relationalAI

Variations on a Causality Theme in Databases

Leo Bertossi

Database Causality

- In data management, we need to understand and compute why certain results are obtained or not

  E.g. query answers, violation of semantic conditions, ...

- A DB system could provide *explanations*

- Want to model, specify and compute causality

- Our research motivated by trying to understand causality in data management from different perspectives

This Presentation

1. Review of causality in DBs
2. The DB repair connection
3. Exploiting the connection
4. ASPs for causality computation
5. Causality under integrity constraints
6. Causal responsibility vs. causal effect
7. Abstracting out causality in DBs
8. Newer developments
9. Extra slides: The abduction connection*

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 $\mathcal{Q}$:

$$\rho_D(\tau) := \frac{1}{|\Gamma| + 1}$$

$|\Gamma|$ = size of smallest contingency set for τ (0 otherwise)

- High responsibility tuples provide more interesting explanations

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: $D \models \mathcal{Q}$

Causes for $\mathcal{Q}$ to be true in D:

R	A	B
	a_4	a_3
	a_2	a_1
	a_3	a_3

S	A
	a_4
	a_2
	a_3

$S(a_3)$ 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 $\frac{1}{2}$

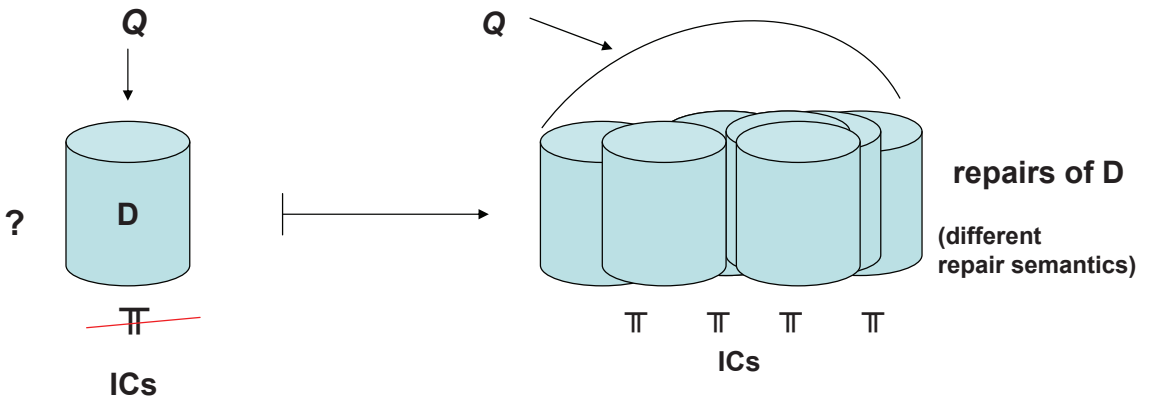
Computational Problems

- Among many of them:
  - Compute causes, and 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) and inconsistent DBs (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 sub-instance)

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

Cardinality-repairs (C-repairs): (maximum-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 database 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 (B&S, op. cit.)

  C-repairs associated to causes, minimum contingency sets, and maximum responsibilities

- Fix a database tuple τ
- τ is actual cause with subset-minimal contingency set $\Gamma \Leftrightarrow D \setminus (\Gamma \cup \{\tau\})$ is S-repair

  In which case, its responsibility is $\frac{1}{1+|\Gamma|}$
- τ is actual cause with min-cardinality contingency set $\Gamma \Leftrightarrow D \setminus (\Gamma \cup \{\tau\})$ is C-repair

  In which case, τ is an MRAC
- (Conversely, repairs can be obtained from causes and their contingency sets)

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 $\rightsquigarrow$

- 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*, non-decision, 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 DB D as set of facts (with tids in 1st attribute):

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

Rules:

$$S'(t_1, x, \mathsf{d}) \vee R'(t_2, x, y, \mathsf{d}) \vee S'(t_3, y, \mathsf{d}) \leftarrow S(t_1, x), R(t_2, x, y), S(t_3, y),$$
$$S'(t, x, \mathsf{s}) \leftarrow S(t, x),\ not\ S'(t, x, \mathsf{d}).$$ 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:

  $D' := \{R(\bar{c}) \mid R'(t, \bar{c}, \mathsf{s}) \in M\}$ (and every S-repair obtained in this way)

  An S-repair D_1 represented by model

  $M_1 = \{R'(1, a_4, a_3, \mathsf{s}), R'(2, a_2, a_1, \mathsf{s}), R'(3, a_3, a_3, \mathsf{s}), S'(4, a_4, \mathsf{s}), S'(5, a_2, \mathsf{s}), S'(6, a_3, \mathsf{d}), \ldots\}$

- For sets of DCs repair programs can be made normal, i.e. non-disjunctive

  Maybe non-stratified, e.g. for FDs and DCs with self-joins

- Consistent query answering becomes certain QA under normal ASPs, which is NP-complete (in data)

  Matching the data complexity of consistent QA under FDs/DCs

- Models corresponding to C-repairs can be obtained by adding weak program constraints (WCs)

Example: (cont.) Add WCs

$$\Leftarrow\ R(t,\bar{x}), R'(t,\bar{x},\mathsf{d})$$
$$\Leftarrow\ S(t,\bar{x}), S'(t,\bar{x},\mathsf{d})$$

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

Specifying Causes with Repair-ASPs

$\rightsquigarrow$ ASPs for causality/responsibility computation (B.; FoIKS'18)

- Cause and responsibility computation become QA on extended repair program

  Causes represented by global tuple identifiers (tids) t

Example: (cont.) DC is $\kappa(Q)$ for $Q: \exists x \exists y (S(x) \wedge R(x,y) \wedge S(y))$

Causes? Add rules:

$$\begin{aligned} Ans(t) &\leftarrow R'(t,x,y,\mathsf{d}) \\ Ans(t) &\leftarrow S'(t,x,\mathsf{d}) \end{aligned}$$

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

$$CauCon(t, t') \leftarrow P_i'(t, \bar{x}_i, \mathsf{d}),\ P_j'(t', \bar{x}_j, \mathsf{d}),\ t \neq t'$$

(t' deleted together with t)

In the example: $CauCon(t, t') \leftarrow S'(t, x, \mathsf{d}), R'(t', u, v, \mathsf{d})$ Etc.

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

$$
\begin{aligned}
preCon(t, \{t'\}) &\leftarrow CauCon(t, t') \\
preCon(t, \#union(C, \{t''\})) &\leftarrow CauCon(t, t''), preCon(t, C), not\ \#member(t'', C) \\
Con(t, C) &\leftarrow preCon(t, C), not\ aux(t, C) \qquad \text{(maximal sets)} \\
aux(t, C) &\leftarrow CauCon(t, t'), \#member(t', C)
\end{aligned}
$$

- Computation of a cause's "responsibility"

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

- Responsibility of a cause t can be obtained through a query to the extended program Π^e: $\Pi^e \models_{brave} rho(t, X)?$ (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

  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) $\mathcal{Q}(x)\colon\ \exists y \exists z (Dep(y, x) \wedge Course(z, x, y))$ $\quad \langle \text{John} \rangle \in \mathcal{Q}(D)$

(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: $\forall x \forall y\, (Dep(x, y) \rightarrow \exists u\, Course(u, y, x))$ (satisfied IND)

t_4 t_8 not actual causes anymore; t_1 still is counterfactual cause

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

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

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

(C) $\mathcal{Q}_2(x)\colon\ \exists y \exists z\, Course(z, x, y)$

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

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 9 with $\mathcal{S}$-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: Database D, query $\mathcal{Q}$: $\exists x \exists y(S(x) \wedge R(x,y) \wedge S(y))$, 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

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

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	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

(Used to hide sensitive information in a DB (B. & Li; TKDE'13))

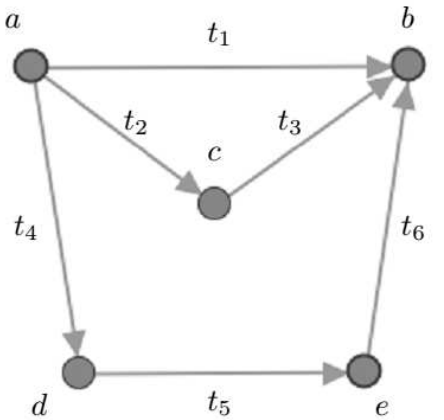
These minimal repairs identify $t_6[1]$ (value in 1st position), $t_1[2]$, $t_3[2]$ as actual causes

Corresponding repair programs can be produced as before (B.; FOIKS'18)

Causal Responsibility and Causal Effect

Example: Boolean Datalog query Π becomes true on E 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



$$
\begin{aligned}
yes &\leftarrow P(a,b) \\
P(x,y) &\leftarrow E(x,y) \\
P(x,y) &\leftarrow P(x,z), E(z,y)
\end{aligned}
$$

$E \cup \Pi \models yes$

All tuples are actual causes: every tuple appears 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 notion to that of causal responsibility: *causal effect*

(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:

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

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 the tuples in D

  A probabilistic database (tuples outside D get probability 0)

- The X_τ and $\mathcal{Q}$ become random variables

  What's the probability that $\mathcal{Q}$ takes a 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 $\{y, x\} \subseteq \{0, 1\}$: $P(\mathcal{Q} = y \mid do(X_\tau = x))$?

  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: $D^p = \{R(a,b;\tfrac{1}{2}), R(a,c;\tfrac{1}{2}), R(c,b;\tfrac{1}{2}), S(b;\tfrac{1}{2}), S(c;\tfrac{1}{2})\}$

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:

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

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

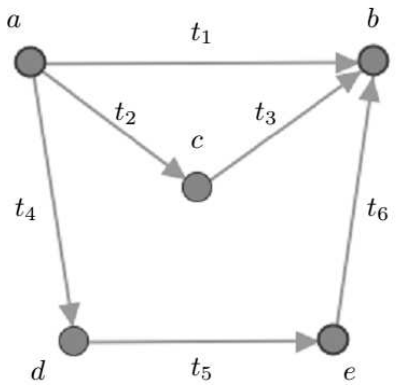
$$\mathbb{E}(\mathcal{Q} \mid do(X_{S(b)}=0)) = P(\mathcal{Q}=1 \mid do(X_{S(b)}=0)) = \tfrac{1}{4}$$

$$\mathbb{E}(\mathcal{Q} \mid do(X_{S(b)}=1)) = \tfrac{13}{16}$$

$$\mathcal{CE}^{D,\mathcal{Q}}(S(b)) = \tfrac{13}{16} - \tfrac{1}{4} = \tfrac{9}{16} > 0$$

an actual cause with this causal effect!

Example: (continued) The Datalog query (here as a union of BCQs) has the lineage:



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

$$\mathcal{CE}^{D,\mathcal{Q}}(t_1) = 0.65625$$

$$\mathcal{CE}^{D,\mathcal{Q}}(t_2) = \mathcal{CE}^{D,\mathcal{Q}}(t_3) = 0.21875$$

$$\mathcal{CE}^{D,\mathcal{Q}}(t_4) = \mathcal{CE}^{D,\mathcal{Q}}(t_5) = \mathcal{CE}^{D,\mathcal{Q}}(t_6) = 0.09375$$

Newer Development: Coalition Games and Causal Contribution

- Initial motivation: By how much a database tuple contributes to the inconsistency of a DB?

  I.e. 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
- Also based on counterfactual intervention: What would happen if we change ...?
- An “alternative” measure is the Banzhaf Index (similar ideas underneath)

  We proved “Causal Effect” coincides with the Banzhaf Index

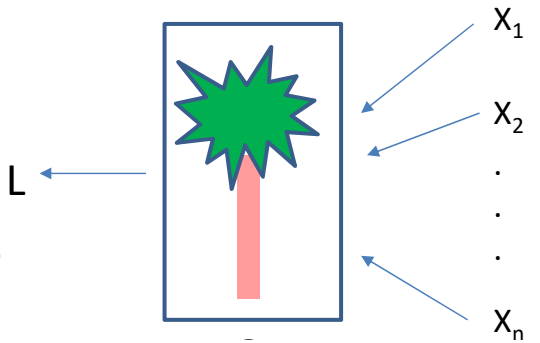
Newer Development: Explanations for Classification

$\mathbf{e} = \langle x_1, \ldots, x_n \rangle$ entity requesting a loan

Black-box classification model returns label $L(\mathbf{e}) = 1$, i.e. rejected

Why???!!!

Similarly if we have the model, e.g. a classification tree



Which are the features (or feature values) that contribute the most?

- Again: counterfactual interventions, measures (feature scores), some based on Shapley values

  Other measures are more model-dependent, etc.

relationalAI

EXTRA SLIDES

Causality and Datalog Abduction

- An *abductive explanation* for an *observation* is a formula that, together with the *background Datalog theory* Π, entails the observation

A recursive Datalog program:

$$\begin{aligned} ancestor(X,Y) &\leftarrow parent(X,Y) \\ ancestor(X,Y) &\leftarrow parent(X,Z), ancestor(Z,Y) \\ parent(mary,john) & \\ parent(mary,john) & \end{aligned}$$

- Datalog programs define monotone queries and actual causality can be applied as above

- A *Datalog abduction problem* (DAP) is of the form $\mathcal{AP} = \langle \Pi, E, Hyp, Obs \rangle$,

  (a) Π: set of Datalog rules

  (b) E: finite set of ground atoms (extensional database)

  (c) Hyp: finite set of ground atoms, the abducible atoms

  (d) Obs: observation, a finite conjunction of ground atoms

$$\Pi \cup E \cup Hyp \models Obs$$

- The *abduction problem* is about computing a subset-minimal $\Delta \subseteq Hyp$, such that $\Pi \cup E \cup \Delta \models Obs$
- Relevance Problem: Deciding if $h \in Hyp$ belongs to some abductive diagnosis is *NP*-complete! (in $|\mathcal{AP}|$)

- Datalog abduction can be represented as an actual causation problem for Datalog queries
- Datalog abduction problem $\mathcal{AP} = \langle \Pi, E, Hyp, Obs \rangle$
- Construct a causality setting:
  - Partition into exogeneous and endogenous tuples, with causes and members of contingency set among the latter:

    $D := D^x \cup D^n$ $D^x := E$, and $D^n := Hyp$
  - Query Datalog program $\Pi' := \Pi \cup \{ans \leftarrow Obs\}$

    (*ans* is fresh propositional atom)
  - Π' is seen as monotone query on D
- It holds: $h \in Hyp$ is relevant for $\mathcal{AP}$ iff h is an actual cause for *ans* wrt. (Π', D)

- These connections enable mutual applications

  In particular, new results about causality for Boolean Datalog queries

- Deciding if a tuple is a counterfactual cause, i.e. with responsibility 1, is in PTIME (in data)

  - Deciding is a tuple is an actual cause is NP-complete (in data)

    (B. & Salimi; IJAR'17)

  - For unions of Boolean conjunctive queries the problem is tractable in data

    (B. & Salimi; TOCS'17)

- Deciding if the a tuple has has a responsibility above a given threshold is NP-complete (in data)