Counterfactual and Score Based Explanations in Data Management and Machine Learning: An Answer-Set Programming Approach

Leopoldo Bertossi

Universidad Adolfo Ibáñez
Faculty of Engineering and Sciences
Santiago, Chile
&
IMFD (Chile)

Reasoning Web School 2021

Explanations in Databases

Receives	$R.1$	$R.2$
	s_2	s_1
	s_3	s_3
	s_4	s_3

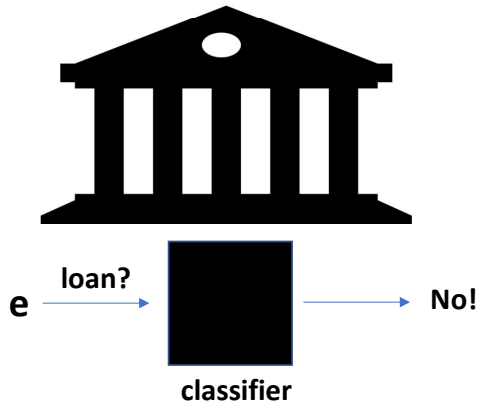
Store	$S.1$
	s_2
	s_3
	s_4

- Query: Are there pairs of official stores in a receiving relationship?
- $\mathcal{Q}$: $\exists x \exists y (Store(x) \wedge Receives(x, y) \wedge Store(y))$

  The query is true in D: $D \models \mathcal{Q}$
- What tuples cause the query to be true?
- How strong are they as causes?
- We would expect tuples $Receives(s_3, s_3)$ and $Receives(s_4, s_3)$ to be causes
- Explanations for a query result ...

- Explanations for violation of semantic conditions (integrity constraints), etc.
- A DB system could provide *explanations*
- Want to model, specify and compute causality
- Large part of our research motivated by trying to understand causality in data management from different perspectives

Explanations in Machine Learning



- Client requesting a loan from a bank using a black-box classifier
- $\mathbf{e} = \langle \text{john}, 18, \text{plumber}, 70\text{K}, \text{harlem} \rangle$

  Record of values for features Name, Age, Income, ...
- Which are the feature values most relevant for the classification outcome, i.e. the label "No"?
- What is the contribution of each feature value to the outcome?
- Questions like these are at the core of Explainable AI

A Score-Based Approach: Responsibility

- Causality has been developed in AI for 3 decades or so
- In particular, Actual Causality
- Also the quantitative notion of Responsibility: a measure of causal contribution
- 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
- Do changes of feature values make the label change to "Yes"?
- We have investigated causality and responsibility in data management and classification
- Semantics, computational mechanisms, intrinsic complexity, logic-based specifications, reasoning, etc.

- There are other explanation scores

  Also called "attribution scores"
- Some of them have been applied in data management and machine learning
- The "causal effect" score
- The Shapley value
- We have done research on them too
- We will present them

- We also want to specify counterfactual interventions
- Reason about them, and explanations
- Compute responsibility scores from the specifications

This Tutorial

1. Review of causality in DBs
2. The DB repair connection
3. ASPs for causality computation
4. Causality under integrity constraints
5. Causal responsibility vs. causal effect
6. Shapley value in DBs
7. Responsibility of explanations for classification
8. Shapley value of explanations for classification
9. Counterfactual Intervention Programs for classification
10. Final remarks

Companion paper: L. Bertossi. "Score-Based Explanations in Data Management and Machine Learning: An Answer-Set Programming Approach to Counterfactual Analysis". Posted as Corr arXiv Paper 2106.10562, 2021.

Causality in Databases

Causality in DBs

- Causality-based explanation for a query result: (Meliou et al., VLDB 2010)
  - A relational instance D and a boolean conjunctive $\mathcal{Q}$
  - A tuple $\tau \in D$ is a counterfactual cause for $\mathcal{Q}$ if $D \models \mathcal{Q}$ and $D \setminus \{\tau\} \not\models \mathcal{Q}$
  - A tuple $\tau \in D$ is an actual cause for $\mathcal{Q}$ if there is a contingency set $\Gamma \subseteq D$, such that τ is a counterfactual cause for $\mathcal{Q}$ in $D \setminus \Gamma$

  Based on (Halpern and Pearl, 2001, 2005)
- The responsibility of an actual cause τ for $\mathcal{Q}$:

$$\rho_D(\tau) := \frac{1}{|\Gamma| + 1}, \; |\Gamma| = \text{size of smallest contingency set for } \tau$$

(0 otherwise)
- High responsibility tuples provide more interesting explanations

Based on (Chockler and Halpern, 2004)

Example

- Database D with relations R and S below

  $\mathcal{Q}$: $\exists x \exists y (S(x) \wedge R(x, y) \wedge S(y))$ Here: $D \models \mathcal{Q}$

R	A	B
	a_4	a_3
	a_2	a_1
	a_3	a_3

S	A
	a_4
	a_2
	a_3

- Causes for $\mathcal{Q}$ to be true in D?
- $S(a_3)$ is counterfactual cause for $\mathcal{Q}$:
  If $S(a_3)$ is removed from D, $\mathcal{Q}$ is no longer an answer
- Its responsibility is $1 = \frac{1}{1+|\emptyset|}$
- $R(a_4, a_3)$ is an actual cause for $\mathcal{Q}$ with contingency set $\{R(a_3, a_3)\}$

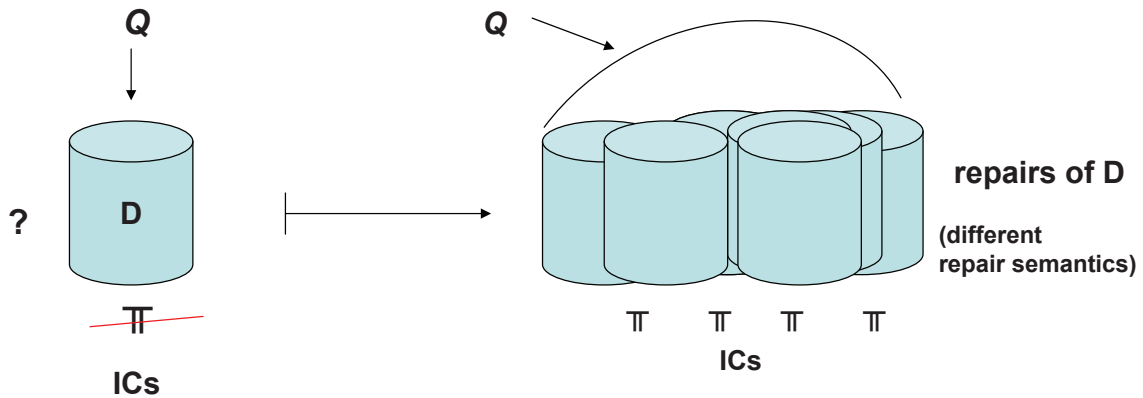
  If $R(a_3, a_3)$ is removed from D, $\mathcal{Q}$ is still true, but further removing $R(a_4, a_3)$ makes $\mathcal{Q}$ false
- Responsibility of $R(a_4, a_3)$ is $\frac{1}{2} = \frac{1}{1+1}$
  Its smallest contingency sets have size 1
- $R(a_3, a_3)$ and $S(a_4)$ are actual causes, with responsibility $\frac{1}{2}$

Computational Problems

- Among many of them:
  - Compute causes
  - Decide if a tuple is a cause
  - Compute responsibilities
  - Compute most responsible causes (MRC)
  - Decide if a tuple has responsibility above a threshold
- Rather complete complexity picture for CQs and UCQs
- Obtained mostly via connection between:
  - causality and database repairs, and
  - causality and consistency-based diagnosis

(B. & Salimi, TOCS'17)

Database Repairs



(Arenas et al., PODS 99)

Example: Denial constraints (DCs) (in particular, FDs)

$\neg\exists x\exists y(P(x) \wedge Q(x,y))$

$\neg\exists x\exists y(P(x) \wedge R(x,y))$

P	A
	a
	e

Q	A	B
	a	b

R	A	C
	a	c

- Subset-repairs (S-repairs): (maximal consistent subinstance)

  $D_1 = \{P(e), Q(a,b), R(a,c)\}$ $D_2 = \{P(e), P(a)\}$

- Cardinality-repairs (C-repairs): (max-cardinality consistent subinstance)

  D_1

The Repair/Causality Connection

- BCQ: $\mathcal{Q}$: $\exists \bar{x}(P_1(\bar{x}_1) \wedge \cdots \wedge P_m(\bar{x}_m))$ and $\mathcal{Q}$ is true in D
  What are the causes for $\mathcal{Q}$ to be true?
- Obtain actual causes and contingency sets from DB repairs
- $\neg\mathcal{Q}$ is logically equivalent to DC

$$\kappa(\mathcal{Q}): \neg\exists \bar{x}(P_1(\bar{x}_1) \wedge \cdots \wedge P_m(\bar{x}_m))$$

- $\mathcal{Q}$ holds in D iff D inconsistent wrt. $\kappa(\mathcal{Q})$
- S-repairs associated to causes and minimal contingency sets
- C-repairs associated to causes, minimum contingency sets, and maximum responsibilities
- Database tuple τ is actual cause with subset-minimal contingency set $\Gamma \iff D \smallsetminus (\Gamma \cup \{\tau\})$ is S-repair
  In which case, its responsibility is $\frac{1}{1+|\Gamma|}$
- τ is actual cause with min-cardinality contingency set Γ
  $\iff D \smallsetminus (\Gamma \cup \{\tau\})$ is C-repair And τ is MRAC

Exploiting the Connection

- Causality problem (CP): Computing/deciding actual causes can be done in polynomial time in data for CQs and UCQs

  (Meliou et al. 2010; B&S'17)

- Most computational problems related to repairs, in particular, C-repairs, are provably hard (data complexity)

  (Lopatenko & B., ICDT'07)

  Techniques and results for repairs can be leveraged

- Responsibility problem: Deciding if a tuple has responsibility above a certain threshold is NP-complete for UCQs (B&S'17)

- Computing $\rho_D(\tau)$ is $FP^{NP(log(n))}$-complete for BCQs
  The *functional* version of the responsibility problem

- Deciding if τ is a most responsible cause is $P^{NP(log(n))}$-complete for BCQs

Answer-Set Programs for Database Repairs

- ASPs can be used to specify, compute and query S- and C-repairs
- Example: (cont.) DC: $\kappa(\mathcal{Q}): \neg\exists x\exists y(S(x) \wedge R(x,y) \wedge S(y))$

  Repair-ASP contains the D as set of facts (with tids):

  $R(1, a_4, a_3), R(2, a_2, a_1), R(3, a_3, a_3), S(4, a_4), S(5, a_2), S(6, a_3)$

  Rules:

$$S'(t_1, x, \mathrm{d}) \vee R'(t_2, x, y, \mathrm{d}) \vee S'(t_3, y, \mathrm{d}) \leftarrow S(t_1, x), R(t_2, x, y), S(t_3, y),$$
$$S'(t, x, \mathrm{s}) \leftarrow S(t, x), \; \mathit{not}\; S'(t, x, \mathrm{d}). \;\text{ etc.}$$

  d, s: annotation constants for "tuple deleted" and "tuple stays in repair", resp.

- A stable model M of the program determines an S-repair D' of D: $D' := \{R(\bar{c}) \mid R'(t, \bar{c}, \mathrm{s}) \in M\}$

  S-repair D_1 represented by: $M_1 = \{R'(1, a_4, a_3, \mathrm{s}), R'(2, a_2, a_1, \mathrm{s}),$
  $R'(3, a_3, a_3, \mathrm{s}), S'(4, a_4, \mathrm{s}), S'(5, a_2, \mathrm{s}), S'(6, a_3, \mathrm{d}), \ldots\}$

- Example: Schema $R(A, B, C, D)$ and FD $A, B \rightarrow C$

$$\neg\exists xyz_1z_2vw(R(x, y, z_1, v) \wedge R(x, y, z_2, w) \wedge z_1 \neq z_2)$$

- Repair program contains the rules: (with global tuple ids, t_i)

$$R'(t_1, x, y, z_1, v, \mathrm{d}) \vee R'(t_2, x, y, z_2, w, \mathrm{d}) \leftarrow R(t_1, x, y, z_1, v), R(t_2, x, y, z_2, w), z_1 \neq z_2$$

$$R'(t, x, y, z, v, \mathrm{s}) \leftarrow R(t, x, y, z, v), \ \mathit{not}\ R'(t, x, y, z, v, \mathrm{d})$$

- For sets of DCs/FDs repair programs can be made normal, i.e. non-disjunctive
- Maybe non-stratified, as with FDs, and with DCs with self-joins
- Certain query answering (QA) under normal ASPs is NP-complete in data
- Matching intrinsic data complexity of consistent query answering under FDs/DCs
- Models corresponding to C-repairs can be obtained by adding weak program constraints (WCs)

- Example: (cont.) Add WCs

  $:\sim\ R(t, \bar{x}), R'(t, \bar{x}, \mathrm{d})$ $:\sim\ S(t, \bar{x}), S'(t, \bar{x}, \mathrm{d})$

- Keep models that minimize the number of violations of the WCs only
- Here: minimize the number of deleted tuples
- C-repairs and WCs useful for capturing most-responsible actual causes
- $\rightsquigarrow$ ASPs for causality/responsibility computation

(B.; KAIS'20)

Specifying Causes with Repair-ASPs

- Repair programs can be used as the basis for specifying causes and reasoning with them
- They provide the right expressive power and complexity for causality-related computations
- Cause and responsibility computation become QA on extended repair program
- Causes represented by global tuple identifiers (tids) t
- Example: (cont.) DC is $\kappa(\mathcal{Q})$ for $\mathcal{Q} : \exists x \exists y (S(x) \wedge R(x, y) \wedge S(y))$

  Causes?
- Add rules:

$$Ans(t) \leftarrow R'(t, x, y, \mathrm{d})$$
$$Ans(t) \leftarrow S'(t, x, \mathrm{d})$$

- QA: $\Pi \models_{brave} Ans(t)$? (true in *some* model of Π)

- For (maximum) responsibility we need contingency sets associated to causes
- New predicate $CauCon(t, t')$:

  "t is actual cause, and t' is a member of the former's contingency set"
- For each pair of predicates P_i, P_j in DC $\kappa(\mathcal{Q})$, the rule

$$CauCon(t, t') \leftarrow P'_i(t, \bar{x}_i, \mathrm{d}),\ P'_j(t', \bar{x}_j, \mathrm{d}),\ t \neq t'$$

  (t' deleted together with t)
- In the example: $CauCon(t, t') \leftarrow S'(t, x, \mathrm{d}), R'(t', u, v, \mathrm{d})$
  Etc.

- Example: (cont.) Π extended with rules for causes with contingency sets

$$CauCon(t, t') \leftarrow S'(t, x, \mathrm{d}), R'(t', u, v, \mathrm{d})$$
$$CauCon(t, t') \leftarrow S'(t, x, \mathrm{d}), S'(t', u, \mathrm{d}), t \neq t'$$
$$CauCon(t, t') \leftarrow R'(t, x, y, \mathrm{d}), S'(t', u, \mathrm{d})$$
$$CauCon(t, t') \leftarrow R'(t, x, y, \mathrm{d}), R'(t', u, v, \mathrm{d}), t \neq t'$$

- From model M_2 corresponding to repair D_2: $CauCon(1, 3)$ and $CauCon(3,1)$
  From repair difference $D \setminus D_2 = \{R(a_4, a_3), R(a_3, a_3)\}$

- Contingency sets computed with extensions of ASP with set-aggregation (e.g. DLV-Complex)

$$preCon(t, \{t'\}) \leftarrow CauCon(t, t')$$
$$preCon(t, \#union(C, \{t''\})) \leftarrow CauCon(t, t''), preCon(t, C), not\ \#member(t'', C)$$
$$Con(t, C) \leftarrow preCon(t, C), not\ aux(t, C) \quad \text{(maximal sets)}$$
$$aux(t, C) \leftarrow CauCon(t, t'), \#member(t', C)$$

- Computation of a cause's "responsibility"

$$\begin{aligned} \textit{pre-rho}(t, n) &\leftarrow \#\textit{count}\{t' : \textit{CauCon}(t, t')\} = n \\ \textit{rho}(t, m) &\leftarrow m * (\textit{pre-rho}(t, m) + 1) = 1 \end{aligned}$$

- Responsibility of a cause t can be obtained through a query to the extended program Π^e:

$$\Pi^e \models_{\textit{brave}} \textit{rho}(t, X)?$$

  Keep minimum value for X

- If WCs are added to the repair program, only maximum-responsibility causes computed

- ASP with WCs computation has exactly required expressive power/complexity needed for maximum-responsibility computation

Causality under Integrity Constraints

- For causality, taking satisfied ICs into account becomes crucial
- In DBs the structural model contains the lineage of the query and now also the ICs (c.f. below)
- Counterfactual interventions become tuple deletions

  (Salimi et al.; TaPP'16)

  Instances obtained from D by tuple deletions should satisfy the ICs (B. & Salimi; IJAR'17)
- In this case, we start assuming that $D \models \Sigma$
- For τ to be actual cause for $\mathcal{Q}(\bar{a})$, the contingency set Γ must satisfy:

$$D \smallsetminus \Gamma \models \Sigma \qquad D \smallsetminus \Gamma \models \mathcal{Q}(\bar{a})$$
$$D \smallsetminus (\Gamma \cup \{\tau\}) \models \Sigma \qquad D \smallsetminus (\Gamma \cup \{\tau\}) \not\models \mathcal{Q}(\bar{a})$$

- Responsibility $\rho^{D,\Sigma}_{\mathcal{Q}(\bar{a})}(\tau)$ defined as before

- Example: DB instance D and CQ, $\mathcal{Q}$ below

	Dep	*DName*	*TStaff*
t_1		Computing	John
t_2		Philosophy	Patrick
t_3		Math	Kevin

	Course	*CName*	*TStaff*	*DName*
t_4		COM08	John	Computing
t_5		Math01	Kevin	Math
t_6		HIST02	Patrick	Philosophy
t_7		Math08	Eli	Math
t_8		COM01	John	Computing

(A) $\mathcal{Q}(x)\colon \exists y \exists z(\mathit{Dep}(y,x) \wedge \mathit{Course}(z,x,y))$

$\langle \text{John} \rangle \in \mathcal{Q}(D)$

(a) t_1 is counterfactual

(b) t_4 with single minimal contingency set $\Gamma_1 = \{t_8\}$

(c) t_8 with single minimal contingency set $\Gamma_2 = \{t_4\}$

- Under IND ψ: $\forall x \forall y\ (\mathit{Dep}(x,y) \rightarrow \exists u\ \mathit{Course}(u,y,x))$
- t_4 t_8 not actual causes anymore: $D \smallsetminus \Gamma_1 \models \psi$, but (satisfied)
  $D \smallsetminus (\Gamma_1 \cup \{t_4\}) \not\models \psi$
- t_1 still is counterfactual cause

(B) $\mathcal{Q}_1(x)\colon\ \exists y\ \mathit{Dep}(y,x)$ $\langle \text{John} \rangle \in \mathcal{Q}_1(D)$

- Under IND: same causes as Q: $\mathcal{Q} \equiv_\psi \mathcal{Q}_1$

(C) $\mathcal{Q}_2(x)$: $\exists y \exists z\, Course(z, x, y)$ $\langle \text{John} \rangle \in \mathcal{Q}_2(D)$

- W/O ψ: t_4 and t_8 only actual causes, with $\Gamma_1 = \{t_8\}$ and $\Gamma_2 = \{t_4\}$, resp.
- Under IND: t_4 and t_8 still actual causes
- Contingency sets?
- We lose Γ_1 and Γ_2

  $D \smallsetminus (\Gamma_1 \cup \{t_4\}) \not\models \psi$, $D \smallsetminus (\Gamma_2 \cup \{t_8\}) \not\models \psi$
- Smallest contingency set for t_4: $\Gamma_3 = \{t_8, t_1\}$
  Smallest contingency set for t_8: $\Gamma_4 = \{t_4, t_1\}$
- Responsibilities of t_4, t_8 decrease: $\rho^{D}_{\mathcal{Q}_2(\text{John})}(t_4) = \frac{1}{2}$, but $\rho^{D,\psi}_{\mathcal{Q}_2(\text{John})}(t_4) = \frac{1}{3}$
- t_1 is still not an actual cause, but affects the responsibility of actual causes

- Some Results:
  - Causes are preserved under logical equivalence of queries under ICs
  - Without ICs, deciding causality for CQs is tractable, but their presence may make complexity grow
  - There are a CQ $\mathcal{Q}$ and an inclusion dependency ψ, for which deciding causality is NP-complete (B & S'17)
  - ASPs for computation of causes and responsibilities under ICs can be produced
- Beyond CQs:
  - What about causality for Datalog queries?
  - For Datalog queries, cause computation can be NP-complete
  - Through a connection to Datalog abduction (B. & Salimi; IJAR'17)

Abstract Causes from Repair Semantics

- Given: DB D, true query $\mathcal{Q}$, and its associated (violated) DC $\kappa(\mathcal{Q})$
- Different repair semantics $\mathcal{S}$ can be considered (not only S-repairs as above)
- A repair semantics identifies a class $Rep^{\mathcal{S}}(D, \kappa(\mathcal{Q}))$ of admissible and consistent instances that "minimally" depart from D
- Now $\mathcal{S}$-related causes can be defined
- $t \in D$ is an $\mathcal{S}$-actual cause for $\mathcal{Q}$ iff as on page 13 with $\mathcal{S}$-repairs instead of S-repairs
- In particular, prioritized repairs (Staworko et al., AMAI'12)
- There are prioritized ASPs that can be used for repair programs (Gebser et al., TPLP'11)

Attribute-Level Causes via Attribute-Based repairs

- Example: D, $\mathcal{Q}$: $\exists x \exists y (S(x) \wedge R(x, y) \wedge S(y))$, and $D \not\models \kappa(\mathcal{Q})$

R	A	B
t_1	a_4	a_3
t_2	a_2	a_1
t_3	a_3	a_3

S	A
t_4	a_4
t_5	a_2
t_6	a_3

Repair by "minimally" changing attribute values by NULL, as in SQL DBs

Cannot be used to satisfy a join

R	A	B
t_1	a_4	a_3
t_2	a_2	a_1
t_3	a_3	a_3

S	A
t_4	a_4
t_5	a_2
t_6	NULL

R	A	B
t_1	a_4	NULL
t_2	a_2	a_1
t_3	a_3	NULL

S	A
t_4	a_4
t_5	a_2
t_6	a_3

Two repairs

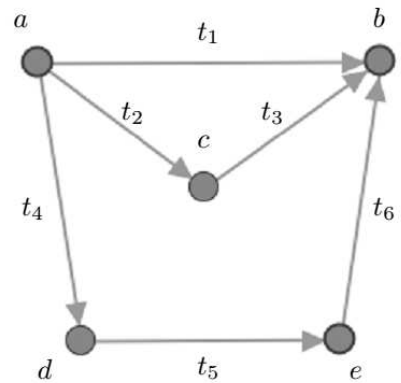
For hiding sensitive information in a DB (B. & Li; TKDE'13))

- These minimal repairs identify $t_6[1]$ (value in 1st position), $t_1[2]$, $t_3[2]$ as actual causes
- Corresponding repair programs can be produced as before (B.; KAIS'21)

Causal Responsibility and Causal Effect

- Causal responsibility can be seen as an explanation score for database tuples in relation to query results
- It is not the only possible score
- Example: Boolean query Π is true if there is a path between a and b

E	X	Y
t_1	a	b
t_2	a	c
t_3	c	b
t_4	a	d
t_5	d	e
t_6	e	b



$$yes \leftarrow P(a,b)$$
$$P(x,y) \leftarrow E(x,y)$$
$$P(x,y) \leftarrow P(x,z), E(z,y)$$

- $E \cup \Pi \models yes$ (query in Datalog, also union of CQs)
- All tuples are actual causes: every tuple in a path from a to b
- All the tuples have the same causal responsibility: $\frac{1}{3}$
- Maybe counterintuitive: t_1 provides a direct path from a to b

- We proposed an alternative to the notion of causal responsibility: *Causal Effect*, a new score (Salimi et al., TaPP'16)
- Causal responsibility has been questioned for other reasons and from different angles
- Retake question about how answer to query $\mathcal{Q}$ changes if τ is deleted/inserted from/into D
- An *intervention* on a *structural causal model*
- In this case provided by the the *lineage* of the query
- Example: $D = \{R(a,b), R(a,c), R(c,b), S(b), S(c)\}$

  BCQ $\mathcal{Q}: \exists x(R(x,y) \wedge S(y))$
- True in D, with lineage instantiated on D given by propositional formula:

$$\Phi_{\mathcal{Q}}(D) = (X_{R(a,b)} \wedge X_{S(b)}) \vee (X_{R(a,c)} \wedge X_{S(c)}) \vee (X_{R(c,b)} \wedge X_{S(b)})$$

- X_τ: propositional variable that is true iff $\tau \in D$

- Want to quantify contribution of a tuple to a query answer
- Assign probabilities uniformly and independently to tuples in D

R^p	A	B	prob
	a	b	$\frac{1}{2}$
	a	c	$\frac{1}{2}$
	c	b	$\frac{1}{2}$

S^p	B	prob
	b	$\frac{1}{2}$
	c	$\frac{1}{2}$

D^p Probabilistic database (tuples outside D get probability 0)

- The X_τ's become independent, identically distributed random variables; and $\mathcal{Q}$ is Bernouilli random variable
- What's the probability that $\mathcal{Q}$ takes a particular truth value when an intervention is done on D?
- Interventions of the form $do(X = x)$: In the *structural equations* make X take value x
- For $y, x \in \{0, 1\}$: $P(\mathcal{Q} = y \mid do(X_\tau = x))$?
- Corresponding to making X_τ false or true
- E.g. $do(X_{S(b)} = 0)$ leaves lineage in the form:

$$\Phi_{\mathcal{Q}}(D)\frac{X_{S(b)}}{0} := (X_{R(a,c)} \wedge X_{S(c)})$$

- The *causal effect* of τ:

$$\mathcal{CE}^{D,\mathcal{Q}}(\tau) \;:=\; \mathbb{E}(\mathcal{Q} \mid do(X_\tau = 1)) - \mathbb{E}(\mathcal{Q} \mid do(X_\tau = 0))$$

- Example: (cont.) When X_τ is made false, probability that the instantiated lineage above becomes true in D^p:

$$P(\mathcal{Q} = 1 \mid do(X_{S(b)} = 0)) = P(X_{R(a,c)} = 1) \times P(X_{S(c)} = 1) = \tfrac{1}{4}$$

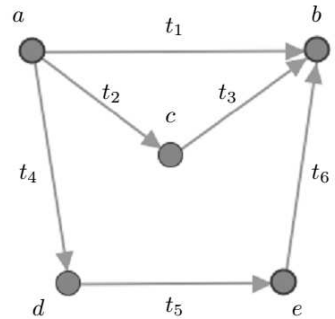
- When X_τ is made true, is probability of this lineage becoming true in D^p:

$$\Phi_{\mathcal{Q}}(D)\frac{X_{S(b)}}{1} := X_{R(a,b)} \vee (X_{R(a,c)} \wedge X_{S(c)}) \vee X_{R(c,b)}$$

$$P(\mathcal{Q} = 1 \mid do(X_{S(b)} = 1)) = P(X_{R(a,b)} \vee (X_{R(a,c)} \wedge X_{S(c)}) \vee X_{R(c,b)} = 1)$$
$$= \cdots = \frac{13}{16}$$

- $\mathbb{E}(\mathcal{Q} \mid do(X_{S(b)} = 0)) = P(\mathcal{Q} = 1 \mid do(X_{S(b)} = 0)) = \frac{1}{4}$
  $\mathbb{E}(\mathcal{Q} \mid do(X_{S(b)} = 1)) = \frac{13}{16}$
- $\mathcal{CE}^{D,\mathcal{Q}}(S(b)) = \frac{13}{16} - \frac{1}{4} = \frac{9}{16} > 0$, an actual cause with this causal effect!

- Example: (cont.) The Datalog query, as a union of BCQs, has the lineage:



$$\Phi_{\mathcal{Q}}(D) = X_{t_1} \vee (X_{t_2} \wedge X_{t_3}) \vee (X_{t_4} \wedge X_{t_5} \wedge X_{t_6})$$

- $\mathcal{CE}^{D,\mathcal{Q}}(t_1) = 0.65625$
  $\mathcal{CE}^{D,\mathcal{Q}}(t_2) = \mathcal{CE}^{D,\mathcal{Q}}(t_3) = 0.21875$
  $\mathcal{CE}^{D,\mathcal{Q}}(t_4) = \mathcal{CE}^{D,\mathcal{Q}}(t_5) = \mathcal{CE}^{D,\mathcal{Q}}(t_6) = 0.09375$
- The causal effects are different for different tuples!
- More intuitive result than responsibility!

- Rather *ad hoc* or arbitrary? (we'll be back ...)

Shapley Value in Databases

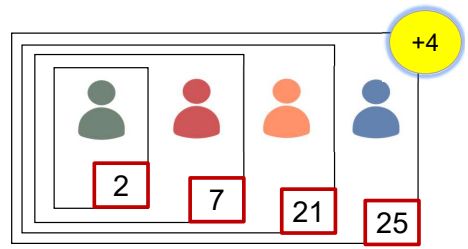
Coalition Games and the Shapley Value

- Initial motivation: By how much a database tuple contributes to the inconsistency of a DB? To the violation of ICs
- Similar ideas can be applied to the contribution to query results (Livshits et al., 2020)
- 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
- Apply standard measures used in game theory, economics, etc.: the Shapley value of tuple
- Implicitly based on counterfactual intervention: What would happen if we change ...?

- Consider a set of players D, and a wealth-distribution (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



- Database tuples and feature values can be seen as players in a coalition game
  Each of them contributing to a shared wealth function
- 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

A Score-Based Approach: Shapley Values in DBs

- Database tuples can be seen as players in a coalition game
- Query $\mathcal{Q}$: $\exists x \exists y (Store(x) \wedge Receives(x, y) \wedge Store(y))$

  It takes values 0 or 1 in a database
- Game function becomes the value of the query
- A set of tuples make it true or not, with some possibly contributing more than others to making it true

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

- Quantifies the contribution of tuple τ to query result
- All possible permutations of subinstances of D
- Average of differences between having τ or not
- Counterfactuals implicitly involved and aggregated

- We investigated algorithmic, complexity and approximation problems
- A dichotomy theorem for Boolean CQs without self-joins

  Syntactic characterization: : PTIME vs. #P-hard
- Extended to aggregate queries
- It has been applied to measure contribution of tuples to inconsistency of a database
- Related and popular score: Bahnzhaf Power Index

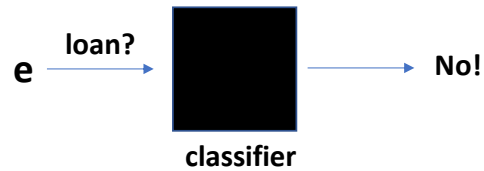
  (order ignored)

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

- Bahnzhaf also difficult to compute: #P-hard in general
- We proved "Causal Effect" coincides with the Banzhaf Index!

Explanations for Classification

A Score-Based Approach: Responsibility



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

- The gist:

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

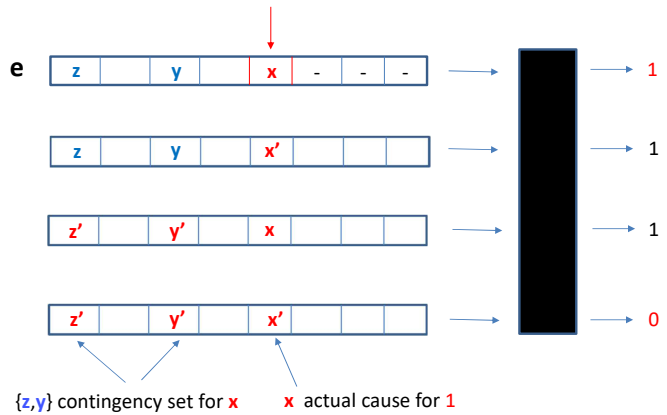
  $\mathbf{e}'' = \langle \text{john}, 18, \text{plumber}, 80\text{K}, \text{brooklyn}, \ldots \rangle$ Yes
- Value for Age is counterfactual cause with $x\text{-}Resp(\text{Age}) = 1$
  Value for Income is actual cause with $x\text{-}Resp(\text{Income}) = \frac{1}{2}$
- Second may be actionable, but not the first

- For binary features this works fine
- We have investigated this case in some detail
- Otherwise, there could be many values that do not change the label, but one of them does
- Better consider all possible values ...

The *Resp* Score: Classification

First a simplified version

- Want explanation for label "1"
- Through changes of feature values, try to get "0"
- Fix a feature value $\mathbf{x} = \mathbf{e}_F$



- $\mathbf{x}$ counterfactual explanation for $L(\mathbf{e}) = 1$ if $L(\mathbf{e}\frac{\mathbf{x}}{\mathbf{x}'}) = 0$, for $\mathbf{x}' \in Dom(F)$
- $\mathbf{x}$ actual explanation for $L(\mathbf{e}) = 1$ if there are values $\mathbf{Y}$ in $\mathbf{e}$, $\mathbf{x} \notin \mathbf{Y}$, and new values $\mathbf{Y}' \cup \{\mathbf{x}'\}$:

  (a) $L(\mathbf{e}\frac{\mathbf{Y}}{\mathbf{Y}'}) = 1$ (b) $L(\mathbf{e}\frac{\mathbf{x}\mathbf{Y}}{\mathbf{x}'\mathbf{Y}'}) = 0$

- If $\mathbf{Y}$ is minimum in size: $x\text{-}Resp(\mathbf{x}) := \frac{1}{1+|\mathbf{Y}|}$

Example:

$\mathcal{C}$

entity (id)	F_1	F_2	F_3	L
$\mathbf{e}_1$	0	1	1	1
$\mathbf{e}_2$	1	1	1	1
$\mathbf{e}_3$	1	1	0	1
$\mathbf{e}_4$	**1**	**0**	1	0
$\mathbf{e}_5$	1	0	0	1
$\mathbf{e}_6$	0	1	0	1
$\mathbf{e}_7$	0	**0**	1	0
$\mathbf{e}_8$	0	0	0	0

- Due to $\mathbf{e}_7$, $F_2(\mathbf{e}_1)$ is counterfactual explanation, with $Resp(\mathbf{e}_1, F_2) = 1$
- Due to $\mathbf{e}_4$, $F_1(\mathbf{e}_1)$ is actual explanation; with $\Gamma = \{F_2(\mathbf{e}_1)\}$ as contingency set:

$$Resp(\mathbf{e}_1, F_1) = \tfrac{1}{2}$$

- Sometimes we may be interested in minimal contingency sets, under set-inclusion

  So as S-repairs vs. C-repairs

- For non-binary features, *Resp* can be expressed as an expected value

A Variation: No contingencies, but average labels

- $\mathbf{e} = \langle \ldots, \mathbf{e}_F, \ldots \rangle, \quad F \in \mathcal{F}$ (B, Li, Schleich, Suciu, Vagena; DEEM@SIGMOD'20)
- $\mathit{Counter}(\mathbf{e}, F) := L(\mathbf{e}) - \mathbb{E}(L(\mathbf{e}') \mid \mathbf{e}'_{\mathcal{F} \setminus \{F\}} = \mathbf{e}_{\mathcal{F} \setminus \{F\}})$
- Easy to compute, and gives reasonable results
- Requires underlying probability space on entity population
- No need to access the internals of the classification model
- Changing one value may not switch the label
  No explanations are obtained
- Bring in contingency sets of feature values!

General Version: Contingencies and average labels

- $\mathbf{e}$ entity under classification, with $L(\mathbf{e}) = 1$, and $F^\star \in \mathcal{F}$
- Local *Resp*-score

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

- $\Gamma \subseteq \mathcal{F} \setminus \{F^\star\}$
- $\mathbf{e}' := \mathbf{e}[\Gamma := \bar{w}] \qquad L(\mathbf{e}') = L(\mathbf{e})$
- $\mathbf{e}'' := \mathbf{e}[\Gamma := \bar{w}, F^\star := v]$, with $v \in dom(F^\star)$
- (When $F^\star(\mathbf{e}) \neq v$, $L(\mathbf{e}'') \neq L(\mathbf{e})$, $F^\star(\mathbf{e})$ is *actual causal explanation* for $L(\mathbf{e}) = 1$ with contingency $\langle \Gamma, \mathbf{e}_\Gamma \rangle$)
- Globally: $Resp(\mathbf{e}, F^\star) := \max_{\substack{|\Gamma| \text{ min.}, (*)>0 \\ \langle \Gamma, \bar{w} \rangle}} Resp(\mathbf{e}, F^\star, \mathcal{F}, \Gamma, \bar{w})$

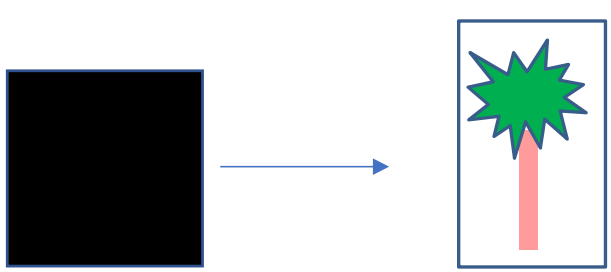
A Score-Based Approach: Shapley Values

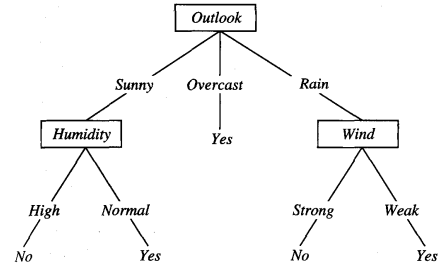
- Feature values can be seen as players in a coalition game
  Each of them contributing to a shared wealth function
- The Shapley value is a established measure of contribution by players to the wealth function
- It emerges as the only measure that enjoys certain desired properties
- For each game one defines an appropriate wealth or game function
- Assume the classifier is binary, with labels 0 and 1
- Set of players $\mathcal{F}$ contain features All relative to $\mathbf{e}$
- Game function: $\mathcal{G}_{\mathbf{e}}(S) := \mathbb{E}(L(\mathbf{e}') \mid \mathbf{e}'_S = \mathbf{e}_S)$ ($\mathbf{e}_S$: projection on S)
  $S \subseteq \mathcal{F}$

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

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

- Quantifies the contribution of feature value $F^\star(\mathbf{e})$ to classification result
- All possible permutations of subsets of $\mathcal{F} \setminus \{F^\star\}$
- Average of differences between having $F^\star(\mathbf{e})$ and not having it
- Counterfactuals implicitly involved and aggregated
- *Shap* score has become popular (Lee & Lundberg; 2017)
- Assumes a probability distribution on entity population
- Both *Resp* and *Shap* may end up considering exponentially many combinations



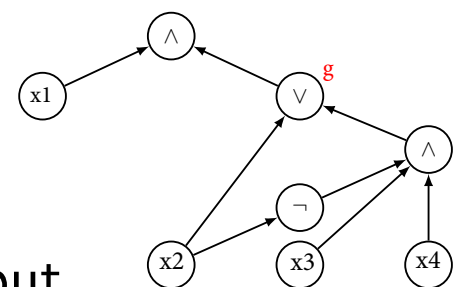


- Can we do better when we have the classification model?
- What if we have a decision tree, or a random forest, or a Boolean circuit?
- Can we compute *Shap* in polynomial time?
- We investigated this problem in detail in a AAAI'21 paper
- Tractable and intractable cases
- Provided algorithms for the former
- In particular, tractable for decision trees and random forests
- Investigated approximation algorithms

d-D Boolean-Circuits

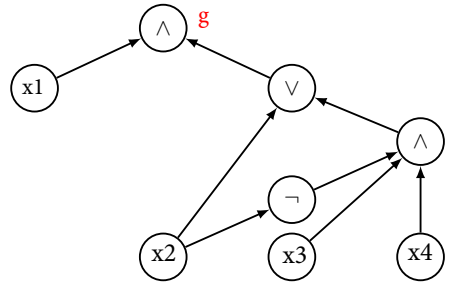
- A Boolean circuit over set of variables X is a DAG $\mathcal{C}$ with:
  - Each node without incoming edges (input) is labeled with either a variable $x \in X$ or a constant in $\{0, 1\}$
  - Each other node is labeled with a gate in $\{\neg, \wedge, \vee\}$
  - There is a single sink node, O, called the output
- $\mathbf{e}\colon X \to \{0,1\}$ (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 $\mathcal{C}(g)$
  $Var(g) = \{x2, x3, x4\}$



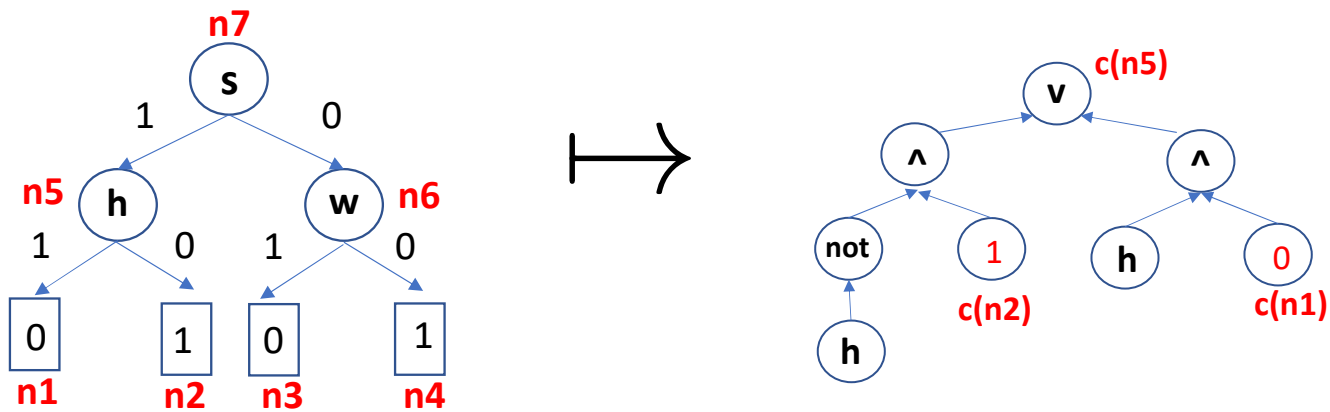
- $\mathcal{C}$ is deterministic if every $\vee$-gate g with input gates g_1, g_2: $\mathcal{C}(g_1)(\mathbf{e}) \neq \mathcal{C}(g_2)(\mathbf{e})$, for every $\mathbf{e}$
- Intuitively, $\vee$-gates behave as $\bar{\vee}$-gates

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



- We will consider $\mathcal{C}$ to be deterministic and decomposable circuit (d-D circuit)
- Several classes of Boolean models can be translated in polynomial time into d-D Boolean circuits:
  - Decision trees
  - Ordered binary decision diagrams (OBDDs)
  - Etc.

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



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

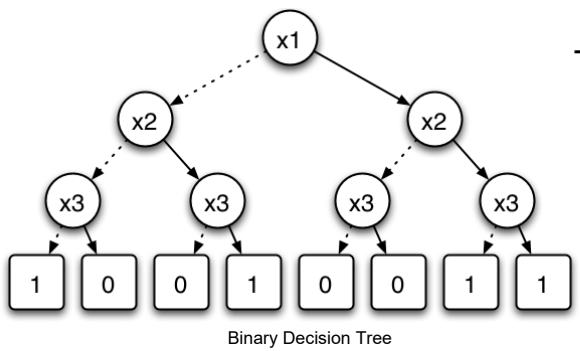
The SHAP Score: d-D Boolean-Circuits

- Theorem: *Shap* can be computed in polynomial time for d-D circuits under the uniform distribution
- Corollary: *Shap* can be computed in polynomial time for decision trees and random forests, OBDDs, etc., under the uniform distribution
- It can be extended to any product distribution on $\{0,1\}^{|X|}$ (uniform is a particular case)

Ordered Binary Decision Diagrams

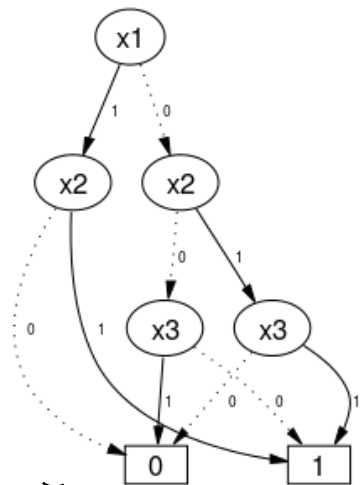
- Our polynomial time algorithm for *Shap* can be applied to *Ordered Binary Decision Diagrams* (OBDDs)
- They are relevant for several reasons in *Knowledge Compilation*
- In particular, to represent "opaque" classifiers as OBDDs, e.g. binary neural networks [Shi, Shih, Darwiche, Choi; KR20]
- Opening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

Idea of the Proof*

- $Shap(\mathcal{F}, \mathcal{G}_{\mathbf{e}}, F) \;=$

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

- Depends on $\mathbf{e}$ and (the classifier behind) L
- $Dom(F_i) = \{0,1\}, \quad F_i \in \mathcal{F},\ i = 1, \ldots, n, \quad \mathbf{e} \in \mathcal{E} := \{0,1\}^n$
  $L(\mathbf{e}) \in \{0,1\}$
- There is also a probability distribution $\mathcal{P}$ on $\mathcal{E}$
- We will identify the Boolean classifier with 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
- Proposition: For the uniform distribution $\mathcal{P}^u$, and $\mathbf{e} \in \mathcal{E}$

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

- $\#SAT \leq_{PTIME}^{Turing} Shap$
- When $\#SAT(L)$ is hard for a Boolean classifier L, computing 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?
- *Shap* computation in polynomial time not precluded

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

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

- So, maybe *Shap* computable in polynomial time ...
- To show that *Shap* can be computed efficiently for d-D circuits, 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 $Shap(\mathcal{F}, \mathcal{G}_{\mathbf{e}}, F_i)$

- Main Result: $\#SAT(\mathcal{C}, \mathbf{e}, \ell)$ can be solved in polynomial time for d-D circuits $\mathcal{C}$, 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)

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

    $\sum_{j+k=\ell} \#SAT(\mathcal{C}(g_1), \mathbf{e}_{Var(g_1)}, j) \times \#SAT(\mathcal{C}(g_2), \mathbf{e}_{Var(g_2)}, k)$

Reasoning about Explanations

Reasoning about Counterfactual Interventions

- Given a classifier, one can reason in answer-set programming (ASP) about counterfactuals
- In interaction with the classifier
- Specified inside the ASP, or invoked as an external predicate
- Have done this for decision-tree and naive-Bayes classifiers
- One can easily impose semantic constraints on counterfactuals
- Each (sensible) counterfactual leading to a change of classification corresponds to a model of the ASP
- Recourses (actionable explanations) can be specified
- Scores can be computed by means of set- and numerical aggregations
- The former for minimal and minimum contingency sets
  The latter for *Resp* scores
- Reasoning is enabled by cautious and brave query answering
- Explanations can be queried

ASPs for Counterfactual Interventions

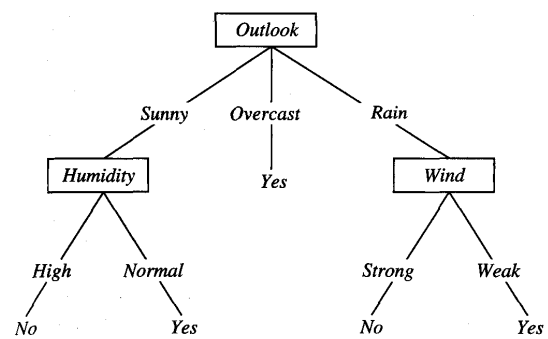
- *Counterfactual Intervention Programs* (CIPs) specify counterfactual interventions on a given entity under classification
- We will use *DLV* and *DLV-Complex* notation
- So as with repair programs, we use annotation constants:

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

- Retake the decision tree on page 47

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 Yes



- 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).

% 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).
```

- Classifier could be invoked as external predicate in Python
- The last is the counterfactual rule
- Only one disjunct in the head becomes true; one per feature
- It uses the non-deterministic choice predicate

  Chooses a new value in last argument for each combination of the first three
- As long as the label does not depart from 1, i.e. yes

- Specification of choice predicate:

```
% definitions of "chosen" predicates:
   chosen1(X,Y,Z,U) :- ent(E,X,Y,Z,tr), cls(X,Y,Z,1), dom1(U), U != X,
                       not diffchoice1(X,Y,Z,U).
   diffchoice1(X,Y,Z, U) :- chosen1(X,Y,Z, Up), U != Up, dom1(U).
   chosen2(X,Y,Z,U) :- ent(E,X,Y,Z,tr), cls(X,Y,Z,1), dom2(U), U != Y,
                       not diffchoice2(X,Y,Z,U).
   diffchoice2(X,Y,Z, U) :- chosen2(X,Y,Z, Up), U != Up, dom2(U).
   chosen3(X,Y,Z,U) :- ent(E,X,Y,Z,tr), cls(X,Y,Z,1), dom3(U), U != Z,
                       not diffchoice3(X,Y,Z,U).

   diffchoice3(X,Y,Z, U) :- chosen3(X,Y,Z, Up), U != Up, dom3(U).

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

- Choice makes the program non-stratified
- Last rule is program constraint prohibiting going back to initial entity

  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

```
% 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.
```

- First rule defines "stop" annotation, when label changes
- Next rules 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" x-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)}
```

- These are the two stable models of the CIP
- Two counterfactual versions with minimal contingency sets
- Only first is minimum counterfactual version: x-Resp($\mathbf{e}$) $= 1$
- Want only maximum responsibility counterfactual versions?

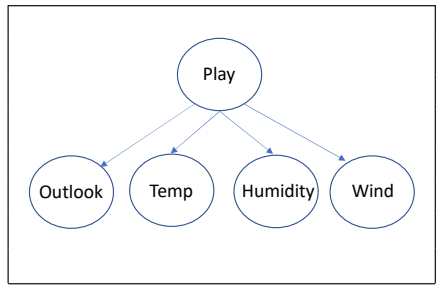
```
% Weak constraints to minimize number of changes:
   :~ ent(E,X,Y,Z,o), ent(E,Xp,Yp,Zp,s), X != Xp.
   :~ ent(E,X,Y,Z,o), ent(E,Xp,Yp,Zp,s), Y != Yp.
   :~ ent(E,X,Y,Z,o), ent(E,Xp,Yp,Zp,s), Z != Zp.
```

- Weak program constraints can be violated, but only a minimum number of times
- Minimize number of feature value differences between $\mathbf{e}$ and counterfactual versions
- Only first model is kept

- Reasoning enabled by query answering
- Under certain and brave 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).
```

Another Example: Naive-Bayes Classification



Outlook	Temperature	Humidity	Wind	Play
sunny	high	high	weak	no
sunny	high	high	strong	no
overcast	high	high	weak	yes
rain	medium	high	weak	yes
rain	low	normal	weak	yes
rain	low	normal	strong	no
overcast	low	normal	strong	yes
sunny	medium	high	weak	no
sunny	low	normal	weak	yes
rain	medium	normal	weak	yes
sunny	medium	normal	strong	yes
overcast	medium	high	strong	yes
overcast	high	normal	weak	yes
rain	medium	high	strong	no

- Classifier is based on Bayesian network on the LHS above
  Associated probabilities learned from data on the RHS:

$P(\text{Play} = \text{yes}) = \frac{9}{14}$	$P(\text{Play} = \text{no}) = \frac{5}{14}$
$P(\text{Outlook} = \text{sunny}\|\text{Play} = \text{yes}) = \frac{2}{9}$	$P(\text{Outlook} = \text{sunny}\|\text{Play} = \text{no}) = \frac{3}{5}$
$P(\text{Outlook} = \text{overcast}\|\text{Play} = \text{yes}) = \frac{4}{9}$	$P(\text{Outlook} = \text{overcast}\|\text{Play} = \text{no}) = 0$
$P(\text{Outlook} = \text{rain}\|\text{Play} = \text{yes}) = \frac{3}{9}$	$P(\text{Outlook} = \text{rain}\|\text{Play} = \text{no}) = \frac{2}{5}$
$P(\text{Temp} = \text{high}\|\text{Play} = \text{yes}) = \frac{2}{9}$	$P(\text{Temp} = \text{high}\|\text{Play} = \text{no}) = \frac{2}{5}$
$P(\text{Temp} = \text{medium}\|\text{Play} = \text{yes}) = \frac{4}{9}$	$P(\text{Temp} = \text{medium}\|\text{Play} = \text{no}) = \frac{2}{5}$
$P(\text{Temp} = \text{low}\|\text{Play} = \text{yes}) = \frac{3}{9}$	$P(\text{Temp} = \text{low}\|\text{Play} = \text{no}) = \frac{1}{5}$
$P(\text{Humidity} = \text{high}\|\text{Play} = \text{yes}) = \frac{3}{9}$	$P(\text{Humidity} = \text{high}\|\text{Play} = \text{no}) = \frac{4}{5}$
$P(\text{Humidity} = \text{normal}\|\text{Play} = \text{yes}) = \frac{6}{9}$	$P(\text{Humidity} = \text{normal}\|\text{Play} = \text{no}) = \frac{1}{5}$
$P(\text{Wind} = \text{strong}\|\text{Play} = \text{yes}) = \frac{3}{9}$	$P(\text{Wind} = \text{strong}\|\text{Play} = \text{no}) = \frac{3}{5}$
$P(\text{Wind} = \text{weak}\|\text{Play} = \text{yes}) = \frac{6}{9}$	$P(\text{Wind} = \text{weak}\|\text{Play} = \text{no}) = \frac{2}{5}$

- Beginning of CIP is as before, with probabilities as facts
- Entity with label Yes: $\mathbf{e} = \langle \text{rain}, \text{high}, \text{normal}, \text{weak} \rangle$

```
% DLV-COMPLEX    #include<ListAndSet>   #maxint = 100000000.
% domains:
   dom_o(sunny). dom_o(overcast). dom_o(rain). dom_t(high). dom_t(medium).
   dom_t(low). dom_h(high). dom_h(normal). dom_w(strong). dom_w(weak).
% original entity that gets label 1:
   ent(e,rain,high,normal,weak,o).
% absolute probabilities for Play (as percentage)
   p(yes, 64). p(no, 36).
% Outlook conditional probabilities (as percentage)
   p_o_c(sunny, yes, 22). p_o_c(overcast, yes, 45). p_o_c(rain, yes, 33).
   p_o_c(sunny, no, 60). p_o_c(overcast, no, 0). p_o_c(rain, no, 40).
% Temperature conditional probabilities (as percentage)
   p_t_c(high, yes, 22). p_t_c(medium, yes, 45). p_t_c(low, yes, 33).
   p_t_c(high, no, 40). p_t_c(medium, no, 40). p_t_c(low, no, 20).
% Humidity conditional probabilities (as percentage)
   p_h_c(normal, yes, 67). p_h_c(high, yes, 33).
   p_h_c(normal, no, 20). p_h_c(high, no, 80).
% Wind conditional probabilities (as percentage)
   p_w_c(strong, yes, 33). p_w_c(weak, yes, 67).
   p_w_c(strong, no, 60). p_w_c(weak, no, 40).
```

- Classification based on probability comparison

$$P(\text{Play} = \text{yes} \mid \text{Outlook} = \text{rain}, \text{Temp} = \text{high}, \text{Humidity} = \text{normal}, \text{Wind} = \text{weak})$$

$$P(\text{Play} = \text{no} \mid \text{Outlook} = \text{rain}, \text{Temp} = \text{high}, \text{Humidity} = \text{normal}, \text{Wind} = \text{weak})$$

```
% spec of the classifier
  cls(E,O,T,H,W,yes) :- ent(E,O,T,H,W,tr),  pb_num(E,O,T,H,W,yes,Fyes),
                        pb_num(E,O,T,H,W,no,Fno), Fyes >= Fno.
  cls(E,O,T,H,W,no)  :- ent(E,O,T,H,W,tr),  pb_num(E,O,T,H,W,yes,Fyes),
                        pb_num(E,O,T,H,W,no,Fno), Fyes < Fno.
```

- Rest of CIP as in previous example, including counterfactual rule, hard and weak constraints, etc.

- We can collect changes of feature values

```
% collecting changed values for each feature:
  expl(E,outlook,O)  :- ent(E,O,T,H,W,o), ent(E,Op,Tp,Hp,Wp,s), O != Op.
  expl(E,temp,T)     :- ent(E,O,T,H,W,o), ent(E,Op,Tp,Hp,Wp,s), T != Tp.
  expl(E,humidity,H) :- ent(E,O,T,H,W,o), ent(E,Op,Tp,Hp,Wp,s), H != Hp.
  expl(E,wind,W)     :- ent(E,O,T,H,W,o), ent(E,Op,Tp,Hp,Wp,s), W != Wp.
```

- We can use DLV-Complex for building contingency sets

```
% building  contingency sets
  cause(E,U)       :- expl(E,U,X).
  cauCont(E,U,I)   :- expl(E,U,X), expl(E,I,Z), U != I.
  preCont(E,U,{I}) :- cauCont(E,U,I).
  preCont(E,U,#union(Co,{I})) :- cauCont(E,U,I), preCont(E,U,Co),
                                 not #member(I,Co).
  cont(E,U,Co)     :- preCont(E,U,Co), not HoleIn(E,U,Co).
  HoleIn(E,U,Co)   :- preCont(E,U,Co), cauCont(E,U,I), not #member(I,Co).
  tmpCont(E,U)     :- cont(E,U,Co), not #card(Co,0).
  cont(E,U,{})     :- cause(E,U), not tmpCont(E,U).
```

- Inverse responsibility computation as before or via cardinality of contingency sets

```
% computing the inverse of x-Resp
  invResp(E,U,R) :- cont(E,U,S), #card(S,M), R = M+1, #int(R).
```

- Without weak constraints we obtain 10 different models

  Each representing a counterfactual version of **e**
- Only three shown here:

```
M1 {ent(e,rain,high,normal,weak,o), ent(e,rain,high,normal,weak,tr),
    cls(e,rain,high,normal,weak,yes), ent(e,rain,high,high,weak,do),
    ent(e,rain,high,high,weak,tr), cls(e,rain,high,high,weak,no),
    ent(e,rain,high,high,weak,s), expl(e,humidity,normal),
    cont(e,humidity,{}),invResp(e,humidity,1),fullExpl(e,humidity,1,{})}
M2 {ent(e,rain,high,normal,weak,o), ent(e,rain,high,high,strong,tr),
    cls(e,rain,high,high,strong,no), ent(e,rain,high,high,strong,s),
    invResp(e,humidity,2), fullExpl(e,humidity,2,{wind}),
    invResp(e,wind,2), fullExpl(e,wind,2,{humidity})}
M3 {ent(e,rain,high,normal,weak,o), ent(e,sunny,high,normal,strong,tr),
    cls(e,sunny,high,normal,strong,no),ent(e,sunny,high,normal,strong,s),
    invResp(e,outlook,2), fullExpl(e,outlook,2,{wind}), ...}
```

- With WCs only the first survives, corresponding to a maximum responsibility counterfactual version

$$\mathbf{e}' = \langle \text{rain}, \text{high}, \text{high}, \text{weak} \rangle$$

- We can add domain knowledge
- If there are functional relationships between feature values for Temperature and Humidity:

  $$\text{high} \mapsto \text{normal}, \quad \{\text{medium}, \text{low}\} \mapsto \text{high}$$

  We can drop disjuncts from counterfactual rule, or qualify body conditions, or use program constraints, for not leading to admissible counterfactuals
- Alternatively, we could use extra rules:

```
ent(E,O,T,normal,W,tr) :- ent(E,O,high,H,W,tr).
ent(E,O,T,high,W,tr)   :- ent(E,O,medium,H,W,tr).
ent(E,O,T,high,W,tr)   :- ent(E,O,low,H,W,tr).
```

- This is very flexible ...
- Reasoning enabled by query answering

  Some queries

```
\DLV>dlcomplex.exe -nofacts -nofdcheck -brave naiveBayes.txt queries.txt
```

- Responsibility of feature value for Outlook?
- Remove weak constraints: Is there a counterfactual with less than three changes?
- Use brave semantics:

```
invResp(e,outlook,R)?                    %Q1
fullExpl(E,U,R,S), R<3?                  %Q2
```

Q1 returns 2,3,4 So, its responsibility is $\frac{1}{2}$

Q2 returns a full explanation: $\mathbf{e}, \mathrm{o}\mathit{outlook}, 2, \{\mathrm{h}\mathit{humidity}\}$

- Under brave semantics: Is there an intervened entity with combination of sunny outlook with strong wind, and its label?
- Or, all intervened entities that obtain label No

```
cls(E,O,T,H,W,_), O = sunny, W = strong?    %Q3
cls(E,O,T,H,W,no)?                          %Q4
```

For Q3 we obtain, e.g., $\mathbf{e}, \mathit{sunny}, \mathit{low}, \mathit{normal}, \mathit{strong}, \mathit{yes}$

For Q4, e.g., $\mathbf{e}, \mathit{sunny}, \mathit{low}, \mathit{high}, \mathit{strong}$

- Does the wind not change under every counterfactual version?
- Under cautions semantics:

```
ent(e,_,_,_,Wp,s), ent(e,_,_,_,W,o), W = Wp?      %Q5
```

  We obtain the empty output

  Meaning Wind is changed in at least one counterfactual version

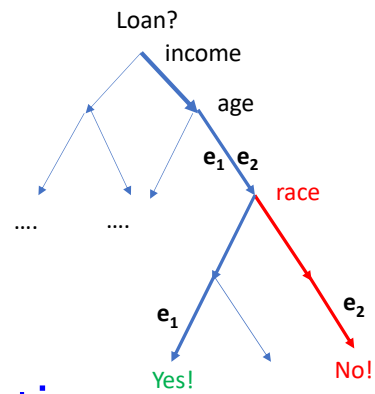
- Different kinds of queries
- For explanatory and exploratory purposes
- For comparison of two different classifiers in a single program
- Etc.

Score-Based Approaches: Final Remarks

- There are many interesting open problems to investigate
- Investigated complexity and algorithmic aspects of *Resp*
- What if we have the classifier?
- Addition of semantic and domain knowledge is important
- Redefinition vs. hacked computation vs. change of distribution?
- Reasoning about counterfactuals
- Connections to model-based diagnosis?
- Explanations vs. Interpretations?
- What is a good explanation?
- What is a good score?
- Maybe emerging from desiderata, so as the Shapley value

- Explanations are at the basis of fairness and bias analysis
- Identifying unexpected or undesirable high-score features becomes relevant
- But possibly not enough
- Understanding the decisions in relation to protected features becomes relevant
- An analytical process that should be characterized, formalized and automated
- Explaining *how* decisions are made

  Here race is a protected feature
  Two entities with same path diverge at that point, getting different labels



- Another promising problem: higher-order analytics on explanations
- What else can be learnt about the population or our classification mechanism?

References (self-references for this presentation)

- Bertossi, L. and Salimi, B. "From Causes for Database Queries to Repairs and Model-Based Diagnosis and Back". Theory of Computing Systems, 2017, 61(1):191-232.

- Bertossi, L. and Salimi, B. "Causes for Query Answers from Databases: Datalog Abduction, View-Updates, and Integrity Constraints". International Journal of Approximate Reasoning, 2017, 90:226-252.

- L. Bertossi. "Specifying and Computing Causes for Query Answers in Databases via Database Repairs and Repair Programs". Knowledge and Information Systems, 2021, 63(1):199-231.

- E. Livshits, L. Bertossi, B. Kimelfeld and M. Sebag. "The Shapley Value of Tuples in Query Answering". In Proc. ICDT 2020. Extended version to appear in Logical Methods in Computer Science, arXiv Paper cs.DB/1904.08679.

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

- 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.

- M. Arenas, P. Barcelo, L. Bertossi, M. Monet. "The Tractability of SHAP-scores over Deterministic and Decomposable Boolean Circuits". Proc. AAAI 2021. Extended version as arXiv Paper 2104.08015, 2021

- L. Bertossi. "Declarative Approaches to Counterfactual Explanations for Classification". Journal submission. arXiv Paper 2011.07423, 2020.

- L. Bertossi. "Score-Based Explanations in Data Management and Machine Learning". Proc. Int. Conf. Scalable Uncertainty Management (SUM 20), Springer LNCS 2322, pp. 17-31.

- L. Bertossi and G. Reyes. "Answer-Set Programs for Reasoning about Counterfactual Interventions and Responsibility Scores for Classification". To appear in Proc. 1st International Joint Conference on Learning and Reasoning (IJCLR'21). Extended version posted as arXiv Paper 2107.10159.