Tractability and Optimization of Shap-Score Computation for Explainable AI

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, ...

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, ...

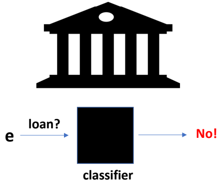
- $\mathbf{e}$ requests a loan from a bank that uses a classifier

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



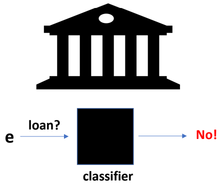
Explanations in Machine Learning

- 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?*

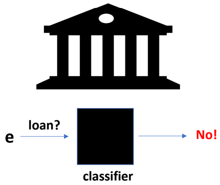


Explanations in Machine Learning

- 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?

- Explanations come in different forms

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

- 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

- 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 score definition and computation

  In data management and machine learning

- 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 score definition and computation

  In data management and machine learning
- Some of them (in data management or ML)
  - Responsibility (in its original and generalized versions)
  - The Causal Effect score
  - The Shapley value (as Shap in ML)

Shap Scores

- Based on the general Shapley value

Shap **Scores**

- Based on the general Shapley value
- Set of players $\mathcal{F}$ contain features, relative to classified entity $\mathbf{e}$

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

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:

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

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 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)

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

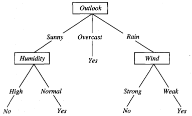
- *Shap* may end up considering exponentially many combinations, and multiple passes through the black-box classifier

  Shap computation is #P-hard in general

- *Shap* may end up considering exponentially many combinations, and multiple passes through the black-box classifier

  Shap computation is #P-hard in general

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

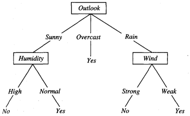


  Exploiting its elements and internal structure?

- *Shap* may end up considering exponentially many combinations, and multiple passes through the black-box classifier

  Shap computation is #P-hard in general

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



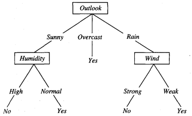
  Exploiting its elements and internal structure?

- What if we have a decision tree, or a random forest, or a Boolean circuit?

- *Shap* may end up considering exponentially many combinations, and multiple passes through the 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?

- 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

- We investigated this problem in detail[1]

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

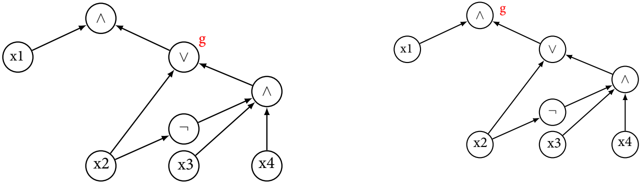
Tractability for BC-Classifiers

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

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

Tractability for BC-Classifiers

- We investigated this problem in detail[1]
- Theorem: *Shap* can be computed in polynomial time for dDBCs under the uniform distribution
- Can be extended to a product distribution on $\mathcal{E} = \{0,1\}^N$



[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

- Corollary: Via polynomial time transformations, under the uniform and product distributions, *Shap* can be computed in polynomial time for
  - Decision trees (and random forests)

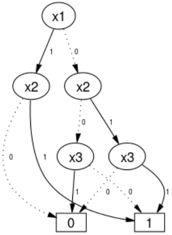
- Corollary: Via polynomial time transformations, under the uniform and product distributions, *Shap* can be computed in polynomial time for
  - Decision trees (and random forests)
  - Ordered binary decision diagrams (OBDDs)

- 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

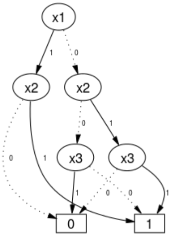


- Corollary: Via polynomial time transformations, under the uniform and product distributions, *Shap* can be computed in polynomial time for
  - Decision trees (and random forests)
  - Ordered binary decision diagrams (OBDDs)

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

    Compatible variable orders along full paths

    Compact representation of Boolean formulas



  - Sentential decision diagrams (SDDs)

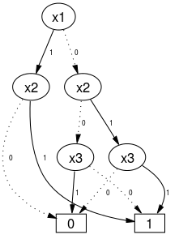
    Generalization of OBDDs

- 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)

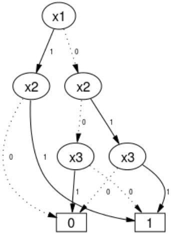
    As dDBC, with negations affecting only input variables

- 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)

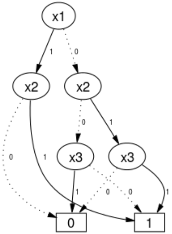
    As dDBC, with negations affecting only input variables
- All the latter relevant in *Knowledge Compilation*

- 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
- 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

[2] Bertossi, Leon; JELIA'23

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

[2] Bertossi, Leon; JELIA'23

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

[2] Bertossi, Leon; JELIA'23

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[2]

[2] Bertossi, Leon; JELIA'23

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[2]
- Even if the compilation is not entirely of polynomial time, it may be worth performing this one-time computation

[2] Bertossi, Leon; JELIA'23

Shap on Neural Networks

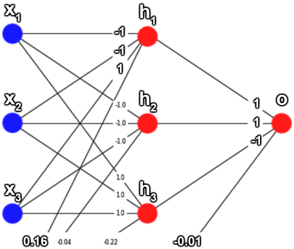
- Binary Neural Networks (BNNs) are commonly considered black-box models
- Naively computing *Shap* on a BNN is bound to be complex
- Better try to compile the BNN into an open-box BC where *Shap* can be computed efficiently
- We have experimented with *Shap* computation with a black-box BNN and with its compilation into a dDBC[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

Shap on Neural Networks

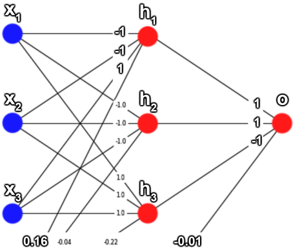
- Binary Neural Networks (BNNs) are commonly considered black-box models
- Naively computing *Shap* on a BNN is bound to be complex
- Better try to compile the BNN into an open-box BC where *Shap* can be computed efficiently
- We have experimented with *Shap* computation with a black-box BNN and with its compilation into a dDBC[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
- We illustrate the approach by means of an example

[2] Bertossi, Leon; JELIA'23



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

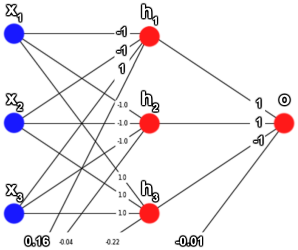
$$:= \begin{cases} 1 & \text{if} \quad \bar{w}_g \bullet \bar{i} + b_g \geq 0, \\ -1 & \text{otherwise,} \end{cases}$$



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

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

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



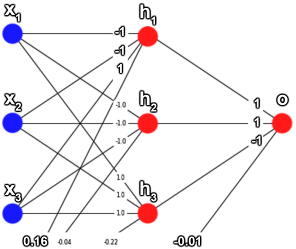
$$
\begin{aligned}
\phi_g(\bar{i}) &= sp(\bar{w}_g \bullet \bar{i} + b_g) \\
&:= \begin{cases} 1 & \text{if } \bar{w}_g \bullet \bar{i} + b_g \geq 0, \\ -1 & \text{otherwise,} \end{cases}
\end{aligned}
$$

- 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)



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

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

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

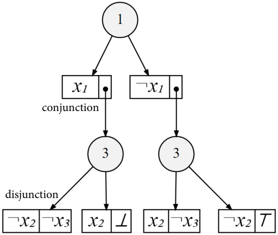
- Done using always CNFs and keeping them "short" ...

(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

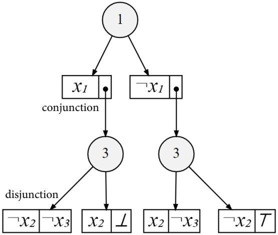


- The CNF is transformed into an SDD

  It succinctly represents the CNF

- The expensive compilation step

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



- The CNF is transformed into an SDD

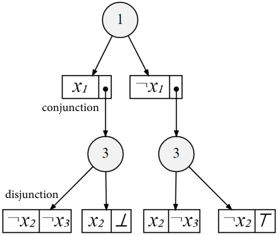
  It succinctly represents the CNF

- The expensive compilation step

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

  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)



- The CNF is transformed into an SDD

  It succinctly represents the CNF

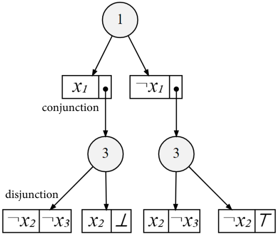
- The expensive compilation step

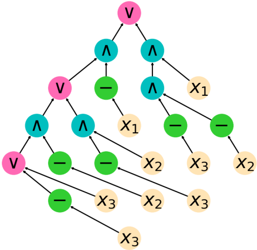
  But upper-bounded by an exponential only in the tree-width of the CNF

  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)

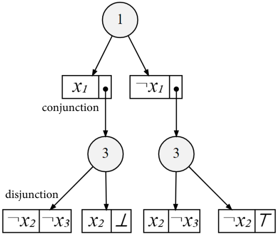
- The SDD is easily transformed into a dDBC





- The CNF is transformed into an SDD

  It succinctly represents the CNF

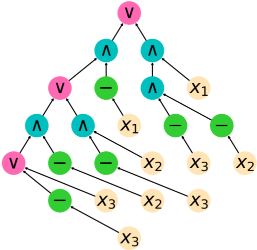


- The expensive compilation step

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

  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)



- The SDD is easily transformed into a dDBC
- On it *Shap* is computed, possibly multiple times

- The CNF is transformed into an SDD

  It succinctly represents the CNF

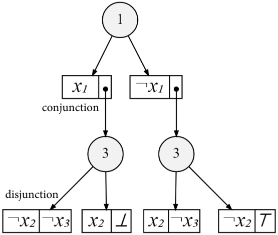
- The expensive compilation step

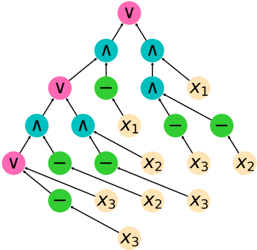
  But upper-bounded by an exponential only in the tree-width of the CNF

  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)

- 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

- In our experiments, we used a BNN with 14 gates
- It was compiled into a dDBC with 18,670 nodes

(room for optimizations)

- In our experiments, we used a BNN with 14 gates
- It was compiled into a dDBC with 18,670 nodes

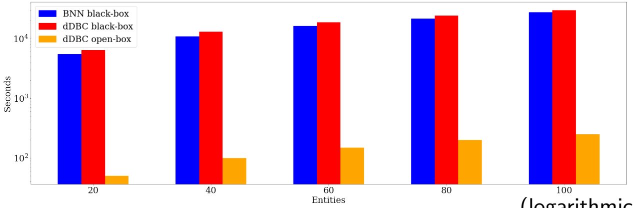
  (room for optimizations)
- A one-time computation that fully replaces the BNN

- 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, open-box dDBC, and black-box dDBC

- In our experiments, we used a BNN with 14 gates
- It was compiled into a dDBC with 18,670 nodes

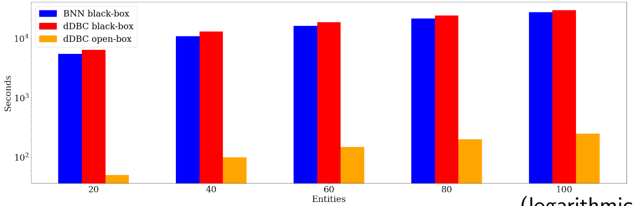
  (room for optimizations)
- A one-time computation that fully replaces the BNN
- We compared *Shap* computation time for black-box BNN, open-box dDBC, and black-box dDBC
- Total time for computing *all Shap scores for all entities*, with increasing numbers of them



(logarithmic scale)

- 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, open-box dDBC, and black-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

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

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

- 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

- 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?

- 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?

- Shapley values satisfy desirable properties for general coalition game theory

- 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?

- Shapley values satisfy desirable properties for general coalition game theory

  Existing scores have been criticized or under-explored in terms of general properties

- 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?

- Shapley values satisfy desirable properties for general coalition game theory

  Existing scores have been criticized or under-explored in terms of general properties

  Specific general and expected properties for Explanations Scores (in AI)?