ILP Systems: A Review

By Chongbing Liu

Outlines

- View of ILP as a search problem
  - search space
  - search strategies
  - search heuristics
- View of ILP as the inverse of deduction
  - inverse entailment
  - PROGOL
- Parallelize ILP

ILP Systems: A Review

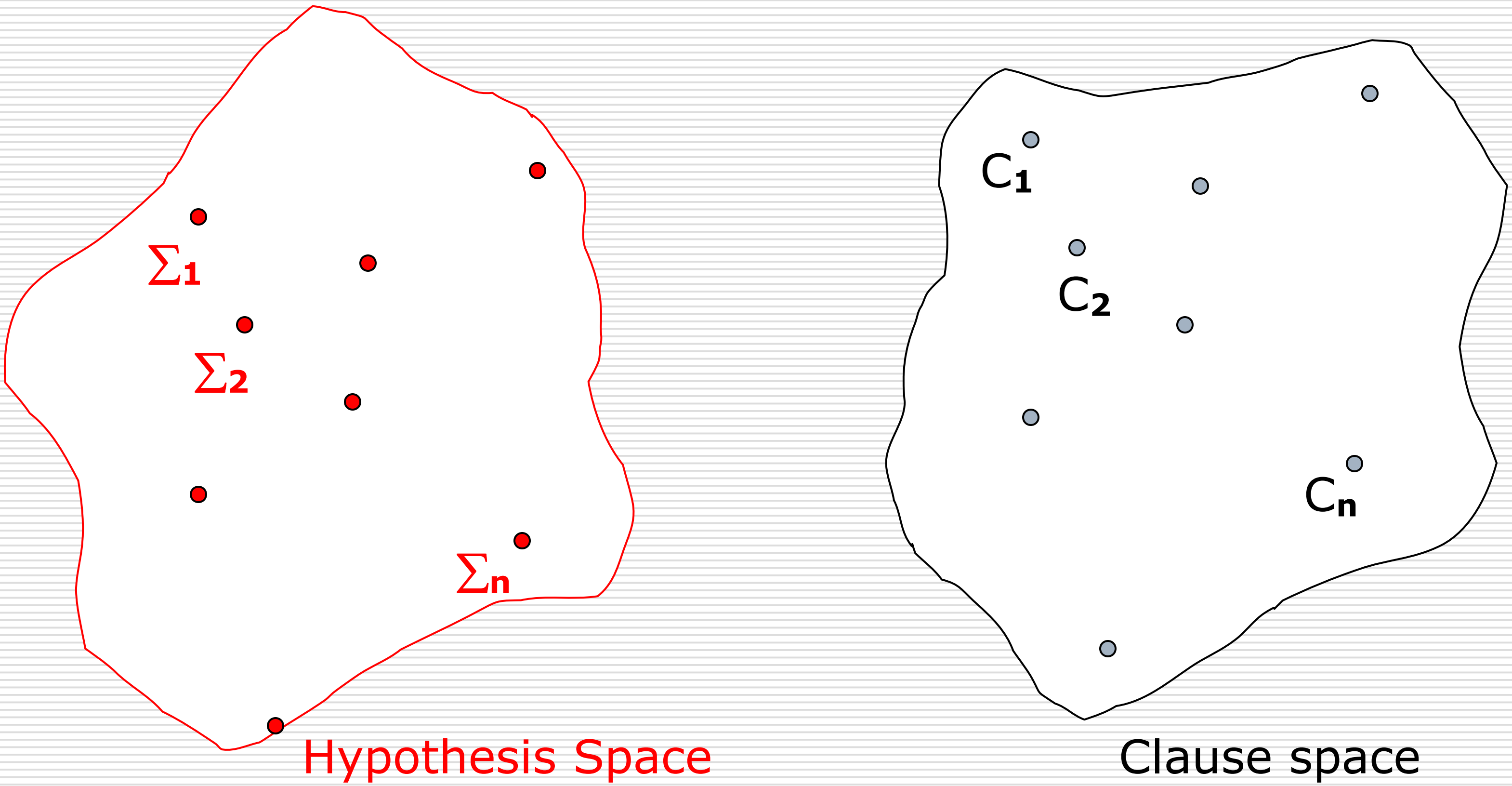
By Chongbing Liu

Outlines

- View of ILP as a search problem
  - search space
  - search strategies
  - search heuristics
- View of ILP as the inverse of deduction
  - Inverse entailment
  - PROGOL
- Parallelize ILP

ILP as a search problem

Search space



ILP as a search problem

Search space

- Definitions
- Structures

ILP as a search problem

Search space (definitions)

Syntactic bias:

Definite clauses, non-recursive clause,

function-free clause, linked clauses,

variable-depth bounded clause, and so on

ILP as a search problem

Search space (definitions)

Semantic bias:

clauses satisfying some mode declarations,

Clauses with given degree of determinacy w.r.t **B**

determinate clauses

ij-determinate clauses

ILP as a search problem

Search space (structures)

Importance of structures (orders):

allows to dynamically generate only part of the space

support pruning the search space

ILP as a search problem

Search space (structures)

Often discussed orders:

subsumption order $C \propto D$ if $C\theta \subseteq D$ for some θ

implication order $C \models D$ if C implies D

relative subsumption order $C \propto_{\mathbf{B}} D$ if $\mathbf{B} \models \forall(C\theta \subseteq D)$ for some θ

relative implication order $C \models_{\mathbf{B}} D$ if $(\mathbf{B} \cup \{C\}) \models D$

generalized subsumption order

$C \geqslant_{\mathbf{B}} D$ if with **B,** C can be used to prove at least as many results as D

ILP as a search problem

Search spaces in ILP systems

system	declarative bias	generality order
MIS	definite clauses	subsumption
FOIL	function-free normal clauses (allowing negative literals in the body) with the target predicate symbol as the head	subsumption
GOLEM	definite clauses having the target predicate symbol as the head, also with ij-determinacy and model constraints	relative subsumption
PROGOL	definite clauses restricted by bounded maximal variable depths, maximal resolution steps and mode declarations	subsumption

ILP Systems: A Review

By Chongbing Liu

Outlines

- View of ILP as a search problem
  - search space
  - search strategies
  - search heuristics
- View of ILP as the inverse of deduction
  - inverse entailment
  - PROGOL
- Parallelize ILP

ILP as a search problem

Search strategies

- Incremental vs. batch learning
- Top-down vs. bottom-up search

ILP as a search problem

Search strategies

- Incremental vs. batch learning

ILP as a search problem

Outline of the *Incremental Learning* Strategy

Initialize Σ to $\{\square\}$

repeat while there are examples available

  read the next (positive or negative) example

  repeat

    if Σ is too strong

      specialized Σ

    if Σ is too weak

      generalize Σ

  until Σ is correct w.r.t. the examples read so far

Search Hypothesis Space

ILP as a search problem

Outline of the *Batch Learning* Strategy

Initialize Σ to $\{\Box\}$

Initialize E_{cur} to E

repeat

find a clause C which covers the most positive example and no negative examples in E_{cur}

update Σ by adding clause C

update E_{cur} by removing positive examples covered by C

until E_{cur} contains no positive examples

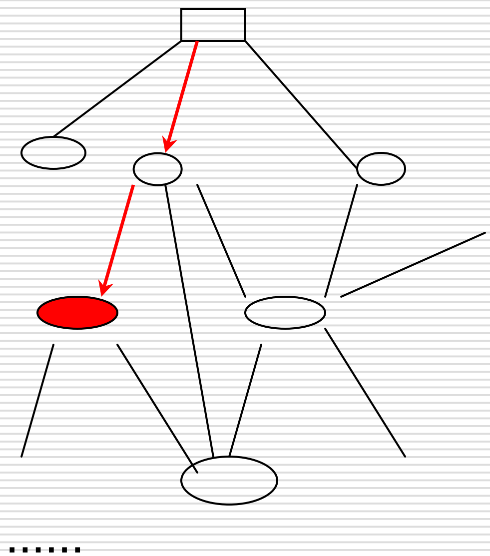
Search Clause Space

ILP as a search problem

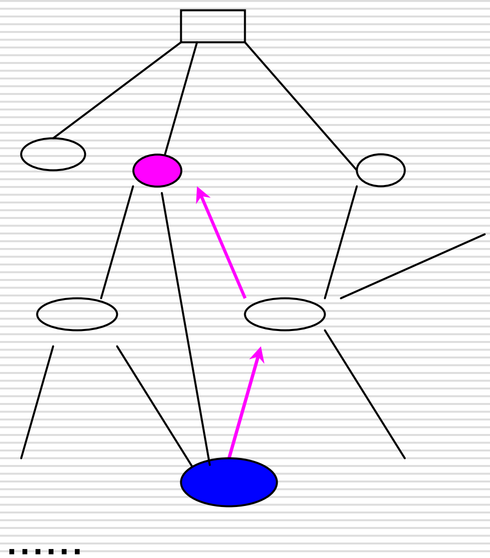
Search strategies

- Incremental vs. batch learning
- Top-down vs. bottom-up search

ILP as a search problem



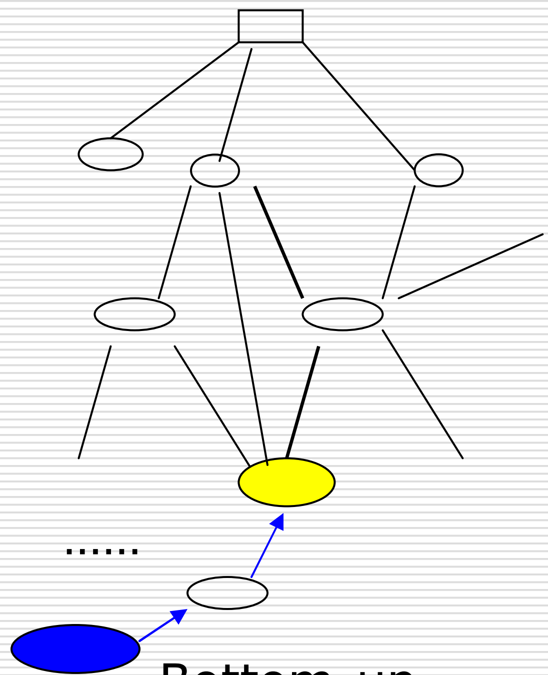
Top-down



Bottom-up

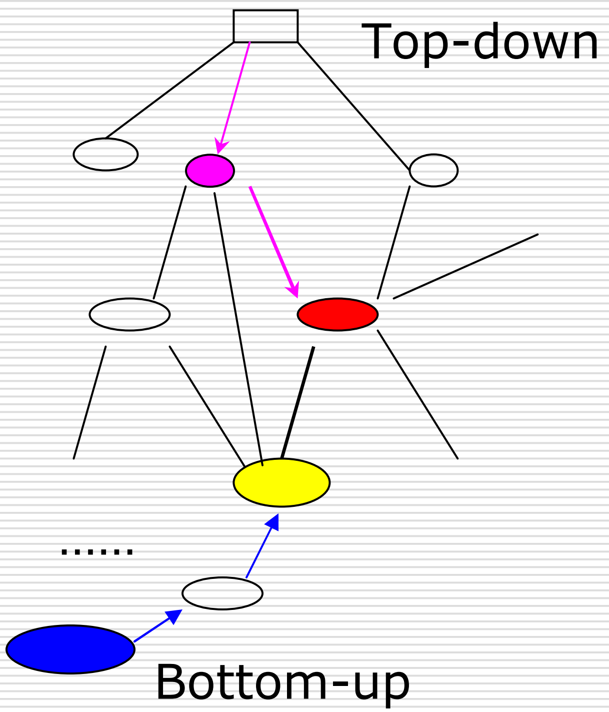
ILP as a search problem

Hybrid search:



ILP as a search problem

Hybrid search:



ILP as a search problem

Search strategies in ILP systems

system	learning mode	search direction	search method
MIS	incremental	mixtured	brute-force search
FOIL	batch	top-down	hill-climing search
GOLEM	batch	bottom-up	?
PROGOL	batch	bottom-up/top-down	A*-like search

ILP Systems: A Review

By Chongbing Liu

Outlines

- View of ILP as a search problem
  - search space
  - search strategies
  - search heuristics
- View of ILP as the inverse of deduction
  - Inverse entailment
  - PROGOL
- Parallelize ILP

ILP as a search problem

Search heuristics

- Any quantities used to guide the search or terminate the search
- Reflect the status of a reached state
- Statistic heuristics

ILP as a search problem

Search heuristics

- posteriori probability of *h* given E: $$P(h|E) = \frac{P(E|h)P(h)}{P(E)}$$
  maximize P(h|E) -→ maximum a posteriori (MAP) hypothesis

$$\begin{aligned} h_{MAP} &\equiv Max_{h\in H}\ P(h|E) \\ &= Max_{h\in H}\ \frac{P(E|h)P(h)}{P(E)} \\ &= Max_{h\in H}\ P(E|h)P(h)\ \text{(since P(E) is constant)} \end{aligned}$$

- likelihood of E given *h :* *P(E|h)*
  if P(h) is constant, then get maximum likelihood (ML) hypothesis

$$h_{ML} \equiv Max_{h\in H}\ P(E|h)$$

ILP as a search problem

Search heuristics

- Transform H_{MAP}, we get :

$$h_{MAP} = Min_{h \in H} \; -log_2 P(E|h) \; -log_2 P(h)$$

H_{MAP} turns to be minimal description length hypothesis (MDL)

$$H_{MDL} = Min_h \; L_{C_H}(h) + L_{C_{E|h}}(E|h)$$

Trade-off !

- Description length of E given *h* : $-log_2 P(E|h)$

if P(h) is constant, then get minimal data description length hypothesis

$$H_{MDL} = Min_h L_{C_{E|h}}(E|h)$$

ILP as a search problem

Search heuristics

Probability approximation:

$$P(E|h) \approx A(h) = P(\oplus \mid h)$$

ILP as a search problem

Search heuristics in ILP systems

system	heuristics	stopping criterion 1	stopping criterion 2
MIS	N/A	$P(E \mid h) = 1$	$P(E \mid h) = 1$
FOIL	$I(C) - I(C')$	$L_{C_H}(h) \geq L_{C_{E\|h}}(E \mid h)$	no e^+ or no more bits available
GOLEM	?	may cover some $\ominus$	?
PROGOL	f=$p - (n + c + h)$	f is minimal	all $\oplus$ covered

ILP Systems: A Review

By Chongbing Liu

Outlines

- View of ILP as a search problem
  - search space
  - search strategies
  - search heuristics
- View of ILP as the inverse of deduction
  - inverse entailment
  - PROGOL
- Parallelize ILP

ILP as the Inverse of Deduction

Inverse entailment

Inverse resolution is not complete. For example,

```
D = f(I,J)← d(I,K),d(K,L),f(L,M),m(K,M,N),m(I,N,J)
C = f(K,N)←        d(K,L),f(L,M),m(K,M,N)
```

C implies D, but C not subsumes D. That is, we can not obtain C from D by inverse resolution.

So we need to try inverting implication. This is called inverse entailment. While doing this, we make use of sub-saturants of D.

ILP as the Inverse of Deduction

Inverse entailment

Sub-saturants: (simplified)

S(D) includes D itself and clauses obtained by replacing the variables in the head with all other variables in the clause.

For function-free clauses, |S(D)| is at most n^k where k is the arity of the head and n is the number of variables in the clause. If not function-free, we need to flatten clause D.

Theorem:

If $C \models D$, then exists A in sub-saturants(D) such that C subsumes A.

ILP as the Inverse of Deduction

Inverse entailment

To compute C from D s.t. C $\models$D, (D is function-free):

1. compute sub-saturants of D, getting S(D)
2. **C**= { }
3. for each s$\in$S(D)

   add all the clauses which subsumes s into **C**
4. Remove $f \in$ **C** for which $f \models$ D is not true

Note: a) step 3 and 4 are decidable since D is function-free.

b) **C** is complete in tat it contains all C which imply D.

c) **C** supersets **C'**={g|g subsumes D}

ILP as the Inverse of Deduction

PROGOL

1. First, for each single positive example e, PROGOL constructs a most specific clause which together with the background knowledge implies e.

$$B \wedge H \models e$$

$$B \wedge \overline{e} \models \overline{H}$$

Let $\overline{\bot}$ be the conjunction of ground literals which are true in all models of $B \wedge \overline{e}$, i.e.,

$$B \wedge \overline{e} \models \overline{\bot}$$

then $$B \wedge e \models \overline{\bot} \models \overline{H}$$

and thus $$H \models \bot$$ ($\bot$ is the most specific clause)

($\bot$ is obtained from $\overline{\bot}$ by replacing terms by unique variable)

ILP as the Inverse of Deduction

PROGOL

2. Second, PROGOL searches for a most general and consistent clause H, which covers the most of other positive examples and no negative examples. Ideally we should search through all the complete set of candidates **C** (computed using inverse entailment technique). But for the sake of simplicity and efficiency, PROGOL only searches $\mathbf{C}'$ where each element subsumes $\perp$. The search is performed top-down.

ILP Systems: A Review

By Chongbing Liu

Outlines

- View of ILP as a search problem
    - search space
    - search strategies
    - search heuristics
- View of ILP as the inverse of deduction
    - inverse entailment
    - PROGOL
- Parallelize ILP

Parallelize ILP

There already exists an implementation of parallel ILP. But

- it is for the non-monotonic problem setting, i.e., for data mining,
- and it is based on Bulk Synchronous Parallelism(BSP) model.

Parallelize ILP

Data Partition

For the non-monotonic setting, there is usually very little background knowledge and negative example. So it makes sense to simply duplicate them to all the processors and only partition the huge set of examples, as that implementation does. In normal problem setting, however, the dominating part of the data is usually the background knowledge (ground literals) instead of examples. Therefore partition should be done on background knowledge as well other than examples, in order to achieve better parallelization. Also ideally a processor should receive background knowledge which is right about the examples it receives. The question is: how to partition background knowledge and examples in coordination?

Thank you.