The Complexity of Inferences and Explanations in Probabilistic Logic Programming

Fabio G. Cozman, Denis D. Mauá
Universidade de São Paulo

July 11, 2017

Overview

1. Probabilistic disjunctive logic programming.
2. The complexity of inferences and explanations.

Probabilistic disjunctive logic programs

- A probabilistic disjunctive logic program is a pair $\langle \mathbf{P}, \mathbf{PF} \rangle$:
  - **P** is a disjunctive logic program (no functions) and
  - **PF** is a set of probabilistic facts.

Probabilistic disjunctive logic programs

- A probabilistic disjunctive logic program is a pair $\langle \mathbf{P}, \mathbf{PF} \rangle$:
  - **P** is a disjunctive logic program (no functions) and
  - **PF** is a set of probabilistic facts.

- Predicate r, atom $\mathrm{r}(\mathrm{t}_1, \ldots, \mathrm{t}_k)$, rule

$$A_1 \vee \cdots \vee A_h :- B_1, \ldots, B_{b'}, \textbf{not } C_{b'+1}, \ldots, \textbf{not } C_b$$

Probabilistic disjunctive logic programs

- A probabilistic disjunctive logic program is a pair $\langle \mathbf{P}, \mathbf{PF} \rangle$:
  - **P** is a disjunctive logic program (no functions) and
  - **PF** is a set of probabilistic facts.

- Predicate r, atom $\mathrm{r}(\mathrm{t}_1, \ldots, \mathrm{t}_k)$, rule

$$A_1 \vee \cdots \vee A_h \mathrel{:-} B_1, \ldots, B_{b'}, \textbf{not } C_{b'+1}, \ldots, \textbf{not } C_b$$

  - A program without disjunction is *normal*.
  - A program without logical variables is *propositional*.

Probabilistic facts

- A probabilistic fact is a fact associated with a probability:

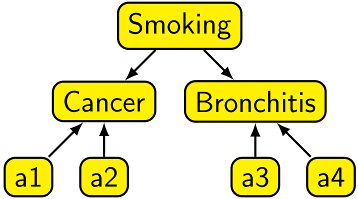
$$\mathbb{P}(A) = \alpha.$$

- Probabilistic facts are assumed independent.

Example: the Bayesian network Asia

- Predicates smoking, cancer, and bronchitis.
- Probabilistic logic program (ProbLog notation):

0.5 :: smoking.
cancer :− smoking, a1.
cancer :− **not** smoking, a2.
bronchitis :− smoking, a3.
bronchitis :− **not** smoking, a4.
0.1 :: a1.   0.01 :: a2.
0.6 :: a3.   0.3 :: a4.



Stratified normal programs:

- ... the grounded dependency graph has no cycle containing a *negative* edge.

- Example:

$$\text{path}(X, Y) :- \text{edge}(X, Y).$$
$$\text{path}(X, Y) :- \text{edge}(X, Z), \text{path}(Z, Y).$$

Stratified normal programs:

- ... the grounded dependency graph has no cycle containing a *negative* edge.

- Example:

$$\text{path}(X, Y) :- \text{edge}(X, Y).$$
$$\text{path}(X, Y) :- \text{edge}(X, Z), \text{path}(Z, Y).$$

$$0.6 :: \text{edge}(1, 2). \quad 0.1 :: \text{edge}(1, 3).$$
$$0.4 :: \text{edge}(2, 5). \quad 0.3 :: \text{edge}(2, 6).$$
$$0.3 :: \text{edge}(3, 4). \quad 0.8 :: \text{edge}(4, 5).$$
$$0.2 :: \text{edge}(5, 6).$$

A word on semantics

- The semantics of acyclic and stratified normal programs is uncontroversial: just take the unique stable model (= answer set = well-founded model).

A word on semantics

- The semantics of acyclic and stratified normal programs is uncontroversial: just take the unique stable model (= answer set = well-founded model).

Stable models:

- Consider logic program **P**.
- For some interpretation $\mathcal{I}$, take the reduct $\mathbf{P}^{\mathcal{I}}$:
  - Ground **P**.
  - Remove rules with subgoal **not** A and $A \in \mathcal{I}$.
  - Remove subgoals **not** A from remaining rules.
- Interpretation $\mathcal{I}$ is stable model if $\mathcal{I}$ is the minimal model of $\mathbf{P}^{\mathcal{I}}$.

Non-stratified program (cycle with negative edge)

- Non-stratified program may have more than one stable model.

The Dilbert example

$$\text{single}(X) \coloneq \text{man}(X), \textbf{not } \text{husband}(X).$$

$$\text{husband}(X) \coloneq \text{man}(X), \textbf{not } \text{single}(X).$$

$$0.9 :: \text{man}(\text{dilbert}).$$

Non-stratified program (cycle with negative edge)

- Non-stratified program may have more than one stable model.

The Dilbert example

$$\text{single}(X) \coloneq \text{man}(X), \textbf{not } \text{husband}(X).$$

$$\text{husband}(X) \coloneq \text{man}(X), \textbf{not } \text{single}(X).$$

$$0.9 :: \text{man}(\text{dilbert}).$$

- man(dilbert) is false: a unique stable model s_1.

Non-stratified program (cycle with negative edge)

- Non-stratified program may have more than one stable model.

The Dilbert example

$$\text{single}(X) :- \text{man}(X), \textbf{not } \text{husband}(X).$$

$$\text{husband}(X) :- \text{man}(X), \textbf{not } \text{single}(X).$$

$$0.9 :: \text{man}(\text{dilbert}).$$

- man(dilbert) is false: a unique stable model s_1.
- man(dilbert) is true: there are two stable models,

$$s_2 = \{\text{husband}(\text{dilbert}) = \text{true}, \text{single}(\text{dilbert}) = \text{false}\},$$

and

$$s_3 = \{\text{husband}(\text{dilbert}) = \text{false}, \text{single}(\text{dilbert}) = \text{true}\}.$$

What could be the semantics of a non-stratified program?

- Probabilities over well-founded models:
  - Sato, Kameya and Zhou (2005),
  - Hadjichristodolou and Warren (2012).
  - Riguzzi (2015).

- Proposal by Lukasiewicz (2005):
  informally, take the set of every possible probability distributions that satisfy the rules and (probabilistic) facts.
  - We adopt name *credal semantics*.
  - Note: another recent semantics based on credal sets by Michels et al. (2015).

An example

The Dilbert example

$$\text{single}(X) \text{ :– } \text{man}(X), \textbf{not } \text{husband}(X).$$

$$\text{husband}(X) \text{ :– } \text{man}(X), \textbf{not } \text{single}(X).$$

$$0.9 :: \text{man}(\text{dilbert}).$$

An example

The Dilbert example

$$\text{single}(X) :- \text{man}(X), \textbf{not } \text{husband}(X).$$

$$\text{husband}(X) :- \text{man}(X), \textbf{not } \text{single}(X).$$

$$0.9 :: \text{man}(\text{dilbert}).$$

- Take any $\gamma \in [0, 1]$:

$$\mathbb{P}(s_1) = 0.1, \quad \mathbb{P}(s_2) = 0.9\gamma, \quad \mathbb{P}(s_3) = 0.9(1 - \gamma).$$

An example: robot navigation (graph coloring...)

$$\text{color}(X, \text{red}) \vee \text{color}(X, \text{green}) \vee \text{color}(X, \text{yellow}) :- \text{site}(X).$$

$$\text{clash} :- \textbf{not}\ \text{clash}, \text{edge}(X, Y), \text{color}(X, C), \text{color}(Y, C).$$

$$\text{path}(X, Y) :- \text{edge}(X, Y). \qquad \text{path}(X, Y) :- \text{edge}(X, Z), \text{path}(Z, Y).$$

An example: robot navigation (graph coloring...)

$$\mathrm{color}(X,\mathrm{red}) \vee \mathrm{color}(X,\mathrm{green}) \vee \mathrm{color}(X,\mathrm{yellow}) :\!-\ \mathrm{site}(X).$$

$$\mathrm{clash} :\!-\ \textbf{not}\ \mathrm{clash}, \mathrm{edge}(X,Y), \mathrm{color}(X,C), \mathrm{color}(Y,C).$$

$$\mathrm{path}(X,Y) :\!-\ \mathrm{edge}(X,Y). \qquad \mathrm{path}(X,Y) :\!-\ \mathrm{edge}(X,Z), \mathrm{path}(Z,Y).$$

$$\mathrm{site}(1). \quad \mathrm{site}(2). \quad \mathrm{site}(3). \quad \mathrm{site}(4). \quad \mathrm{site}(5).$$

$$\mathrm{color}(2,\mathrm{red}). \quad \mathrm{color}(5,\mathrm{green}).$$

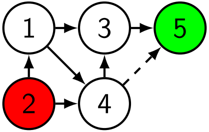
$$0.5 :: \mathrm{edge}(4,5).$$

$$\mathrm{edge}(1,3). \quad \mathrm{edge}(1,4). \quad \mathrm{edge}(2,1). \quad \mathrm{edge}(2,4). \quad \mathrm{edge}(3,5). \quad \mathrm{edge}(4,3).$$

An example: robot navigation (graph coloring...)

$$\text{color}(X, \text{red}) \vee \text{color}(X, \text{green}) \vee \text{color}(X, \text{yellow}) :- \text{site}(X).$$
$$\text{clash} :- \textbf{not}\ \text{clash}, \text{edge}(X, Y), \text{color}(X, C), \text{color}(Y, C).$$
$$\text{path}(X, Y) :- \text{edge}(X, Y). \qquad \text{path}(X, Y) :- \text{edge}(X, Z), \text{path}(Z, Y).$$

site(1). site(2). site(3). site(4). site(5).
color(2, red). color(5, green).
0.5 :: edge(4, 5).
edge(1, 3). edge(1, 4). edge(2, 1). edge(2, 4). edge(3, 5). edge(4, 3).



Inferences

- Inference: whether $\underline{\mathbb{P}}(\mathbf{Q}|\mathbf{E}) > \gamma$.

Inferences

- Inference: whether $\mathbb{P}(\mathbf{Q}|\mathbf{E}) > \gamma$.

- MPE: whether there is an interpretation $\mathbf{Q}$ that agrees with literals $\mathbf{E}$, such that $\mathbb{P}(\mathbf{Q}) > \gamma$.

Inferences

- Inference: whether $\mathbb{P}(\mathbf{Q}|\mathbf{E}) > \gamma$.
- MPE: whether there is an interpretation $\mathbf{Q}$ that agrees with literals $\mathbf{E}$, such that $\mathbb{P}(\mathbf{Q}) > \gamma$.
- MAP: whether there is a partial interpretation $\mathbf{Q}$ that agrees with literals $\mathbf{E}$, such that $\mathbb{P}(\mathbf{Q}|\mathbf{E}) > \gamma$.

Results

	Propositional			Bounded arity		
	Inferential	MPE	MAP	Inferential	MPE	MAP
Acyclic normal	PP	NP	NP^{PP}	PP^{NP}	Σ_2^P	NP^{PP}
No negation, normal	PP	NP	NP^{PP}	PP^{NP}	Σ_2^P	NP^{PP}
Stratified normal	PP	NP	NP^{PP}	PP^{NP}	Σ_2^P	NP^{PP}
Normal, credal	PP^{NP}	Σ_2^P	NP^{PP}	$PP^{\Sigma_2^P}$	Σ_3^P	NP^{PP}
Normal, well-founded	PP	NP	NP^{PP}	PP^{NP}	Σ_2^P	NP^{PP}
Disjunctive, credal	$PP^{\Sigma_2^P}$	Σ_3^P	NP^{PP}	$PP^{\Sigma_3^P}$	Σ_4^P	NP^{PP}

(Complexity class $\Sigma_i^P = NP^{\Sigma_{i-1}^P}$.)
(Complexity class PP: class of problems solved by a probabilistic polynomial-time Turing machine.)

(In orange: PGM2016, WPLP2016, ENIAC2016.)

Results

	Propositional			Bounded arity		
	Inferential	MPE	MAP	Inferential	MPE	MAP
Acyclic normal	PP	NP	$\mathsf{NP}^{\mathsf{PP}}$	$\mathsf{PP}^{\mathsf{NP}}$	Σ_2^P	$\mathsf{NP}^{\mathsf{PP}}$
No negation, normal	PP	NP	$\mathsf{NP}^{\mathsf{PP}}$	$\mathsf{PP}^{\mathsf{NP}}$	Σ_2^P	$\mathsf{NP}^{\mathsf{PP}}$
Stratified normal	PP	NP	$\mathsf{NP}^{\mathsf{PP}}$	$\mathsf{PP}^{\mathsf{NP}}$	Σ_2^P	$\mathsf{NP}^{\mathsf{PP}}$
Normal, credal	$\mathsf{PP}^{\mathsf{NP}}$	Σ_2^P	$\mathsf{NP}^{\mathsf{PP}}$	$\mathsf{PP}^{\Sigma_2^P}$	Σ_3^P	$\mathsf{NP}^{\mathsf{PP}}$
Normal, well-founded	PP	NP	$\mathsf{NP}^{\mathsf{PP}}$	$\mathsf{PP}^{\mathsf{NP}}$	Σ_2^P	$\mathsf{NP}^{\mathsf{PP}}$
Disjunctive, credal	$\mathsf{PP}^{\Sigma_2^P}$	Σ_3^P	$\mathsf{NP}^{\mathsf{PP}}$	$\mathsf{PP}^{\Sigma_3^P}$	Σ_4^P	$\mathsf{NP}^{\mathsf{PP}}$

(Complexity class $\Sigma_i^P = \mathsf{NP}^{\Sigma_{i-1}^P}$.)
(Complexity class PP: class of problems solved by a probabilistic polynomial-time Turing machine.)

Results

	Propositional			Bounded arity		
	Inferential	MPE	MAP	Inferential	MPE	MAP
Acyclic normal	PP	NP	NP^{PP}	PP^{NP}	Σ_2^P	NP^{PP}
No negation, normal	PP	NP	NP^{PP}	PP^{NP}	Σ_2^P	NP^{PP}
Stratified normal	PP	NP	NP^{PP}	PP^{NP}	Σ_2^P	NP^{PP}
Normal, credal	PP^{NP}	Σ_2^P	NP^{PP}	$PP^{\Sigma_2^P}$	Σ_3^P	NP^{PP}
Normal, well-founded	PP	NP	NP^{PP}	PP^{NP}	Σ_2^P	NP^{PP}
Disjunctive, credal	$PP^{\Sigma_2^P}$	Σ_3^P	NP^{PP}	$PP^{\Sigma_3^P}$	Σ_4^P	NP^{PP}

(Complexity class $\Sigma_i^P = NP^{\Sigma_{i-1}^P}$.)
(Complexity class PP: class of problems solved by a probabilistic polynomial-time Turing machine.)

Results

	Propositional			Bounded arity		
	Inferential	MPE	MAP	Inferential	MPE	MAP
Acyclic normal	PP	NP	NP^{PP}	PP^{NP}	Σ_2^P	NP^{PP}
No negation, normal	PP	NP	NP^{PP}	PP^{NP}	Σ_2^P	NP^{PP}
Stratified normal	PP	NP	NP^{PP}	PP^{NP}	Σ_2^P	NP^{PP}
Normal, credal	PP^{NP}	Σ_2^P	NP^{PP}	$PP^{\Sigma_2^P}$	Σ_3^P	NP^{PP}
Normal, well-founded	PP	NP	NP^{PP}	PP^{NP}	Σ_2^P	NP^{PP}
Disjunctive, credal	$PP^{\Sigma_2^P}$	Σ_3^P	NP^{PP}	$PP^{\Sigma_3^P}$	Σ_4^P	NP^{PP}

(Complexity class $\Sigma_i^P = NP^{\Sigma_{i-1}^P}$.)
(Complexity class PP: class of problems solved by a probabilistic polynomial-time Turing machine.)

Results

	Propositional			Bounded arity		
	Inferential	MPE	MAP	Inferential	MPE	MAP
Acyclic normal	PP	NP	NP^{PP}	PP^{NP}	Σ_2^P	NP^{PP}
No negation, normal	PP	NP	NP^{PP}	PP^{NP}	Σ_2^P	NP^{PP}
Stratified normal	PP	NP	NP^{PP}	PP^{NP}	Σ_2^P	NP^{PP}
Normal, credal	PP^{NP}	Σ_2^P	NP^{PP}	$PP^{\Sigma_2^P}$	Σ_3^P	NP^{PP}
Normal, well-founded	PP	NP	NP^{PP}	PP^{NP}	Σ_2^P	NP^{PP}
Disjunctive, credal	$PP^{\Sigma_2^P}$	Σ_3^P	NP^{PP}	$PP^{\Sigma_3^P}$	Σ_4^P	NP^{PP}

(Complexity class $\Sigma_i^P = NP^{\Sigma_{i-1}^P}$.)
(Complexity class PP: class of problems solved by a probabilistic polynomial-time Turing machine.)

Results

	Propositional			Bounded arity		
	Inferential	MPE	MAP	Inferential	MPE	MAP
Acyclic normal	PP	NP	NP^{PP}	PP^{NP}	Σ_2^P	NP^{PP}
No negation, normal	PP	NP	NP^{PP}	PP^{NP}	Σ_2^P	NP^{PP}
Stratified normal	PP	NP	NP^{PP}	PP^{NP}	Σ_2^P	NP^{PP}
Normal, credal	PP^{NP}	Σ_2^P	NP^{PP}	$PP^{\Sigma_2^P}$	Σ_3^P	NP^{PP}
Normal, well-founded	PP	NP	NP^{PP}	PP^{NP}	Σ_2^P	NP^{PP}
Disjunctive, credal	$PP^{\Sigma_2^P}$	Σ_3^P	NP^{PP}	$PP^{\Sigma_3^P}$	Σ_4^P	NP^{PP}

(Complexity class $\Sigma_i^P = NP^{\Sigma_{i-1}^P}$.)
(Complexity class PP: class of problems solved by a probabilistic polynomial-time Turing machine.)

Results

	Propositional			Bounded arity		
	Inferential	MPE	MAP	Inferential	MPE	MAP
Acyclic normal	PP	NP	NP^{PP}	PP^{NP}	Σ_2^P	NP^{PP}
No negation, normal	PP	NP	NP^{PP}	PP^{NP}	Σ_2^P	NP^{PP}
Stratified normal	PP	NP	NP^{PP}	PP^{NP}	Σ_2^P	NP^{PP}
Normal, credal	PP^{NP}	Σ_2^P	NP^{PP}	$PP^{\Sigma_2^P}$	Σ_3^P	NP^{PP}
Normal, well-founded	PP	NP	NP^{PP}	PP^{NP}	Σ_2^P	NP^{PP}
Disjunctive, credal	$PP^{\Sigma_2^P}$	Σ_3^P	NP^{PP}	$PP^{\Sigma_3^P}$	Σ_4^P	NP^{PP}

(Complexity class $\Sigma_i^P = NP^{\Sigma_{i-1}^P}$.)
(Complexity class PP: class of problems solved by a probabilistic polynomial-time Turing machine.)

Conclusion

- Main goal was to map the complexity of probabilistic disjunctive logic programming (and its sub-languages) and credal and well-founded semantics.
- Future work: remove bounds on arity, and consider query complexity.

- Thanks to support by CNPq and FAPESP.