Efficient Computation of SHAP Values for Piecewise-Linear Decision Trees

Alexey Guryanov
Lomonosov Moscow State University
Moscow, Russian Federation
0000-0001-9365-9995

2021 International Conference on Information Technology and Nanotechnology (ITNT) | 978-1-6654-3217-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/ITNT52450.2021.9649051

***Abstract*—The interpretability of machine learning models is important in many applied data analysis problems. In recent years, a universal SHAP interpretation paradigm based on Shapley values has become popular. However, to interpret the values using SHAP directly, it is necessary to calculate the model's prediction the number of times that is exponential to the size of the feature space, which leads to the prohibitive complexity of this calculation method for complex models over complex feature spaces, for example, for neural networks or gradient boosting ensembles. It was shown that there are efficient algorithms for calculating SHAP values for decision trees and additive ensembles based on them, using structure of trees for an optimized calculation. One of interesting new classes of machine learning models is piecewise linear decision trees and gradient boosting ensembles based on them. In this paper, an efficient algorithm for computing SHAP values for piecewise linear decision trees and additive ensembles based on them was proposed.**

Keywords—Machine Learning, Machine Learning Model Interpretation, SHAP, Piecewise-Linear Decision Trees, Gradient Boosting

I. INTRODUCTION

In recent years, machine learning algorithms have been used in an increasing number of industries. However, spread of such algorithms is hampered in certain industries by the inability to explain why models make specific predictions or decisions. Interpretation algorithms have been actively used with varying degrees of success to solve this problem. Such techniques help explain predictions of models, as well as gain a greater understanding of the operation of the system, which is useful both for maintaining systems and for their further development. One of the most interesting technologies for interpreting machine learning models that have been developed in recent years is the so-called. SHAP values allow us to assess each factor's contribution to the final decision, and at the same time, have a number of important and valuable properties. For an arbitrary machine learning model, the structure of which is unknown in advance, calculation of SHAP values is computationally expensive, but for a number of machine learning models, including linear models and models based on decision trees, methods of efficient calculation have been proposed.

Piecewise linear decision trees are an attractive new machine learning model. Such models achieve better performance without significantly increasing the prediction time than regular decision trees, which is an important factor in some application areas. In recent years, several effective ways of training such models have been proposed. Within this paper framework, a justification of the correctness of a practical algorithm for calculating SHAP values of standard decision trees is provided, and a new algorithm is proposed for efficient calculation of SHAP values.

The work is structured as follows: in the second chapter, the problem of interpreting machine learning models is described, and the definition of the SHAP interpretation method based on Shapley values is given; in the third chapter, the structure of different types of decision trees are given; the fourth chapter justifies an efficient computation the SHAP values of different types of decision trees; in the fifth chapter experimental results are described.

II. SHAP VALUES

The ability to correctly interpret machine learning models' predictions is essential in a large number of industries. Interpretation can help prove the reliability of the model to the client, provide additional information based on which the model can be improved, and help to understand the modeling process. In some areas, simple models are often preferred because of their ease of interpretation, even though they may be less accurate than complex models.

One of the new ways to interpret machine learning models is the so-called SHAP values [1]. This method of interpretation has important theoretically substantiated properties and generalizes the approaches proposed in several previous works [2], [3]. One of the properties is additivity - SHAP values of a sum of a model can be computed as a sum of SHAP values of individual models.

To calculate SHAP values, the function:

$$f_x(S) = f(h_x(z') = E[f(x)|x_S]$$

is introduced, where S is the set of nonzero positions in the simplified feature representation z', and $E[f(x)|x_S$ is the mathematical expectation of this function due to the set S of the simplified feature representation. SHAP values combine these conditional mathematical expectations with the classical Shapley values from game theory in order to attribute ϕ_i to each of the features for $i \in 1..M$:

$$\phi_i = \sum_{i \in S \setminus \{i\}} \left[\frac{|S|!\,(M - |S| - 1)!}{M!} [f_x(S \cup \{i\}) - f_x(S)] \right].$$

In recent works [4] it was shown, that it is possible to efficiently compute SHAP values for decision trees and additive ensembles over decision trees..

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

Authorized licensed use limited to: Ericsson. Downloaded on September 24,2022 at 02:09:11 UTC from IEEE Xplore. Restrictions apply.

III. Decision tree types

A. Decision trees

Decision trees are one of the most wide-spread machine learning algorithms [5]. They are actively used to solve classification, regression and ranking tasks in various domain areas. Decision tree is a binary tree, the internal vertices contain a decision rule, and leaves contain predictions, which can be either a class category, or a real number. Additive ensembles of machine learning models (often based on decision trees) are often used to solve tasks in many domains, such as medical screening [6] or optical character recognition [7]. One of the most popular implementation of additive ensembles over decision trees is described in [8].

B. Piecewise-linear decision trees

One of the possible developments of standard decision tree is a piecewise-linear decision tree. Let us introduce several different types of decision trees based on different limitations on linear coefficients in the leaves.

Definition 1 (Piecewise-linear decision tree type 1). A piecewise-linear decision tree type 1 is a binary decision tree where predictions in the leaves are linear functions' overall features.

Training of such trees was studied in [9].

Definition 2 (Piecewise-linear decision tree type 2). A piecewise-linear decision tree type 2 is a piecewise decisive tree type 1, with an additional condition that linear functions in the leaves can only use features that participated in the decision rules of all nodes that lead to this leaf.

Training of such trees was studied in [10].

Definition 3 ([Piecewise linear decision tree type 3]). A piecewise-linear decision tree type 3 is a piecewise-linear decision tree type 2, in which the linear coefficients for a specific feature in the leaf are tied to the node that leads to this leaf, which uses a decision rule over this feature, and linear coefficients for a feature are equal in all leaves of left and right subtree of this node.

The ability to train such trees was investigated in [11].

C. Decomposition of decision trees into base predictors

Let us introduce notation to describe the structure of decision trees: D describes tree depth, L - number of leaves, and M - number of features, and let us introduce two types of basic predictors, constant leaf predictor, and linear leaf predictor.

Definition 4 (Constant leaf predictor). Constant leaf predictor is a decision tree, which has precisely one non-zero leaf value.

Definition 5 (Linear leaf predictor). Linear leaf predictor is a piecewise linear decision tree type 1, which has precisely one non-zero linear coefficient in all leaves, and the leaf with a non-zero linear coefficient has an expected output value of zero.

Statement *1*. It is possible to represent a decision tree as a sum of no more than L constant leaf predictors with a depth no more than D.

Statement 2. It is possible to represent a piecewise linear decision tree type 1 as a sum of no more than L constant leaf predictors and no more than M*L linear leaf predictors, type 2 as a sum of no more than L constant leaf predictors and no more than D*L linear leaf predictors, type 3 as a sum of no more than L constant leaf predictors and no more than 2*L - 2 linear leaf predictors, all with depth no more than D.

The representation from these remarks is easy to construct if matching every leaf of a decision tree to a constant leaf prediction and every non zero linear coefficient in a piecewise-linear decision tree to a linear leaf predictor, structures of different types of piecewise-linear decision trees impose a limit on several such coefficients. It is also possible to construct such representations using only a single traversal of a decision tree, which means that such an operation's complexity will be O(L).

IV. Efficient interpretation of decision trees

Statement 3. It is possible to compute SHAP values of a constant leaf predictor with depth D with an algorithm with complexity $O(D^2)$.

To prove this statement let's establish formulas for prediction value of constant leaf predictor conditioned on a specific known pool of variables. Let F be a set of features that are used in decision rules in the nodes on the path to the leaf j in the constant leaf predictor, $P_j(x,e)$ be a probability that in the node, corresponding to feature e, decision rule will put an object on path towards leaf j and $Ind_j(x,e)$ be an indicator that a decision rule in node with feature e will put object x on path towards leaf j then:

$$w_j(x,S) = \prod_{e \in F} \left[e \in S; P_j(e); Ind_j(x,e)\right] \quad (1)$$

$$f_x(S) = w_j(x,S) v_j.$$

Combining these formulas with formulas for computation of SHAP values it is possible to achieve the following representation:

$$\phi_i = v_j \cdot \sum_{S \subseteq N \setminus \{i\}} \left[\frac{|S|!\,(M - |S| - 1)!}{M!} [w_j(x, S \cup \{i\}) - w_j(x,S)] \right].$$

It is possible to simplify this formula using several techniques, such as separation of set S into two parts, $I = S \cap F$ and $O = S \cap (N \setminus F)$ and combining sums for all subsets of I with a common size:

$$\phi_i = v_j \cdot \sum_{O \subseteq N \setminus \{F,i\}} \sum_{I \subseteq F \setminus \{i\}} \left[\frac{(|I| + |O|)!\,(M - |I| - |O| - 1)!}{M!} [w_j(x, I \cup \{i\}) - w_j(x,I)] \right] =$$

$$v_j \cdot \sum_{\substack{T \in \mathbb{Z} \\ 0 \le T \le |F|}} \sum_{\substack{I \subseteq F \setminus \{i\} \\ |I| = T}} \sum_{O \subseteq N \setminus \{F,i\}} \left[\frac{(T + |O|)!\,(M - T - |O| - 1)!}{M!} [w_j(x, I \cup \{i\}) - w_j(x,I)] \right].$$

It is possible to prove, that $\sum_{O \subseteq N \setminus \{F,i\}} \frac{(T+|O|)!(M-T-|O|-1)!}{M!} = \frac{T!(|F|-T-1)!}{|F|!}$, which means:

Identify applicable funding agency here. If none, delete this text

Authorized licensed use limited to: Ericsson. Downloaded on September 24,2022 at 02:09:11 UTC from IEEE Xplore. Restrictions apply.

$$\phi_i = v_j \cdot \sum_{\substack{T\in\mathbb{Z} \\ 0\le T\le|F|}} \sum_{\substack{I\subseteq F\setminus\{i\} \\ |I|=T}} \left[\frac{T!\,(|F|-T-1)!}{|F|!}[w_j(x, I\cup\{i\}) - w_j(x,I)]\right] =$$

$$v_j \cdot \sum_{\substack{T\in\mathbb{Z} \\ 0\le T\le|F|}} \frac{T!\,(|F|-T-1)!}{|F|!}\left[\sum_{\substack{I\subseteq F\setminus\{i\} \\ |I|=T}} [w_j(x, I\cup\{i\}) - w_j(x,I)]\right]$$

Let us introduce a $V(K,T)$, that corresponds to a sum of all combinations of weights from previous equation with an additional condition - you can only use no more than the first K features for this calculation according to a specific ordering of these features. It is possible to notice based on equation 1, that these values are bound by recurrent dependencies, which allow us to use a dynamic programming approach to compute values of $V(|F|,T)$ for all values of T for single feature importance ϕ_i in $(O(D^2)$:

$$V(K,T) = \sum_{\substack{I\subseteq F^K\setminus\{i\} \\ |I|=T}} [w_j(x, I\cup\{i\}) - w_j(x,I)]$$

$$\phi_i = v_j \cdot \sum_{\substack{T\in\mathbb{Z} \\ 0\le T\le|F|}} \frac{T!\,(|F|-T-1)!}{|F|!} \cdot V(|F|,T)$$

$$\begin{aligned} V(0,0) &= Ind_j(x,i) - P_j(i) \\ V(K,T) &= 0, T > K \\ V(K+1,K+1) &= V(K,K) * Ind_j(x, e_{K+1}) \\ V(K+1,T+1) &= V(K,T) * Ind_j(x, e_{K+1}) + V(K,T+1) * P_j(e_{K+1}) \end{aligned}$$

Since the recurrent dependencies are linear, it is also possible to compute $V(|F|,T)$ for all ϕ_i in a single run of dynamic programming algorithm by computing dynamic programming without filtering out feature i to compute a table of size DxD. Then to use backward inference to compute all relevant $V(|F|,T)$ for every feature in $O(D)$ by assuming i was the last feature in order.

Statement 4. It is possible to compute SHAP values of a linear leaf predictor with depth D with an algorithm with complexity $O(D^2)$.

It is possible to adapt formulas from the previous section for the case of a linear function. Let k be the index of feature used for a linear function, c_k be the value of the linear coefficient, and Ex_k^j be the expectation of the feature's value in the leaf j. Then

$$f_x(S) = w_j(x,S) \cdot c_k^j \cdot [k \in S; x_k - Ex_k^j; 0]).$$

$$\phi_i = \sum_{S\subseteq N\setminus\{i\}} \frac{|S|!\,(M-|S|-1)!}{M!} \cdot c_k^j \cdot$$
$$(w_j(x, S\cup\{i\}) \cdot [k \in S\cup\{i\}; x_k - Ex_k^j; 0] - w_j(x,S) \cdot [k \in S; x_k - Ex_k^j; 0])];$$

We can notice, that ϕ_i can be simplified if we know that index k of linear feature and i of the feature we are trying to compute coincide or not. If $i \neq k$ then conditions in ternary operators $k \in S$ and $k \in S\cup\{i\}$ are equivalent. If $i = k$ then for every set $S \subseteq N\setminus\{i\}$ condition $k \in S$ is not satisfied and condition $k \in S\cup\{i\}$ is satisfied:

$$\phi_i^{i\neq k} = c_k^j \cdot (x_k - Ex_k^j) \cdot \sum_{S\subseteq N\setminus\{i,k\}} [\frac{(|S|+1)!\,(M-|S|-2)!}{M!}$$
$$\cdot (w_j(x, S\cup\{i,k\} - w_j(x, S\cup\{k\}))]].$$

$$\phi_i^{i=k} = c_k^j \cdot (x_k - Ex_k^j) \cdot \sum_{S\subseteq N\setminus\{i\}} \frac{|S|!\,(M-|S|-1)!}{M!} \cdot (w_j(x, S\cup\{i\});$$

These representations are almost identical to the representation achieved in the constant leaf predictor section, and it is possible to introduce a similar algorithm with complexity $O(D^2)$ that computes SHAP values for all features for linear leaf predictors.

Statement 5. It is possible to compute SHAP values of regular decision trees with an algorithm with algorithmic complexity $O(LD^2)$, piecewise-linear decision tree type 1 - $O(MLD^2)$, piecewise-linear decision tree type 2 - $O(LD^3)$, piecewise-linear decision tree type 3 - $O(LD^2)$.

This statement follows from the representation of different types of trees as sums of different leaf predictors and suggested algorithms for computation of SHAP values of leaf predictors in $O(D^2)$.

V. Experimental Results

Suggested algorithm was implemented using python as a programming language. Experimental evaluation was conducted on a publicly available dataset HIGGS [12] which consists of 1100000 objects with 28 real-valued features. An ensemble of 100 piecewise-linear decision trees [11] were trained with different number of first features chosen and different max tree depth. For every value of number of features and max depth from 1 to 8 both a runtime of implemented efficient algorithm and a brute force algorithm

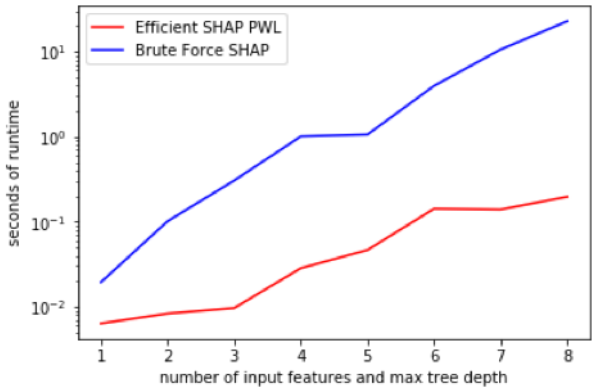

Fig. 1. Runtime improvement of an efficient SHAP computation for piecewise-linear decision trees compared to brute force algorithm.

was computed. The results are shown on figure 1. As can be seen from the figure, suggested algorithm allows for an orders of magnitude speedup compared to a brute force algorithm for computation of SHAP values.

VI. Conclusion

Interpretation of machine learning algorithms is an essential task in many industries. Many exciting and powerful models are not being used because they are impossible or infeasible to be interpreted. This paper introduces an efficient

Authorized licensed use limited to: Ericsson. Downloaded on September 24,2022 at 02:09:11 UTC from IEEE Xplore. Restrictions apply.

algorithm to interpret a new class of machine learning models - piecewise-linear decision trees. The suggested algorithm reduces the computational complexity of interpretation of such models exponentially regarding the number of features to $O(LD^2)$ for a specific type of tree.

REFERENCES

1. S. M. Lundberg, and S. I. Lee, "A unified approach to interpreting model predictions," Advances in neural information processing systems, 2017, pp. 4768-4777.
2. A. Shrikumar, P. Greenside, A. Shcherbina, A. Kundaje, "Not just a black box: Learning important features through propagating activation differences," arXiv preprint arXiv:1605.01713, 2016.
3. M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why should I trust you?" Explaining the predictions of any classifier," Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 2016, pp. 1135–1144.
4. S. M. Lundberg, G. G. Erion, and S. I. Lee, "Consistent individualized feature attribution for tree ensembles," arXiv preprint arXiv:1802.03888, 2018.
5. J. R. Quinlan, "Induction of decision trees," Machine learning, vol. 1(1), pp. 81–106, 1986.
6. A. Ogunleye and Q. G. Wang, "XGBoost model for chronic kidney disease diagnosis," IEEE/ACM transactions on computational biology and bioinformatics, vol. 17(6), pp. 2131–2140, 2019.
7. K. S. Sarin and I. A. Hodashinsky, "Bagged ensemble of fuzzy classifiers and feature selection for handwritten signature verification," Computer Optics, vol. 43(5), pp.833–845, 2019.
8. C. Tianqi and C. Guestrin, "Xgboost: A scalable tree boosting system," Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, 2016.
9. L. de Vito, "LinXGBoost: extension of XGBoost to generalized local linear models," arXiv preprint arXiv:1710.03634, 2017.
10. Y. Shi, J. Li, and Z. Li, "Gradient boosting with piece-wise linear regression trees," arXiv preprint arXiv:1802.05640, 2018.
11. A. Guryanov, "Histogram-based algorithm for building gradient boosting ensembles of piecewise linear decision trees," International Conference on Analysis of Images, Social Networks and Texts. Springer, Cham, 2019.
12. P. Baldi, P. Sadowski, D. Whiteson, "Searching for exotic particles in high-energy physics with deep learning," Nature communications, vol. 5(1), pp. 1–9, 2014.

Authorized licensed use limited to: Ericsson. Downloaded on September 24,2022 at 02:09:11 UTC from IEEE Xplore. Restrictions apply.