The Tractability of SHAP-Score-Based Explanations over Deterministic and Decomposable Boolean Circuits

Marcelo Arenas, Pablo Barceló, Leopoldo Bertossi, **Mikaël Monet**

AAAI'21 conference, held online, February 6th 2021

Millennium Institute Foundational Research on Data

Inria

Pontificia Universidad Católica de Chile

Universidad Adolfo Ibáñez

One minute summary (1/2)

SHAP-score in explainable AI: a notion used to explain the decisions of AI models. Let:

- M be a **model** (example: a classifier used by a bank to decide when clients can be given a loan)
- e be an **entity** (example: a client)
- x a **feature** (example: "has_stable_job")

→ The SHAP-score $\mathrm{SHAP}(M, \mathrm{e}, x)$ represents the influence of the feature value $\mathrm{e}(x)$ on the output $M(\mathrm{e})$

One minute summary (2/2)

We focus on binary classifiers $M : \{0,1\}^n \to \{0,1\}$ (features are binary, and the output is yes/no)

Main result

The SHAP-score $\mathrm{SHAP}(M, \mathrm{e}, x)$ can be computed in polynomial time when the model M is given as a **deterministic and decomposable Boolean circuit**

These classifiers are studied in the field of knowledge compilation and generalize binary decision trees, Binary Decision Diagrams (OBDDs, FBDDs), d-DNNFs, etc.

Outline

Shapley values and SHAP-score

Knowledge compilation: deterministic and decomposable Boolean circuits

Results and proof sketch

Shapley values and SHAP-score

Shapley values (1/2)

Notion from **cooperative game theory**. Let X be a set of players and $\mathcal{G} : 2^X \to \mathbb{R}$ be a game on X. We wish to assign to every player $p \in X$ a contribution $s_X(\mathcal{G}, p)$. Some reasonnable axioms:

Shapley values (1/2)

Notion from **cooperative game theory**. Let X be a set of players and $\mathcal{G} : 2^X \to \mathbb{R}$ be a game on X. We wish to assign to every player $p \in X$ a contribution $s_X(\mathcal{G}, p)$. Some reasonnable axioms:

1. Symmetry: For every game $\mathcal{G}$ on X and players $p_1, p_2 \in X$, if we have $\mathcal{G}(S \cup \{p_1\}) = \mathcal{G}(S \cup \{p_2\})$ for every $S \subseteq X$, then $s_X(\mathcal{G}, p_1) = s_X(\mathcal{G}, p_2)$

Shapley values (1/2)

Notion from **cooperative game theory**. Let X be a set of players and $\mathcal{G} : 2^X \to \mathbb{R}$ be a game on X. We wish to assign to every player $p \in X$ a contribution $s_X(\mathcal{G}, p)$. Some reasonnable axioms:

1. Symmetry: For every game $\mathcal{G}$ on X and players $p_1, p_2 \in X$, if we have $\mathcal{G}(S \cup \{p_1\}) = \mathcal{G}(S \cup \{p_2\})$ for every $S \subseteq X$, then $s_X(\mathcal{G}, p_1) = s_X(\mathcal{G}, p_2)$
2. Null player: A player p is null if $\mathcal{G}(S \cup \{x\}) = \mathcal{G}(S)$ for every $S \subseteq X$. For every null player we have $s_X(\mathcal{G}, x) = 0$

Shapley values (1/2)

Notion from **cooperative game theory**. Let X be a set of players and $\mathcal{G} : 2^X \to \mathbb{R}$ be a game on X. We wish to assign to every player $p \in X$ a contribution $s_X(\mathcal{G}, p)$. Some reasonnable axioms:

1. Symmetry: For every game $\mathcal{G}$ on X and players $p_1, p_2 \in X$, if we have $\mathcal{G}(S \cup \{p_1\}) = \mathcal{G}(S \cup \{p_2\})$ for every $S \subseteq X$, then $s_X(\mathcal{G}, p_1) = s_X(\mathcal{G}, p_2)$
2. Null player: A player p is null if $\mathcal{G}(S \cup \{x\}) = \mathcal{G}(S)$ for every $S \subseteq X$. For every null player we have $s_X(\mathcal{G}, x) = 0$
3. Linearity: For every $a, b \in \mathbb{R}$, games $\mathcal{G}_1, \mathcal{G}_2$ on X and player p we have $s_X(a\mathcal{G}_1 + b\mathcal{G}_2, p) = a \cdot s_X(\mathcal{G}_1, p) + b \cdot s_X(\mathcal{G}_2, p)$

Shapley values (1/2)

Notion from **cooperative game theory**. Let X be a set of players and $\mathcal{G} : 2^X \to \mathbb{R}$ be a game on X. We wish to assign to every player $p \in X$ a contribution $s_X(\mathcal{G}, p)$. Some reasonnable axioms:

1. Symmetry: For every game $\mathcal{G}$ on X and players $p_1, p_2 \in X$, if we have $\mathcal{G}(S \cup \{p_1\}) = \mathcal{G}(S \cup \{p_2\})$ for every $S \subseteq X$, then $s_X(\mathcal{G}, p_1) = s_X(\mathcal{G}, p_2)$
2. Null player: A player p is null if $\mathcal{G}(S \cup \{x\}) = \mathcal{G}(S)$ for every $S \subseteq X$. For every null player we have $s_X(\mathcal{G}, x) = 0$
3. Linearity: For every $a, b \in \mathbb{R}$, games $\mathcal{G}_1, \mathcal{G}_2$ on X and player p we have $s_X(a\mathcal{G}_1 + b\mathcal{G}_2, p) = a \cdot s_X(\mathcal{G}_1, p) + b \cdot s_X(\mathcal{G}_2, p)$
4. Efficiency: For every game $\mathcal{G}$ on X we have $\sum_{p \in X} s_X(\mathcal{G}, p) = \mathcal{G}(X) - \mathcal{G}(\varnothing)$

Shapley values (2/2)

Theorem [Shapley, 1953]

There is a unique function $s_X(\cdot,\cdot)$ that satisfies all four axioms.

$$\mathrm{Shapley}_X(\mathcal{G}, p) \stackrel{\mathrm{def}}{=} \sum_{S \subseteq X \setminus \{p\}} \frac{|S|!(|X| - |S| - 1)!}{|X|!} (\mathcal{G}(S \cup \{p\}) - \mathcal{G}(S))$$

Has found many applications in computer science.

Next slide: the SHAP-score for XAI

SHAP-score for explainable AI

Let X be a set of features, e an entity (that has a value $\mathrm{e}(x)$ for every feature $x \in X$), M a model (that assigns a value to each entity), $\mathcal{D}$ a probability distribution over the set of entities, and x a feature.

SHAP-score for explainable AI

Let X be a set of features, e an entity (that has a value $\mathsf{e}(x)$ for every feature $x \in X$), M a model (that assigns a value to each entity), $\mathcal{D}$ a probability distribution over the set of entities, and x a feature.

The **SHAP score** $\mathsf{SHAP}_{\mathcal{D}}(M, \mathsf{e}, x)$ is the Shapley value of x in the following game function $\mathcal{G}$:

$$\mathcal{G}(S) \stackrel{\text{def}}{=} \mathbb{E}_{\mathsf{e}' \sim \mathcal{D}}[M(\mathsf{e}') \mid \mathsf{e}'(y) = \mathsf{e}(y) \text{ for all } y \in S]$$

SHAP-score for explainable AI

Let X be a set of features, e an entity (that has a value $\mathsf{e}(x)$ for every feature $x \in X$), M a model (that assigns a value to each entity), $\mathcal{D}$ a probability distribution over the set of entities, and x a feature.

The **SHAP score** $\mathrm{SHAP}_{\mathcal{D}}(M, \mathsf{e}, x)$ is the Shapley value of x in the following game function $\mathcal{G}$:

$$\mathcal{G}(S) \stackrel{\text{def}}{=} \mathbb{E}_{\mathsf{e}' \sim \mathcal{D}}[M(\mathsf{e}') \mid \mathsf{e}'(y) = \mathsf{e}(y) \text{ for all } y \in S]$$

In other words,

$$\mathrm{SHAP}_{\mathcal{D}}(M, \mathsf{e}, x) \stackrel{\text{def}}{=} \sum_{S \subseteq X \setminus \{x\}} \frac{|S|!(|X| - |S| - 1)!}{|X|!} (\mathcal{G}(S \cup \{x\}) - \mathcal{G}(S))$$

When is it tractable?

Question: For which kind of models/probability distributions can we compute it efficiently?

When is it tractable?

Question: For which kind of models/probability distributions can we compute it efficiently?

Theorem [Lundberg et al., 2020]

The SHAP-score can be computed in polynomial time for decision trees

When is it tractable?

Question: For which kind of models/probability distributions can we compute it efficiently?

Theorem [Lundberg et al., 2020]

The SHAP-score can be computed in polynomial time for decision trees

→ We generalize this result to more powerful classes of models, from the field of **knowledge compilation**

Knowledge compilation: deterministic and decomposable Boolean circuits

Knowledge compilation

Knowledge compilation: a field of AI that studies various formalisms to represent Boolean functions...

→ examples: truth tables, Boolean formulas in DNF/CNF, Boolean circuits, binary decision diagrams (OBDDs), binary decision trees, etc.

Knowledge compilation

Knowledge compilation: a field of AI that studies various formalisms to represent Boolean functions...

→ examples: truth tables, Boolean formulas in DNF/CNF, Boolean circuits, binary decision diagrams (OBDDs), binary decision trees, etc.

... and the tasks that these allow to solve efficiently

→ examples: satisfiability in $O(n)$ for truth tables or DNFs but NP-c for CNFs, model counting in $O(n)$ for OBDDs but #P-hard for DNFs, etc.

Knowledge compilation

Knowledge compilation: a field of AI that studies various formalisms to represent Boolean functions...

→ examples: truth tables, Boolean formulas in DNF/CNF, Boolean circuits, binary decision diagrams (OBDDs), binary decision trees, etc.

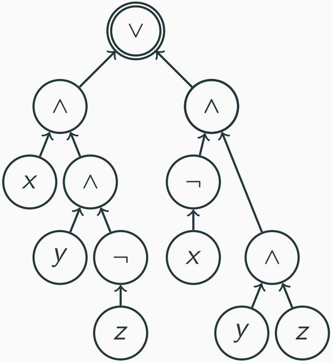
... and the tasks that these allow to solve efficiently

→ examples: satisfiability in $O(n)$ for truth tables or DNFs but NP-c for CNFs, model counting in $O(n)$ for OBDDs but #P-hard for DNFs, etc.

Deterministic and decomposable Boolean circuits: the less restricted formalism of knowledge compilation that allows tractable model counting

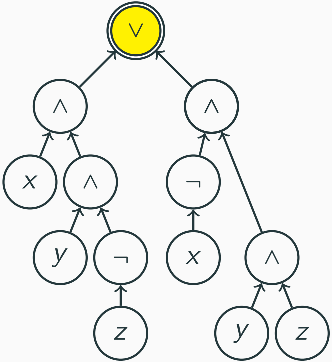
Deterministic and decomposable Boolean circuits

(also called "**tractable Boolean circuits**")



Deterministic and decomposable Boolean circuits

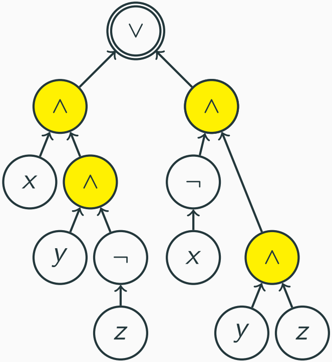
(also called "**tractable Boolean circuits**")



- Deterministic: inputs of ∨-gates are **mutually exclusive**

Deterministic and decomposable Boolean circuits

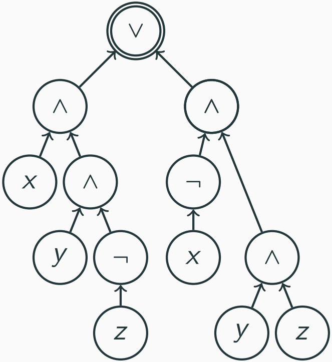
(also called "**tractable Boolean circuits**")



- Deterministic: inputs of ∨-gates are **mutually exclusive**
- Decomposable: inputs of ∧-gates are **independent** (no variable has a path to two different inputs of the same ∧-gate)

Deterministic and decomposable Boolean circuits

(also called "tractable Boolean circuits")



- Deterministic: inputs of ∨-gates are **mutually exclusive**
- Decomposable: inputs of ∧-gates are **independent** (no variable has a path to two different inputs of the same ∧-gate)

→ model counting or even probability evaluation can be solved in linear time

Results and proof sketch

Results

- Set X of binary features; so an entity e is a function from X to $\{0,1\}$
- A deterministic and decomposable circuit M
- An entity e and a feature $x \in X$
- We assume that the distribution $\mathcal{D}$ is such that each feature $y \in X$ has an **independent probability** p_y of being 1

Results

- Set X of binary features; so an entity e is a function from X to $\{0, 1\}$
- A deterministic and decomposable circuit M
- An entity e and a feature $x \in X$
- We assume that the distribution $\mathcal{D}$ is such that each feature $y \in X$ has an **independent probability** p_y of being 1

Main result

Given as input M, e, x and p_y for every $y \in X$, we can compute the SHAP-score $\mathrm{SHAP}_{\mathcal{D}}(M, \mathrm{e}, x)$ in time $O(|M| \cdot |X|^2)$

Results

- Set X of binary features; so an entity e is a function from X to $\{0,1\}$
- A deterministic and decomposable circuit M
- An entity e and a feature $x \in X$
- We assume that the distribution $\mathcal{D}$ is such that each feature $y \in X$ has an **independent probability** p_y of being 1

Main result

Given as input M, e, x and p_y for every $y \in X$, we can compute the SHAP-score $\mathrm{SHAP}_{\mathcal{D}}(M, \mathrm{e}, x)$ in time $O(|M| \cdot |X|^2)$

Secondary result (easy)

For any class $\mathcal{C}$ of models and under the uniform distribution, model counting for $\mathcal{C}$ reduces to the problem of computing SHAP-scores for $\mathcal{C}$

Proof sketch of main result (1/3)

Recall that $\mathrm{SHAP}_{\mathcal{D}}(M, \mathrm{e}, x)$ is defined as

$$\sum_{S \subseteq X \setminus \{x\}} \frac{|S|!(|X| - |S| - 1)!}{|X|!} \Big(\mathbb{E}_{\mathrm{e}' \sim \mathcal{D}}[M(\mathrm{e}') \mid \mathrm{e}'(y) = \mathrm{e}(y) \text{ for all } y \in S \cup \{x\}]$$

$$- \mathbb{E}_{\mathrm{e}' \sim \mathcal{D}}[M(\mathrm{e}') \mid \mathrm{e}'(y) = \mathrm{e}(y) \text{ for all } y \in S]\Big)$$

Proof sketch of main result (1/3)

Recall that $\mathrm{SHAP}_{\mathcal{D}}(M, \mathrm{e}, x)$ is defined as

$$\sum_{S \subseteq X \setminus \{x\}} \frac{|S|!(|X|-|S|-1)!}{|X|!} \Big(\mathbb{E}_{\mathrm{e}' \sim \mathcal{D}}[M(\mathrm{e}') \mid \mathrm{e}'(y) = \mathrm{e}(y) \text{ for all } y \in S \cup \{x\}] \\ - \mathbb{E}_{\mathrm{e}' \sim \mathcal{D}}[M(\mathrm{e}') \mid \mathrm{e}'(y) = \mathrm{e}(y) \text{ for all } y \in S]\Big)$$

Lemma

Computing SHAP-score can be reduced in polynomial time to the following problem.

INPUT: binary features X, entity e, deterministic and decomposable circuit M, integer k.

OUTPUT: $\sum_{\substack{S \subseteq X \\ |S| = k}} \mathbb{E}_{\mathrm{e}' \sim \mathcal{D}}[M(\mathrm{e}') \mid \mathrm{e}'(y) = \mathrm{e}(y) \text{ for all } y \in S]$

Proof sketch of main result (2/3)

Goal: compute $\sum_{\substack{S \subseteq X \\ |S|=k}} \mathbb{E}_{\mathrm{e}' \sim \mathcal{D}}[M(\mathrm{e}') \mid \mathrm{e}'(y) = \mathrm{e}(y) \text{ for all } y \in S]$.

Proof sketch of main result (2/3)

Goal: compute $\sum_{\substack{S \subseteq X \\ |S|=k}} \mathbb{E}_{\mathsf{e}' \sim \mathcal{D}}[M(\mathsf{e}') \mid \mathsf{e}'(y) = \mathsf{e}(y) \text{ for all } y \in S]$.

- **Step 1**: smooth the circuit. A Boolean circuit is *smooth* if for every $\vee$-gate g, every input gate of g sees the same set of variables. We can smooth M in $O(|M| \cdot |X|^2)$

Proof sketch of main result (2/3)

Goal: compute $\sum_{\substack{S \subseteq X \\ |S|=k}} \mathbb{E}_{\mathsf{e}' \sim \mathcal{D}}[M(\mathsf{e}') \mid \mathsf{e}'(y) = \mathsf{e}(y) \text{ for all } y \in S]$.

- **Step 1**: smooth the circuit. A Boolean circuit is *smooth* if for every $\vee$-gate g, every input gate of g sees the same set of variables. We can smooth M in $O(|M| \cdot |X|^2)$
- **Step 2**: for every gate g of the circuit and $\ell \in \{0, \ldots, |\mathrm{var}(g)|\}$, define the value

$$\alpha_g^\ell \stackrel{\text{def}}{=} \sum_{\substack{S \subseteq \mathrm{var}(g) \\ |S|=\ell}} \mathbb{E}_{\mathsf{e}' \sim \mathcal{D}}[M_g(\mathsf{e}') \mid \mathsf{e}'(y) = \mathsf{e}(y) \text{ for all } y \in S]$$

Proof sketch of main result (2/3)

Goal: compute $\sum_{\substack{S\subseteq X\\|S|=k}} \mathbb{E}_{\mathrm{e}'\sim\mathcal{D}}[M(\mathrm{e}') \mid \mathrm{e}'(y)=\mathrm{e}(y) \text{ for all } y\in S]$.

- **Step 1**: smooth the circuit. A Boolean circuit is *smooth* if for every $\vee$-gate g, every input gate of g sees the same set of variables. We can smooth M in $O(|M|\cdot|X|^2)$
- **Step 2**: for every gate g of the circuit and $\ell\in\{0,\ldots,|\mathrm{var}(g)|\}$, define the value

$$\alpha_g^\ell \stackrel{\text{def}}{=} \sum_{\substack{S\subseteq \mathrm{var}(g)\\|S|=\ell}} \mathbb{E}_{\mathrm{e}'\sim\mathcal{D}}[M_g(\mathrm{e}') \mid \mathrm{e}'(y)=\mathrm{e}(y) \text{ for all } y\in S]$$

and compute the values α_g^ℓ by bottom-up induction on the circuit

Proof sketch of main result (3/3)

Compute $\alpha_g^\ell \stackrel{\text{def}}{=} \sum_{\substack{S \subseteq \mathsf{var}(g) \\ |S|=\ell}} \mathbb{E}_{\mathsf{e}' \sim \mathcal{D}}[g(\mathsf{e}') \mid \mathsf{e}'(y) = \mathsf{e}(y) \text{ for all } y \in S]$
for every gate g and integer $\ell \in \{0, \ldots, |\mathsf{var}(g)|\}$

- g is a variable gate with variable y. Then $\alpha_g^0 = p_y$ and $\alpha_g^1 = \mathsf{e}(y)$
- g is an OR gate with inputs g_1, g_2. Then $\alpha_g^\ell = \alpha_{g_1}^\ell + \alpha_{g_2}^\ell$
- g is an AND gate with inputs g_1, g_2. Then $\alpha_g^\ell = \sum_{\substack{\ell_1 \in \{0,\ldots,|\mathsf{var}(g_1)|\} \\ \ell_2 \in \{0,\ldots,|\mathsf{var}(g_2)|\} \\ \ell_1+\ell_2=\ell}} \alpha_{g_1}^{\ell_1} \cdot \alpha_{g_2}^{\ell_2}$
- g is a $\neg$-gate with input g_1. Then $\alpha_g^\ell = \binom{|\mathsf{var}(g)|}{\ell} - \alpha_{g_1}^\ell$

$\rightarrow$ We can compute all the values α_g^ℓ in time $O(|M| \cdot |X|^2)$

Conclusion

- We prove that the **SHAP-score can be computed in PTIME** for deterministic and decomposable Boolean circuits under product distributions
  - → this generalizes a result of [Lundberg et al., 2020]
- We show that computing SHAP-scores is always as hard as the model counting problem
  - → computing SHAP-score is actually PTIME-equivalent to the problem of computing expectations! Check out the [AAA'21 paper by Van den Broeck, Lykov, Schleich, and Suciu] :)

Conclusion

- We prove that the **SHAP-score can be computed in PTIME** for deterministic and decomposable Boolean circuits under product distributions
  - → this generalizes a result of [Lundberg et al., 2020]
- We show that computing SHAP-scores is always as hard as the model counting problem
  - → computing SHAP-score is actually PTIME-equivalent to the problem of computing expectations! Check out the [AAA'21 paper by Van den Broeck, Lykov, Schleich, and Suciu] :)

Future work:

- Other probability distributions?
- When the problem is intractable, can we still approximate/sort the values efficiently?
- Practical implementation of our algorithm

Conclusion

- We prove that the **SHAP-score can be computed in PTIME** for deterministic and decomposable Boolean circuits under product distributions
  - → this generalizes a result of [Lundberg et al., 2020]
- We show that computing SHAP-scores is always as hard as the model counting problem
  - → computing SHAP-score is actually PTIME-equivalent to the problem of computing expectations! Check out the [AAA'21 paper by Van den Broeck, Lykov, Schleich, and Suciu] :)

Future work:

- Other probability distributions?
- When the problem is intractable, can we still approximate/sort the values efficiently?
- Practical implementation of our algorithm

Thanks for your attention!

Scott M Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee.
From local explanations to global understanding with explainable ai for trees.
Nature machine intelligence, 2(1):2522–5839, 2020.

Lloyd S Shapley.
A value for n-person games.
Contributions to the Theory of Games, 2(28):307–317, 1953.

Guy Van den Broeck, Anton Lykov, Maximilian Schleich, and Dan Suciu.
On the tractability of shap explanations.
arXiv preprint arXiv:2009.08634, 2020.