x}_i \in \mathbb{R}^{d+1}$ is the concatenation of x_i with a leading 1 to take the intercept into account, and ϵ_i is an error term. Let us now follow Merkle and Zeileis [25] and make an i.i.d. Gaussian assumption on the ϵ_is, giving rise to the likelihood

$$\mathcal{L}(\beta; x_i) = \frac{1}{(2\pi\sigma^2)^{d/2}} \exp\left(\frac{-1}{2\sigma^2}(y_i - \beta^\top \tilde{x}_i)^2\right), \tag{3}$$

where $\sigma^2 > 0$ is the variance of ϵ_i for all $i \in [n]$. Taking the log in Eq. (3), the log-likelihood of a single observation is given by

$$\ell(\beta; x_i) = \frac{-1}{2}\left\{\frac{1}{\sigma^2}(y_i - \beta^\top \tilde{x}_i)^2 + d \log 2\pi\sigma^2\right\}. \tag{4}$$

The individual scores are obtained by taking the partial derivatives of the log-likelihood with respect to the parameters, that is, for any $i \in [n]$,

$$s(\beta; x_i) = \left(\frac{\partial \ell(\beta; x_i)}{\partial \beta_0}, \frac{\partial \ell(\beta; x_i)}{\partial \beta_1}, \ldots, \frac{\partial \ell(\beta; x_i)}{\partial \beta_d}\right)^\top. \tag{5}$$

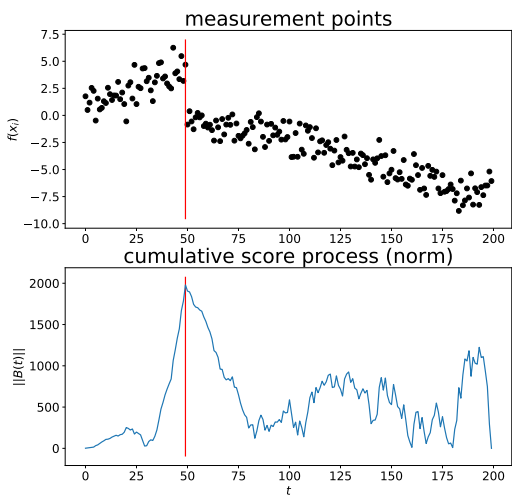

Figure 3: Evolution of the norm of the cumulative score process. *Top panel:* measurements of a piecewise-linear model (dark dots) visualized along one axis. *Bottom panel:* evolution of $\|B(t)\|_1$ as a function of t (solid blue line). The process presents a clear maximum, which gives us a candidate split for this axis (vertical red line).

A straightforward computation yields

$$s(\beta; x_i) = \left(y_i \cdot \tilde{x}_i - \tilde{x}_i \tilde{x}_i^\top \beta\right) \sigma^{-2}. \tag{6}$$

The maximum likelihood estimate $\hat{\beta}$ is obtained by solving $\sum_i s(\beta; x_i) = 0$: in this case this coincides with the *ordinary least-squares* estimate, in a sense the best linear fit on the entire leaf.

However, what we want is to split this leaf if the deviations from the linear model computed on the whole leaf are too strong. Therefore, we compute the *cumulative score process* defined by

$$\forall t \in [0,1], \quad B^{(j)}(t;\hat{\beta}) = \frac{1}{\sqrt{n}} \sum_{i=1}^{\lfloor nt \rfloor} s(\hat{\beta}; x_i)\,, \tag{7}$$

and take as a criterion **the maximum of the L^1 norm of $B^{(j)}(t)$, across all features.** The computation of the best split is described in Algorithm 1 and we provide an example in Figure 3.

Note that Eq. (7) corresponds to Eq. (13) in Merkle and Zeileis [25], with the notable difference that Eq. (13) is normalized (by the square root of the estimated covariance matrix of the scores). We observe empirically that normalizing yields worse results when detecting the splits.

Algorithm 1 `GetBestSplit`: Get the best split for a given leaf.

Require: $x_1, \dots, x_n$: Sobol points in current leaf; $y_1, \dots, y_n$: corresponding f values

for $j = 1$ to d **do**
  re-order $x_1, \dots, \dots, x_n$ according to feature j
  re-order $y_1, \dots, y_n$ accordingly
  compute $\hat{\beta} = \mathrm{OLS}(y; x)$
  compute predictions $\hat{y}_i$
  set $\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
  compute scores $s(\hat{\beta}; x_i)$ according to Eq. (6)
  compute $B^{(j)}$ according to Eq. (7)
  store $m_j = \max_{t\in[0,1]} \left\|B^{(j)}(t)\right\|_1$
  store $t_j = \arg\max_{t\in[0,1]} \left\|B^{(j)}(t)\right\|_1$
return $j^\star = \arg\max_{j\in[d]} m_j$, feature to split
return $t_j^\star$, value of the split
return $\hat{\beta}$, local linear model

Algorithm 2 `RecTree`: Recursive construction of $\hat{f}$.

Require: R: hyper-rectangle; S: Sobol points inside R; V: values of f on S

initialize $T \leftarrow \mathcal{X}$, tree storing partition and models
for $L \in T$.leaves **do**
  if $R^2(L) \leq \rho$ and $n(L) \geq 2n_{\text{min}}$ **then**
    set $x = \{x_1, \dots, x_n\} = S \cap L$
    set $y = \{y_1, \dots, y_n\}$ accordingly
    $(j, t_j) = \texttt{GetBestSplit}(x, y)$
    $L^\ell \leftarrow L \cap \{x : x_j \leq t_j\}$
    $L^r \leftarrow L \cap \{x : x_j > t_j\}$
    L.left_child $\leftarrow \texttt{RecTree}(L^\ell, S, V)$
    L.right_child $\leftarrow \texttt{RecTree}(L^r, S, V)$
return T

Algorithm 3 `GlobalSurrogate`: Compute the global surrogate model used by GLEAMS.

Require: f: black-box model; $\mathcal{X}$ or its boundaries; N: number of Sobol points; ρ: R^2 threshold; n_{min}: min. number of points per leaf

1: generate $S = \{x_1, \dots, x_N\}$ Sobol points in $\mathcal{X}$
2: compute $y_i = f(x_i)$
3: $V \leftarrow \{y_1, \dots, y_N\}$
4: **return** $T \leftarrow \texttt{RecTree}(\mathcal{X}, S, V)$

C. Evaluation

Evaluating interpretability methods is quite challenging: for a given machine learning model, there is rarely a ground-truth available, telling us which features were used for a given prediction. Zhou et al. [26] outline two main evaluation methodologies: *human-centered* evaluations, and *functionality-grounded* (quantitative, note that the terminology was introduced by Doshi-Velez and Kim [27]). We focus on the latter, since human-centered evaluations are hard to obtain and can be unreliable. Among those, we choose to compare GLEAMS to existing methodology using monotonicity and recall. Below we briefly describe these two metrics.

Monotonicity is defined as the correlation between the absolute value of the attributions at a given point and the expected loss of the model restricted to the corresponding feature. Intuitively, this takes high values if the model is very imprecise when it does not know the feature at hand. This is Metric 2.3 in Nguyen and Martínez [15]. Following their notation, for each interpretability method to evaluate, for each point in the test set, we compute on the one hand a vector of attributions $\alpha \in \mathbb{R}^d$ and on the other hand a vector of expected loss $e \in \mathbb{R}^d$. Monotonicity is then defined as the Spearman rank correlation [28] between $|\alpha|$ and e. Formally, for all $j \in [d]$, e_j is defined as

$$e_j = \int_{a_j}^{b_j} \ell(f(\xi), f_j(x))p(x)\mathrm{d}x\,, \tag{8}$$

where f_j is the restriction of f to coordinate j (keeping all other coordinates equal to those of ξ), p is a probability distribution on $[a_j, b_j]$ (recall that a_j and b_j denote the boundaries of $\mathcal{X}$ along dimension j, see Assumption 1), and ℓ is the squared loss.

Let us define $\mathcal{U}(C)$ the uniform distribution on a compact set C. In our experiments, we consider two natural choices of p in Eq. (8). First, $p \sim \mathcal{U}([\ell_j, r_j])$, where ℓ_j (resp. r_j) are the left (resp. right) boundaries of R along feature j, where R is the cell containing ξ. This corresponds to **local monotonicity**: how well the attributions reflect the variations in prediction locally, that is, from the point of view of GLEAMS, on R. Second, $p \sim \mathcal{U}([a_j, b_j])$, which corresponds to **global monotonicity**: how well the attributions reflect the variations in prediction on the whole input space along feature j.

Let us conclude this section by defining formally the **Recall of Important Features**. For certain models, we can actually be certain that some feature attributions should be 0. This is the case, for instance, if we force our model f to use only a subset of the features. As a measure of the quality of explanations, one can then compare the top features selected by an attribution method to the set of true features, which we know. Let us call T the set of true features. For any

given example ξ, we can define T^{ξ} the set of the top $|T|$ features, ranked by $|\alpha_j|$. Originally proposed by Ribeiro et al. [5], the recall of important features is then defined as

$$\rho(\xi) := \frac{\left|T \cap T^{\xi}\right|}{|T|} \,. \tag{9}$$

Intuitively, $\rho(\xi)$ is large (close to one) if the considered attribution method gives large attributions to the true features. In our experiments, we report the average recall over all points of the test set unless otherwise mentioned.

D. Toy data experiments

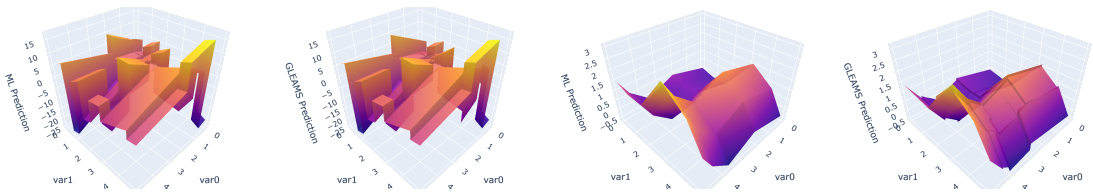

Figure 4: Approximating piecewise-linear models with GLEAMS surrogate model. *Left panels:* example (i), model on the left and GLEAMS surrogate on the right. *Right panels:* example (ii), model on the left and GLEAMS surrogate on the right.

As a first sanity check, we ran the partition scheme of GLEAMS on simple models and checked whether the surrogate model $\hat{f}$ was indeed close to the true model f. In order to have a ground-truth for the partition, we chose partition-based models, *i.e.*, there exist a partition $\mathcal{P}$ of $\mathcal{X}$ such that the parameters of f are constant on each rectangular cell $R \in \mathcal{P}$. We considered a simple setting where the model is piecewise-linear with rectangular cells, a situation in which GLEAMS could, in theory, recover perfectly both the underlying partition and the coefficients of the local models. We present here two examples: (i) a discontinuous piecewise-linear model with arbitrary coefficients, and (ii) a piecewise-linear model with continuity constraints. Example (i) is a generalization of the surface produced by a Random Forest regressor, while example (ii) is reminiscent of a fully-connected, feedforward, ReLU activated neural network. Note however that for the latter, the cells of the partition are more complicated than simple rectangle, see Montufar et al. [29] (in particular Figure 2). In each case, GLEAMS recovers perfectly the underlying model, as demonstrated in Figure 2.

It is interesting to notice that, despite having a stopping criterion on the coefficient of determination, not all leaves satisfy this criterion since the other stopping criterion is a minimal number of points per leaf. Thus it is possible to end up in situations where the R^2 on a given leaf is sub par. Nevertheless, these leaves are by construction quite small. For instance, in example (i), the average R^2 is equal to 1, and in example (ii) it is equal to $.98$.