EcoShap: Save Computations by Only Calculating Shapley Values for Relevant Features

Samaneh Jmashidi[1], Sławomir Nowaczyk[1] and Mahmoud Rahat[1]

***Abstract*— A popular solution for eXplainable Artificial Intelligence (XAI) is using Shapley values (SVs). Although SVs have a solid mathematical foundation derived from cooperative game theory, they suffer from a large computational cost. SV calculation is NP-hard, requiring approximations, especially when features exceed twenty. However, users are usually interested in the most important features, so it is not essential to calculate SVs for all features. This paper introduces the Economic Hierarchical Shapley Values (ecoShap) method for calculating SVs for the most crucial features only.**

As EcoShap iteratively expands disjoint groups of features, it avoids expensive computations for most of the less important features. The proposed method identifies top features efficiently on eight datasets. Three to seven of the most important features can be determined with half the computation cost.

I. INTRODUCTION

Due to their ever-growing practical applications in industry, business, society, healthcare, and justice, researchers are increasingly focused on explaining machine learning models.

The output of a prediction model must be interpreted correctly, especially in safety-critical systems. As a result, humans gain trust in the model, understand how it makes decisions, and gain insight into its potential improvements.

In eXplainable Artificial Intelligence (XAI), explanations based on feature importance are arguably the most popular approach. Other approaches include prototype explanations, rule-based systems, counterfactual analysis, and model distillation.

The reason is that feature importance provides a straightforward and intuitive approach to understanding the relationship between input features and prediction targets. Especially for users without technical backgrounds, this approach can increase transparency and build trust. Using it, data scientists can identify and revise biased, irrelevant, or redundant features to improve model accuracy. Lastly, AI can only provide knowledge discovery through actionable and robust insights if humans understand how and why machine learning makes its decisions.

Cooperative game theory (CGT) traces have gained recognition recently as one approach to feature importance. The CGT approach is axiomatically motivated. Shapley value (SV) is an example of this type of solution, built on a very strong theoretical foundation and characterized by fairness, symmetry, and efficiency.

Although significant progress has been made in calculating approximate Shapley values, the computational complexity still limits potential applications.

[1]Center for Applied Intelligent Systems Research (CAISR), Halmstad University, Halmstad, Sweden

This paper proposes a method that calculates a limited number of the highest Shapley values instead of all of them. This solution builds on a recent idea, where SVs are calculated for groups of features rather than individually. We exploit the (typically assumed to hold) superadditivity property and the lower computational cost associated with calculating SVs for groups of features at once. By doing so, we can calculate SVs at fractions of the cost (for eight popular machine learning datasets, we always find the single most important feature in 30% of calculations, and in half the time, we can compute three to five of the top features).

In most applications, one only needs a few of the most important features; the rest is unimportant.

II. ECOSHAP ALGORITHM

Our proposed approach follows the binary search tree idea. In the first step, all features F are randomly split into two disjoint subsets of equal sizes, G_1 and G_2. By recursively identifying the subset containing the most important feature and ignoring the rest, we could find f^* in $log_2|F|$ steps.

Although this is not possible in most cases, it is easy to identify which branch is more likely to contain the most important feature(s) based on its SV.

Groups have an SV greater than or equal to their maximum features' SV. Suppose the first subgroup has value $\phi^{Sh}_{G_1}$ and the second has value $\phi^{Sh}_{G_2}$, and without loss of generality $\phi^{Sh}_{G_1} < \phi^{Sh}_{G_2}$. So we split G_2 because we assume that it contains f^*. G_1 is also stored in a priority queue called $\mathcal{G}$. In general, it is expected that f^* belongs to the group with the highest SV, G^*. Throughout each step, G^* will split into two disjoint groups until we find the single feature f_i. f_i has the highest SV in the current tree leaves (note that the fully-expanded tree is unneeded); we can be sure that it has a higher SV than the features belonging to other leaves. Hence, it is the most important feature, i.e., $f_i = f^*$.

$$f_i \equiv f^*, \quad if \quad \forall_{G \in \mathcal{G}} \quad \phi^{Sh}_{f_i} > \phi^{Sh}_{G}. \qquad (1)$$

To find the second important feature, we expand the next-in-line G^* on the remainder of the tree. And when we find a feature with the second-highest value among all leaves, we can be sure it is the second-important feature. At any point in the search, whenever the SV of one feature exceeds that of other groups within $\mathcal{G}$, we can be sure that this feature is more important than all "yet unexplored" features.

III. RESULTS

In this section, by comparing ecoShap's results with those from MCshap, we demonstrate the computational efficiency

Algorithm 1 EcoShap Algorithm

```
Require:
    F - list of all features
    K - the number of most important features wants to find
Ensure:
    Top features - The top K most important features
 1: Top features ← []
 2: $G^* \leftarrow [F]$
 3: $\mathcal{G} \leftarrow [G^*]$
 4: $count_k \leftarrow 0$
 5: while $count_k < K$ do
 6:     if $len(G^*) == 1$ then
 7:         Top features$[count_k] \leftarrow G^*$
 8:         $count_k += 1$
 9:     else
10:         $G_1, G_2 \leftarrow Divide(G^*)$
11:         Calculate $\phi^{Sh}_{G_1}, \phi^{Sh}_{G_2}$
12:         $\mathcal{G}.add(G_1)$
13:         $\mathcal{G}.add(G_2)$
14:     end if
15:     $\mathcal{G}.remove(G^*)$
16:     $G^* \leftarrow$ The member of $\mathcal{G}$ with the highest SV
17: end while
18: Return Top features
```

of the ecoShap algorithm and verify their precision.

A. EcoShap Performance on Budget

This section examines how ecoShap can calculate SVs for many top-ranked features while still saving computations compared to MCshap. Accordingly, we consider the MCshap computational costs as the “full budget.” Table I shows how many features ecoShap can identify using different budget percentages (from 10% to 100%) for eight datasets.

TABLE I: The average number of features found based on the budget percentage

Budget Percentage	Dataset							
	HP	WEC_A	WEC_P	WEC_S	WEC_T	ONP	SC	MSD
10%	0.18	0.0	0.0	0.0	0.0	0.12	1.18	0.42
20%	1.32	0.0	0.0	0.0	0.0	1.04	2.0	3.02
30%	2.18	0.22	0.38	0.18	0.22	2.16	2.6	4.32
40%	3.24	1.8	2.1	1.52	1.64	4.54	3.9	5.98
50%	4.44	3.44	4.3	3.08	3.74	7.1	5.64	6.02
60%	5.46	5.64	7.12	5.82	6.22	8.94	7.58	7.54
70%	6.68	8.14	10.24	9.14	8.96	11.86	11.98	11.4
80%	8.2	12.5	12.5	12.9	11.8	15.04	14.28	15.26
90%	10.7	15.76	15.4	15.22	14.76	18.88	17.06	18.52
100%	13.68	16.6	15.98	16.26	16.14	24.18	19.18	19.64

According to these experiments, Using up to half of the budget, often much less, ecoShap can always identify the top three most important features. In some cases, it is possible to identify up to seven of the most important features with half of the budget. These experiments demonstrate the computational advantages of ecoShap in identifying essential features on a limited budget.

B. The Accuracy of the ecoShap

The final step is to examine ecoShap’s accuracy. According to our assumptions, ecoShap should not introduce any errors in calculations. Despite this, errors can accumulate unfavorable in practice due to the stochastic nature of all the algorithms involved. These effects are negligible, as shown in this section.

Given that the SVs have no definitive ground truth, we use MCshap to construct a close-to-ground-truth (CtGT). To show that our proposed method accurately approximates SVs without significant deviation from the baseline, we compare ecoShap results with their CtGT MCshap counterparts.

We utilize the "features on the whole budget" (FoB) metric, which denotes the most significant features identified by ecoShap using the "full budget." As a measure of accuracy, we use the sum of absolute errors (SAE) of the FoB features, defined as:

$$SAE = \sum_{f \in FoB} |ecoShap(f) - CtGT(f)| \tag{2}$$

TABLE II: The mean and standard deviation of the SAE for each dataset.

Dataset	#feature	#FoB	SAE
HP	36	14	0.0011 ± 0.0002
WEC_A	48	17	0.0011 ± 0.0002
WEC_P	48	16	0.0010 ± 0.0002
WEC_S	48	16	0.0011 ± 0.0002
WEC_T	48	16	0.0010 ± 0.0002
ONP	59	24	0.0005 ± 0.0000
SC	81	19	0.0019 ± 0.0003
MSD	90	20	0.0934 ± 0.0171

The mean and standard deviation of the SAE values are presented in Table II for each dataset, showing that ecoShap does not cause approximation errors. For example, in the HP dataset, the sum of errors for 14 features is 0.0011, which is quite negligible.

IV. CONCLUSIONS

We propose the economic Shapley value (ecoShap) method, which is capable of finding the most important features with low computational costs. By calculating SVs for groups of features at the beginning of the search, this method acts as a filter and avoids wasting the budget on unimportant features.

The proposed algorithm found the top three to seven most important features in all experimented datasets by spending less than half the budget. To verify the accuracy of ecoShap, we compared it with ground-truth results from a baseline, which demonstrated that there is no significant approximation error.

It is worth noting that features are randomly split into groups in ecoShap’s current version. Future work should focus on developing more effective grouping approaches to eliminate unnecessary divisions and improve search efficiency.