Carleton UNIVERSITY

Millennium Institute Foundational Research on Data

Computing Attribution Scores for Classification in Explainable AI

Prof. Leopoldo Bertossi (PhD Math)

Explanations in Machine Learning

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

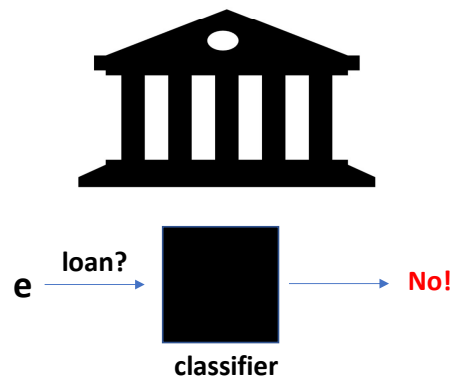
  An entity as a record of feature values: Name, Age, Activity, Income, ...

- **e** requests a loan from a bank that uses a classifier
- Loan is denied
- 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 and local (feature value in an entity)

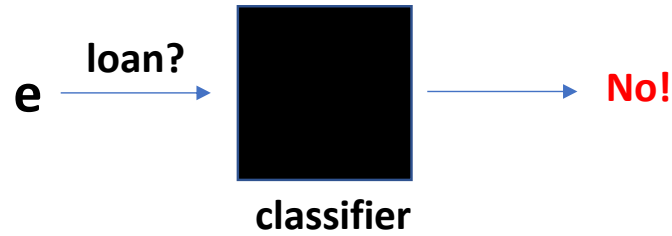
A Score-Based Approach: Responsibility

- 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 a great explanation

  Similarly for features in a potential contingency set
- Better consider average labels obtained via counterfactual interventions
- *Resp*, our extended version of responsibility, 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 $F^\star$ (whose values vary)

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

  Label different from $L(\mathbf{e})$, in average; and with contingencies

- We are interested in maximum-score feature values

  Associated to minimum (cardinality) contingency sets

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

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

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.
- 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)
- 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) ...

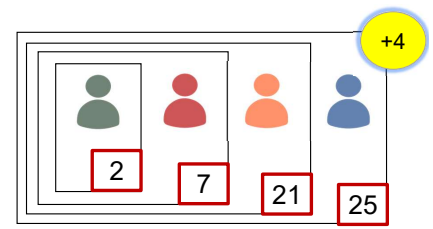
- Set of players D, and game function $\mathcal{G}: \quad \mathcal{P}(D) \longrightarrow \mathbb{R}$
  ($\mathcal{P}(D)$ the power set of D)

- 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:

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

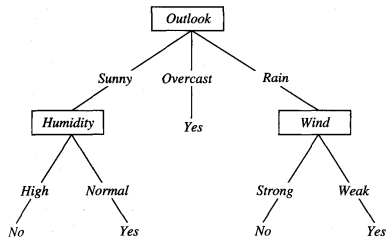
- For a feature $F^\star \in \mathcal{F}$, compute: $\mathit{Shap}(\mathcal{F}, \mathcal{G}_{\mathbf{e}}, F^\star)$

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

- *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

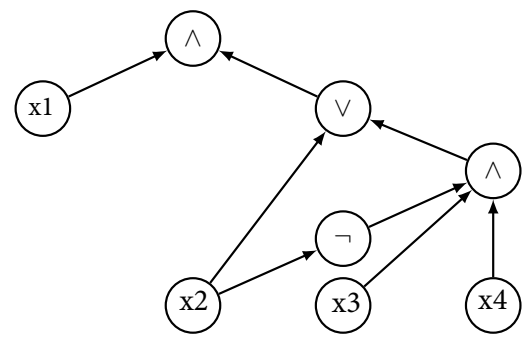
  For decision trees, the conjecture was: Yes!

  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

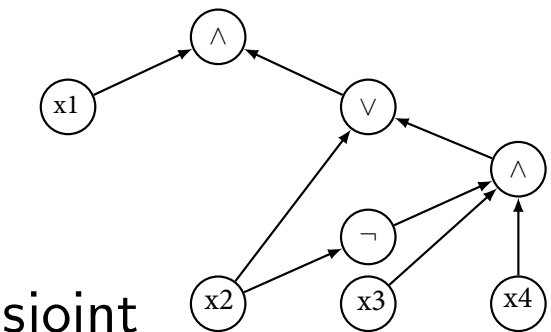
- We established that *Shap* is at least as hard as model counting for the BC:

  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:** (dDBCs)
  - **Determinism**: Inputs to an $\vee$-gate cannot be both true
  - **Decomposability**: Variables hanging from different inputs to an $\wedge$-gate are disjoint



- *Shap* computation in PTIME not initially precluded
- dDBCs include -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 $\mathcal{C}$

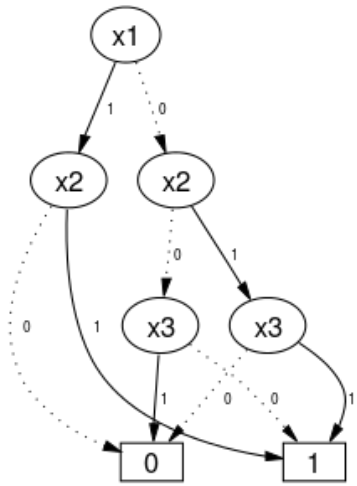
- $\not\Longrightarrow$ the same for *Shap*, but gives us some hope ...
- To show that *Shap* can be computed efficiently for dDBCs, we need a detailed analysis
- We assume the uniform distribution for the moment
- Theorem: *Shap* can be computed in polynomial time for dDBCs under the uniform distribution
- 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)

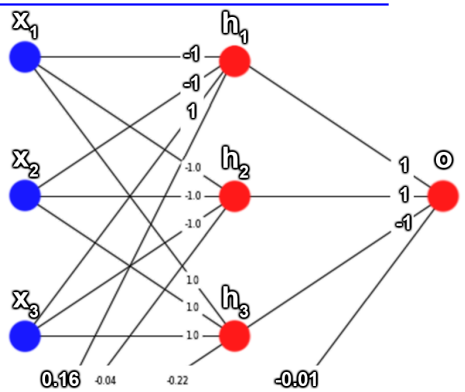
    Generalization of OBDDs
  - Deterministic-decomposable negation normal-form (dDNNFs)

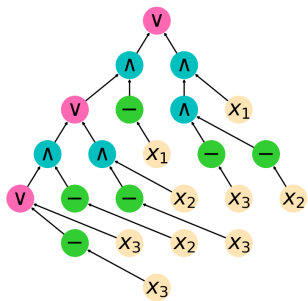
    As dDBC, with negations affecting only input variables
- All the latter relevant in *Knowledge Compilation*
- An optimized efficient algorithm for *Shap* computation
- NNs not in the list above: What if we try?

SHAP on Binary Neural Networks

- NNs considered as black-boxes
- We experimented with SHAP computation on a BNN via compilation into a dDBC
  1. BNN $\mapsto$ CNF (parsimonious and optimized)
  2. CNF $\mapsto$ SDD (non-polynomial, but FPT)
  3. SDD $\mapsto$ dDBC (straightforward)

  Still worth this one-time computation
  (target dDBC may be used multiple times)

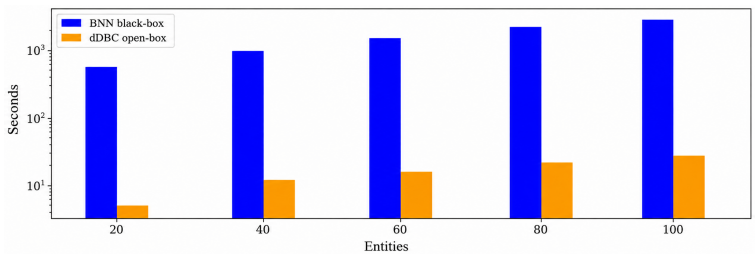




- Experiments: BNN with 14 gates, dDBC with 18,670 nodes

  Compared SHAP computation for: black-box BNN and open-box dDBC

- All SHAP scores for all entities, with increasing numbers of them[4]



(logarithmic scale)

[4] Bertossi, Leon; JELIA'23

The Need for Reasoning

- Counterfactual entities can be interesting *per se*

  To explore alternative scenarios, with different outcomes

- Need for tools for imposing domain knowledge (domain semantics), e.g. an age never decreases

  Specify and compute counterfactuals that make sense, and can be used in practice: actionable, recourses

  Some combinations of feature values may not be allowed

  Some changes may "trigger" other changes

  To impose preferences on counterfactuals

  Conditions on feature values

- We need tools for logical reasoning

- For posing and answering queries about explanations

- What if I leave some feature values fixed?
- Do I get same high-score feature with this "similar" entity?

  So as "actionability", similarity can be declaratively specified
- Is there a high-score counterfactual that changes an specific feature? Or never changes that one?
- On-the-fly interaction with ML models
- We developed a *declarative*, logic-based, method to *reason* with and about counterfactuals, and compute *Resp*-scores
- We used *Answer-Set Programming*, a form of logic programming[5]
- Much in the direction of Neuro-Symbolic AI!

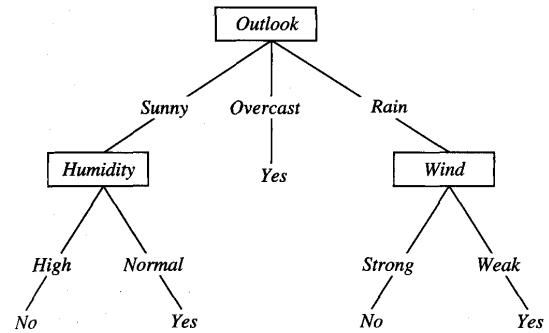
[5] L. Bertossi. "Declarative Approaches to Counterfactual Explanations for Classification". *TPLP*, 2023.

Counterfactual ASPs: An Example

- 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)
- 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
- 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).
```

- Rules for collecting changes, leading to score computation:

```
% 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.
```

- Last rule contains aggregation for counting number of feature value changes

- Sets of changes (in each model) is minimal (for free with ASP)
- 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
- Two answer-sets of the CIP:

```
{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 counterfactuals with minimal contingency sets
- Only first is minimum counterfactual version: $Resp(\mathbf{e}) = 1$

  More precisely: $Resp(\mathbf{e}[\text{Humidity}]) = 1$

- Want only maximum-responsibility counterfactuals?
- Introduce weak program constraints

  They can be violated, but only a minimum number of times

```
% 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.
```

- Minimize number of feature value differences between **e** and its counterfactuals
- Only first model is kept
- 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).
```

- Reasoning enabled by query answering

  Under cautious and brave semantics:

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

    ```
    invResp(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)