Designing of Nonmonotonic Inductive Logic Programming Systems

Chongbing Liu

October 24, 2005

Outlines

- Basic Algorithms and Properties

- Sequential Learning Algorithm

- Parallelization

Basic Algorithms and Properties

Necessary conditions

Learn from a single positive example

Learn from a single negative example

Learn from a set of examples

General properties

Necessary Conditions

Given

B: a program, H: a rule, E: a ground literal.

Proposition

$B \cup \{H\} \models E \ and \ B \models H \Longrightarrow B \models E \quad (i)$

From (i), we can prove

$B \not\models E \ and \ B \cup \{H\} \models E \Longrightarrow B \not\models H \quad (ii)$ (E: positive example)

$B \models E \ and \ B \cup \{H\} \not\models E \Longrightarrow B \not\models H \quad (iii)$ (E: negative example)

Trivial hypothese

- 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.
- $B \not\models H$ implies $M^+ \not\models H$.
- let $\Gamma = \{K \in M^+ \mid K\ is\ relevant\ to\ E\ and\ isinvolved\ inB \cup \{E\}\}$.
- since $M^+ \models \Gamma$, we have $M^+ \not\models r_0$ where $r_0 = \leftarrow \Gamma$.
- integrity constraint $r_0 = \leftarrow \Gamma$ is a trivial valid (ground) candidate of H.

Learning from a single positive example

algorithm: Learn-single-pos

Input: a categorical program B, a ground atom E (positive)

Output: a rule R

1. compute the answer set M of B and its expansion set M^+;
2. construct the integrity constraint $\leftarrow \Gamma$ from M^+;
3. produce a rule $E \leftarrow \Gamma'$ by shifting $not\ E$ in Γ;
4. generate a general rule R where $R\theta = (L \leftarrow \Gamma')$ for some $theta$.

Learning from a single positive example

Illustration

Given:

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

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

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

Steps:

1. compute answer set M of B and expansion set M^+:
   - stable model of B (same as that of B $not\;E$):
     $M = \{\; bird(tweety).\;\; bird(polly).\;\; penguin(polly).\;\}$
   - expansion of M:
     $M^+ = \{\; bird(tweety).\; bird(polly).\; penguin(polly).$
     $not\; penguin(tweety).\; not\;\; flies(tweety).\; not\;\; flies(polly).\;\}$
2. construct the integrity constraint Γ from M^+:
   $\leftarrow\;\; bird(tweety), not\; penguin(tweety),\;\; not\;\; flies(tweety).$
3. produce a rule $E \leftarrow \Gamma'$ by shifting $not\;\; E$ in Γ:
   $r_0 =\;\; flies(tweety) \leftarrow\;\; bird(tweety), not\; penguin(tweety).$
4. generate a general rule :
   $H = flies(X) \leftarrow\;\; bird(X), not\; penguin(X).$
   (simplified as $ab(x) \leftarrow\;\; penguin(x)$.

Learning from a single positive example

Properties

B: categorical program,

E: positive example,

R: learned rule by algorithm Learnsingle-pos

Properties:

- $B \not\models R$.
- $pred(head(R)) = pred(E)$.
- if R is negative-cycle-free and its head predicate appears nowhere in B, then $B \cup \{R\}$ is also categorical.
- if R is negative-cycle-free and its head predicate appears nowhere in B, then $B \cup \{R\} \models E$.

Learning from a single negative example

algorithm: Learn-single-neg

Input:
a categorical program B, a ground atom E (negative example), a target predicate $K(\ldots)$ on which $pred(E)$ strongly and negatively depends in B.

Output: a rule R

1. compute the answer set M of B and its expansion set M^+;
2. construct the integrity constraint $\leftarrow \Gamma$ from M^+;
3. produce the rule $K(\ldots) \leftarrow \Gamma'$ by shifting $not\ K(\ldots)$ in Γ;
4. obtain Γ'' by dropping from Γ' every literal l whose predicate $pred(l)$ strongly and netagively depends on $K(\ldots)$ in B.
5. generate a general rule R from $K(\ldots) \leftarrow \Gamma''$ such that $R\theta = K(\ldots) \leftarrow \Gamma''$ for some θ.

Learning from a single negative example

Illustration

Given:

$$\begin{array}{ll} B: & flies(x) \leftarrow bird(x), not\ ab(x), \\ & bird(x) \leftarrow penguin(x), \\ & bird(tweety), \\ & penguin(polly). \\ E: & \ominus flies(polly). \end{array}$$

target predicate: **ab**

Note: $B \models flies(polly)$.

Steps:

1. compute answer set M of B and expansion set M^+:
   ... omitted ...

2. construct the integrity constraint $\leftarrow \Gamma$ from M^+:
   $\leftarrow\ bird(polly),\ penguin(polly),\ flies(polly),\ not\ ab(polly)$.

3. produce the rule $K(\ldots) \leftarrow \Gamma'$ by shifting $not\ K(\ldots)$ in Γ:
   $ab(polly) \leftarrow\ bird(polly),\ penguin(polly),\ flies(polly)$.

4. dropping from Γ' every literal l whose predicate $pred(l)$ strongly and netagively depends on predicate ab:
   $ab(polly) \leftarrow\ bird(polly),\ penguin(polly)$.

5. generate a general rule H:
   $ab(x) \leftarrow\ bird(x),\ penguin(x)$.
   (simplified as $ab(x) \leftarrow penguin(x)$.

Note: Now since $ab(polly)$ is $true$, $not\ ab(penguin)$ is $false$. Therefore, the newly learned theory prevents the first rule in B from deriving $flies(polly)$.

Learning from a single negative example

Properties

B: categorical program,

E: negative example,

K: target predicate,

R: learned rule by algorithm Learnsingle-pos

Properties:

- $B \not\models R$.
- $pred(head(R)) \neq pred(E)$, instead, $pred(head(R)) = K$.
- if R is negative-cycle-free, then $B \cup \{R\}$ is not necessarily categorical.
- if $B \cup \{R\theta\} \models R$ and $B \cup \{R\}$ is consistent, then $B \cup \{R\} \not\models E$.

Learning from a set of examples

all positive examples

1. Let B be a categorical program, and R_i is a rule learned from B and a positive example E_i, $1 \leq i \leq n$.

   If each R_i is negativ-cycle-free and $pred(E_i)$ appears nowhere in B, then $B \cup \{R_1, \ldots, R_n\} \models E_i$.

2. Let B be a categorical program, E_1 and E_2 be positive examples such that $pred(E_1)$ and $pred(E_2)$ appear nowhere in B.

   Suppose rule R_1 learned from B and E_1 is negative-cycle-free, and rule R_2 learned from $B \cup \{R_1\}$ and E_2 is negative-cycle-free.

   Then $B \cup \{R_1, R_2\} \models E_i (i = 1, 2)$. (monotonicity)

Learning from a set of examples

all negative examples

... ... omitted ... ...

Learning from a set of examples

mixed set of positive and negative examples

1. may not necessarily produce a solution which satisfies both positive and negative examples.

2. in incremental learning mode, the order in which the examples are taken, does matter. (obvious in multiple-predicate learning, less obvious in single-predicate learning)

General Properties

- Both positive and negative examples may lead to new rules learned.
- Based on answer set semantics, so have both abductive and inductive nature.
- Example-driven learning, therefore bottom-up search in general.
- **(Induction in noncategorical programs)** Suppose program B has anser sets $S_1, \ldots, S_n$, and rule R_i is obtained by algorithm Learn-single-pos using B and a same positiv example E. If each R_i is negative-cycle-free and $pred(E)$ appears nowhere in B, then $B \cup \{R_1, \ldots, R_n\} \models E$.
- No modifications to the rules as background knowledge. But the result of induction often has the same effect as modifying rules in a program, given appropriate program transformation techniques. For instance, let $B = \{p \leftarrow q, r. \quad q.\}$ and $E = p$. Then algorithm Learn-single-pos will learn a rule

  $p \leftarrow q,\ not\ r.$

  However, this rule and the first rule in B can be merged as $p \leftarrow q$, which is equivalent to the rule obtained by dropping r from the first rule in B.
- Since the learned theory may contain a lot of redundencies, it seems that we really need some robust program transformation procedures.
- This feature allows the batch learning systems to incorporate some prior knowledge, which was not allowed in traditional batch learning.
- batch learning is preferred to incremental learning, since it leads to less redundant theories.

Sequential Learning Algorithms

Incremental Learning

Initialize Σ to $\{\Box\}$ or some prior knowledge
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. $\Sigma \models e^-$
      learn a rule from Σ and e^- using Learn-single-neg
        add the learned rule to Σ
    if $\exists e^+$ s.t. $\Sigma \not\models e^+$
      learn a rule from Σ and e^+ using Learn-single-pos
        add the learned rule to Σ
  simplify Σ
until no examples left to read.

Sequential Learning Algorithms

Batch Learning

Initialize Σ to $\{\Box\}$ or some prior knowledge

    while there are positive examples uncovered by Σ

        learn a rule R from Σ and a randomly selected e^+

        find a best consistent rule R' between $\Box$ and R

            using algorithm Learn-single-pos

        remove positive examples covered by R'

        add the rule R' to Σ

        simplify Σ

    while there are negative examples

        learn a rule R from Σ and a randomly selected e^-

            using algorithm Learn-single-neg

        remove e^-

        add the rule R to Σ

        simplify Σ

Parallel Learning Algorithms

master processor :

Initialize Σ to $\{\Box\}$ or some prior knowledge

partition the positive examples to p processors

replicate all negative examples to all processors

broadcast Σ to all the worker processors

collect learned Σ_i from processor i

merge Σ_i's and simplify them into a new Σ

worker processor i :

receive its partition of positive examples

and all the negative examples

learn a theory Σ_i sequentially

send Σ_i to the master