Imperial College London

Learning Weak Constraints in Answer Set Programming

Mark Law, Alessandra Russo and Krysia Broda

September 4, 2015

Preference Learning

- Learning to rank is an approach to preference learning where the goal is to learn to order objects given pairwise examples of a user's preferences.
- For example, learning academic's preferences about interview scheduling.

	1	2	3
M	c1	c2	c2
T	c2	c2	c2
W	c2	c1	c2

is preferred to

	1	2	3
M	c1	c2	c2
T	c2	c2	c2
W	c2	c1	c2

Although traditional machine learning techniques can be used to predict preferences, their reasoning is not easily human readable.

Inductive Logic Programming

- Given a set of positive examples E^+, negative examples E^- and a background knowledge B, the goal is to find a hypothesis H such that:
  - $\forall e \in E^+ : B \cup H \models e$
  - $\forall e \in E^- : B \cup H \not\models e$

Imperial College London

Inductive Logic Programming

- Given a set of positive examples E^+, negative examples E^- and a background knowledge B, the goal is to find a hypothesis H such that:
  - $\forall e \in E^+ : B \cup H \models e$
  - $\forall e \in E^- : B \cup H \not\models e$
- The key advantages are that:
  - The hypotheses are human readable.
  - Can define useful concepts in the background knowledge.
  - Can give a very structured language bias to guide the search.

Learning from Answer Sets (ILP_{LAS})

- In ILP_{LAS} (Law et al. 2014), examples are *partial interpretations*.
- A partial interpretation e is a pair of sets of atoms $\langle e^{inc}, e^{exc} \rangle$.
- An answer set A extends e iff $e^{inc} \subseteq A$ and $e^{exc} \cap A = \emptyset$.

	1	2	3
M	c1	c2	c2
T	c2	c2	c2
W	c2	c1	c2

is extended by

	1	2	3
M	c1	c2	c2
T	c2	c2	c2
W	c2	c1	c2

Learning from Answer Sets (ILP_{LAS})

Definition 1

An ILP_{LAS} task is a tuple $T = \langle B, S_M, E^+, E^- \rangle$. A hypothesis $H \subseteq S_M$ is in $ILP_{LAS}(T)$, the set of all inductive solutions of T, if and only if:

- $\forall e^+ \in E^+ \; \exists A \in AS(B \cup H)$ *such that A extends e^+*
- $\forall e^- \in E^- \; \nexists A \in AS(B \cup H)$ *such that A extends e^-.*

(Law et al. 2014)

Imperial College London

Weak Constraints in ASP

:∼ assign(D, S),type(D, S, c1).[1@2, D, S]

:∼ assign(D, S1),assign(D, S2),neq(S1, S2).[1@1, D, S1, S2]

	1	2	3
M	c1	c2	c2
T	c2	c2	c2
W	c2	c1	c2

is preferred to

	1	2	3
M	c1	c2	c2
T	c2	c2	c2
W	c2	c1	c2

Weak Constraints in ASP

:∼ assign(D, S),type(D, S, c1). [1@2, D, S]

:∼ assign(D, S1),assign(D, S2),neq(S1, S2). [1@1, D, S1, S2]

	1	2	3
M	c1	c2	c2
T	c2	c2	c2
W	c2	c1	c2

is preferred to

	1	2	3
M	c1	c2	c2
T	c2	c2	c2
W	c2	c1	c2

Weak Constraints in ASP

```
:~ assign(D, S),type(D, S, c1).[1@2, D, S]
:~ assign(D, S1),assign(D, S2),neq(S1, S2).[1@1, D, S1, S2]
```

	1	2	3
M	c1	c2	c2
T	c2	c2	c2
W	c2	c1	c2

is preferred to

	1	2	3
M	c1	c2	c2
T	c2	c2	c2
W	c2	c1	c2

Weak Constraints in ASP

:~ assign(D, S),type(D, S, c1).[1@2, D, S]

:~ assign(D, S1),assign(D, S2),neq(S1, S2).[1@1, D, S1, S2]

	1	2	3
M	c1	c2	c2
T	c2	c2	c2
W	c2	c1	c2

is preferred to

	1	2	3
M	c1	c2	c2
T	c2	c2	c2
W	c2	c1	c2

Learning from Ordered Answer Sets

- As previous ILP frameworks could only give examples of what should (or shouldn't) be an answer set of the program, no existing framework could incentivise learning weak constraints.
- We now present an extension of ILP_{LAS} (Law et al. 2014), with a new type of example aimed at learning weak constraints.

Ordering Examples

Definition 3

An ordering example is a tuple $o = \langle e_1, e_2 \rangle$ where e_1 and e_2 are partial interpretations.

Brave Scheduling Example

```
:~ assign(D, S1),assign(D, S2),neq(S1, S2).[1@1, D, S1, S2]
:~ assign(D, S),type(D, S, c1).[1@2, D, S]
```

	1	2	3
M	c1	c2	c2
T	c2	c2	c2
W	c2	c1	c2

is sometimes preferred to

	1	2	3
M	c1	c2	c2
T	c2	c2	c2
W	c2	c1	c2

Definition 3

- *An ASP program P* bravely respects o *iff* $\exists A_1, A_2 \in AS(P)$ *such that* A_1 *extends* e_1, A_2 *extends* e_2 *and* $A_1 \succ_P A_2$.

Cautious Scheduling Example

```
:~ assign(D, S1),assign(D, S2),neq(S1, S2).[1@1, D, S1, S2]
:~ assign(D, S),type(D, S, c1).[1@2, D, S]
```

	1	2	3
M	c1	c2	c2
T	c2	c2	c2
W	c2	c1	c2

is always preferred to

	1	2	3
M	c1	c2	c2
T	c2	c2	c2
W	c2	c1	c2

Definition 3

- P cautiously respects o *iff* $\forall A_1, A_2 \in AS(P)$ *such that* A_1 *extends* e_1 *and* A_2 *extends* e_2*, it is the case that* $A_1 \succ_P A_2$.

Learning from Ordered Answer Sets (ILP_{LOAS})

Definition 4

An ILP_{LOAS} task is a tuple $T = \langle B, S_M, E^+, E^-, O^b, O^c \rangle$.
A hypothesis $H \subseteq S_M$ is in $ILP_{LOAS}(T)$, the inductive solutions of T, if and only if:

1. $\forall e^+ \in E^+ \; \exists A \in AS(B \cup H)$ *such that A extends e^+*
2. $\forall e^- \in E^- \; \nexists A \in AS(B \cup H)$ *such that A extends e^-.*
3. $\forall o \in O^b \;\; B \cup H$ *bravely respects o*
4. $\forall o \in O^c \;\; B \cup H$ *cautiously respects o*

Imperial College London

Complexity

Theorem 3

Let T be any propositional ILP_{LAS} or ILP_{LOAS} task. Deciding whether T has at least one inductive solution is NP^{NP}-complete.

Algorithm

- Our new algorithm ILASP2 (Inductive Learning of Answer Set Programs) is sound and complete wrt the optimal (shortest) solutions of any ILP_{LOAS} task.
- It is available for download at `https://www.doc.ic.ac.uk/~ml1909/ILASP`.
- It extends our previous ILASP1 algorithm for solving ILP_{LAS} tasks (Law et al. 2014).

Positive and Violating Hypotheses

Definition 5/6

Let $T = \langle B, S_M, E^+, E^- \rangle$ *be an* ILP_{LAS} *task. Any* $H \subseteq S_M$ *is a* positive hypothesis *iff* $\forall e \in E^+ \; \exists A \in AS(B \cup H)$ *such that* A *extends* e*.*

A positive hypothesis H *is violating iff* $\exists e^- \in E^-, \exists A \in AS(B \cup H)$ *such that* A *extends* e^-*.*

(Law et al. 2014)

Positive and Violating Hypotheses

$E^+ = \{$

	1	2
M	c1	c2
T	c2	c2

$\}$ $E^- = \{$

	1	2
M	c1	c2
T	c2	c2

,

	1	2
M	c1	c2
T	c2	c2

$\}$

$$B = \left\{ \begin{array}{l} \texttt{slot(m, 1..2). slot(t, 1..2). type(m, 1, c1).} \\ \texttt{type(m, 2, c2). type(t, 1, c2). type(t, 2, c2).} \\ \texttt{0\{assign(D, S)\}1:-slot(D, S).} \end{array} \right\}$$

$$H = \emptyset$$

This is a positive hypothesis, but is also violating.

Positive and Violating Hypotheses

$E^{+} =$ {

	1	2
M	c1	c2
T	c2	c2

}

$E^{-} =$ {

	1	2
M	c1	c2
T	c2	c2

,

	1	2
M	c1	c2
T	c2	c2

}

$B =$ {

```
slot(m,1..2). slot(t,1..2). type(m,1,c1).
type(m,2,c2). type(t,1,c2). type(t,2,c2).
0{assign(D,S)}1:-slot(D,S).
```

}

$H =$ {

```
:-assign(m,S).
```

}

This is a positive hypothesis which is not violating. Hence, it is an inductive solution.

Positive and Violating Hypotheses

Definition 5/6

Let $T = \langle B, S_M, E^+, E^-, O^b, O^c \rangle$ *be an* ILP_{LOAS} *task. Any* $H \subseteq S_M$ *is a* positive hypothesis *iff* $\forall e \in E^+ \; \exists A \in AS(B \cup H)$ *such that* A *extends* e*, and* $\forall o \in O^b$ $B \cup H$ *bravely respects* o*.*

A positive hypothesis H *is violating iff at least one of the following cases is true:*

- $\exists e^- \in E^-, \exists A \in AS(B \cup H)$ *such that* A *extends* e^-*.*
- $\exists A_1, A_2 \in AS(B \cup H)$ *and* $\exists \langle e_1, e_2 \rangle \in O^c$ *such that* A_1 *extends* e_1*,* A_2 *extends* e_2 *and* $A_1 \not\succ_P A_2$ *with respect to* $B \cup H$*.*

ILASP1

procedure $\mathrm{ILASP1}(T)$
    solutions $= []$
    $n = 0$
    while *solutions* $== []$ **do**
        $V = violating\_hypotheses(T, n)$
        $solutions = remaining\_positive\_hypothses(T, n, V)$
        $n = n + 1$
    end while
    return *solutions*
end procedure

(Law et al. 2014)

Positive Hypotheses and Violating Reasons

Definition 5/6

Let $T = \langle B, S_M, E^+, E^-, O^b, O^c \rangle$ *be an* ILP_{LOAS} *task. Any* $H \subseteq S_M$ *is a* positive hypothesis *iff* $\forall e \in E^+ \; \exists A \in AS(B \cup H)$ *such that* A *extends* e*, and* $\forall o \in O^b$ $H \cup B$ *bravely respects* o*.*

A positive hypothesis H *is violating iff at least one of the following cases is true:*

- $\exists e^- \in E^-, \exists A \in AS(B \cup H)$ *such that* A *extends* e^-*.*
  In this case we call A *a violating interpretation of* T*.*
- $\exists A_1, A_2 \in AS(B \cup H)$ *and* $\exists \langle e_1, e_2 \rangle \in O^c$ *such that* A_1 *extends* e_1*,* A_2 *extends* e_2 *and* $A_1 \not\succ_P A_2$ *with respect to* $B \cup H$*.*
  In this case, we call $\langle A_1, A_2 \rangle$ *a violating pair of* T*.*

Meta Representation: Properties

Given a task T and a set of violating reasons VR, we defined a meta program $\mathcal{M}(T, VR)$.

- The A's $\in AS(\mathcal{M}(T, VR))$ map to the positive hypotheses $\mathcal{M}^{-1}_{hyp}(A)$ which do not violate any $vr \in VR$.
- If H is violating, there is an $A \in AS(\mathcal{M}(T, VR))$ st $H = \mathcal{M}^{-1}_{hyp}(A)$ and $\texttt{violating} \in A$.
- The optimality of each answer set A is $2|\mathcal{M}^{-1}_{hyp}(A)|$ if $\texttt{violating} \in A$ and $2|\mathcal{M}^{-1}_{hyp}(A)| + 1$ otherwise.

ILASP2

procedure $\text{ILASP2}(T)$
  $VR = []$
  $A = solve(\mathcal{M}(T, VR))$
  while $A \neq$ `nil` && `violating` $\in A$ **do**
    $VR \mathrel{+}= \mathcal{M}_{vr}^{-1}(A)$
    $A = solve(\mathcal{M}(T, VR))$
  end while
  return $\{\mathcal{M}_{hyp}^{-1}(A) \mid A \in AS^*(\mathcal{M}(T, VR))\}$
end procedure

- $solve(P)$ returns an optimal answer set of P (it returns `nil` if $AS(P) = \emptyset$).
- $\mathcal{M}_{hyp}^{-1}$ maps an answer set of $\mathcal{M}(T, VR)$ to the corresponding hypothesis.
- $\mathcal{M}_{vr}^{-1}$ maps an answer set of $\mathcal{M}(T, VR)$ to the corresponding violating reason.

Imperial College London

Soundness and Completeness

Theorem 5

Let T be an ILP_{LOAS} task. If $ILASP2(T)$ terminates, then $ILASP2(T)$ returns the set of optimal inductive solutions of $ILP_{LOAS}(T)$.

Sudoku Experiment

- In (Law et al. 2014) we showed that ILASP1 could be used to learn the rules of a 4x4 version of sudoku.
- This took 486.2s (over 8 minutes) to solve with ILASP1 as there were 332437 violating hypotheses found before the first inductive solution.
- ILASP2 only needs 9 violating reasons and so solves the same task in 0.69s (less than 1 second).

Experiments

Our main experiments were on the interview scheduling example.

- For each experiment, we randomly generated hypotheses H with up to 3 weak constraints from S_M.
- We used each H to randomly generate orderings.
- We then used ILASP2 to learn a hypothesis H' which covered these examples.
- We checked the accuracy of H' at predicting the orderings of the timetables given by H.

Experiments

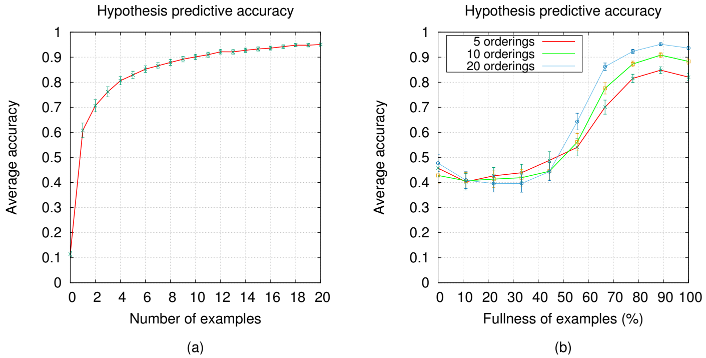

Figure : Accuracy with varying (a) numbers of examples; (b) fullness of examples

Imperial College London

Experiments

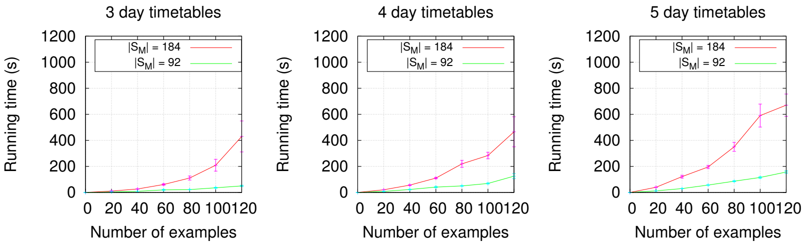

Figure : Average running time of ILASP2 with varying numbers of examples

Comparison of Non-monotonic ILP systems

Learning Task	Normal Rules	Choice Rules	Constraints	Brave	Cautious	Weak Constraints	Algorithm for optimal solutions
Brave Induction [Sakama, Inoue 2009]	✔	✔	✖	✔	✖	✖	✖
Cautious Induction [Sakama, Inoue 2009]	✔	✔	✖	✖	✔	✖	✖
XHAIL [Ray 2009] & ASPAL [Corapi et al 2011]	✔	✖	✖	✔	✖	✖	✔
Induction of Stable Models [Otero 2001]	✔	✖	✖	✔	✖	✖	✖
Induction from Answer Sets [Sakama 2005]	✔	✖	✔	✔	✔	✖	✖
LAS [Law et al 2014]	✔	✔	✔	✔	✔	✖	✔
LOAS	✔	✔	✔	✔	✔	✔	✔

Other Related Work in ILP

- ILP systems have previously been used for preference learning (Dastani et al. 2001, Horváth 2012). This addressed learning ratings, such as *good*, *poor* and *bad*, rather than rankings over the examples.

Summary

- Presented ILP_{LOAS}, which is the first framework capable of learning weak constraints.
- Presented ILASP2, which is sound and complete wrt ILP_{LOAS}.
- Proved the complexity of deciding the existence of solutions for ILP_{LAS} and ILP_{LOAS}.
- Showed that ILASP2 is more efficient than ILASP1 for the previous ILP_{LAS} task.
- Gave experimental results of ILASP2 in the setting of learning accademic's preferences for interview scheduling.

Future Work

- Support noisy examples.
- Improve the performance with larger numbers of examples.

CORAPI, D., RUSSO, A., AND LUPU, E. 2012.
Inductive logic programming in answer set programming.
In *Inductive Logic Programming*. Springer, 91–97.

DASTANI, M., JACOBS, N., JONKER, C. M., AND TREUR, J. 2001.
Modeling user preferences and mediating agents in electronic commerce.
In *Agent Mediated Electronic Commerce*. Springer, 163–193.

HORVÁTH, T. 2012.
A model of user preference learning for content-based recommender systems.
Computing and informatics 28, 4, 453–481.

LAW, M., RUSSO, A., AND BRODA, K. 2014.

Inductive learning of answer set programs.
In *Logics in Artificial Intelligence (JELIA 2014)*. Springer.

Otero, R. P. 2001.
Induction of stable models.
In *Inductive Logic Programming*. Springer, 193–205.

Ray, O. 2009.
Nonmonotonic abductive inductive learning.
Journal of Applied Logic 7, 3, 329–340.

Sakama, C. 2005.
Induction from answer sets in nonmonotonic logic programs.
ACM Transactions on Computational Logic (TOCL) 6, 2, 203–231.

Sakama, C. and Inoue, K. 2009.

Brave induction: a logical framework for learning from incomplete information.
Machine Learning 76, 1, 3–35.