Computing the Shapley Value of Facts in query Answering

SIGMOD 22 Session 22: Provenance and Uncertainty

By 钟瑞峥 林俊光

May 10, 2023

Presentation Overview

- **Background**
  - Introduction to Shapley Value
  - General Problems
  - Problems remained in ICDT 2020
- **Solution**
  - Overview
  - Theoretical Analysis
  - Exact Computation
  - Inexact Computation
- **Conclusion**
- **References**

Background - Shapley Value

Shapley value is a game-theoretic notion for wealth distribution that is nowadays extensively used to explain complex data-intensive computation.

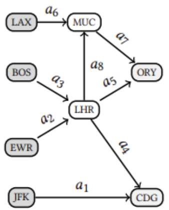
$$\text{Shapley}\left(q, D_{\mathrm{n}}, D_{\mathrm{x}}, f\right) \stackrel{\text{def}}{=} \sum_{E \subseteq D_{\mathrm{n}} \setminus \{f\}} \frac{|E|!\left(|D_{\mathrm{n}}| - |E| - 1\right)!}{|D_{\mathrm{n}}|!} \left(q\left(D_{\mathrm{x}} \cup E \cup \{f\}\right) - q\left(D_{\mathrm{x}} \cup E\right)\right)$$

- Research show that query evaluation over relational databases fits well in this explanation paradigm.
- Airport problem

	Flights (endo)	
	Src	**Dest**
a_1	JFK	CDG
a_2	EWR	LHR
a_3	BOS	LHR
a_4	LHR	CDG
a_5	LHR	ORY
a_6	LAX	MUC
a_7	MUC	ORY
a_8	LHR	MUC

	Airports (exo)	
	Name	**Country**
b_1	JFK	USA
b_2	EWR	USA
b_3	BOS	USA
b_4	LAX	USA
b_5	LHR	EN
b_6	MUC	GR
b_7	ORY	FR
b_8	CDG	FR

(a) Database of flights and airports



(b) Flights in graph view. Dark and light gray depict "USA" and "FR" airports respectively

Background - Problems in Calculating Shapley Value

- Calculation of the shapley value is NP-hard in general
- The number of possible coalitions is exponential in the number of facts

ICDT 2020

- Showed mainly lower bounds on the complexity of computing the Shapley value of facts in query answering
  - gives polynomial-time algorithm for self-join-free conjunctive queries
- Need a large number of executions of the query over database subsets
- Does not provide sufficient evidence

Solution - Overview

Two Approaches

- Exact Computation
  - Capture the dependence of query answers using boolean expressions
  - Transform it to a d-DNNF circuit form in which we devise an algorithm for computing shapley value
  - May be too costly in certain circumstances

Note: Given a d-DNNF circuit, we can compute the shapley value in polynomial time with the help of c2d and Provsql

- Inexact Computation
  - Not necessarily compute exact Shapley values, determine the order of facts according to their Shapley values
  - Faster yet inexact approach that transforms it into CNF Proxy

d-DNNF(Decomposable Deterministic Negation Normal Form)

NNF Definition

- A formula is in NNF if negation only appears in literals

$(A \vee B) \wedge C$

$\neg(A \vee \neg C)$

DNNF Definition

- A formula in NNF is in DNNF form if the decompositional property holds

$(A \wedge B) \vee (A \wedge ((\neg B \vee E) \wedge F)$

d-DNNF(Decomposable Deterministic Negation Normal Form)

d-DNNF Definition

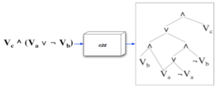
- A DNNF is called deterministic if operands of a disjunction do not share models

$$(A \wedge B) \vee (A \wedge ((\neg B \vee E) \wedge F)$$

This is not in d-DNNF form as two disjunction operands share the model

A = 1, B = 1, E = 1, F = 1

The c2d Compiler



Description

c2d is a system that compiles CNF into d-DNNF (deterministic, decomposable negation normal form). d-DNNF is a tractable logical form that allows operations such as model counting, clausal entailment, and mo
enumeration and minimization to be performed in linear time. d-DNNF is also a strict superset of OBDD and is strictly more succinct than OBDD.

Read the paper that describes the c2d compiler.
Read the paper that describes d-DNNF and related languages.
See results of using c2d on some benchmarks.

Download

Download c2d and instructions for using it.

Related Software

miniC2D: A software package for knowledge compilation and model counting based on exhaustive DPLL.

Ace: A system for compiling Bayesian networks which is based on c2d.

The SDD Package: A system for constructing, manipulating, and optimizing Sentential Decision Diagrams (SDDs).

B+E: A CNF preprocessor. See results on c2d in:
J.-M. Lagniez, E. Lonca, P. Marquis. "Improving Model Counting by Leveraging Definability". Proceedings of IJCAI'16, pages 751-757. paper

Contact

Send questions and comments to darwiche at cs.ucla.edu.

Solution - Overview

Shapley value can be computed in polynomial time whenever the query can be evaluated in polynomial time over tuple-independent probablistic databases.

- Algortihm 1: compiling to a deterministic and decomposable circuit
- Algorithm 2: resort to CNF proxy which is fast yet inexact if timeout reaches
- Hybrid approach: Algorithm 1 + Algorithm 2

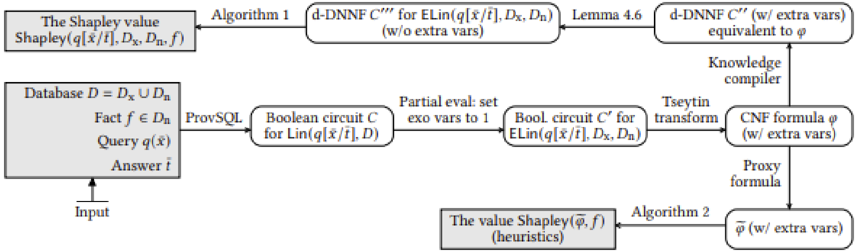


Figure 3: Our implementation architecture.

Note: For non-boolean queries, we are intereseted in the Shapley Value of the fact f for every individual tuple in the output. Therefore, the computational challenge reduces to that of the boolean queries.

Tseytin Transformation

- Knowledge Compiler usually takes boolean formulas in CNF form, and not arbitrary boolean circuits.
- Tseytin transformation is a method for converting a boolean circuit into an equisatisfiable CNF formula.
  $((p \vee q) \wedge r) \rightarrow (\neg s)$

  Consider all subformulas(excluding simple variables)

$\neg s$

$p \vee q$

$(p \vee q) \wedge r$

$((p \vee q) \wedge r) \rightarrow (\neg s)$

$x_1 \leftrightarrow \neg s$

$x_2 \leftrightarrow p \vee q$

$x_3 \leftrightarrow x_2 \wedge r$

$x_4 \leftrightarrow x_3 \rightarrow x_1$

All substitutions can be transformed into CNF form

$x_2 \leftrightarrow p \vee q \equiv (\neg x_2 \vee p \vee q) \wedge (\neg p \vee x_2) \wedge (\neg q \vee x_2)$

Theoretical Analysis - Reduction

PROBLEM:	Shapley(q)
INPUT:	A database $D = D_x \cup D_n$ and an endogenous fact $f \in D_n$.
OUTPUT:	The value Shapley(q, D_n, D_x, f).

PROBLEM:	PQE(q)
INPUT:	A tuple-independent database (D, π).
OUTPUT:	The value $\Pr(q, (D, \pi))$.

For every self-join free boolean queries, either q is hierarchical and shapley(q) can be solved in polynomial time, or q is not hierarchical and shapley(q) is intractable

Proposition: For every Boolean query q, we have that Shapley $(q) \leq_{\mathrm{T}}^{\mathrm{p}} \mathrm{PQE}(q)$

Proof:

$$\#\,\text{Slices}\,(q, D_{\mathrm{x}}, D_{\mathrm{n}}, k) \stackrel{\text{def}}{=} |\,\{E \subseteq D_{\mathrm{n}} \,|\, |E| = k \text{ and } q\,(D_{\mathrm{x}} \cup E) = 1\}\,|.$$

$$\text{Shapley}\,(q, D_{\mathrm{n}}, D_{\mathrm{x}}, f) =$$

$$\sum_{k=0}^{|D_{\mathrm{n}}|-1} \frac{k!\,(|D_{\mathrm{n}}| - k - 1)!}{|D_{\mathrm{n}}|!} (\#\,\text{Slices}\,(q, D_{\mathrm{x}} \cup \{f\}, D_{\mathrm{n}} \backslash \{f\}, k) \quad - \#\,\text{Slices}\,(q, D_{\mathrm{x}}, D_{\mathrm{n}} \backslash \{f\}, k))\,.$$

Arithmetic terms can be computed in polynomial time

Theoretical Analysis - Reduction

We want to prove that #Slices can be computed in polynomial time

$$(1+z)^n \, \mathbf{Pr}\left(q, (D_z, \pi_z)\right) = \sum_{i=0}^{n} z^i \# \, \mathbf{Slices}\left(q, D_{\mathrm{x}}, D_{\mathrm{n}}, i\right)$$

We can now call an oracle to PQE(q) on n+1 databases $D_{z_0}, \ldots, D_{z_n}$ for distinct values $z_0 = 0, z_1 = 1, \ldots, z_n = n$.

Exact Computation

Recently, research proved that when the models are given as circuits from knowledge compilation can be computed in polynomial time(SHAP scores).

The same approach can be applied to the computation of Shapley value.

Proposition: Given as input a deterministic and decomposable circuit C representing $\mathrm{ELin}\left(q, D_{\mathrm{n}}, D_{\mathrm{x}}\right)$ for a database $D = D_{\mathrm{X}} \cup D_{\mathrm{n}}$ and Boolean query q, and an endogenous fact $f \in D_{\mathrm{n}}$, we can compute in polynomial time (in $|C|$) the value Shapley $\left(q, D_{\mathrm{n}}, D_{\mathrm{x}}, f\right)$.

We can rewrite the equation as following:

$$\mathrm{Shapley}\left(q, D_{\mathrm{n}}, D_{\mathrm{x}}, f\right) = \sum_{k=0}^{|D_{\mathrm{n}}|-1} \frac{k!\left(|D_{\mathrm{n}}|-k-1\right)}{|D_{\mathrm{n}}|}\left(\#\mathrm{SAT}_{k}\left(C_{1}\right) - \#\mathrm{SAT}_{k}\left(C_{2}\right)\right).$$

Lemma: Given as input a d-DNNF Boolean Circuit C, and an integer k, we can compute in polynomial time the quantity $SAT_k(C)$.

Exact Computation

Proof: Let X = Vars(C) and n = |X| We denote by

ϕ_g the boolean function over the variables Var(g) that is represented by this gate

a_l^g = $\#SAT_l(\phi_g)$ the number of assignments of size l to Vars(g) that satisfy ϕ_g

$$\mathrm{SAT}_k(\varphi) \stackrel{\text{def}}{=} \mathrm{SAT}(\varphi) \cap \{v \subseteq X || v |= k\}$$

Variable gate.

- α_g^0 is 0 and α_g^1 is 1 .

$\neg gate$

- If g is a $\neg$-gate with input gate g', then $\alpha_g^\ell = \binom{|\operatorname{Vars}(g)|}{l} - \alpha_{g'}^\ell$ for every $\ell \in \{0, \ldots, |\operatorname{Vars}(g)|\}$.

Exact Computation

Deterministic $\vee$-gate

Define $S_1 \overset{\text{def}}{=} \mathrm{Vars}\left(g_2\right) \setminus \mathrm{Vars}\left(g_1\right)$ and similarly $S_2 \overset{\text{def}}{=} \mathrm{Vars}\left(g_1\right) \setminus \mathrm{Vars}\left(g_2\right)$. Since g is deterministic, we have:

$$\mathrm{SAT}\left(\varphi_g\right) = \left(\mathrm{SAT}\left(\varphi_{g_1}\right) \otimes 2^{S_1}\right) \cup \left(\mathrm{SAT}\left(\varphi_{g_2}\right) \otimes 2^{S_2}\right)$$

with the union being disjoint. By intersecting with the assignments of $\mathrm{Vars}(g)$ of size ℓ, we obtain:

$$\begin{aligned}\mathrm{SAT}_\ell\left(\varphi_g\right) = &\left[\left(\mathrm{SAT}\left(\varphi_{g_1}\right) \otimes 2^{S_1}\right) \cap \{v \subseteq \mathrm{Vars}(g) || v \mid= \ell\}\right] \\ &\cup \left[\left(\mathrm{SAT}\left(\varphi_{g_2}\right) \otimes 2^{S_2}\right) \cap \{v \subseteq \mathrm{Vars}(g) || v \mid= \ell\}\right]\end{aligned}$$

$$\begin{aligned}\#\,\mathrm{SAT}_\ell\left(\varphi_g\right) = &\left|\left(\mathrm{SAT}\left(\varphi_{g_1}\right) \otimes 2^{S_1}\right) \cap \{v \subseteq \mathrm{Vars}(g) || v \mid= \ell\}\right| \\ &+ \left|\left(\mathrm{SAT}\left(\varphi_{g_2}\right) \otimes 2^{S_2}\right) \cap \{v \subseteq \mathrm{Vars}(g) || v \mid= \ell\}\right|\end{aligned}$$

$$\left|\left(\mathrm{SAT}\left(\varphi_{g_1}\right) \otimes 2^{S_1}\right) \cap \{v \subseteq \mathrm{Vars}(g) || v \mid= \ell\}\right| = \sum_{i=\max(0,\ell-|S_1|)}^{\min(\ell,\mathrm{Vars}(g_1)|)} \alpha_{g_1}^i \times \binom{|S_1|}{\ell - i}.$$

Exact Computation

Decomposable $\wedge$-gate.

$$\mathrm{SAT}\left(\varphi_g\right) = \mathrm{SAT}\left(\varphi_{g_1}\right) \otimes \mathrm{SAT}\left(\varphi_{g_2}\right)$$

We now intersect with the set of assignments of $\mathbf{Vars}(g)$ of size l to obtain

$$\alpha_g^\ell = \#\,\mathrm{SAT}_\ell\left(\varphi_g\right) = \sum_{i=\max(0,\ell-|\mathrm{Vars}(g_2)|)}^{\min(\ell,|\mathrm{Vars}(g_1)|)} \alpha_{g_1}^i \times \alpha_{g_2}^{\ell-i}$$

Exact Computation

Algorithm 1: Shapley values from deterministic and decomposable Boolean circuits

Input : Deterministic and decomposable Boolean circuit C with output gate g_{output} representing $\mathsf{ELin}(q, D_{\mathrm{x}}, D_{\mathrm{n}})$ and an endogenous fact $f \in D_{\mathrm{n}}$.
Output: The value $\mathsf{Shapley}(q, D_{\mathrm{x}}, D_{\mathrm{n}}, f)$.

Complete C so that $\mathsf{Vars}(g_{\mathrm{output}}) = D_{\mathrm{n}}$;
Compute $C_1 = C[f \to 1]$ and $C_2 = C[f \to 0]$;
`// Partial evaluations of` C `by setting` f `to 1 and to 0`
$\Gamma = \mathrm{ComputeAll\#SAT}_k(C_1)$; `// As an array`
$\Delta = \mathrm{ComputeAll\#SAT}_k(C_2)$; `// As an array`
return $\displaystyle\sum_{k=0}^{|D_{\mathrm{n}}|-1} \frac{k!\,(|D_{\mathrm{n}}| - k - 1)!}{|D_{\mathrm{n}}|!} \cdot (\Gamma[k] - \Delta[k])$;

Def $\mathtt{ComputeAll\#SAT}_k(C)$:
  Preprocess C so that each $\vee$-gate and $\wedge$-gate has fan-in exactly 0 or 2;
  Compute the set $\mathsf{Vars}(g)$ for every gate g in C;
  Compute values α_g^ℓ for every gate g in C and $\ell \in \{0, \ldots, |\mathsf{Vars}(g)|\}$ by bottom-up induction on C using the inductive relations from the proof of Lemma 4.5;
  return $[\alpha_{g_{\mathrm{output}}}^0, \ldots, \alpha_{g_{\mathrm{output}}}^{|\mathsf{Vars}(C)|}]$;

Inexact Computation

Based on the observation that having a high shapley score is correlated with

- Appearing many times of the provenance
- Having few alternatives

Shapley $(h, x) \stackrel{\text{def}}{=} \sum_{S \subseteq X \setminus \{x\}} \frac{|S|!(|X|-|S|-1)!}{|X|!}(h(S \cup \{x\}) - h(S)).$

$$\Phi(\psi_i, x) = \begin{cases} \frac{1}{(a_i+b_i)\cdot\binom{a_i+b_i-1}{b_i}} & \textit{if } x \textit{ appears in } \psi_i \textit{ in positive form;} \\ \frac{-1}{(a_i+b_i)\cdot\binom{a_i+b_i-1}{a_i}} & \textit{if } x \textit{ appears in } \psi_i \textit{ in negative form;} \\ 0 & \textit{otherwise.} \end{cases}$$

Inexact Computation

Algorithm 2: CNF Proxy

Input : CNF φ and a set of endogenous facts D_n.
Output: The value Shapley$(\widetilde{\varphi}, x)$ for each $x \in D_\mathrm{n}$.

$n \leftarrow |\varphi.clauses()|$;
$v \leftarrow 0^{|D_\mathrm{n}|}$; `// As an array`
for $\psi \in \varphi.clauses()$ **do**
  $\mathcal{L} \leftarrow \psi.literals()$;
  $m \leftarrow |\mathcal{L}|$;
  $pos \leftarrow \{\ell \in \mathcal{L} \mid \ell \text{ is positive}\}$;
  $neg \leftarrow \{\ell \in \mathcal{L} \mid \ell \text{ is negative}\}$;
  for $\ell \in pos \cap D_\mathrm{n}$ **do**
    $v[\ell.var()] \leftarrow v[\ell.var()] + \frac{1}{nm \cdot \binom{m-1}{|neg|}}$;
  end
  for $\ell \in neg \cap D_\mathrm{n}$ **do**
    $v[\ell.var()] \leftarrow v[\ell.var()] - \frac{1}{nm \cdot \binom{m-1}{|pos|}}$;
  end
end
return v

Conclusion - Experimental Results

- knowledge compilation using the c2d compiler
- datasets
  - TPC-H (removing nested queries and queries with aggregation)
  - IMDB
- Algorithm1 yields an overall 84.43% success rate for TPC-H and 99.96% for IMDB,
- Algorithm2 is very efficient and that the ranking based on CNF proxies is very close to the ranking based on the exact Shapley value