Nonmonotonic Inductive Logic Programming (NMILP)

Chongbing Liu

October 24, 2005

Outlines

- Why NMILP?
- SLDNF Based Approaches
- Moving from ILP to NMILP
- Stable Models Based Approaches

Why NMILP?

Nonmonotonic Logic Programming (NMLP)

- normal logic programs (CAW by NAF)
  $A_0 \leftarrow A_1, \ldots, A_m, notA_{m+1}, \ldots, notA_n$

- mainly, stable model semantics (beliefs)

- default reasoning on incomplete knowledge (defaults + observation $\rightsquigarrow$ conclusion)

- rules acts as contraints or derivation rules (not as definitions)

- nonmonotonicity (addition of new info may contradict previous conclusions)

- No learning mechanisms are provided

Inductive Logic Programming (ILP)

- Given :
  - Background Knowledge $\mathcal{B}$ and
  - Examples $E = E^+ \cup E^-$ ($\mathcal{B} \not\models E$)

  Find a theory H such that
  - $\mathcal{B} \cup H \models e$ for every $e \in E^+$
  - $\mathcal{B} \cup H \not\models f$ for every $f \in E^-$

- Present ILP uses Horn clauses for $\mathcal{B}$ and H
  - less expressive language
  - monotonic reasoning

- armed with various learning mechanisms
  - incremental learning (non-monotonic learning)
  - batch learning (monotonic learning)
  - top-down search and bottom-up search
  - inverse resolution
  - inverse entailment

Nonmonotonic Inductive Logic Programming (NMILP)

NMLP: expressive language, human commonsense reasoning, but no learning mechanisms

ILP: language with limited expressiveness, armed with learning mechanisms, but does not simulate human commonsense reasoning

NMILP: hopefully takes advantages of both paradigms

NMILP = NMLP + ILP

SLDNF Based Approaches

Representive Efforts

- Non-monotonic learning, M. Bain, S.Muggleton, 1992
- Learning Logic Programs with negation as failure, 1996
- Learning nonmonotonic logic program: learning exceptions, 1995
- Normal programs and multiple predicate learning, 1998
- Learning extended logic programs, 1997
- A three-valued framework for the induction of general programs, 1996

Incremental Learning

Initialize Σ to $\{\Box\}$

repeat

  read the next (positive or negative) example

  while Σ is not correct w.r.t. the examples read so far

    if $\exists e^-$ s.t. $\Sigma \models e^-$

      specialize Σ by identifying a false clause

      and delete it from Σ

    if $\exists e^+$ s.t. $\Sigma \not\models e^+$

      generalize Σ by constructing a clause $C \models e$

      and add it to Σ

until no examples left to read.

=============================

An Example

$\mathcal{B} = \{bird(swan).\quad bird(eagle).\quad bird(penguin).\quad bird(pigeon).\}$

$E = \{\oplus flies(swan).\quad \oplus\ flies(eagle).\quad \ominus\ flies(penguin).\}$

$$\Sigma_0 = \Box$$

$$\downarrow \oplus{:}flies(swan)$$

$$\Sigma_1 = \{flies(X) \leftarrow bird(X).\}$$

$$\downarrow \oplus{:}flies(eagle)$$

$$\Sigma_2 = \{flies(X) \leftarrow bird(X).\}$$

$$\downarrow \ominus{:}flies(penguin)$$

$$\Sigma_3 = \{flies(swan).\quad flies(eagle).\}$$

An Example

$\mathcal{B} = \{bird(swan).\ \ bird(eagle).\ \ bird(penguin).\ \ bird(pigeon).\}$

$E = \{\oplus flies(swan).\ \ \oplus\ flies(eagle).\ \ \ominus\ flies(penguin).\}$

$$\Sigma_0 = \Box$$

$$\downarrow \oplus : flies(swan)$$

$$\Sigma_1 = \{flies(X) \leftarrow bird(X).\}$$

$$\downarrow \oplus : flies(eagle)$$

$$\Sigma_2 = \{flies(X) \leftarrow bird(X).\}$$

$$\downarrow \ominus : flies(penguin)$$

$$\Sigma_3 = \{flies(swan).\ \ flies(eagle).\}$$

================================

comments

1. monotonic reasoning (Horn clauses based)
2. non-monotonic learning (correct info not preserved, e.g., both Σ_1 and Σ_2 imply $flies(pegion)$, but Σ_3 does not.)
3. may result in poor learning quality
4. due to problem of "overly(drastic)-specialization"
5. we desire to preserve correct info
6. can not be achieved by any forms of "incremental-specialization" within classical logic framework
7. SOLUTION: introducing negation !

Closed World Specialization

Input:

set of clauses T (possibly with negation) and ground atom A s.t. $T \models A$ and A is incorrect

Operations:

Generate proof of $T \models A$ using SLDNF-resolution

Assume $C \in T$ resolved with $\leftarrow A$

Let $C = Hd : -Bd$

Let θ be the substitution for variables in C

If literal $notB \in Bd$

Let $T' = T \cup \{B\theta\}$

else

Let $\{V_1, \ldots, V_n\}$ be the domain of θ
Let q be a predicate symbol not found in T
Let $B = q(V_1, \ldots, V_n)$
Let $T' = T - \{C\} \cup \{Hd : -(Bd \cup notB\} \cup \{B\theta\}$

Output: T'

Note: T' specializes T, but not in traditional sense, since T' has a new predicate symbol.

==================================

In our example, the following theory will be learned

$\{flies(X) \leftarrow bird(X), not\ flightless(X).\ \ flightless(penguin).\}$

Now since $flightless(penguin)$ is $true$, $not\ flightless(penguin)$ is $false$. Therefore, the newly learned theory does not derive $flies(penguin)$ any more.

Moving from ILP to NMILP

Inverse Resolution is not directly applicable in NMILP !

Inverse resolution:(absorption)

$C_1 : q \leftarrow A$ $\qquad\qquad$ $C_2 : p \leftarrow q, B$

$C_3 : p \leftarrow A, B$

- Σ_1 generalizes Σ_2 if $\Sigma_2 \models a$ implies $\Sigma_1 \models a$
- Denote $\Sigma = \{C_1, C_3\}, A(\Sigma) = \{C_1, C_2\}$.
- $A(\Sigma)$ generalizes Σ in Horn clausal logic

In NMLP, however

- $A(\Sigma)$ does not necessarily generalizes Σ
  $\Sigma = \{p \leftarrow \neg q,\ q \leftarrow r,\ s \leftarrow r,\ s \leftarrow\}$ (V:3,2,2)
  $A(\Sigma) = \{p \leftarrow \neg q,\ q \leftarrow s,\ s \leftarrow r,\ s \leftarrow\}$
  Then, $\Sigma \models p$ but $A(\Sigma) \not\models p$.

- It may be the case that Σ is consistent, but $A(\Sigma)$ is not.
  $\Sigma = \{p \leftarrow q, \neg p,\ q \leftarrow r,\ s \leftarrow r,\ s \leftarrow\}$ (V:3,2,2))
  $A(\Sigma) = \{p \leftarrow q, \neg q,\ q \leftarrow s,\ s \leftarrow r,\ s \leftarrow\}$
  Then, Σ is consistent,but $A(\Sigma)$ is not.

- ...

Inverse Entailment is not directly applicable to NMILP !

Deduction Theorem (Horn clausal logic)

For any formula A, we have

$$P \cup \{R\} \models A \Longleftrightarrow P \models R \rightarrow A$$

Inverse entailment:

Given Horn program B and an example E, deduction theorem gives:

$$\begin{aligned} B \cup \{H\} \models E &\Longleftrightarrow B \models (H \rightarrow E) & (1) \\ &\Longleftrightarrow B \models (\neg E \rightarrow \neg H) & (2) \\ &\Longleftrightarrow B \cup \{\neg E\} \models \neg H & (3) \end{aligned}$$

$B \wedge \neg E \models \neg H$ serves as a necessary condition for constructing H.

In NMLP, however

- Deduction theorem in Eq. (1) and (3) does not hold in general
- Contrapositive implication in Eq. (2) is undefined

Stable Model Based Approaches

Main Results (by Chiaka Sakama)

Deduction Theorem (Horn clausal logic)

For any formula A, we have

$$P \cup \{R\} \models A \Longleftrightarrow P \models R \rightarrow A$$

Entailment Theorem (NMLP)

For any ground literal A, we have

$$P \cup \{R\} \models_S A \Longrightarrow P \models_S R \rightarrow A \qquad \text{(i)}$$

$$P \cup \{R\} \models_S A \Longleftarrow P \models_S R \rightarrow A \text{ and } P \models_S R \qquad \text{(ii)}$$

Contrapositive rule in NMLP

$R : A_0 \leftarrow A_1, \ldots, A_m, not\ A_{m+1}, \ldots, not\ A_n$

$R^c : not\ A_1; \ldots, not\ A_m; not\ not A_{m+1}, \ldots, not\ not\ A_n \leftarrow not\ A_0$

$R^c : \leftarrow A_1, \ldots, A_m, not\ A_{m+1}, \ldots, not\ A_n, not\ A_0$

We can prove that $P \models_S R \Longleftrightarrow P \models_S R_C$ (iii)

Inverse Entailment in NMLP

Given normal program B and a positive example E such that

$$B \models_S\ not\ E \qquad \text{(iv)}$$

Then

$$B \cup \{H\} \models_S E \quad \overset{by\ (i)}{\Longleftrightarrow} \quad B \models_S (H \rightarrow E)$$

$$\overset{by\ (iii)}{\Longleftrightarrow} \quad B \models_S (not\ E \rightarrow\ not\ H)$$

$$consider\ H = p(x_1, \ldots, x_k) \quad where \quad p\ is\ a\ new\ atom$$

$$\overset{by\ (ii)\ and\ (iv)}{\Longrightarrow} \quad B \cup \{not\ E\} \models_S\ not\ H$$

So $B \cup \{not\ E\} \models_S\ not\ H$ serves as a necessary condition for H.

This necessary condition can be simplified as $B \models_S\ not\ H$.

Learning from a single positive example

Classical Inverse Entailment(IE):

- necessary condition for H: $B \wedge \neg E \models \neg H$ (*)
- let Bot be the conjunction of ground literals which are true in every model of $B \wedge \neg E$.
- we **consider** $Bot \models \neg H$ (but note: this IE is not complete since condition (*) does not imply $Bot \models \neg H$).
- $H_0 = \neg Bot$ is a trivial valid (ground) candidate of H.
- organize H_0 s.t. target predicate atom A is left to "$\leftarrow$".
- generalizing H_0 by replacing constants with variables, we get a most specific hypothesis with variables.

NMLP Inverse Entailment(NMLP_IE):

- necessary condition for H: $B \models_S not\ H$ (**)
  (same as $B \cup \{not\ E\} \models_S not\ H$)
- let $M^+ = M \cup \{not\ l \mid l \notin M\ and\ l \in \mathcal{HB}\}$, where M is the stable model of B and $\mathcal{HB}$ is the Herbrand model of B.
- condition (**) **implies** $M^+ \models not\ H$.
- let $\Gamma = \{K \in M^+ \mid K\ is\ relevant\ to\ L\ and\ isinvolved\ in B \cup \{E\}\}$.
- since $M^+ \models \Gamma$, we have $M^+ \models not\ r_0$ where $r_0 = \leftarrow \Gamma$.
- integrity constraint $r_0 = \leftarrow \Gamma$ is a trivial valid (ground) candidate of H.
- shift the target predicate atom to the left of "$\leftarrow$" in r_0.
- generalizing r_0 by replacing constants with variables, we get a most specific hypothesis with variables.

Illustration

Given:

$\mathcal{B} = \{bird(X) \leftarrow penguin(X).\ \ bird(tweety).\ \ penguin(polly).\}$

$E = \{\oplus flies(tweety).\}$

Note: $B \models\ not\ flies(tweety)$.

Steps:

stable model of B (same as that of $B\ not\ E$):
$M = \{\ bird(tweety).\ \ bird(polly).\ \ penguin(polly).\ \}$

expansion of M:
$M^+ = \{\ bird(tweety).\ bird(polly).\ penguin(polly).$
$not\ penguin(tweety).\ not\ flies(tweety).\ not\ flies(polly).\ \}$

integrity constraint:
$r_0 = \leftarrow\ bird(tweety), not\ penguin(tweety),\ not\ flies(tweety).$

shift the atom with target predicate to the left side:
$r_0 =\ flies(tweety) \leftarrow\ bird(tweety), not\ penguin(tweety).$

generalize r_0 by replacing constant with variables:
$H = flies(X) \leftarrow\ bird(X), not\ penguin(X).$