Shapley Value $\rightleftarrows$ Model Counting

Ahmet Kara (OTH Regensburg)

Joint work with Dan Olteanu and Dan Suciu

AIMoTh'26, Passau

Shapley Value

- The Shapley value quantifies the fair contribution of a player to a wealth function shared by a set of players in a cooperative game
- It was introduced by Lloyd Shapley in 1951

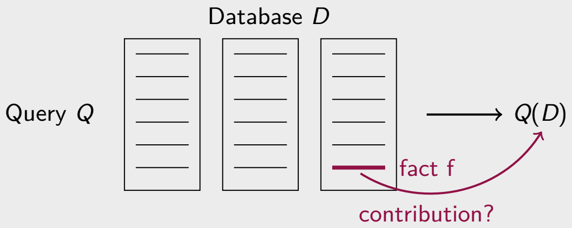
Some applications of the Shapley Value:

- Measuring the importance of features in machine learning
- Measuring the centrality and power of genes
- Finding key players in social networks
- Explaining query results

Explaining Query Results Using Shapley Value

Livshits et al. quantify the contribution of database facts to query results using the Shapley value

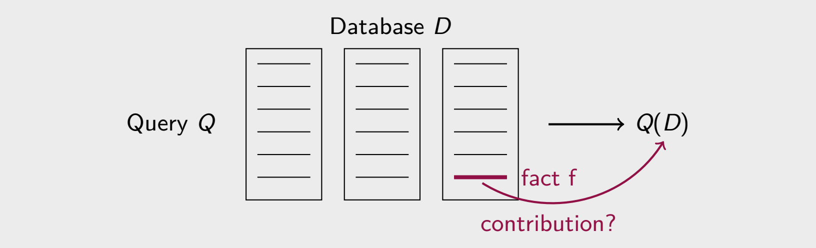
Problem Setting



Explaining Query Results Using Shapley Value

Livshits et al. quantify the contribution of database facts to query results using the Shapley value

Problem Setting



High-level recipe of computing the Shapley value of a database fact f:

- Interpret database facts as players and sub-databases as coalitions
- Evaluate Q on sub-databases $D' \subseteq D \setminus \{f\}$ and $D' \cup \{f\}$ and compare the results

Our work is about the complexity of this computation

Explaining Query Results Using Shapley Value - Prior Results

Theorem [Livshits et al., 21]

Let Q be a self-join-free Boolean query.

- If Q is hierarchical, Shapley value computation is in PTIME.
- Otherwise, Shapley value computation is #P-hard.

- Authors ask: Is there a simple proof based on model counting?

Explaining Query Results Using Shapley Value - Prior Results

Theorem [Livshits et al., 21]

Let Q be a self-join-free Boolean query.

- If Q is hierarchical, Shapley value computation is in PTIME.
- Otherwise, Shapley value computation is #P-hard.

- Authors ask: Is there a simple proof based on model counting?

Theorem [Deutch et. al., 22]

There is a polynomial-time reduction from Shapley value computation to probabilistic query evaluation.

- Authors ask: Are the two problems polynomial-time equivalent?

Explaining Query Results Using Shapley Value - Prior Results

Theorem [Livshits et al., 21]

Let Q be a self-join-free Boolean query.

- If Q is hierarchical, Shapley value computation is in PTIME.
- Otherwise, Shapley value computation is #P-hard.

- Authors ask: Is there a simple proof based on model counting?

Theorem [Deutch et. al., 22]

There is a polynomial-time reduction from Shapley value computation to probabilistic query evaluation.

- Authors ask: Are the two problems polynomial-time equivalent?

Our results give answers to both questions

Overview of Our Work

- We use the Shapley value to quantify the contribution of

  Boolean variables to the models of Boolean functions

- We give polynomial-time reductions between

  computing the Shapley value of Boolean variables

  and

  counting the models of Boolean functions

- We show applications of our result to the computation of Shapley values of database facts in query evaluation

Shapley Value of Boolean Variables

Shapley Value of Boolean Variables - Example

$$F = X \wedge (Y \vee \neg Z)$$

What is the contribution of Y to the models of F?

Shapley Value of Boolean Variables - Example

$$F = X \wedge (Y \vee \neg Z)$$

What is the contribution of Y to the models of F?

permutations of variables
XYZ
XZY
YXZ
YZX
ZYX
ZXY

Shapley Value of Boolean Variables - Example

$$F = X \wedge (Y \vee \neg Z)$$

What is the contribution of Y to the models of F?

permutations of variables	Impact of setting Y to 1 to the Boolean value of F		
XYZ	$F[XY] - F[X]$	$=$	$1 - 1 = 0$
XZY			
YXZ			
YZX			
ZYX			
ZXY			

$F[\boldsymbol{X}]$: Boolean value of F, when only the variables in $\boldsymbol{X}$ are set to 1

Shapley Value of Boolean Variables - Example

$$F = X \wedge (Y \vee \neg Z)$$

What is the contribution of Y to the models of F?

permutations of variables	Impact of setting Y to 1 to the Boolean value of F		
XYZ	$F[XY] - F[X]$	$=$	$1 - 1 = 0$
XZY	$F[XZY] - F[XZ]$	$=$	$1 - 0 = 1$
YXZ	$F[Y] - F[\emptyset]$	$=$	$0 - 0 = 0$
YZX	$F[Y] - F[\emptyset]$	$=$	$0 - 0 = 0$
ZYX	$F[ZY] - F[Z]$	$=$	$0 - 0 = 0$
ZXY	$F[ZXY] - F[ZX]$	$=$	$1 - 0 = 1$

$F[\boldsymbol{X}]$: Boolean value of F, when only the variables in $\boldsymbol{X}$ are set to 1

Shapley Value of Boolean Variables - Example

$$F = X \wedge (Y \vee \neg Z)$$

What is the contribution of Y to the models of F?

permutations of variables	Impact of setting Y to 1 to the Boolean value of F		
XYZ	$F[XY] - F[X]$	$=$	$1 - 1 = 0$
XZY	$F[XZY] - F[XZ]$	$=$	$1 - 0 = 1$
YXZ	$F[Y] - F[\emptyset]$	$=$	$0 - 0 = 0$
YZX	$F[Y] - F[\emptyset]$	$=$	$0 - 0 = 0$
ZYX	$F[ZY] - F[Z]$	$=$	$0 - 0 = 0$
ZXY	$F[ZXY] - F[ZX]$	$=$	$1 - 0 = 1$

sum $= 2$

$F[\boldsymbol{X}]$: Boolean value of F, when only the variables in $\boldsymbol{X}$ are set to 1

Shapley Value of Boolean Variables - Example

$$F = X \wedge (Y \vee \neg Z)$$

What is the contribution of Y to the models of F?

$$\begin{array}{lllll}
\text{permutations} & \text{Impact of setting } Y \text{ to } 1 & & & \\
\text{of variables} & \text{to the Boolean value of } F & & & \\
\hline
XYZ & F[XY] - F[X] & = & 1 - 1 = 0 & \\
XZY & F[XZY] - F[XZ] & = & 1 - 0 = 1 & \\
YXZ & F[Y] - F[\emptyset] & = & 0 - 0 = 0 & \\
YZX & F[Y] - F[\emptyset] & = & 0 - 0 = 0 & \\
ZYX & F[ZY] - F[Z] & = & 0 - 0 = 0 & \\
ZXY & F[ZXY] - F[ZX] & = & 1 - 0 = 1 &
\end{array} \left.\vphantom{\begin{array}{l}1\\1\\1\\1\\1\\1\end{array}}\right\} \text{sum} = 2$$

$F[\boldsymbol{X}]$: Boolean value of F, when only the variables in $\boldsymbol{X}$ are set to 1

Shapley value of Y in F: $Shap(Y, F) = \frac{\text{sum of contributions of } F}{\text{number of permutations}} = \frac{2}{6}$

Shapley Value of Boolean Variables

- Let F be a Boolean function with variables $X_1, \ldots, X_n$
- S_n: Set of all permutations of $X_1, \ldots, X_n$
- $\Pi^{<i}$: set of variables that appear before X_i in the permutation Π

Shapley value of variable X_i in F

$$Shap(X_i, F) = \frac{1}{n!} \sum_{\Pi \in S_n} (F[\Pi^{<i} \cup \{X_i\}] - F[\Pi^{<i}])$$

Problems and Main Results

Computational Problems

Let $\mathcal{C}$ be a class of Boolean functions

Shap($\mathcal{C}$) (Shapley Computation)

Given:	$F \in \mathcal{C}$ over variable set $\boldsymbol{X}$
Compute:	$Shap(X, F)$, $\forall X \in \boldsymbol{X}$

$\#\mathcal{C}$ (Model Counting)

Given:	$F \in \mathcal{C}$ over variable set $\boldsymbol{X}$
Compute:	$\sum_{\boldsymbol{Y} \subseteq \boldsymbol{X}} F[\boldsymbol{Y}]$ (number of models of F)

$\#_*\mathcal{C}$ (k-Model Counting)

Given:	$F \in \mathcal{C}$ over variable set $\boldsymbol{X}$
Compute:	$\sum_{\substack{\boldsymbol{Y} \subseteq \boldsymbol{X} \\ \lvert\boldsymbol{Y}\rvert = k}} F[\boldsymbol{Y}]$, $\forall k \leq \lvert\boldsymbol{X}\rvert$ (number of models of F of size k)

OR-Substitutions

$F \xrightarrow{\text{OR}} G$ (function F OR-substitutes into function G) if:

- G results from F by substituting each variable by a (possibly empty) disjunction of fresh variables.

Example

$$X \wedge (Y \vee \neg Z) \xrightarrow{\text{OR}} X^1 \wedge (\bot \vee \neg(Z^1 \vee Z^2 \vee Z^3))$$

$$\widetilde{\mathcal{C}} \stackrel{\text{def}}{=} \{G \mid \exists F \in \mathcal{C} \text{ with } F \xrightarrow{\text{OR}} G\}$$

Our Main Result

Theorem

For any class $\mathcal{C}$ of Boolean functions, the following polynomial-time reductions hold:

$$\begin{aligned} \mathit{Shap}(\mathcal{C}) &\leq^P \#_*\widetilde{\mathcal{C}} \\ \#_*\mathcal{C} &\leq^P \#\widetilde{\mathcal{C}} \\ \#\mathcal{C} &\leq^P \mathit{Shap}(\widetilde{\mathcal{C}}) \end{aligned}$$

Corollary

If $\widetilde{\mathcal{C}} = \mathcal{C}$, then $\mathit{Shap}(\mathcal{C}) \equiv^P \#_*\mathcal{C} \equiv^P \#\mathcal{C}$

Implications of Our Main Result

For the following classes $\mathcal{C}$, it holds $\widetilde{\mathcal{C}} = \mathcal{C}$ and $\#\mathcal{C}$ is in PTIME:

- Positive β-acyclic CNF-formulas
- Deterministic and decomposable Boolean circuits

For each such class $\mathcal{C}$, our result implies:

- *Shap*($\mathcal{C}$) is in PTIME.

Application:
Explaining Query Results

Query Lineage

- The lineage of a Boolean query over a database is a propositional DNF formula over variables associated with database facts
- The lineage contains one clause for each tuple of joining facts

Example

Query $Q = R(A, B), S(B, C)$

Database D

R	A	B	
	a_1	b_1	X_1
	a_2	b_1	X_2
	a_1	b_2	X_3

S	B	C	
	b_1	c_1	Y_1
	b_2	c_1	Y_2

lineage variables

Lineage of Q over D: $F_{Q,D} = (X_1 \wedge Y_1) \vee (X_2 \wedge Y_1) \vee (X_3 \wedge Y_2)$

$$\mathcal{C}_Q = \{F_{Q,D} \mid D \text{ is a database}\}$$

Query Lineage

- The lineage of a Boolean query over a database is a propositional DNF formula over variables associated with database facts
- The lineage contains one clause for each tuple of joining facts

Example

Query $Q = R(A, B), S(B, C)$

Database D

R	A	B	
	a_1	b_1	X_1
	a_2	b_1	X_2
	a_1	b_2	X_3

S	B	C	
	b_1	c_1	Y_1
	b_2	c_1	Y_2

lineage variables

Lineage of Q over D: $F_{Q,D} = (X_1 \wedge Y_1) \vee (X_2 \wedge Y_1) \vee (X_3 \wedge Y_2)$

$$\mathcal{C}_Q = \{F_{Q,D} \mid D \text{ is a database}\}$$

The Shapley value of a lineage variable quantifies the contribution of the corresponding fact to the query result

OR-Substitution of Lineage Expressions

- The stretching of a query is obtained by adding a fresh variable to each atom

Example

Query	$Q =$	$R(A, B),$	$S(B, C)$
Stretching	$\widetilde{Q} =$	$R(Z_1, A, B),$	$S(Z_2, B, C)$

OR-Substitution of Lineage Expressions

- The stretching of a query is obtained by adding a fresh variable to each atom

Example

Query	$Q =$	$R(A, B)$,	$S(B, C)$
Stretching	$\widetilde{Q} =$	$R(Z_1, A, B)$,	$S(Z_2, B, C)$

- OR-substitution of lineage corresponds to stretching of queries

$$\begin{tikzcd} Q \arrow[r, "\text{stretch}"] \arrow[d, "D"'] & \widetilde{Q} \arrow[d, "\widetilde{D}"] \\ F_{Q,D} \arrow[r, "\text{OR-substitute}"'] & \widetilde{F_{Q,D}} = F_{\widetilde{Q},\widetilde{D}} \end{tikzcd}$$

$$\Longrightarrow \mathcal{C}_{\widetilde{Q}} = \widetilde{\mathcal{C}}_Q$$

Back to the Open Questions in Prior Work

Shapley Values of Database Facts 1/2

Theorem [Livshits et al., 21]

Let Q be a self-join-free Boolean query.

- If Q is hierarchical, $\mathit{Shap}(\mathcal{C}_Q)$ is in PTIME
- Otherwise, it is #P-hard.

- Authors ask: Is there a simple proof based on model counting?

Shapley Values of Database Facts 1/2

Theorem [Livshits et al., 21]

Let Q be a self-join-free Boolean query.

- If Q is hierarchical, $\mathit{Shap}(\mathcal{C}_Q)$ is in PTIME
- Otherwise, it is #P-hard.

- Authors ask: Is there a simple proof based on model counting?
- Our answer: Yes, our results allow for a simple proof!
- Main ingredients of the proof:
  - $\mathcal{C}_{\widetilde{Q}} = \widetilde{\mathcal{C}}_Q$
  - $\widetilde{Q}$ is hierarchical for any hierarchical query Q
  - $\#\mathcal{C}_Q$ is in PTIME for hierarchical queries [Olteanu & Huang, 2008]
  - $\#\mathcal{C}_Q$ is #P-hard for non-hierarchical queries [Provan & Ball, 1983]

Shapley Values of Database Facts 2/2

Theorem [Deutch et. al., 22]

For any Boolean query Q:

$$Shap(\mathcal{C}_Q) \leq^P \mathrm{PQE}(Q)$$

PQE(Q): Evaluation of Q over probabilistic databases

- Authors ask: $Shap(\mathcal{C}_Q) \equiv^P \mathrm{PQE}(Q)$?

Shapley Values of Database Facts 2/2

Theorem [Deutch et. al., 22]

For any Boolean query Q:

$$Shap(\mathcal{C}_Q) \leq^P \text{PQE}(Q)$$

PQE(Q): Evaluation of Q over probabilistic databases

- Authors ask: $Shap(\mathcal{C}_Q) \equiv^P \text{PQE}(Q)$?
- Our answer: $Shap(\mathcal{C}_Q) \equiv^P \#\mathcal{C}_{\widetilde{Q}}$

Shapley Computation in Practice

We implemented systems for computing the Shapley value

[SIGMOD'24]

- Shapley (Banzhaf) computation for select-project-join-union queries
  - Exact algorithm
  - Anytime deterministic approximation algorithm
  - Algorithm for ranking and top-k
- Our algorithms significantly outperform algorithms from prior work on real-world datasets
- Lower bound: Shapley-based ranking is tractable only for hierarchical queries

Shapley Computation in Practice

We implemented systems for computing the Shapley value

[SIGMOD'24]

- Shapley (Banzhaf) computation for select-project-join-union queries
  - Exact algorithm
  - Anytime deterministic approximation algorithm
  - Algorithm for ranking and top-k
- Our algorithms significantly outperform algorithms from prior work on real-world datasets
- Lower bound: Shapley-based ranking is tractable only for hierarchical queries

[VLDB'25]

- First practical approach for queries with aggregates

Proof Techniques for the Main Result

Expressing Shapley Value Using k-Model Counting

Lemma [Livshits et al., 21]

The Shapley value of a variable X of a Boolean function F is:

$$Shap(X, F) = \sum_{k=0}^{n-1} c_k \big(\#_k F[X := 1] - \#_k F[X := 0]\big)$$

where $c_k = \frac{k!(n-k-1)!}{n!}$.

Vandermonde Matrices

Definition

Let $x_1, \ldots, x_n$ be scalars. The $n \times n$ Vandermonde matrix generated by these numbers is:

$$\begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ & & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{pmatrix}$$

Lemma [Golub et al., 96]

Let $x_1, \ldots, x_n$ pairwise distinct scalars. Then, the $n \times n$ Vandermonde matrix generated by these numbers is invertible.

From Shapley Computation to Model Counting 1/2

Lemma

For any class $\mathcal{C}$ of Boolean functions, it holds:

$$\#\mathcal{C} \;\leq^{P} \mathit{Shap}(\widetilde{\mathcal{C}})$$

Given

$F \in \mathcal{C}$

Oracle for $\mathit{Shap}(\widetilde{\mathcal{C}})$

Compute

$\sum_{\boldsymbol{Y} \subseteq \boldsymbol{X}} F[\boldsymbol{Y}]$ (number of models of F)

From Shapley Computation to Model Counting 1/2

Lemma

For any class $\mathcal{C}$ of Boolean functions, it holds:

$$\#\mathcal{C} \ \leq^P \mathit{Shap}(\widetilde{\mathcal{C}})$$

Given

$F \in \mathcal{C}$

Oracle for $\mathit{Shap}(\widetilde{\mathcal{C}})$

Compute

$\sum_{\boldsymbol{Y} \subseteq \boldsymbol{X}} F[\boldsymbol{Y}]$ (number of models of F)

Proof steps:

- Let $F \in \mathcal{C}$ with variables $X_1, \ldots, X_n$
- For $i \in \{1, \ldots, n\}$, let $F^{(\ell,i)} = F[X_p := \bigvee_{j \in \{1,\ldots,\ell\}} X_p^j]_{p \neq i}$
- We have $F^{(\ell,i)} \in \widetilde{\mathcal{C}}$
- Compute $\mathit{Shap}(F^{(\ell,i)}, X_i)$ using oracle for $\mathit{Shap}(\widetilde{\mathcal{C}})$

From Shapley Computation to Model Counting 2/2

- Show:

$$Shap(F^{(\ell,i)}, X_i) = \sum_{k=0}^{n-1} (2^\ell - 1)^k \underbrace{c_k(\#_k F[X_i := 1] - \#_k F[X_i := 0])}_{\Gamma_k F}$$

From Shapley Computation to Model Counting 2/2

- Show:

$$Shap(F^{(\ell,i)}, X_i) = \sum_{k=0}^{n-1} (2^\ell - 1)^k \underbrace{c_k(\#_k F[X_i := 1] - \#_k F[X_i := 0])}_{\Gamma_k F}$$

- Create one equation for each $\ell \in \{1, \ldots, n\}$:

$$\underbrace{\begin{pmatrix} Shap(F^{(1,i)}, X_i) \\ \vdots \\ Shap(F^{(n,i)}, X_i) \end{pmatrix}}_{\text{known}} = \underbrace{\begin{pmatrix} (2^1 - 1)^1 & \cdots & (2^1 - 1)^{n-1} \\ \vdots & \vdots & \vdots \\ (2^{n+1} - 1)^1 & \cdots & (2^{n+1} - 1)^{n-1} \end{pmatrix}}_{\text{Vandermonde}} \underbrace{\begin{pmatrix} \Gamma_0 F \\ \vdots \\ \Gamma_n F \end{pmatrix}}_{\text{unknown}}$$

From Shapley Computation to Model Counting 2/2

- Show:

$$Shap(F^{(\ell,i)}, X_i) = \sum_{k=0}^{n-1} (2^\ell - 1)^k \underbrace{c_k(\#_k F[X_i := 1] - \#_k F[X_i := 0])}_{\Gamma_k F}$$

- Create one equation for each $\ell \in \{1, \ldots, n\}$:

$$\underbrace{\begin{pmatrix} Shap(F^{(1,i)}, X_i) \\ \vdots \\ Shap(F^{(n,i)}, X_i) \end{pmatrix}}_{\text{known}} = \underbrace{\begin{pmatrix} (2^1 - 1)^1 & \cdots & (2^1 - 1)^{n-1} \\ \vdots & \vdots & \vdots \\ (2^{n+1} - 1)^1 & \cdots & (2^{n+1} - 1)^{n-1} \end{pmatrix}}_{\text{Vandermonde}} \underbrace{\begin{pmatrix} \Gamma_0 F \\ \vdots \\ \Gamma_n F \end{pmatrix}}_{\text{unknown}}$$

- Compute all differences $\#_k F[X_i := 1] - \#_k F[X_i := 0])$

From Shapley Computation to Model Counting 2/2

- Show:

$$Shap(F^{(\ell,i)}, X_i) = \sum_{k=0}^{n-1} (2^\ell - 1)^k \underbrace{c_k(\#_k F[X_i := 1] - \#_k F[X_i := 0])}_{\Gamma_k F}$$

- Create one equation for each $\ell \in \{1, \ldots, n\}$:

$$\underbrace{\begin{pmatrix} Shap(F^{(1,i)}, X_i) \\ \vdots \\ Shap(F^{(n,i)}, X_i) \end{pmatrix}}_{\text{known}} = \underbrace{\begin{pmatrix} (2^1-1)^1 & \cdots & (2^1-1)^{n-1} \\ \vdots & \vdots & \vdots \\ (2^{n+1}-1)^1 & \cdots & (2^{n+1}-1)^{n-1} \end{pmatrix}}_{\text{Vandermonde}} \underbrace{\begin{pmatrix} \Gamma_0 F \\ \vdots \\ \Gamma_n F \end{pmatrix}}_{\text{unknown}}$$

- Compute all differences $\#_k F[X_i := 1] - \#_k F[X_i := 0])$
- Show

$$\sum_{i=1}^{n} (\#_k F[X_i := 1] - \#_k F[X_i := 0]) = (k+1)\#_{k+1} F - (n-k)\#_k F \quad (*)$$

From Shapley Computation to Model Counting 2/2

- Show:
$$Shap(F^{(\ell,i)}, X_i) = \sum_{k=0}^{n-1} (2^\ell - 1)^k \underbrace{c_k(\#_k F[X_i := 1] - \#_k F[X_i := 0])}_{\Gamma_k F}$$
- Create one equation for each $\ell \in \{1, \ldots, n\}$:
$$\underbrace{\begin{pmatrix} Shap(F^{(1,i)}, X_i) \\ \vdots \\ Shap(F^{(n,i)}, X_i) \end{pmatrix}}_{\text{known}} = \underbrace{\begin{pmatrix} (2^1 - 1)^1 & \cdots & (2^1 - 1)^{n-1} \\ \vdots & \vdots & \vdots \\ (2^{n+1} - 1)^1 & \cdots & (2^{n+1} - 1)^{n-1} \end{pmatrix}}_{\text{Vandermonde}} \underbrace{\begin{pmatrix} \Gamma_0 F \\ \vdots \\ \Gamma_n F \end{pmatrix}}_{\text{unknown}}$$
- Compute all differences $\#_k F[X_i := 1] - \#_k F[X_i := 0])$
- Show
$$\sum_{i=1}^{n} (\#_k F[X_i := 1] - \#_k F[X_i := 0]) = (k+1)\#_{k+1} F - (n-k)\#_k F \qquad (*)$$
- Compute $\#_0 F$ and then inductively $\#_k F$ for $k \in \{1, \ldots, n\}$ using (*)