Interpreting Predictions of Tree Ensembles via SHAP Values

Applications and Principle

Pei Liu

May 9, 2019

Department of Computer Science @UESTC

Table of contents

Introduction

What is SHAP values?

SHAP(**SH**apley **A**dditive ex**P**lanation) is a unified approach to explain the output of any machine learning model.

The properties of *SHAP values* are as follows:

- fully individualized
- only possible consistent
- locally accurate

What can SHAP values bring to us?

As the linear model does, the *SHAP values* also can show features each contributing to push the model output from the base value (the average model output) to the model output.

$$\hat{Y} = Prob(\textit{The fruit is an apple}) = 0.4 \cdot Color + \cdots + 0.2 \cdot Shape + 0.1 \cdot Size$$

Gallery

Gallery of SHAP values

The individualized *SHAP values*:

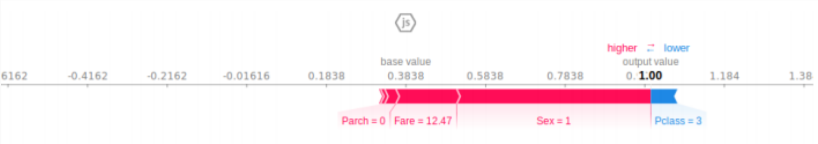

Figure 1: Instance model prediction of Titanic Data

Gallery of SHAP values

To summarize the effects of all the features via *SHAP values*:

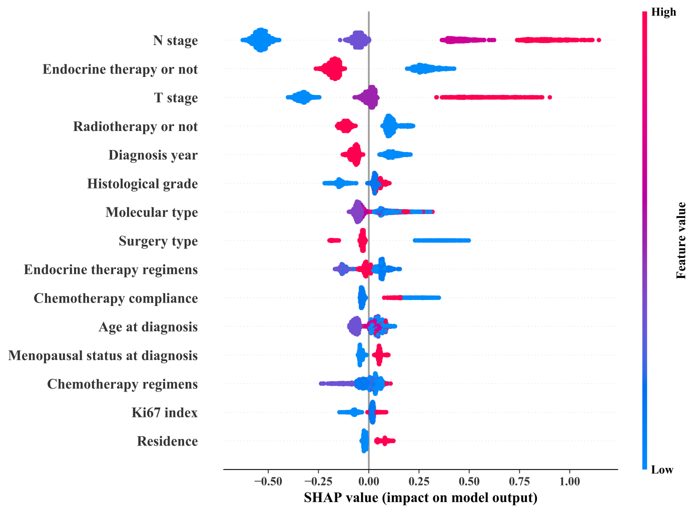

Figure 2: Summary SHAP Plot of West China Hospital Breast Cancer

Gallery of SHAP values

To summarize the effects of all the features via *SHAP values*:

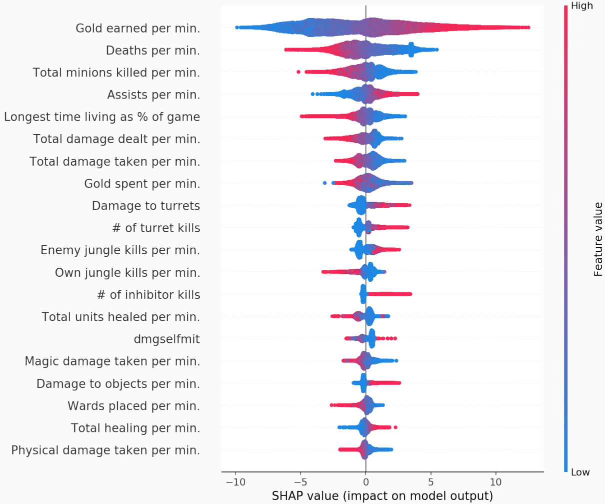

Figure 3: Summary SHAP Plot of LOL Win Prediction

Let us dive into SHAP values

What are the *principles* behind *SHAP values*?

Background

Problem Description

Given that

- a set of players: $N = \{x_1, x_2, \cdots, x_n\}$
- value function: $v(S)$ for any $S \subseteq N$

then what is the payoff for each player i, i.e. $\psi_i(N, v) =?$

Constraints

Accuracy: all values are assigned out for each player.

$$\sum_{i \in N} \psi_i(N, v) = v(N)$$

Constraints

Symmetry: the contribution of play i and j is same if they are *interchangeable*.

$$\psi_i(N, v) = \psi_j(N, v)$$

if

$$v(S \cup \{i\}) = v(S \cup \{j\})$$

for any

$$S \subseteq N \setminus \{i, j\}$$

Interchangebale agents should receive the same shares/payments.

Constraints

Dummy player: i is a dummy player if the amount that i contributes to any coalition is 0.

$$\psi_i(N, v) = 0$$

if

$$v(S \cup \{i\}) = v(S)$$

for all S.

Dummy players should receive nothing.

Constraints

Additivity: if we can separate game into two parts $v = v_1 + v_2$, then we should be able to decompose the payments.

For any two v_1 and v_2,

$$\psi_i(N, v_1 + v_2) = \psi_i(N, v_1) + \psi_i(N, v_2)$$

for each i, where the game is defined by

$$(v_1 + v_2)(S) = v_1(S) + v_2(S)$$

for every coalition S.

Shapley Value in Game Theory

Shapley Value gives the unique solution under those constraints.

> *Given a coalitional game* (N, v)*, the Shapley Value divides payoffs among players according to:*
>
> $$\psi_i(N, v) = \frac{1}{N!} \sum_{S \subseteq N \setminus \{i\}} |S|!(|N| - |S| - 1)!\big[v(S \cup \{i\}) - v(S)\big]$$
>
> *for each player* i*.*

How to understand it?

Principles

Next, questions come to us:

- how would **Shapley value** help us to interprete model's output?
- how does *SHAP values* exploit **Shapley value** in game theory?

The questions above had been solved by the paper titled "A Unified Approach to Interpreting Model Predictions" from **NIPS 2017**.

principles

Since the original model $f(x)$ is so complex that it's hard to be interpreted, we must use a simpler *explanation model* $g(x^{'})$, which we define as any interpretable approximation of the original model.

Definition 1. input space mapping function h_x that mapps the *simplified inputs* $x^{'}$ to the original inputs x, i.e.,

$$x = h_x(x^{'})$$

Some examples of mapping function h_x?

Definition 2. an explaination/approximation model g that is a linear function of binary varibales.

$$g(x^{'}) = \phi_0 + \sum_{i=1}^{M} \phi_i x_i^{'}$$

where $x^{'} \in \{0, 1\}^M$, M is the number of simplified input features, and $x = h_x(x^{'})$ (as defined by **Definition 1** before).

Now, assuming that the mapping function h_x and explaination model g is known for us, i.e. ϕ_i is given.

Perfectly! We can use g to interprete the original model output!

principles

Definition 2. an explaination/approximation model g.

$$g(x^{'}) = \phi_0 + \sum_{i=1}^{M} \phi_i x_i^{'}$$

But with the known explaination model g, we can infer its properties naturally:

- local accuracy
- missingness
- additivity
- interchangeable?

principles

Definition 2. an explaination/approximation model g.

$$g(x^{'}) = \phi_0 + \sum_{i=1}^{M} \phi_i x_i^{'}$$

As a result, based on the game theory, we can know that **Shapley Value** is the only possible solution of the explaination model g.

$$\phi_i(f, x) = \frac{1}{M!} \sum_{z^{'} \subseteq x^{'}} |z^{'}|!(|M| - |z^{'}| - 1)! \left[f_x(z^{'}) - f_x(z^{'} \setminus i) \right]$$

How to calculate $f_x(z^{'})$? Or how to calculate contributions of the observed feature set $z^{'}$ w.r.t model's output?

principles

Using a conditional expectation function of the original model:

$$f_x(z') = f(h_x(z')) = f(z_S) \approx E[f(z)|z_S]$$

where S is the set of non-zero indexes in z'.

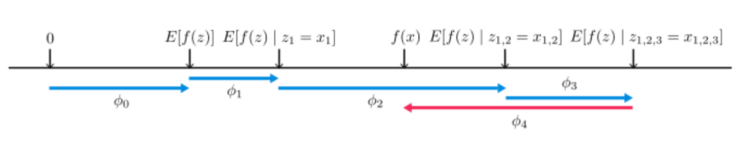

Figure 4: SHAP values calculations

In tree ensemble model, estimating *SHAP values* directly in $O(TL2^M)$ time.

Algorithm 1 Estimating $E[f(x) \mid x_S]$

procedure EXPVALUE($x, S, tree = \{v, a, b, t, r, d\}$)
    procedure G(j, w)
        if $v_j \neq internal$ **then**
            return $w \cdot v_j$
        else
            if $d_j \in S$ **then**
                return G(a_j, w) **if** $x_{d_j} \leq t_j$ **else** G(b_j, w)
            else
                return G($a_j, wr_{a_j}/r_j$) + G($b_j, wr_{b_j}/r_j$)
            end if
        end if
    end procedure
    return G(1, 1)
end procedure

Figure 5: Algorithm of estimating the conditional expectations

principles

More details we can find and discuss in paper:

- Kernel SHAP versus LIME
- Deep SHAP versus DeepLift
- Estimating *SHAP values* in $O(TLD^2)$ time (D, depth of tree)
- SHAP interaction values

Questions?

References

References

References:

- Original paper: `http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions`
- Tree explanation: `https://arxiv.org/abs/1802.03888`
- SHAP library: `https://github.com/slundberg/shap`
- BEAMER template for presentation: `https://github.com/matze/mtheme`