skema BUSINESS SCHOOL | AI R&D CENTER CANADA

Computing Shapley Values as Explanation Scores

in

Data Management and Machine Learning

Leopoldo Bertossi
leopoldo.bertossi@skema.edu

Coalition Games and the Shapley Value

- Initial motivation: How much does a database tuple contribute to the inconsistency of a DB? To the violation of ICs
- Similarly: Contribution to a query result [1,2]
- Usually several tuples together are necessary to violate an IC or produce a query result
- Like players in a coalition game, some may contribute more than others

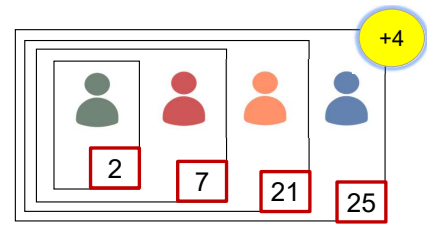
  As represented by an application-dependent numeric game or wealth function
- Apply standard measures used in game theory: the Shapley value of tuple

  As an attribution score for its contribution

- Consider a set of players D, and a wealth-distribution function $\mathcal{G}: \mathcal{P}(D) \longrightarrow \mathbb{R}$ ($\mathcal{P}(D)$ the power set of D)
- The Shapley value of player p among a set of players D:

$$Shapley(D, \mathcal{G}, p) := \sum_{S \subseteq D \setminus \{p\}} \frac{|S|!(|D| - |S| - 1)!}{|D|!} (\mathcal{G}(S \cup \{p\}) - \mathcal{G}(S))$$

- $|S|!(|D| - |S| - 1)!$ is number of permutations of D with all players in S coming first, then p, and then all the others
- Expected contribution of player p under all possible additions of p to a partial random sequence of players followed by a random sequence of the rest of the players



- Implicit counterfactual intervention:
  What would happen if we change ...?
  Or having p vs. not having it?

- The Shapley value is a established measure of contribution by players to the wealth function
- It emerges as the only measure that enjoys certain desired properties
- For each game one defines an appropriate wealth or game function
- Shapley difficult to compute: #P-hard in general
- Evidence of difficulty: $\#SAT$ is #P-hard

  About counting satisfying assignments for propositional formulas

  At least as difficult as SAT
- There had been research in KR on the Shapley-value to measure the inconsistency of a propositional KB

Shapley as Score for QA

- Back to QA in DBs, tuples in DB D can be seen as players in a coalition game

  Each of them contributing to a shared wealth function

- Concentrated on BCQs and aggregation on CQs

- For a Boolean query $\mathcal{Q}$, and $S \subseteq D$: $\mathcal{Q}(S) = \begin{cases} 1 & \text{if } S \models \mathcal{Q} \\ 0 & \text{if } S \not\models \mathcal{Q} \end{cases}$

  $$Shapley(D, \mathcal{Q}, \tau) := \sum_{S \subseteq D \setminus \{\tau\}} \frac{|S|!(|D|-|S|-1)!}{|D|!} \left[\mathcal{Q}(S \cup \{\tau\}) - \mathcal{Q}(S) \right]$$

  Quantifies the contribution of tuple τ to query result

- Players (tuples) can be split into endogenous and exogenous

  One wants to measure the contribution of endogenous tuples

  They could be those in a particular table or particular source

- Dichotomy Theorem: $\mathcal{Q}$ BCQ without self-joins

  If $\mathcal{Q}$ hierarchical, then $Shapley(D, \mathcal{Q}, \tau)$ can be computed in PTIME

  Otherwise, the problem is $FP^{\#P}$-complete

- $\mathcal{Q}$ is hierarchical if for every two existential variables x and y:
  - $Atoms(x) \subseteq Atoms(y)$, or
  - $Atoms(y) \subseteq Atoms(x)$, or
  - $Atoms(x) \cap Atoms(y) = \emptyset$

- Example: $\mathcal{Q}: \exists x \exists y \exists z (R(x, y) \wedge S(x, z))$

  $Atoms(x) = \{R(x, y), S(x, z)\}$, $Atoms(y) = \{R(x, y)\}$,
  $Atoms(z) = \{S(x, z)\}$ Hierarchical!

- Example: $\mathcal{Q}^{nh}: \exists x \exists y (R(x) \wedge S(x, y) \wedge T(y))$

  $Atoms(x) = \{R(x), S(x, y)\}$, $Atoms(y) = \{S(x, y), T(y)\}$

  Not hierarchical!

- Same criteria as for QA over prob DBs (Dalvi & Suciu; 2004)

  But new proof techniques required

- Positive case: reduced to counting subsets of D of fixed size that satisfy $\mathcal{Q}$

  A dynamic programming approach works

- Negative case: Use query $\mathcal{Q}^{nh}$ above

  Reduction from counting independent sets in a bipartite graph

- Dichotomy extends to summation over CQs; same conditions and cases

  Shapley value is an expectation, that is linear

- Hardness extends to aggregate non-hierarchical queries: max, min, avg

- What to do in hard cases?

- Approximation:

  For every fixed BCQ $\mathcal{Q}$, there is a multiplicative fully-polynomial randomized approximation scheme (FPRAS)

$$P(\tau \in D \mid \frac{Sh(D,\mathcal{Q},\tau)}{1+\epsilon} \leq A(\tau,\epsilon,\delta) \leq (1+\epsilon)Sh(D,\mathcal{Q},\tau)\}) \geq 1-\delta$$

  Also applies to summations

- A related and popular score is the Bahnzhaf Power Index (order ignored)

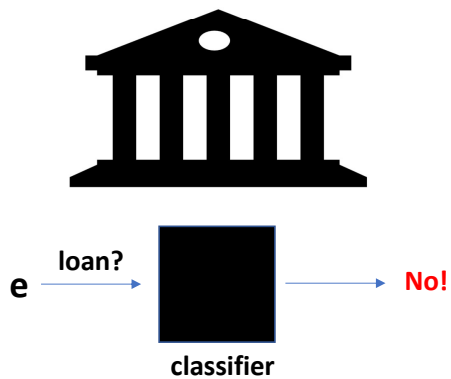
$$Banzhaf(D,\mathcal{Q},\tau) := \frac{1}{2^{|D|-1}} \cdot \sum_{S \subseteq (D \setminus \{\tau\})} (\mathcal{Q}(S \cup \{\tau\}) - \mathcal{Q}(S))$$

  Bahnzhaf also difficult to compute; provably #P-hard in general

Explanations in Machine Learning

- Bank client $\mathbf{e} = \langle \text{john}, 18, \text{plumber}, 70\text{K}, \text{harlem}, \ldots \rangle$

  As an entity represented as a record of values for features Name, Age, Activity, Income, ...
- $\mathbf{e}$ requests a loan from a bank, which uses a classifier



- The client asks *Why?*
- What kind of *explanation?*
  How?
  From what?

Explanations in AI

- Users and those affected by results from AI systems, the stakeholders, request explanations

  Assessments (e.g. a credit score), classifications (good/bad client), decisions (approve/reject loan), etc.

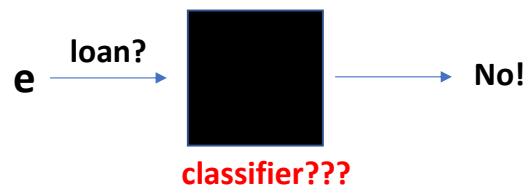
- A whole new area of AI has emerged: *Explainable AI* (XAI)

  A whole discipline has emerged: *Ethical AI*

- It touches Law, Sociology, Philosophy, ...

- Motivated by the need for more *transparent, trustable, fair, unbiased, ...* and *interpretable* AI systems

- New legislation forces AI systems affecting users to provide explanations and guarantee all the above



It may really be a "black box"!

- Different kinds of explanations have been proposed [6]
- Here we concentrate on attribution scores as explanations for a classification result

  They quantify the relevance of a particular feature value in a given entity under classification for the result of the latter
- There are other attribution scores

  Among them, the causal responsibility score [3,5,6]

  Based on actual causality, it explicitly appeals to counterfactuals
- Here, we concentrate on the Shapley value as an attribution score for explainable ML

Shap Scores

- Based on the general Shapley value
- Set of players $\mathcal{F}$ contain features, relative to classified entity $\mathbf{e}$
- We need an appropriate $\mathbf{e}$-dependent game function that maps (sub)sets of players to real numbers
- For $S \subseteq \mathcal{F}$, and $\mathbf{e}_S$ the projection of $\mathbf{e}$ on S:

$$\mathcal{G}_{\mathbf{e}}(S) \;:=\; \mathbb{E}(L(\mathbf{e}') \mid \mathbf{e}' \in \mathcal{E} \;\;\&\;\; \mathbf{e}'_S = \mathbf{e}_S)$$

- For a feature $F^\star \in \mathcal{F}$, compute: $Shap(\mathcal{F}, \mathcal{G}_{\mathbf{e}}, F^\star)$

$$\sum_{S \subseteq \mathcal{F} \setminus \{F^\star\}} \frac{|S|!(|\mathcal{F}|-|S|-1)!}{|\mathcal{F}|!} [\underbrace{\mathbb{E}(L(\mathbf{e}') | \mathbf{e}'_{S \cup \{F^\star\}} = \mathbf{e}_{S \cup \{F^\star\}})}_{\mathcal{G}_{\mathbf{e}}(S \cup \{F^\star\})} - \underbrace{\mathbb{E}(L(\mathbf{e}') | \mathbf{e}'_S = \mathbf{e}_S)}_{\mathcal{G}_{\mathbf{e}}(S)}]$$

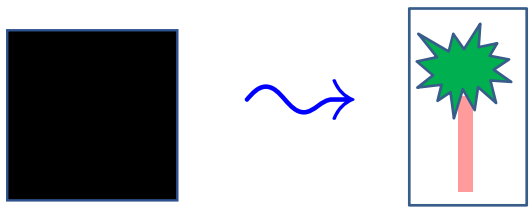
- *Shap* score has become popular (Lee & Lundberg, 2017)
- Assumes a probability distribution on entity population

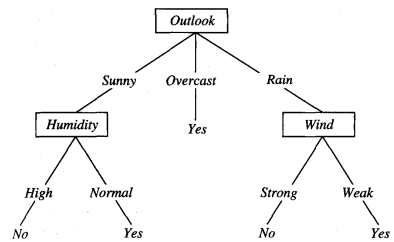
Shap Tractability?

- *Shap* may end up considering exponentially many combinations

  And multiple passes through the black-box classifier

- Can we do better with an open-box classifier?





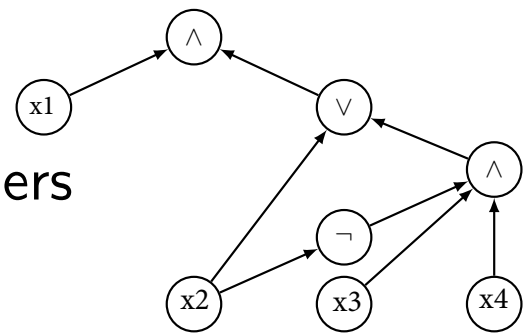
  Exploiting its elements and internal structure?

- What if we have a decision tree, or a random forest, or a Boolean circuit?
- Can we compute *Shap* in polynomial time?

Tractability for BC-Classifiers: Big Picture

- We investigated this problem in detail [4]
- Tractable and intractable cases, with algorithms for the former

  Investigated good approximation algorithms
- Choosing the right abstraction (model) is crucial
- We considered Boolean-Circuit Classifiers (BCCs), i.e. propositional formulas with (binary) output gate



- It was known already that *Shap* is intractable for "Monotone 2CNF"-classifiers under the product distribution [3]
- So, it had to be a broad and interesting class of BCs

Shap for Boolean-Circuit Classifiers

- Features $F_i \in \mathcal{F},\ i = 1, \ldots, n, \quad Dom(F_i) = \{0, 1\}$,
  $\mathbf{e} \in \mathcal{E} := \{0, 1\}^n, \quad L(\mathbf{e}) \in \{0, 1\}$
- There is also a probability distribution P on $\mathcal{E}$
- For BC-classifier L: $\quad Shap(\mathcal{F}, G_{\mathbf{e}}, F^\star) \ =$

  $\sum_{S \subseteq \mathcal{F} \setminus \{F^\star\}} \frac{|S|!(|\mathcal{F}|-|S|-1)!}{|\mathcal{F}|!} [\mathbb{E}(L(\mathbf{e}'|\mathbf{e}'_{S \cup \{F^\star\}} = \mathbf{e}_{S \cup \{F^\star\}}) - \mathbb{E}(L(\mathbf{e}')|\mathbf{e}'_S = \mathbf{e}_S)]$

  Depends on $\mathbf{e}$ and L
- $SAT(L) := \{\mathbf{e}' \mid L(\mathbf{e}') = 1\} \qquad\qquad \#SAT(L) := |SAT(L)|$

  Counting the number of inputs that get label 1
- We established that *Shap* is at least as hard as model counting for the BC:

  Proposition: For the uniform distribution P^u, and $\mathbf{e} \in \mathcal{E}$

  $$\#SAT(L) \ = \ 2^{|\mathcal{F}|} \times (\ L(\mathbf{e}) - \textstyle\sum_{i=1}^{n} Shap(\mathcal{F}, G_{\mathbf{e}}, F_i)\ )$$

- Then: $\#SAT \leq^{Turing}_{PTIME} Shap$

  When $\#SAT(L)$ is hard for a Boolean classifier L, $Shap$ is also hard

- Negative Corollary: Computing $Shap$ is $\#P$-hard for
  - Linear perceptron classifier
    By reduction from $\#Knapsack$ (with weights in binary)
  - Boolean classifiers defined by Monotone 2DNF or Monotone 2CNF [Provan & Ball, 1983]

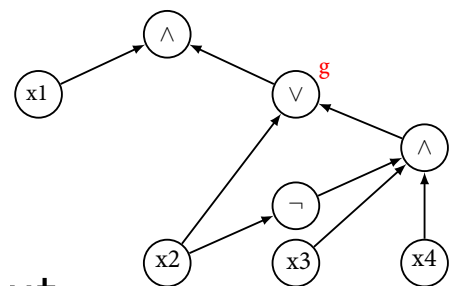
- Can we do better for other classes of binary classifiers?

  Other classes of Boolean-circuit classifiers?

Deterministic and Decomposable BCs

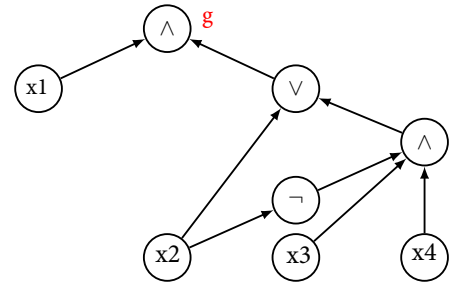
- A Boolean circuit over set of variables X is a DAG $\mathcal{C}$ with:
  - Each node without incoming edges (input) is labeled with either a variable $x \in X$ or a constant in $\{0, 1\}$
  - Each other node is labeled with a gate in $\{\neg, \wedge, \vee\}$
  - There is a single sink node, O, called the output
- $\mathbf{e}: X \to \{0, 1\}$ (equivalently $\mathbf{e} \in \{0, 1\}^{|X|}$) is accepted by $\mathcal{C}$, written $\mathcal{C}(\mathbf{e}) = 1$, iff O takes value 1
- For a gate g of $\mathcal{C}$, $\mathcal{C}(g)$ is the induced subgraph containing gates on a path in $\mathcal{C}$ to g

  $Var(g)$ is the set of variables of $\mathcal{C}(g)$
  $Var(g) = \{x2, x3, x4\}$



- $\mathcal{C}$ is deterministic if every $\vee$-gate g with input gates g_1, g_2: $\mathcal{C}(g_1)(\mathbf{e}) \neq \mathcal{C}(g_2)(\mathbf{e})$, for every $\mathbf{e}$

- $\mathcal{C}$ is decomposable if every $\wedge$-gate g with input gates g_1, g_2: $Var(g_1) \cap Var(g_2) = \emptyset$



- We concentrated on the class of deterministic and decomposable Boolean circuits (dDBCs)
- *Shap* computation in polynomial time not initially precluded
- A class of BCCs that includes -via efficient (knowledge) compilation- many interesting ones, syntactic and not ...
  - Decision trees (and random forests)
  - Ordered binary decision diagrams (OBDDs)
  - Sentential decision diagrams (SDDs)
  - Deterministic-decomposable negation normal-form (dDNNFs)

Shap for dDBCs

- Proposition: For dDBCs $\mathcal{C}$, $\#SAT(\mathcal{C})$ can be computed in polynomial time ($\not\Rightarrow$ the same for *Shap*)

  Idea: Bottom-up procedure that inductively computes $\#SAT(\mathcal{C}(g))$, for each gate g of $\mathcal{C}$
- To show that *Shap* can be computed efficiently for dDBCs, we need a detailed analysis
- We assume the uniform distribution for the moment
- A related problem: "satisfiable circle of an entity"

  $$SAT(\mathcal{C}, \mathbf{e}, \ell) := SAT(\mathcal{C}) \cap \{\ e' \mid \underbrace{\|\mathbf{e} - \mathbf{e}'\|_1 = \ell}_{\ell \text{ value discrepancies}}\ \}$$

  $$\#SAT(\mathcal{C}, \mathbf{e}, \ell) := |SAT(\mathcal{C}, \mathbf{e}, \ell)|$$
- Proposition: If computing $\#SAT(\mathcal{C}, \mathbf{e}, \ell)$ is tractable, so is $Shap(X, \mathcal{G}_{\mathbf{e}}, x)$

- Main Lemma: $\#SAT(\mathcal{C}, \mathbf{e}, \ell)$ can be solved in polynomial time for dDBCs $\mathcal{C}$, entities $\mathbf{e}$, and $1 \leq \ell \leq |X|$

  Idea: Inductively compute $\#SAT(\mathcal{C}(g), \mathbf{e}_{Var(g)}, \ell)$ for each gate $g \in \mathcal{C}$ and integer $\ell \leq |Var(g)|$

  - Input gate: immediate
  - $\neg$-gate:

    $\#SAT(\mathcal{C}(\neg g), \mathbf{e}_{Var(g)}, \ell) = \binom{Var(g)}{\ell} - \#SAT(\mathcal{C}(g), \mathbf{e}_{Var(g)}, \ell)$
  - $\vee$-gate: (uses determinism)

    $\#SAT(\mathcal{C}(g_1 \vee g_2), \mathbf{e}_{Var(g_1) \cup Var(g_2)}, \ell) =$

    $\#SAT(\mathcal{C}(g_1), \mathbf{e}_{Var(g_1)}, \ell) + \#SAT(\mathcal{C}(g_2), \mathbf{e}_{Var(g_2)}, \ell)$
  - $\wedge$-gate: (uses decomposition)

    $\#SAT(\mathcal{C}(g_1 \wedge g_2), \mathbf{e}_{Var(g_1) \cup Var(g_2)}, \ell) =$

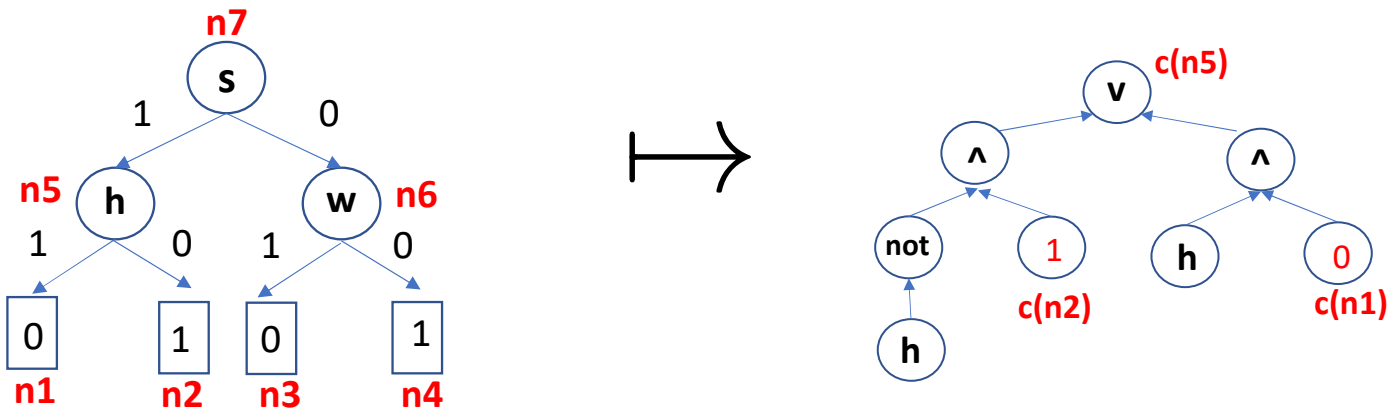
    $\sum_{j+k=\ell} \#SAT(\mathcal{C}(g_1), \mathbf{e}_{Var(g_1)}, j) \times \#SAT(\mathcal{C}(g_2), \mathbf{e}_{Var(g_2)}, k)$
- Theorem: *Shap* can be computed in polynomial time for dDBCs under the uniform distribution
- It can be extended to any product distribution on $\{0,1\}^{|X|}$

Shap from dDBCs

- Corollary: Via polynomial time transformations, under the uniform and product distributions, *Shap* can be computed in polynomial time for
  - Decision trees (and random forests)
  - Ordered binary decision diagrams (OBDDs)
  - Sentential decision diagrams (SDDs)
  - Deterministic-decomposable negation normal-form (dDNNFs)
- An optimized efficient algorithm for *Shap* computation can be applied to any of these [4]

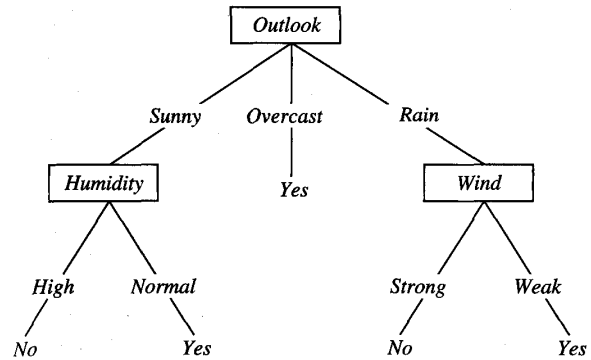
Shap for Decision Trees and ...

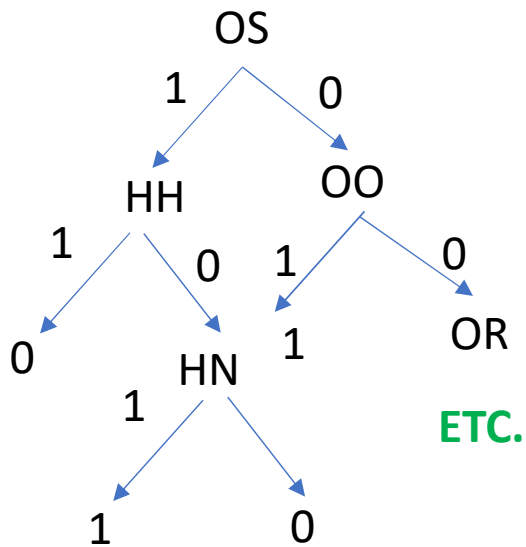
- Compiling binary decision trees into dDBCs
- An inductive construction starting from the bottom of the DT
- Leaves of DT become constant binary gates in dDBC
- By induction one can prove the resulting circuit is dDBC
- Final dDBC is the compilation $c(r)$ of root node r of DT



- Final equivalent dDBC: $c(n7)$
- Computable in linear time

- Beyond binary features?
- "Binarize" features
- *OutlookSunny* (OS)
  OutlookOvercast, *OutlookRain*, etc.
  become propositional features





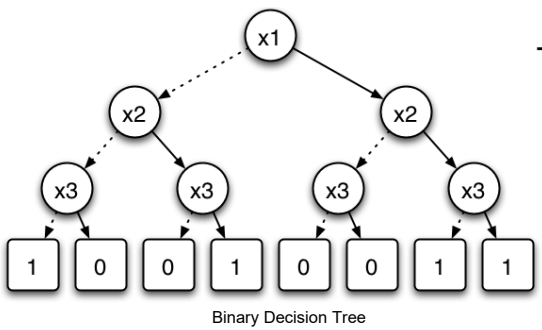
Certain entities become impossible (probability 0)

$\mathbf{e} = \langle \underbrace{0, 1, 1}_{\text{for OS, OO, OR}} , \ldots \rangle$ ×

$\mathbf{e} = \langle \underbrace{0, 1, 0}_{\text{for OS, OO, OR}} , \ldots \rangle$ *ok*

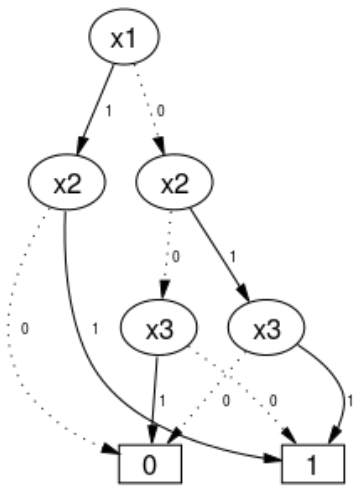
- Our polynomial time algorithm for *Shap* can be applied to *Ordered Binary Decision Diagrams* (OBDDs)
- They are relevant for several reasons in *Knowledge Compilation*
- In particular, to represent "opaque" classifiers as OBDDs, e.g. binary neural networks [Shi, Shih, Darwiche, Choi; KR20]
- Opening the ground for efficiently applying *Shap* to them

$f(x_1, x_2, x_3) = (\neg x_1 \wedge \neg x_2 \wedge \neg x_3) \vee (x_1, \wedge x_2) \vee (x_2 \wedge x_3)$



Binary Decision Tree

x1	x2	x3	f
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

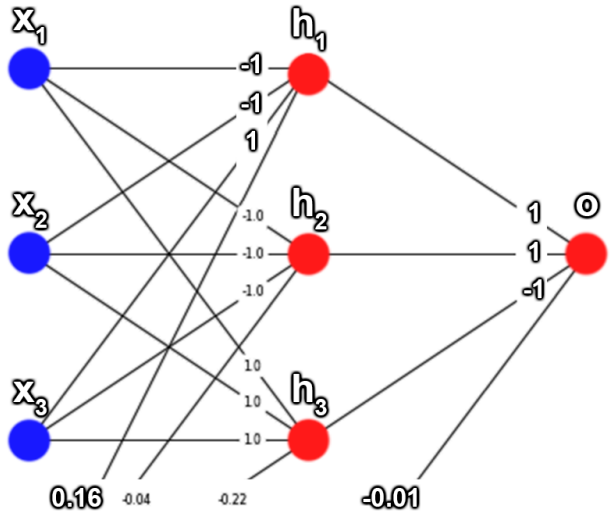


OBDD

Same variable order along full paths

Shap on Neural Networks

- Binary Neural Networks (BNNs) are commonly considered black-box models
- Naively computing *Shap* on a BNN is bound to be complex
- Better try to compile the BNN into an open-box BC where *Shap* can be computed efficiently
- We have experimented with *Shap* computation with a black-box BNN and with its compilation into a dDBC [7]
- Even if the compilation is not entirely of polynomial time, it may be worth performing this one-time computation
- Particularly if the target dDBC will be used multiple times, as is the case for explanations
- We illustrate the approach by means of an example



$$\phi_g(\bar{i}) \quad = \quad sp(\bar{w}_g \bullet \bar{i} + b_g)$$
$$:= \quad \begin{cases} 1 & \text{if } \bar{w}_g \bullet \bar{i} + b_g \geq 0, \\ -1 & \text{otherwise,} \end{cases}$$

- The BNN is described by means of a propositional formula, which is further transformed and optimized into CNF

$$\begin{aligned} o \longleftrightarrow\ & (-[(x_3 \wedge (x_2 \vee x_1)) \vee (x_2 \wedge x_1)] \wedge \\ & ([(-x_3 \wedge (-x_2 \vee -x_1)) \vee (-x_2 \wedge -x_1)] \vee \\ & [(x_3 \wedge (-x_2 \vee -x_1)) \vee (-x_2 \wedge -x_1)])) \vee \\ & ([(-x_3 \wedge (-x_2 \vee -x_1)) \vee (-x_2 \wedge -x_1)] \wedge \\ & [(x_3 \wedge (-x_2 \vee -x_1)) \vee (-x_2 \wedge -x_1)]). \end{aligned}$$

In CNF:

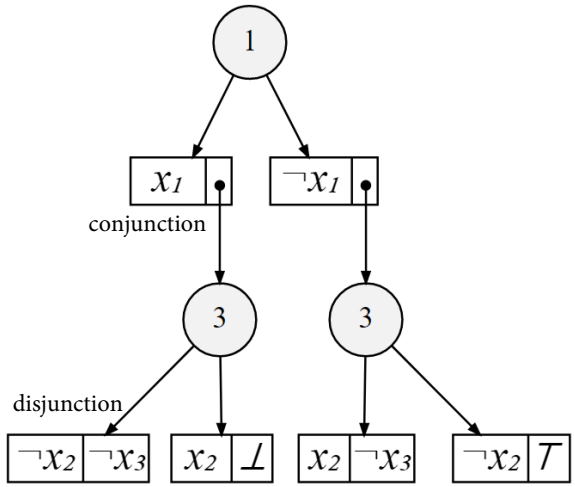
$$o \longleftrightarrow (-x_1 \vee -x_2) \wedge (-x_1 \vee -x_3) \wedge (-x_2 \vee -x_3)$$

- The CNF is transformed into an SDD

  It succinctly represents the CNF

- The expensive compilation step

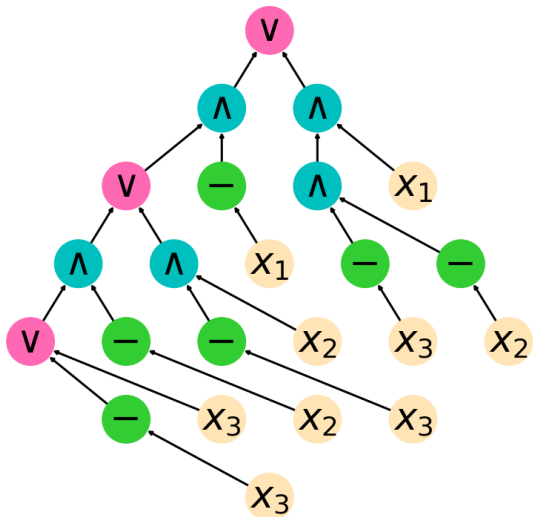
  But upper-bounded by an exponential only in the tree-width of the CNF



  A measure of how close to a tree is the undirected graph associated to the CNF

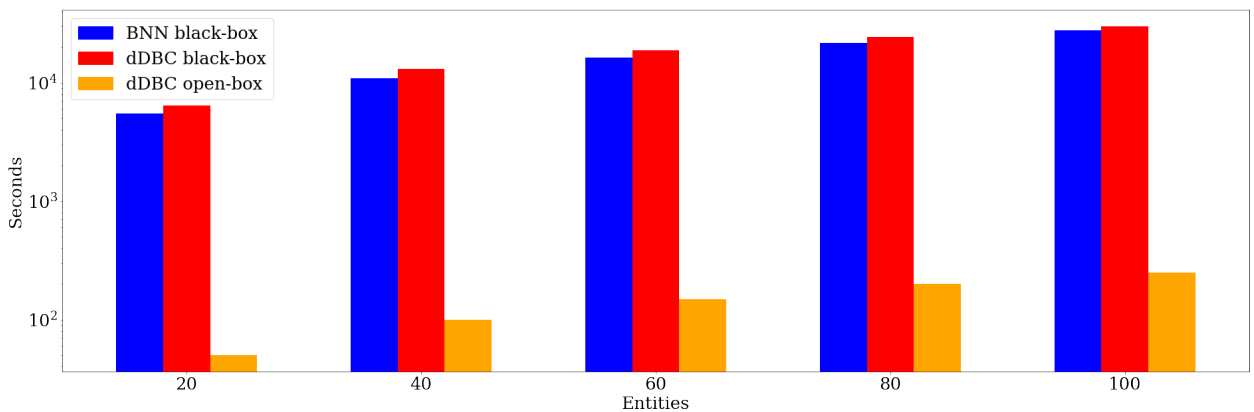
  An edge between variables if together in a clause

- Finally, the SDD is easily transformed into a dDBC
- On it *Shap* is computed, possibly multiple times
- With considerable efficiency gain



- In our experiments, we used a BNN with 14 gates
- It was compiled into a dDBC with 18,670 nodes
- A one-time computation that fully replaces the BNN
- We compared *Shap* computation time for black-box BNN, open-box dDBC, and black-box dDBC

  Total time for computing all Shap scores with increasing number of classification inputs



(logarithmic scale)

- The uniform distribution was used

Look Ahead

- The above results on *Shap* computation hold under the uniform and product distributions

  The latter imposes independence among features

- Other distributions have been considered for *Shap* and other scores

  The empirical and product-empirical distributions [3]

  They naturally arise when no more information available about the distribution

- Imposing domain semantics (domain knowledge) is relevant to explore

- Can we modify *Shap* definition and computation accordingly?

  Or the distribution? [5]

- Do we still have an efficient algorithm?
- In the case of databases, do complexity results change under integrity constraints (ICs)?

  That is, the implicit counterfactuals must respect the ICs
- In the case of causal responsibility, there is a change under ICs [6]

Self-References

[1] E. Livshits, L. Bertossi, B. Kimelfeld and M. Sebag. "The Shapley Value of Tuples in Query Answering". *Logical Methods in Computer Science*, 17(3):22.1-22.33.

[2] E. Livshits, L. Bertossi, B. Kimelfeld, M. Sebag. "Query Games in Databases". *ACM Sigmod Record*, 2021, 50(1):78-85.

[3] L. Bertossi, J. Li, M. Schleich, D. Suciu and Z. Vagena. "Causality-based Explanation of Classification Outcomes". Proc. 4th International Workshop on "Data Management for End-to-End Machine Learning" (DEEM) at ACM SIGMOD/PODS, 2020, pp. 6.1-6.10.

[4] M. Arenas, P. Barcelo, L. Bertossi, M. Monet. "The Tractability of SHAP-scores over Deterministic and Decomposable Boolean Circuits". *Journal of Machine Learning Research*, 2023, 24(63):1-58. (extended version of AAAI 2021 paper).

[5] L. Bertossi. "Declarative Approaches to Counterfactual Explanations for Classification". *Theory and Practice of Logic Programming*, 2022. (forthcoming) arXiv Paper 2011.07423, 2021.

[6] L. Bertossi. "Attribution-Scores and Causal Counterfactuals as Explanations in Artificial Intelligence". In 'Reasoning Web: Causality, Explanations and Declarative Knowledge'. Springer LNCS 13759, 2023. Corr arXiv Paper 2303.02829.

[7] L. Bertossi and Jorge E. Leon. "Opening Up the Neural Network Classifier for Shap Score Computation". Corr arXiv Paper 2303.06516.