Shapley Values for Databases and Machine Learning

Mikaël Monet

VALDA seminar, February 25th 2022

Slides available online at mikael-monet.net/en/talks.html

Outline

The Shapley value

Shapley values in databases: explaining query results

Shapley values in ML: SHAP-score

The Shapley value

Cooperative games

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

Cooperative games

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

1. Null player: A player p is null if $\mathcal{G}(S \cup \{p\}) = \mathcal{G}(S)$ for every $S \subseteq X$. For every null player we have $s_X(\mathcal{G}, p) = 0$

Cooperative games

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

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

Cooperative games

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

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

Cooperative games

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

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

Theorem [Shapley, 1953]

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

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

Shapley values in databases: explaining query results

My co-authors

This part of the talk is based on joint work with Daniel Deutch, Nave Frost and Benny Kimelfeld.

(Paper accepted at SIGMOD'22)

Shapley values for databases

- Framework introduced by Livshits, Bertossi, Kimelfeld, and Sebag [LBKS'20]
- Let D be a relational database, that we see as a set of *facts* of the form $R(a_1, ..., a_k)$, and q be a Boolean query that takes as input a database D and outputs $q(D) \in \{0, 1\}$.

Shapley values for databases

- Framework introduced by Livshits, Bertossi, Kimelfeld, and Sebag [LBKS'20]
- Let D be a relational database, that we see as a set of *facts* of the form $R(a_1, \dots, a_k)$, and q be a Boolean query that takes as input a database D and outputs $q(D) \in \{0, 1\}$.
- We want to define the "contribution" of every fact $f \in D$ for the (non-)satisfaction of q. We use the Shapley value where the players are the facts of D and the game maps $S \subseteq D$ to $q(S) \in \{0, 1\}$

$$\text{Shapley}(q, D, f) \stackrel{\text{def}}{=} \sum_{S \subseteq D \setminus \{f\}} \frac{|S|!(|D| - |S| - 1)!}{|D|!} \big(q(S \cup \{f\}) - q(S) \big)$$

Complexity?

When can it be computed efficiently?

Definition: problem Shapley(q)

Input: A database D and a fact $f \in D$

Output: The value Shapley(q, D, f)

Complexity?

When can it be computed efficiently?

Definition: problem Shapley(q)

Input: A database D and a fact $f \in D$

Output: The value Shapley(q, D, f)

We consider the data complexity (query q is *fixed*)

Theorem [LBKS'20]

Let q be a self-join–free conjunctive query. If q is hierarchical then Shapley(q) is PTIME, otherwise it is $\mathrm{FP}^{\#\mathrm{P}}$-hard

Link to probabilistic databases?

Theorem [LBKS'20]

Let q be a self-join–free conjunctive query. If q is hierarchical then Shapley(q) is PTIME, otherwise it is $\mathrm{FP}^{\#\mathrm{P}}$-hard

This is the same dichotomy as for probabilistic query evaluation...

Is there a more general connection?

Link to probabilistic databases?

Theorem [LBKS'20]

Let q be a self-join–free conjunctive query. If q is hierarchical then Shapley(q) is PTIME, otherwise it is $\mathrm{FP}^{\#\mathrm{P}}$-hard

This is the same dichotomy as for probabilistic query evaluation...
Is there a more general connection?

Answer: yes, we show that Shapley(q) reduces to probabilistic query evaluation, for every Boolean query q

Tuple-independent probabilistic database (TID)

Probabilistic databases

Tuple-independent probabilistic database (TID)

D =

WorksAt		π
Bob	Inria	0.9
Alice	CNRS	0.5
John	ENS	0.7
Mary	Inria	0.2

Probabilistic databases

Tuple-independent probabilistic database (TID)

$D' =$

WorksAt		π
~~Bob~~	~~Inria~~	~~0.9~~
Alice	CNRS	0.5
~~John~~	~~ENS~~	~~0.7~~
Mary	Inria	0.2

Probabilistic databases

Tuple-independent probabilistic database (TID)

$D' =$

WorksAt		π
~~Bob~~	~~Inria~~	~~0.9~~
Alice	CNRS	0.5
~~John~~	~~ENS~~	~~0.7~~
Mary	Inria	0.2

$\Pr(D') = (1 - 0.9) \times 0.5 \times (1 - 0.7) \times 0.2$

Probabilistic databases

Tuple-independent probabilistic database (TID)

D =

WorksAt		π
Bob	Inria	0.9
Alice	CNRS	0.5
John	ENS	0.7
Mary	Inria	0.2

q = « there are two people who work at the same institution »

Probabilistic databases

Tuple-independent probabilistic database (TID)

D =

WorksAt		π
Bob	Inria	0.9
Alice	CNRS	0.5
John	ENS	0.7
Mary	Inria	0.2

q = « there are two people who work at the same institution »

$$\Pr((D,\pi) \models q) = \sum_{\substack{D' \subseteq D \\ D' \models q}} \Pr(D')$$

$\mathrm{PQE}(q)$ and Shapley(q)

Definition: problem $\mathrm{PQE}(q)$

Input: A tuple-independent database (D, π)

Output: The probability $\Pr((D, \pi) \models q)$ that (D, π) satisfies q

$\mathrm{PQE}(q)$ and Shapley(q)

Definition: problem $\mathrm{PQE}(q)$

Input: A tuple-independent database (D, π)

Output: The probability $\Pr((D, \pi) \models q)$ that (D, π) satisfies q

Theorem (ours)

For every Boolean query q, Shapley(q) reduces in PTIME to $\mathrm{PQE}(q)$

→ In particular, this implies that Shapley(q) is PTIME whenever $\mathrm{PQE}(q)$ is PTIME (and we know a lot about this)

Next: proof of this result

Reduction from $\mathrm{Shapley}(q)$ to $\mathrm{PQE}(q)$ (1/4)

We wish to compute Shapley$(q, D, f) \stackrel{\text{def}}{=}$

$$\sum_{S \subseteq D \setminus \{f\}} \frac{|S|!(|D| - |S| - 1)!}{|D|!} \big(q(S \cup \{f\}) - q(S)\big).$$

Reduction from Shapley(q) to PQE(q) (1/4)

We wish to compute Shapley$(q, D, f) \stackrel{\text{def}}{=}$

$$\sum_{S \subseteq D \setminus \{f\}} \frac{|S|!(|D| - |S| - 1)!}{|D|!}\big(q(S \cup \{f\}) - q(S)\big).$$

For an integer $k \in \{0, \ldots, |D|\}$, define

$$\#\mathrm{Slices}(q, D, k) \stackrel{\text{def}}{=} |\{S \subseteq D \mid |S| = k \text{ and } q(S) = 1\}|$$

Reduction from $\mathrm{Shapley}(q)$ to $\mathrm{PQE}(q)$ (1/4)

We wish to compute $\mathrm{Shapley}(q, D, f) \stackrel{\text{def}}{=}$

$$\sum_{S \subseteq D \setminus \{f\}} \frac{|S|!(|D| - |S| - 1)!}{|D|!} \big(q(S \cup \{f\}) - q(S) \big).$$

For an integer $k \in \{0, \ldots, |D|\}$, define

$$\#\mathrm{Slices}(q, D, k) \stackrel{\text{def}}{=} |\{S \subseteq D \mid |S| = k \text{ and } q(S) = 1\}|$$

Regroup the terms by size to obtain $\mathrm{SHAP}(q, D, f) =$

$$\sum_{k=0}^{|D|-1} \frac{k!(|D| - k - 1)}{|D|} \Big(\#\mathrm{Slices}(q_{+f}, D \setminus \{f\}, k) - \#\mathrm{Slices}(q_{-f}, D \setminus \{f\}, k) \Big)$$

In other words, $\mathrm{Shapley}(q)$ reduces to the problem of computing $\#\mathrm{Slices}(q)$, so it suffices to reduce $\#\mathrm{Slices}(q)$ to $\mathrm{PQE}(q)$

Reduction from $\mathrm{Shapley}(q)$ to $\mathrm{PQE}(q)$ (2/4)

We wish to compute $\#\mathrm{Slices}(q, D, k) \stackrel{\mathrm{def}}{=}$

$$|\{S \subseteq D \mid |S| = k \text{ and } q(S) = 1\}|$$

Reduction from $\mathrm{Shapley}(q)$ to $\mathrm{PQE}(q)$ (2/4)

We wish to compute $\#\mathrm{Slices}(q, D, k) \stackrel{\mathrm{def}}{=}$

$$|\{S \subseteq D \mid |S| = k \text{ and } q(S) = 1\}|$$

For $z \in \mathbb{Q}$, we define a TID database (D_z, π_z) as follows: D_z contains all the facts of D, and for a fact f of D we define $\pi_z(f) \stackrel{\mathrm{def}}{=} \frac{z}{1+z}$.

Reduction from Shapley(q) to PQE(q) (2/4)

We wish to compute $\#\mathrm{Slices}(q, D, k) \stackrel{\text{def}}{=}$

$$\left|\{S \subseteq D \mid |S| = k \text{ and } q(S) = 1\}\right|$$

For $z \in \mathbb{Q}$, we define a TID database (D_z, π_z) as follows: D_z contains all the facts of D, and for a fact f of D we define $\pi_z(f) \stackrel{\text{def}}{=} \frac{z}{1+z}$. Then:

$$\Pr(q, (D_z, \pi_z)) \stackrel{\text{def}}{=} \sum_{S \subseteq D_z \text{ s.t. } q(S)=1} \Pr(S)$$

$$= \sum_{i=0}^{n \stackrel{\text{def}}{=} |D|} \sum_{\substack{S \subseteq S \text{ s.t.} \\ |S|=i \text{ and } q(S)=1}} \Pr(S)$$

Reduction from Shapley(q) to $\mathrm{PQE}(q)$ (3/4)

$$
\begin{aligned}
\Pr(q,(D_z,\pi_z)) &= \sum_{i=0}^{n} \sum_{\substack{S\subseteq D \text{ s.t.}\\ |S|=i \text{ and } q(S)=1}} \Pr(S) \\
&= \sum_{i=0}^{n} \sum_{\substack{S\subseteq S \text{ s.t.}\\ |S|=i \text{ and } q(S)=1}} \left(\frac{z}{1+z}\right)^{i} \left(1-\frac{z}{1+z}\right)^{n-i} \\
&= \sum_{i=0}^{n} \left(\frac{z}{1+z}\right)^{i} \left(\frac{1}{1+z}\right)^{n-i} \sum_{\substack{S\subseteq S \text{ s.t.}\\ |S|=i \text{ and } q(S)=1}} 1 \\
&= \frac{1}{(1+z)^n} \sum_{i=0}^{n} z^i \#\mathrm{Slices}(q,D,i)
\end{aligned}
$$

Reduction from $\text{Shapley}(q)$ to $\mathrm{PQE}(q)$ (3/4)

Hence we have

$$(1+z)^n \Pr(q, (D_z, \pi_z)) = \sum_{i=0}^{n} z^i \, \#\mathrm{Slices}(q, D, i).$$

This suffices to conclude. Indeed, we now call an oracle to $\mathrm{PQE}(q)$ on $n+1$ databases $D_{z_0}, \ldots, D_{z_n}$ for $n+1$ arbitrary distinct values $z_0, \ldots, z_n$, forming a system of linear equations as given by the relation above. Since the corresponding matrix is a Vandermonde with distinct coefficients, it is invertible, so we can compute in polynomial time the value $\#\mathrm{Slices}(q, D, k)$.

So $\text{Shapley}(q)$ reduces in PTIME to $\mathrm{PQE}(q)$

Open problem

Do we have the other direction? We don't know

Open problem

For every Boolean query q, is it the case that $\mathrm{PQE}(q)$ reduces in PTIME to Shapley(q)?

Using provenance and knowledge compilation to solve Shapley(q) (1/2)

- An approach to probabilistic query evaluation: compute the provenance of the query q on the database D in a formalism from knowledge compilation, and then use this representation to compute the probability

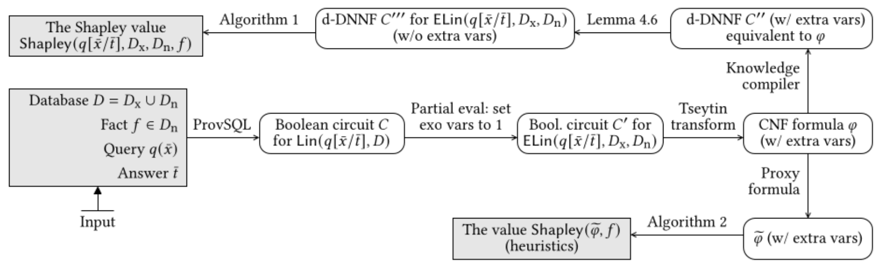
→ We can do the same for computing Shapley values

Proposition (ours)

Given as input a deterministic and decomposable circuit C representing the provenance, we can compute in time $O(|C| \cdot |D|^2)$ the value $\mathrm{SHAP}(q, D, f)$.

Using provenance and knowledge compilation to solve Shapley(q) (2/2)

Implementation, experiments on TPC-H and IMDB datasets.



Shapley values in ML: SHAP-score

My co-authors

This part of the talk is based on the preprint “On the Complexity of SHAP-Score-Based Explanations: Tractability via Knowledge Compilation and Non-Approximability Results” [Arxiv] with Marcelo Arenas, Pablo Barceló, and Leopoldo Bertossi
(Conference version at AAAI’21)

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.

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 $\mathcal{G}_{\mathsf{e}}$:

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

SHAP-score for explainable AI

Let X be a set of features, e an entity (that has a value $\mathrm{e}(x)$ 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.

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

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

In other words,

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

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

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.

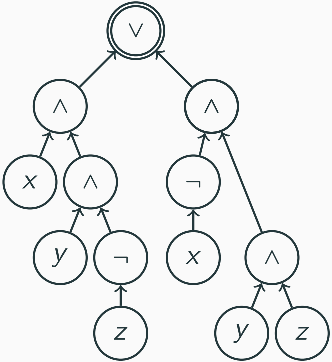
... and the tasks that these allow to solve efficiently

→ 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

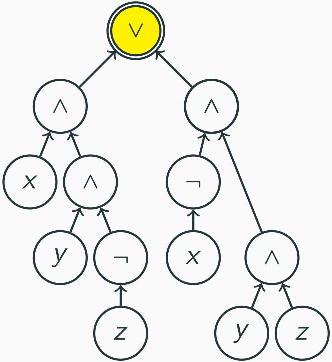
Deterministic and decomposable Boolean circuits

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



Deterministic and decomposable Boolean circuits

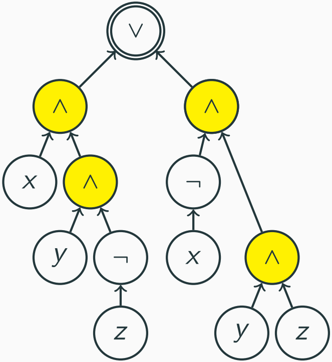
(also called "**tractable Boolean circuits**")



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

Deterministic and decomposable Boolean circuits

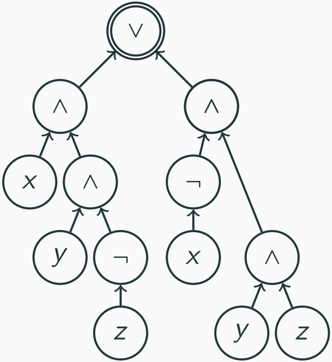
(also called "**tractable Boolean circuits**")



- 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

- 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 $\mathrm{SHAP}_{\mathcal{D}}(M, \mathsf{e}, x)$ in time $O(|M| \cdot |X|^2)$

Proof sketch of main result (1/3)

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

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

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

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

- **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 $\ell \in \{0, \dots, |\mathrm{var}(g)|\}$, define the value

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

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 $O(|M| \cdot |X|^2)$

Reduction from computing expectations

Computing expectations problem for a class $\mathcal{C}$: Given as input a model $M \in \mathcal{C}$ and independent probability values on the features, what is the expected value of M?

Reduction (folklore)

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

→ (One application of the efficiency axiom. Notice the difference with the open problem on Shapley(q))

Reduction from computing expectations

Computing expectations problem for a class $\mathcal{C}$: Given as input a model $M \in \mathcal{C}$ and independent probability values on the features, what is the expected value of M?

Reduction (folklore)

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

$\rightarrow$ (One application of the efficiency axiom. Notice the difference with the open problem on Shapley(q))

$\Longrightarrow$ Computing SHAP-score is #P-hard for CNF or DNF formulas, for instance

Reduction from computing expectations

Computing expectations problem for a class $\mathcal{C}$: Given as input a model $M \in \mathcal{C}$ and independent probability values on the features, what is the expected value of M?

Reduction (folklore)

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

$\rightarrow$ (One application of the efficiency axiom. Notice the difference with the open problem on Shapley(q))

$\Longrightarrow$ Computing SHAP-score is #P-hard for CNF or DNF formulas, for instance

- When a problem is hard, try to approximate it
- We will use the notion of **Fully Polynomial-time Randomized Approximation Scheme** (FPRAS).

FPRAS

Let Σ be a finite alphabet and $f : \Sigma^* \to \mathbb{R}$ be a problem. Then f is said to have an FPRAS if there is a randomized algorithm $\mathcal{A} : \Sigma^* \times (0,1) \to \mathbb{N}$ and a polynomial $p(u,v)$ such that, given $x \in \Sigma^*$ and $\epsilon \in (0,1)$, algorithm $\mathcal{A}$ runs in time $p(|x|, 1/\epsilon)$ and satisfies the following condition:

$$\Pr\big(|f(x) - \mathcal{A}(x,\epsilon)| \le \epsilon f(x)\big) \;\; \ge \;\; \frac{3}{4}.$$

- Example: model counting for DNF formulas has a FPRAS [KLM89]

No FPRAS for DNFs

Lemma

Computing the SHAP-score for models given as monotone DNF formulas has no FPRAS unless NP=RP

This is in contrast to model counting (computing expectaions) for DNFs which has a FPRAS!

No FPRAS for DNFs

Lemma

Computing the SHAP-score for models given as monotone DNF formulas has no FPRAS unless NP=RP

This is in contrast to model counting (computing expectaions) for DNFs which has a FPRAS!

- (We did not identify a class of models for which computing the SHAP-score is intractable but where it can be approximated)

Thanks for your attention!

Bibliography i

Marcelo Arenas, Pablo Barceló, Leopoldo E. Bertossi, and Mikaël Monet.
On the complexity of shap-score-based explanations: Tractability via knowledge compilation and non-approximability results.

Richard M Karp, Michael Luby, and Neal Madras.
Monte-carlo approximation algorithms for enumeration problems.
Journal of algorithms, 10(3):429–448, 1989.

Ester Livshits, Leopoldo E. Bertossi, Benny Kimelfeld, and Moshe Sebag.
The shapley value of tuples in query answering.
In *ICDT*, volume 155, pages 20:1–20:19. Schloss Dagstuhl, 2020.

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.