Computing Attribution Scores for Classification in Explainable AI

Prof. Leopoldo Bertossi (PhD Math)

AMMCS, Waterloo, Canada, August 2026

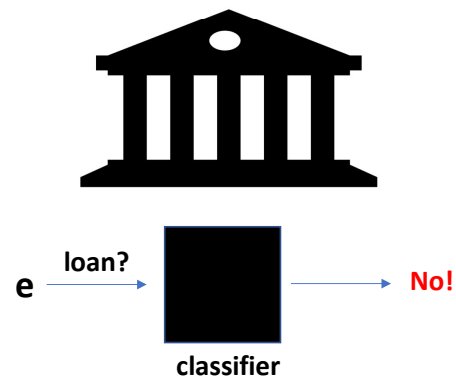
www.scs.carleton.ca/∼bertossi

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, ...

- **e** requests a loan from a bank that uses a classifier



- The client asks *Why?*
- What kind of *explanation?*

  How?

  From what?

- Explanations come in different forms
- Some of them are *causal explanations*, some are *explanation scores* a.k.a. *attribution-scores*
- They are sometimes related

  E.g. actual causality leads to responsibility scores
- Large part of our recent research is about the use of causality, and the definition and computation of attribution-scores
- Some of them: In data management and XAI
  - Responsibility (in its original and generalized versions)
  - The Causal Effect score
  - The Shapley value (as Shap in ML)
- They are model-agnostic Really?

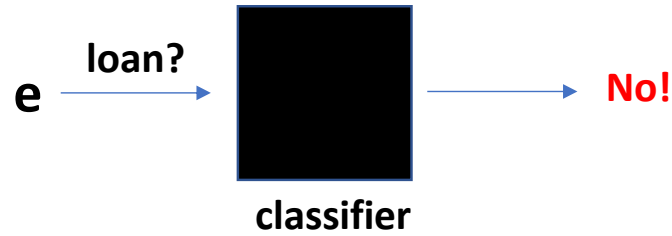
A Score-Based Approach: Responsibility

- Causality has been developed in AI for three decades or so
- In particular: Actual Causality
- Also the quantitative notion of Responsibility: a measure of causal contribution (the *Resp*-score)
- Both based on Counterfactual Interventions
- Hypothetical changes of values in a causal model to detect other changes

  "What would happen if we change ..."?

  By so doing identify actual causes
- Does the deletion of the DB tuple invalidates the query?
- Does a change of this feature value leads to label "Yes"?

- Simplified Case:



$$\mathbf{e} = \langle \text{john}, 18, \text{plumber}, 70\text{K}, \text{harlem}, \ldots \rangle \quad \text{No}$$

- Counterfactual versions:

$$\mathbf{e}' = \langle \text{john}, 25, \text{plumber}, 70\text{K}, \text{harlem}, \ldots \rangle \quad \text{Yes}$$

$$\mathbf{e}'' = \langle \text{john}, 18, \text{plumber}, 80\text{K}, \text{brooklyn}, \ldots \rangle \quad \text{Yes}$$

- For the gist:

1. Value for feature Age is counterfactual cause with explanatory responsibility $\mathit{Resp}(\mathbf{e}, \text{Age}) = 1$
2. Value change Income $:=$ 80K needs an additional, minimum contingent change: $\Gamma = \{\text{Area} := \text{brooklin}\}$

   Income $:=$ 70K is actual cause with $\mathit{Resp}(\mathbf{e}, \text{Income}) = \frac{1}{1+|\Gamma|} = \frac{1}{2}$

The Generalized *Resp* Score

- For binary (two-valued) features the previous "definition" works fine (previous example is non-binary)
- Otherwise, there may be many values for a feature that do not change the label

  The original value is not great explanation

  Similarly for features in a potential contingency set

- Better consider average labels obtained via counterfactual interventions

  Resp, our extended version of responsibility, will be expressed in terms of an expected value[1]

- Assume classifier L is binary; and, w.l.o.g: $L(\mathbf{e}) = 1$

[1] Bertossi, Li, Schleich, Suciu,Vagena; SIGMOD Deem WS'20

- Start with local score: (contingent $\Gamma \subseteq (\mathcal{F} \setminus \{F^\star\})$ and values $\bar{w}$ for Γ)

$$Resp(\mathbf{e}, F^\star, \underline{\Gamma, \bar{w}}) := \frac{L(\mathbf{e}) - \mathbb{E}(\ L(\mathbf{e}') \mid F(\mathbf{e}') = F(\mathbf{e}^{\Gamma,\bar{w}}),\ \forall F \in (\mathcal{F} \setminus \{F^\star\})\ )}{1 + |\Gamma|} \quad (*)$$

  - $\mathbf{e}^{\Gamma,\bar{w}}$ is as input $\mathbf{e}$, with values for features in Γ as in $\bar{w}$
  - All e' are as $\mathbf{e}^{\Gamma,\bar{w}}$ for all features, except $F^\star$

- Global score, with "best" contingencies $(\Gamma, \bar{w})$

$$Resp(\mathbf{e}, F^\star) := \max_{\langle \Gamma, \bar{w} \rangle,\ |\Gamma|\ min.,\ (*) > 0} Resp(\mathbf{e}, F^\star, \Gamma, \bar{w})$$

- Condition "$(*) > 0$" accounts for counterfactual reasoning:

  Label different from $L(\mathbf{e})$, in average

- We are interested in maximum-score feature values

  Associated to minimum (cardinality) contingency sets

Responsibility: Some Remarks

- We have investigated actual causality and responsibility in data management and ML-based classification
- Semantics, computational mechanisms, intrinsic complexity, logic-based specifications, reasoning, etc.
- Assign scores to database tuples to capture their relevance for query-answering

  To features values in an entity under classification

  To capture their causal, or, more generally, explanatory strength
- It can be applied without knowing "the internals" of a classifier Only input/output relation needed

  It can be a "black box", or treated as such (a complex NN)

- We have experimentally compared responsibility scores with other local attribution-scores in ML
  - *Shap*
  - *Ad hoc* scores, such as for FICO data on "open-box" model (connected logistic regressions)
- Promising results in terms of values obtained, and computation
- However, already with binary domains, *Resp* is intractable[2]
- Can we compute it faster when we have access to the internals?

  This kind of research was done for *Shap*

[2] Bertossi; TPLP'23

Coalition Games and the Shapley Value

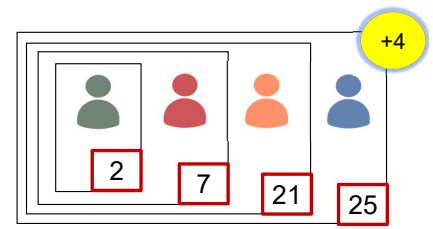
- Usually *several tuples together* produce a query result

  And *several feature values* lead to a classification label
- Like players in a *coalition game* contributing, possibly differently, to a shared wealth-distribution function
- Apply standard measures used in game theory: the Shapley value of a player (as a measure of its contribution)
- The Shapley value is a established measure of contribution by players to a wealth function
- It emerges as the only measure enjoying certain properties
- We need a game (function) ...

- Set of players D, and game function $\mathcal{G}: \quad \mathcal{P}(D) \longrightarrow \mathbb{R}$
  ($\mathcal{P}(D)$ 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



- For each application one defines an appropriate game function

- Shapley is difficult to compute

  Naive approach: exponentially many counterfactual combinations

- Actually, Shapley computation is #P-hard in general
- A complexity class of (possibly implicitly) computational counting problems
- Being #P-hard is evidence of difficulty: $\#SAT$ is #P-hard

  Counting satisfying assignments for a propositional formula

  At least as difficult as SAT

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:

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

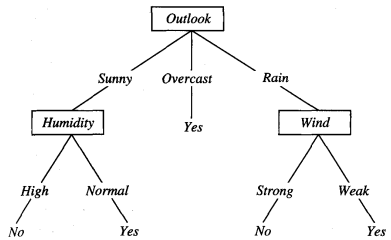
- For a feature $F^\star \in \mathcal{F}$, compute: $\mathit{Shap}(\mathcal{F}, \mathcal{G}_{\mathbf{e}}, F^\star)$

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

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

- *Shap* 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?
- This was an open problem

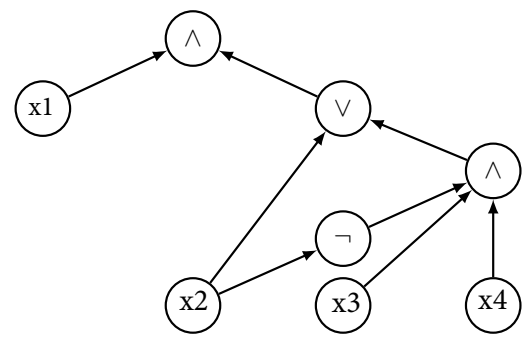
  For decision trees, the conjecture was: Yes!

  There had been wrong proofs ...

Tractability for BC-Classifiers: Big Picture

- We investigated this problem in detail[3]
- Tractable and intractable cases, with algorithms for the former

  Investigated approximation algorithms
- Choosing the right abstraction (model) is crucial
- We considered Boolean-Circuit Classifiers (BCCs), i.e. propositional formulas with a (binary) output gate
- We had shown before that *Shap* is intractable for "Monotone 2CNF" classifiers under the product distribution
  (at most 2 variables per clause, and positive)
- So, it had to be a broad and interesting class of BCs



[3] Arenas, Bertossi, Barcelo, Monet; AAAI'21; JMLR'23

Shap for Boolean-Circuit Classifiers

- Features: $F_i \in \mathcal{F},\ i = 1, \ldots, n$ $\qquad Dom(F_i) = \{0, 1\}$

  Input entity: $\mathbf{e} \in \mathcal{E} := \{0, 1\}^n$ $\qquad L(\mathbf{e}) \in \{0, 1\}$

- There is 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 $\mathbf{e}$, $F^\star$, and L)

- $SAT(L) := \{\mathbf{e}' \in \mathcal{E} \mid L(\mathbf{e}') = 1\}$ $\qquad \#SAT(L) := |SAT(L)|$

  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 $\mathbf{e} \in \mathcal{E}$

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

- When $\#SAT(L)$ is hard for a BC L, *Shap* also has to be hard
- Corollary: Computing *Shap* is $\#P$-hard for Boolean classifiers defined by Monotone 2DNF or Monotone 2CNF
  (Provan & Ball, 1983)
- Can we do better for other classes of binary classifiers?

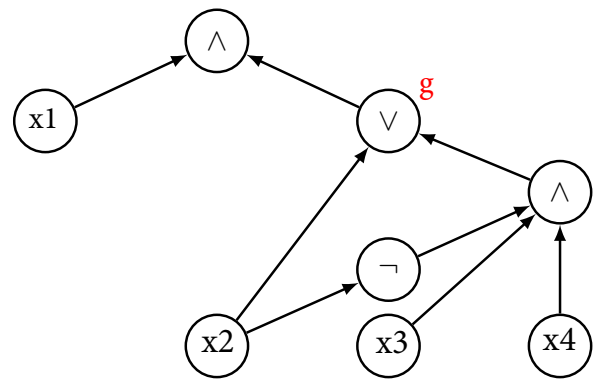
  Other classes of Boolean-circuit classifiers?

Deterministic and Decomposable BCs

- A Boolean circuit over set of variables X is a DAG $\mathcal{C}$ with:
  - Each input (source) node labeled with a variable or a constant in $\{0, 1\}$
  - Other nodes labeled with a gate in $\{\neg, \wedge, \vee\}$
  - Single sink node, O, the output
- For gate g of $\mathcal{C}$, $\mathcal{C}(g)$ is the induced subgraph containing gates on a path in $\mathcal{C}$ to g

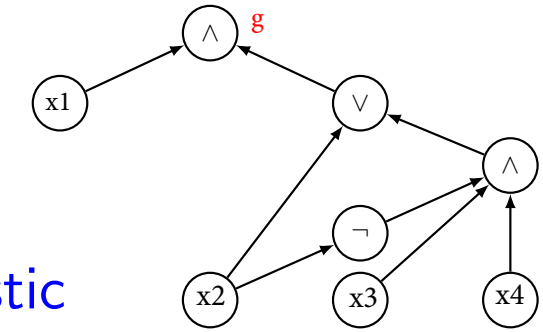
  $Var(g)$ is the set of variables of $\mathcal{C}(g)$

  $Var(g) = \{x2, x3, x4\}$



- $\mathcal{C}$ 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 ... (more coming)

Shap for dDBCs

- Proposition: For dDBCs $\mathcal{C}$, $\#SAT(\mathcal{C})$ can be computed in polynomial time

  Idea: Bottom-up procedure that inductively computes $\#SAT(\mathcal{C}(g))$, for each gate g of $\mathcal{C}$

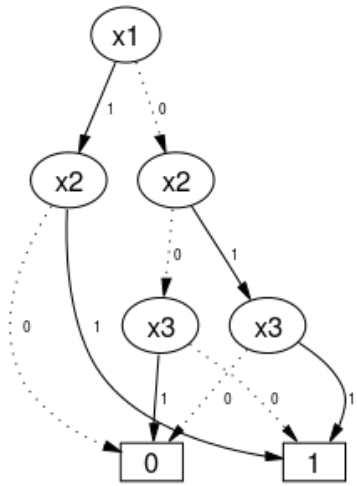
- $\not\Longrightarrow$ the same for *Shap*, but gives us some hope ...
- To show that *Shap* can be computed efficiently for dDBCs, we need a detailed analysis
- We assume the uniform distribution for the moment
- Theorem: *Shap* can be computed in polynomial time for dDBCs under the uniform distribution
- It can be extended to any product distribution on $\mathcal{E}$

- 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)

    $(\neg x_1 \wedge \neg x_2 \wedge \neg x_3) \vee (x_1 \wedge x_2) \vee (x_2 \wedge x_3)$

    Compact representation of Boolean formulas

    Compatible variable orders along full paths



  - Sentential decision diagrams (SDDs)

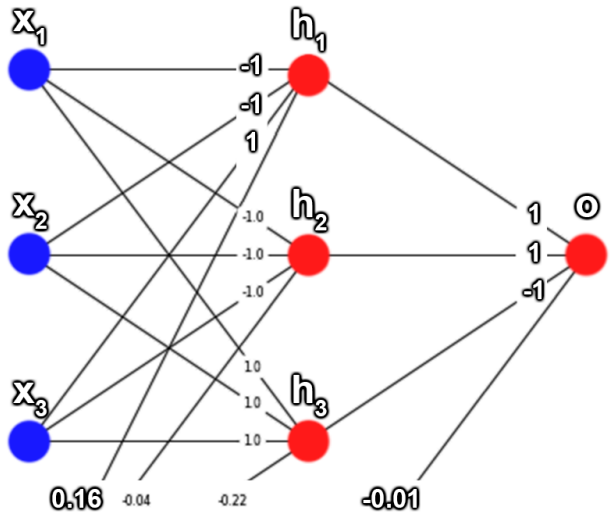
    Generalization of OBDDs
  - Deterministic-decomposable negation normal-form (dDNNFs)

    As dDBC, with negations affecting only input variables
- All the latter relevant in *Knowledge Compilation*
- An optimized efficient algorithm for *Shap* computation can be applied to any of these

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[4]
- Even if the compilation is not entirely in 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

[4] Bertossi, Leon; JELIA'23



$$\phi_g(\bar{i}) \quad = \quad sp(\bar{w}_g \bullet \bar{i} + b_g)$$

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

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

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

- Done using always CNFs and keeping them "short" ...

(room for optimizations)

- 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

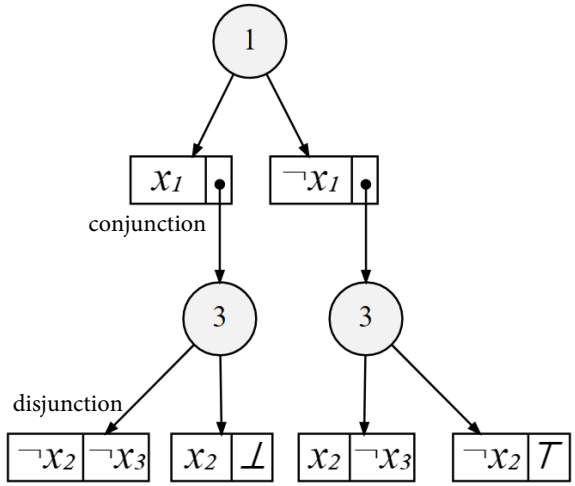
  It succinctly represents the CNF

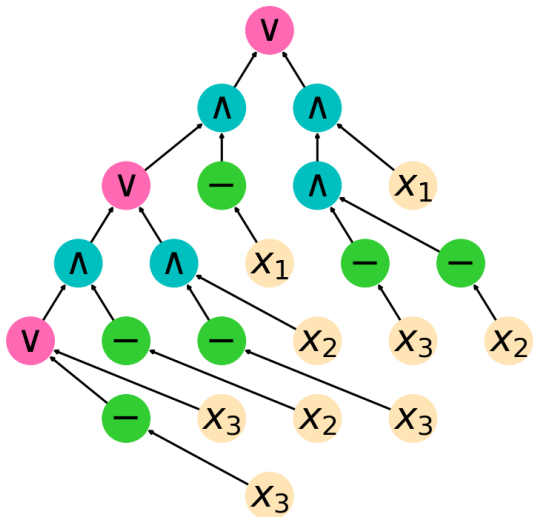
- The expensive, provably intractable compilation step

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

  TW of the associated undirected graph: an edge between variables if together in a clause

  A measure of how close it is to a tree
  (In example, graph is clique, TW is #vars -1 =2)





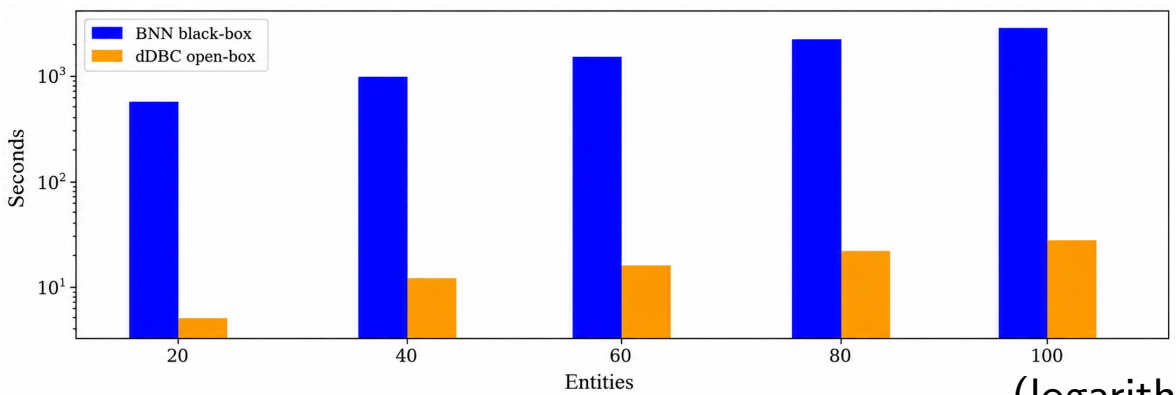
- SDD is easily transformed into a dDBC

- 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

  (room for optimizations)
- A one-time computation that fully replaces the BNN
- We compared *Shap* computation time for black-box BNN and open-box dDBC
- Total time for computing *all Shap scores for all entities*, with increasing numbers of them



(logarithmic scale)
- The uniform distribution was used

Some Research Directions

- 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

  They naturally arise when no more information available about the distribution

  How far can we go with other distributions?

  Do we still have an efficient algorithm?

- Explanation scores commonly use the classifier plus a probability distribution over the underlying entity population

  Imposing or using explicit and additional domain semantics or domain knowledge is relevant to explore

  Can we modify *Shap*'s definition and computation accordingly?

  Or the probability distribution?

- All of the above apply to *Resp*

The Need for Reasoning

- Counterfactual entities can be interesting *per se*

  They allow us to explore alternative scenarios, with different outcomes

- How can we rank them?

  Apart from the obvious: size of associated contingency sets

- What else can be do with attribution scores and counterfactual explanations?

- We can reason about/with them, analyze them, select some of them, aggregate them, etc.

  In interaction with both attribution-score model/algorithm or classifier, for further exploration

  For global understanding of the classifier or application domain

- We need tools for conveying or imposing domain knowledge (domain semantics), e.g. an age never decreases

  Only some counterfactuals may make sense

  Some combinations of feature values may not be allowed

  Some changes may "trigger" other changes

  To impose preferences on counterfactuals

- We need tools for doing this kind of logical reasoning

- We need tools for posing and answering queries about explanations

  Are there explanations with this particular property?

  Or any two that differ by ...?

- Specification and computation of actionable counterfactuals, a.k.a. recourses

  With them one can do something in practice

  Or others with a different preferred property

- On-the-fly interaction with different ML models and scores

  Do I get same score with this different ML system?

  Or this other attribution score (definition, algorithm or implementation)?

- Imposing conditions on feature values

  What if I leave some feature values fixed?

  Do I get same high-score feature with this "similar" entity?

  Is there a high-score counterfactual version of the entity that changes this specific feature?

  Or never changes that one?

- We have devised *declarative* (logic-based) methods to *reason* with and about counterfactuals, and compute *Resp* scores

  In interaction with classifiers "called" from the program

- We have used *Answer-Set Programming*, a form of logic programming

  > L. Bertossi. "Declarative Approaches to Counterfactual Explanations for Classification". Theory and Practice of Logic Programming, 23 (3): 559–593, 2023.

Much in the direction of Neuro-Symbolic AI!