Shapley Values for Relational Databases

Mikaël Monet

ENS Paris-Saclay visit at Cristal, December 6th 2021

L'équipe LINKS (Linking Dynamic Data)

Joint team between Inria Lille, university of Lille, and the CNRS CRIStAL lab. Members :

- 9 permanent members (1 *directeur de recherche*, 2 *professeurs*, 5 *maîtres de conférence*, 1 *chargé de recherche*)
- 5 PhD students
- 1 research engineer

Research themes

- Store, query, update, integrate heterogeneous data...
  - → relational databases, graph databases, RDF, hybrid formats, etc.
- that can be linked and constrained...
  - → *schema mappings*, integrity constraints, *ontologies*, etc.
- that is potentially voluminous...
  - → "big data", *streaming* algorithms, usage of RDBMS for graphs, etc.
- and can also contain uncertainty
  - → databases with missing values, *probabilistic* databases

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 reasonnable 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 reasonnable axioms:

1. 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 $s_X(\mathcal{G}, x) = 0$

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 reasonnable axioms:

1. 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 $s_X(\mathcal{G}, x) = 0$
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 reasonnable axioms:

1. 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 $s_X(\mathcal{G}, x) = 0$
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 reasonnable axioms:

1. 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 $s_X(\mathcal{G}, x) = 0$
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 $\sum_{p \in X} s_X(\mathcal{G}, p) = \mathcal{G}(X)$

The Shapley value

Theorem [Shapley, 1953]

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

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

Shapley values in databases: explaining query results

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 $R(a_1, ..., a_k)$, 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

Probabilistic databases

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 $\Pr((D, \pi) \vDash q)$ 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 $\mathsf{Shapley}(q, D, f) \stackrel{\text{def}}{=}$

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

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

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

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

For an integer $k \in \{0, \dots, |D|\}$, 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 $\mathrm{Shapley}(q, D, f) \stackrel{\mathrm{def}}{=}$

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

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

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

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

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

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 $\text{Shapley}(q)$ to $\text{PQE}(q)$ (2/4)

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

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

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

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

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

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 $\pi_z(f) \stackrel{\text{def}}{=} \frac{z}{1+z}$.

Reduction from $\mathrm{Shapley}(q)$ to $\mathrm{PQE}(q)$ (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 $\text{Shapley}(q)$ 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 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 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.

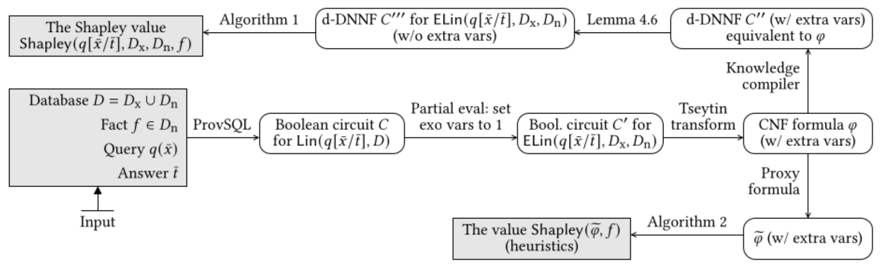
→ We can do the same for computing Shapley values

Proposition (ours)

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

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

Implementation, experiments on TPC-H and IMDB datasets.



The End

- Thanks for your attention!
- (Contact us for research internships)

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.

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