Applying Logic to the Specification and Computation of Attribution Scores in Explainable Machine Learning

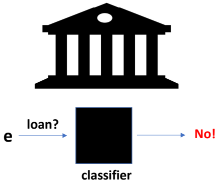
Leopoldo Bertossi

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, ...

- $\mathbf{e}$ requests a loan from a bank that uses a classifier



- The client asks *Why?*

- What kind of *explanation?*

  How?

  From what?

- Some of them are *causal explanations*, some are *explanation scores* a.k.a. *attribution scores*
- They quantify the relevance of each feature value in $\mathbf{e}$ for the assigned label
- Here two of them:
  - *Shap* (based on Shapley value of Coalition Game Theory)
  - *Resp* (Responsibility, based on Actual Causality)
- We will consider only binary features and a binary classifier

  Entity population $\mathcal{E} = \{0,1\}^N$

  Classifier $L\colon\ \mathcal{E} \to \{0,1\}$

Shap **Score**

- Set of players $\mathcal{F}$ contain features, relative to classified entity $\mathbf{e}$
- An appropriate $\mathbf{e}$-dependent game function (shared wealth-function) mapping subsets of players to real numbers
- For $S \subseteq \mathcal{F}$, and $\mathbf{e}_S$ the projection of $\mathbf{e}$ on S:

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

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

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

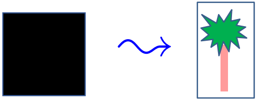
(Lee & Lundberg, 2017)

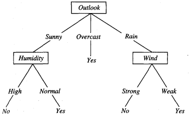
- Assumes a probability distribution on entity population $\mathcal{E}$

- *Shap*: Exponentially many subsets of players, and multiple passes through a possibly black-box classifier

  Shap computation is #P-hard in general

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





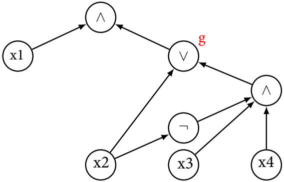
  Exploiting its elements and internal structure?

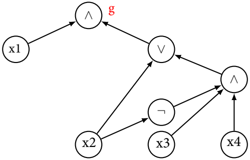
- A decision tree, or a random forest, or a Boolean circuit?

- Can we compute *Shap* in polynomial time?

Tractability for BC-Classifiers

- Theorem: *Shap* can be computed in polynomial time for dDBCs under the uniform distribution[1]





Deterministic and Decomposable Boolean Circuit

- Can be extended to a product distribution on $\mathcal{E} = \{0, 1\}^N$
- They (and related models) are relevant in *Knowledge Compilation*

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

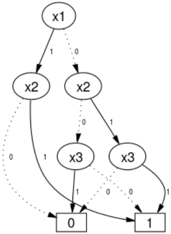
- Corollary: Via polynomial time transformations, under the uniform and product distributions, *Shap* can be computed in polynomial time for

  - Decision trees (and random forests)
  - Ordered binary decision diagrams (OBDDs)

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

    Compatible variable orders along full paths

    Compact representation of Boolean formulas



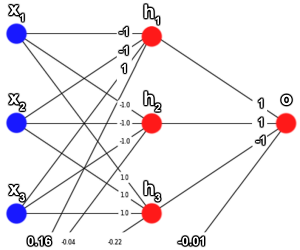
  - Sentential decision diagrams (SDDs)
    Generalization of OBDDs
  - Deterministic-decomposable negation normal-form (dDNNFs)
    As dDBC, with negations affecting only input variables

- An optimized efficient algorithm for *Shap* computation can be applied to all of these

Shap on Neural Networks

- Binary Neural Networks (BNNs) are commonly considered black-box models
- We experimented with *Shap* computation with a black-box BNN and with its compilation into a dDBC[2]
- 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

[2] Bertossi, Leon; JELIA'23



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

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

- BNN described by a propositional formula, which is further transformed into an optimized CNF
- Actually, 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

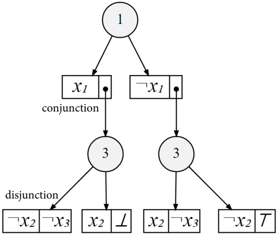
  Succinctly representing the CNF

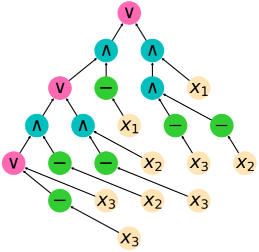
- The expensive compilation step

  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

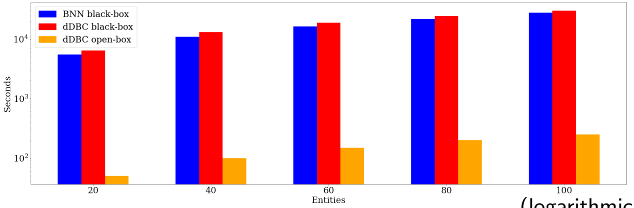
  (In example, graph is clique, TW is #vars -1 =2)





- The SDD is easily transformed into a dDBC
- *Shap* computed on it, possibly multiple times
- The uniform distribution was used

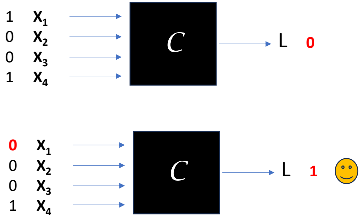
- In our experiments, we used a BNN with 14 gates
- Compiled into dDBC with 18,670 nodes (room for optimizations)
- A one-time computation that fully replaces the BNN
- Compared *Shap* computation time for: black-box BNN, open-box dDBC, and black-box dDBC
- Total time for computing *all Shap scores for all entities*, with increasing numbers of them



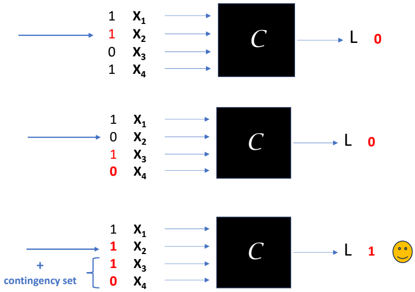
(logarithmic scale)

Resp: Causal Responsibility

- Actual Causality is based on counterfactual interventions
  (Halpern & Pearl, 2001)
- Hypothetical changes of values in a causal model to detect other changes ... identifying then actual causes
- Do changes of feature values change the label from 0 to 1?
- A measure of causal contribution: Responsibility
  (Chockler & Halpern, 2004)



Counterfactual cause for
$L = 0$: $\mathbf{x_1} = 1$
$Resp(\mathbf{x_1}) := 1$
maximum responsibility



concentrate on $\mathbf{x_2}$: not counterfactual cause

changes on $\mathbf{x_3}, \mathbf{x_4}$ do not change label

change on $\mathbf{x_2}$ *accompanied by* changes on $\mathbf{x_3}, \mathbf{x_4}$ does change label!

- $\Gamma = \{\mathbf{x_3}, \mathbf{x_4}\}$ is contingency set for $\mathbf{x_2}$
- $\mathbf{x_2}$ is actual cause for $L = 0$
- If Γ is minimum-size contingency set for $\mathbf{x_2}$:

$$Resp(\mathbf{x_2}) := \frac{1}{1+|\Gamma|} = \frac{1}{3}$$

- We call $\langle 1, 1, 1, 0 \rangle$ a counterfactual (version) of original entity

The Need for Reasoning

- Logical specification of counterfactual interventions and *Resp*
- Logical reasoning for interaction with attribution-score spec/algorithm and classifier for further exploration
- Reason about interventions and counterfactuals
- Compute responsibility scores, and reason about them
- Impose semantic constraints on counterfactuals
- Counterfactuals can be queried or reasoned about
  - Specification of actionable counterfactuals?
  - Some actionable high-score feature value?
  - Specs of other counterfactuals of interest? Computing them?
  - What if I leave some feature values fixed?
  - Do I get same high-score feature with this "similar" entity?
  - Any high-score counterfactual version that changes this feature?

    Or never changes that one? ETC.

- Usually interested in maximum-responsibility feature values (associated to minimum-cardinality contingency sets)
- We have used Answer-Set Programming (ASP)
  - Declarative language, and reasoning via QA
  - Possibly several answer-sets (models)
  - Each counterfactual version leading to a new label corresponds to an answer set (model)
  - Minimality of answer-sets, and closed-world assumption
  - Non-monotonicity, and commonsense reasoning (persistence)
  - Program and semantic constraints (the latter on counterfactuals)
  - Required expressive power and computational complexity
  - Weak constraints (useful for specifying minimum cardinalities)
  - Set and numerical aggregations (useful for score computation)
  - Predicates for interaction with external classifiers
  - Reasoning is enabled by cautious and brave query answering

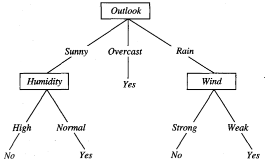
    True in all models vs. true in some model

ASPs for Counterfactual Interventions

- Here, decision-trees (also done for external naive-Bayes via Python)
- A decision tree

  Features $\mathcal{F} = \{\text{Outlook}, \text{Humidity}, \text{Wind}\}$
  $Dom(\text{Outlook}) = \{\text{sunny}, \text{overcast}, \text{rain}\}$
  $Dom(\text{Humidity}) = \{\text{high}, \text{normal}\}$
  $Dom(\text{Wind}) = \{\text{strong}, \text{ weak}\}$

  Entity $\mathbf{e} = ent(\text{sunny}, \text{normal}, \text{weak})$ gets label 1



- *Counterfactual Intervention Programs* (CIPs) specify counterfactual interventions on a given entity under classification
- We use *DLV* and *DLV-Complex* notation (the system we used)
- Use annotation constants:

Annotation	Intended Meaning
o	original entity
do	do counterfactual intervention
tr	entity in transition
s	stop, label has changed (single change of feature value)

- Specifying domains, entity, classification tree, annotations:

```
% facts:
   dom1(sunny). dom1(overcast). dom1(rain). dom2(high). dom2(normal).
   dom3(strong). dom3(weak).
   ent(e,sunny,normal,weak,o).   % original entity at hand

% specification of the decision-tree classifier:
   cls(X,Y,Z,1) :- Y = normal, X = sunny, dom1(X), dom3(Z).
   cls(X,Y,Z,1) :- X = overcast, dom2(Y), dom3(Z).
   cls(X,Y,Z,1) :- Z = weak, X = rain, dom2(Y).
   cls(X,Y,Z,0) :- dom1(X), dom2(Y), dom3(Z), not cls(X,Y,Z,1).

% transition rules: the initial entity or one affected by a value change
   ent(E,X,Y,Z,tr) :- ent(E,X,Y,Z,o).
   ent(E,X,Y,Z,tr) :- ent(E,X,Y,Z,do).
```

- The main, counterfactual rule:

```
% counterfactual rule: alternative single-value changes
   ent(E,Xp,Y,Z,do) v ent(E,X,Yp,Z,do) v ent(E,X,Y,Zp,do) :-
                      ent(E,X,Y,Z,tr), cls(X,Y,Z,1), dom1(Xp), dom2(Yp),
                      dom3(Zp), X != Xp, Y != Yp, Z!= Zp,
                      chosen1(X,Y,Z,Xp), chosen2(X,Y,Z,Yp),
                      chosen3(X,Y,Z,Zp).
```

- Only one disjunct in the head becomes true; one per feature

- Uses three non-deterministic choice predicates

  Chooses a new value in last argument for each combination of the first three

- While the label stays 1

- Choice predicate can be specified

  Choice makes the program non-stratified

- A program constraint prohibiting going back to initial entity:

```
% Not going back to initial entity (program constraint):
   :- ent(E,X,Y,Z,do), ent(E,X,Y,Z,o).
```

  Acts by eliminating models that violate it

  Also contributes to non-stratification

- Non-stratified negation is what makes ASP necessary
- Each counterfactual version represented by a model
- Next rule defines "stop" annotation, when label becomes 0

```
% stop when label has been changed:
   ent(E,X,Y,Z,s) :- ent(E,X,Y,Z,do), cls(X,Y,Z,0).
```

```
% collecting changed values for each feature:
   expl(E,outlook,X)   :- ent(E,X,Y,Z,o), ent(E,Xp,Yp,Zp,s), X != Xp.
   expl(E,humidity,Y)  :- ent(E,X,Y,Z,o), ent(E,Xp,Yp,Zp,s), Y != Yp.
   expl(E,wind,Z)      :- ent(E,X,Y,Z,o), ent(E,Xp,Yp,Zp,s), Z != Zp.

   entAux(E) :- ent(E,X,Y,Z,s).        % auxiliary predicate to
                                       % avoid unsafe negation
                                       % in the constraint below
   :- ent(E,X,Y,Z,o), not entAux(E).   % discard models where
                                       % label does not change

% computing the inverse of x-Resp:
   invResp(E,M) :- #count{I: expl(E,I,_)} = M, #int(M), E = e.
```

- Rules above for collecting changes, leading to score computation
- Sets of changes (in each model) is minimal (for free with ASP)
- Second last is program constraint: gets rid of models with unchanged label
- Last rule contains aggregation for counting number of feature value changes
- For each counterfactual version (or model) this is a "local" *Resp*-score associated to a minimal set of changes

  Not necessarily the "global" *Resp*-score yet

```
{ent(e,sunny,normal,weak,o), cls(sunny,normal,strong,1),
 cls(sunny,normal,weak,1), cls(overcast,high,strong,1),
 cls(overcast,high,weak,1), cls(rain,high,weak,1),
 cls(overcast,normal,weak,1), cls(rain,normal,weak,1),
 cls(overcast,normal,strong,1), cls(sunny,high,strong,0),
 cls(sunny,high,weak,0), cls(rain,high,strong,0),
 cls(rain,normal,strong,0), ent(e,sunny,high,weak,do),
 ent(e,sunny,high,weak,tr), ent(e,sunny,high,weak,s),
 expl(e,humidity,normal),invResp(e,1)}

{ent(e,sunny,normal,weak,o), cls(sunny,normal,strong,1),...,
 cls(rain,normal,strong,0), ent(e,rain,normal,strong,do),
 ent(e,rain,normal,strong,tr), ent(e,rain,normal,strong,s),
 expl(e,outlook,sunny), expl(e,wind,weak), invResp(e,2)}
```

- Two stable models of the CIP
- Two counterfactuals with minimal contingency sets
- Only first is minimum counterfactual version: $\mathit{Resp}(\mathbf{e}) = 1$

  More precisely: $\mathit{Resp}(\mathbf{e}{\downarrow}\text{Humidity}) = 1$
- Want only maximum responsibility counterfactuals?

- Introduce weak program constraints

```
% Weak constraints to minimize number of changes:
   :~ ent(E,X,Y,Z,o), ent(E,Xp,Yp,Zp,s), X != Xp.
   :~ ent(E,X,Y,Z,o), ent(E,Xp,Yp,Zp,s), Y != Yp.
   :~ ent(E,X,Y,Z,o), ent(E,Xp,Yp,Zp,s), Z != Zp.
```

- Weak program constraints can be violated, but only a minimum number of times
- Minimize number of feature value differences between **e** and its counterfactuals
- Only first model is kept (as on preceding slide)

- Reasoning enabled by query answering

  Under cautious and brave semantics:

  - *Responsibility of feature* Outlook*?*
  - *A counterfactual version with less than 3 changes?*

```
invRresp(e,outlook,R)?
fullExpl(E,U,R,S), R<3?
```

(brave semantics)

  - *An intervened entity with combination of sunny outlook and strong wind, and its label?*
  - *All intervened entities that obtain label* No*?*

```
cls(E,O,T,H,W,_), O = sunny, W = strong?
cls(E,O,T,H,W,no)?
```

  - *Does the wind not change under every counterfactual version?*

```
ent(e,_,_,_,Wp,s), ent(e,_,_,_,W,o), W = Wp?
```

(cautious semantics)

- Adding domain knowledge very easy
- In a particular domain, there may never be rain with strong wind — Discard such a model!

```
% hard constraint disallowing a particular combination
   :- ent(E,rain,X,strong,tr).
```