# Why Shapley Value and Its Variants Are Useful in Machine Learning and Other Applications

Laxman Bokati[1], Olga Kosheleva[2],
Vladik Kreinovich[1,3], and Nguyen Ngoc Thach[4]
[1]Computational Science Program
Departments of [2]Teacher Education and [3]Computer Science
University of Texas at El Paso, El Paso, Texas 79968, USA
lbokati@miners.utep.edu, olgak@utep.edu, vladik@utep.edu
[4]Institute for Research Science and Banking Technology
Banking University Ho Chi Minh City, Vietnam, thachnn@buh.edu.vn

## 1. Outline

- Shapley value is a useful way to allocate gains in cooperative games.
- It has been very successful in machine learning (and in other applications beyond cooperative games).
- This success is somewhat puzzling:
  - the usual derivation of the Shapley value is based on requirements like additivity
  - these requirements re natural in cooperative games and but not in machine learning.
- In this talk, we provide a new simple derivation of the Shapley value,.
- This derivation does not use game-specific requirements like additivity.
- It is, thus, applicable in the machine learning case as well.

## 2. How to distribute gain between the agents: the problem for which Shapley value was invented

- In a cooperating scenario, we have $n$ collaborating agents $1, \ldots, n$.
- We assume that:
  - for each set $S \subseteq N \stackrel{\text{def}}{=} \{1, \ldots, n\}$ of agents,
  - we know the largest gain $v(S)$ that agents from this set can get with guarantee is they act together.
- The best strategy is for everyone to get together and get the gain $v(N)$.
- The question is: how to divide this resulting gain $v(N)$ between the agents?

## 3. Let us reformulate this problem in more precise terms

- In mathematical terms, what we need is a function $\varphi$ that assigns:
  - to each function $v : 2^N \to \mathbb{R}_0^+$ from the set $2^N$ of all subsets of $N$ to the set $\mathbb{R}_0^+$ of all non-negative real numbers,
  - an $n$-dimensional vector $(\varphi_1(v), \ldots, \varphi_n(v))$ of non-0negative values $v_i$ for which

$$\varphi_1(v) + \ldots + \varphi_n(v) = v(N).$$

## 4. Natural requirements

- A solution to the above problem was proposed in 1951 by Lloyd S. Shapley.
- Shapley received the 2012 Nobel Prize in Economics for this discovery.
- Shapley considered the following two natural requirements.

## 5. First requirement: fairness

- Suppose that in some situations, agents $i$ and $j$ contribute equally, i.e., we have $v(S) = v(\pi_{i \leftrightarrow j}(S))$ for all sets $S$.
- Here, $\pi_{i \leftrightarrow j}$ is a permutation that swaps $i$ and $j$ and leaves all other elements intact.
- Then these two agents should get the exact same amount:

$$\varphi_i(v) = \varphi_j(v).$$

## 6. Second requirement: additivity

- Suppose that we have two different independent situations with the same set of agents:
  - one situation characterized by a function $u$, and
  - another situation characterized by a function $v$.
- Then the overall amount that each agent $i$ gets in both situations is

$$\varphi_i(u) + \varphi_i(v).$$

- Alternatively, we can consider these two situations as a single situation, with $w(S) = u(S) + v(S)$ for all $S$.
- It is reasonable to require that:
  - since we did not change anything by simply considering the two situation as one,
  - the overall gain of each player in this new situation should be the same:

$$\varphi_i(w) = \varphi_i(u + v) = \varphi_i(u) + \varphi_i(v).$$

## 7. Resulting formula

- Shapley showed that these two requirements uniquely determine the function $\varphi_i$:

$$\varphi_i(v) = \sum_{S:\ i \notin S} \frac{|S|! \cdot (n - |S| - 1)!}{n!} \cdot (v(S \cup \{i\}) - v(S)).$$

- Here, $|S|$ denotes the number of elements in the set $S$.

## 8. Shapley value is easy to compute

- When $n$ is small, we can simply use the above formula.
- For large $n$, we can use the equivalent description of the Shapley value $\varphi_i(v)$ in terms of permutations $\pi : N \to N$ of the set $N$.
- Namely, each permutation sorts the elements of the set $N$ as

$$\pi(1) < \pi(2) < \ldots$$

- We can then add these elements one by one.
- For the case when we add the agent $i$, we can compute the difference between the new and the previous value of $v$.
- The expected value of this difference over random permutations is exactly the Shapley value.

## 9. Shapley value is easy to compute (cont-d)

- It is easy to simulate a random permutation:
  - as $\pi(1)$, we select each of elements $1, \ldots, n$ with the same probability $1/n$;
  - then, as $\pi(2)$, we select each of the $n-1$ remaining elements with the same probability $1/(n-1)$, etc.
- Thus, by using such Monte-Carlo simulations, we can estimate the Shapley value as accurately as possible.

## 10. Variants of the Shapley value

- In some applications, it is useful to use variants of the Shapley value, of the type

$$\phi_i(v) = \sum_{S\colon\ i \notin S} a(|S|) \cdot (v(S \cup \{i\}) - v(S)).$$

- Here the function $a(|S|)$ is such that

$$\sum_{S\colon\ i \notin S} a(|S|) = 1.$$

- This condition is equivalent to

$$\sum_{k=0}^{n-1} \frac{(n-1)!}{(n-1-k)! \cdot k!} \cdot a(k) = 1.$$

## 11. Successful use of Shapley value in machine learning

- Lately, the Shapley value has been successfully used in machine learning and in other applications.
- It is used to describe the importance of different inputs.
- In this case:
  - instead of agents, we gave inputs, and
  - instead of a gain $v(S)$, we have a different characteristic,
  - e.g., classification efficiency – corresponding to the case when we only use inputs from the set $S$.

## 12. This success is somewhat puzzling

- The usual derivation of the Shapley value is based on additivity.
- However, for classification efficiency, adding two efficiencies makes no sense.
- We therefore need a different explanation for the empirical success of Shapley value in these applications.

## 13. What we do in this talk

- In this talk, we provide a simple alternative derivation of Shapley value and its variants,
- This derivation that does not use additivity.
- It can, therefore, explain the success of Shapley value and its variants in machine learning applications.

## 14. Main idea

- We want to come up with a value $\varphi_i$ that describes:
  - how much adding an input $i$ improves the desired result,
  - e.g., how much it improves the classification efficiency.
- In other words, we want this value to describe the difference $v(S \cup \{i\}) - v(S)$ between:
  - the result $v(S \cup \{i\})$ obtained by adding $i$ and
  - the result $v(S)$ that we get without adding the input $i$.
- So, we want to have to make sure that:
  - for each set $S$ that does not contain the input $i$,
  - this difference is close to the desired value $\varphi_i$:

$$v(S \cup \{i\}) - v(S) \approx \varphi_i.$$

## 15. From the idea to the exact formulation of the problem

- In mathematical terms, we have several equations for determining a single unknown $\varphi_i$.
- In other words, we have an over-determined system of linear equations.
- In data processing, a usual way to deal with such systems is to the use the Least Squares approach.
- So, we find the value $\varphi_i$ for which the following sum attains its smallest possible value:

$$\sum_{S:\ i \notin S} \frac{((v(S \cup \{i\}) - v(S)) - \varphi_i)^2}{\sigma^2(S)}.$$

- Here the coefficients $\sigma^2(S)$ describe the weight that we assign to each equation.

## 16. Requiring permutation-invariance

- A priori, there is usually no reason to believe that some inputs and more important than others.
- Thus, it makes sense to require:
  - as in the original derivation of the Shapley value,
  - that the weights $\sigma^2(S)$ should not depend on which exactly inputs are included in the set $S$.
- These weights should be permutation-invariant.
- Thus, they should depend only on the size $|S|$ of the corresponding set $S$.

## 17. Requiring permutation-invariance (cont-d)

- So, we must have $\sigma^2(S) = b(|S|)$ for some function

$$b : \{0, \ldots, n-1\} \to \mathbb{R}_0^+.$$

- Thus, we arrive at the need to minimize the following expression:

$$\sum_{S:\ i \notin S} \frac{((v(S \cup \{i\}) - v(S)) - \varphi_i)^2}{b(|S|)}.$$

## 18. Solving the resulting optimization problem

- Let us differentiate the above expression and equate the derivative to 0.
- Then we get

$$2 \cdot \sum_{S:\ i \notin S} \frac{\varphi_i - (v(S \cup \{i\}) - v(S))}{b(|S|)} = 0$$

- This is equivalent to:

$$\varphi_i \cdot \sum_{S:\ i \notin S} \frac{1}{b(|S|)} = \sum_{S:\ i \notin S} \frac{1}{b(|S|)} \cdot (v(S \cup \{i\}) - v(S)).$$

- Thus, we conclude that

$$\varphi_i = \sum_{S:\ i \notin S} a(|S|) \cdot (v(S \cup \{i\}) - v(S)).$$

## 19. Solving the resulting optimization problem (cont-d)

- Here, we denoted

$$a(k)=\frac{\dfrac{1}{b(k)}}{\displaystyle\sum_{S:\ i\notin S}\frac{1}{b(|S|)}}.$$

- This is exactly the above formula for the variants of Shapley value.
- Vice versa, each variant of the Shapley value corresponding to the values $a(k)$ can be obtained this way.
- Namely, it is sufficient to take

$$b(k)=\frac{1}{a(k)}.$$

## 20. Acknowledgments


- This work was supported in part by the National Science Foundation grants:
  - 1623190 (A Model of Change for Preparing a New Generation for Professional Practice in Computer Science), and
  - HRD-1834620 and HRD-2034030 (CAHSI Includes).
- It was also supported by the AT&T Fellowship in Information Technology.
- It was also supported by the program of the development of the Scientific-Educational Mathematical Center of Volga Federal District No. 075-02-2020-1478.