

Il Ragionamento Logico come Forma di Apprendimento

Sviluppo di Un Framework per ILP

Giovanni Maria Speciale

Relatore - Prof. Andrea Omicini

Correlatori - Prof.ssa Roberta Calegari,

Dott. Giovanni Ciatto

Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

Dipartimento di Informatica – Scienza e Ingegneria (DISI)

ALMA MATER STUDIORUM – Università di Bologna - Campus di Cesena

Tesi di Laurea Magistrale, Luglio 2021

Intelligenza Artificiale

- sistemi con intelligenza trasparente
- ragionamento automatizzato
- capacità di commonsense reasoning

Tecniche

- Machine Learning
- Inductive Logic Programming

Definizione: Inductive Logic Programming

- è un paradigma di programmazione
- concettualizza il ragionamento comune
- adotta un processo d'inferenza dal particolare al generale

Differenze

ILP:

- inferenza logica
- procedura ispezionabile
- poco tollerante a input ambiguo o non elaborato

ML:

- inferenza statistica
- procedura implicita
- tolleranza a dati grezzi e non elaborati

1° Sistematizzazione dello stato dell'arte

- chiarezza
- intelligibilità

2° Realizzazione di un Modulo ILP

- usabilità
- eterogeneità

Aspetti Fondamentali

- Impostazione dell'apprendimento
- Linguaggio di rappresentazione
- Bias linguistico
- Metodo di ricerca:
 - Top-Down
 - Bottom-Up
 - Meta-Level
 - Conflict-Driven

Tecniche d'Induzione

RLGG: *Relative-Least General Generalization*

IE: *Inverse-Entailment*

MIL: *Meta-Interpretative Learning*

BCP: *Bottom-Clause Propositionalization*

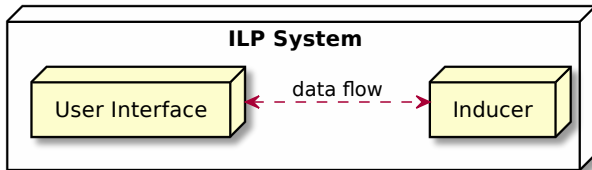
Principali Algoritmi d'Induzione:

- Golem → RLGG → Bottom-Up
- Progol → IE → Bottom-Up/Top-Down
- Metagol → MIL → Meta-Level
- ILASP → MIL → Conflict-Driven
- CILP++ → BCP → Artificial Neural Network

Pattern Architetturale

Nel pattern software architetturale sono preponderanti:

- il comportamento, o parte algoritmica del sistema
- l'interazione o user interface (UI)



I *Inducer*

- `golem(): Inducer`
- `progol(): Inducer`
- `metagol(): Inducer`

- `induce(): Theory`

Metodo

- `induce()`

Metodi Factory

- `golem()`
- `progol()`
- `metagol()`

Design - Passo d'Induzione I

Esempi Positivi

- `son(john, joseph).`
- `son(dario, mary).`
- `son(joseph, rose).`

Esempi Negativi

- `son(dario, joseph).`
- `son(john, mary).`

Background Knowledge

- | | |
|--------------------------------------|------------------------------|
| • <code>parent(joseph, john).</code> | • <code>male(joseph).</code> |
| • <code>parent(mary, dario).</code> | • <code>male(dario).</code> |
| • <code>parent(rose, joseph).</code> | • <code>female(mary).</code> |
| • <code>male(john).</code> | • <code>female(rose).</code> |

Risultato Atteso

`son(X,Y):- male(X),parent(Y,X).`

Golem

- non utilizza nessun bias sintattico
- seleziona casualmente due esempi positivi e ne applica la RLGG

Progol

sfrutta le dichiarazioni di modalità come bias sintattico:

- `modeh(1,son(-person,+person)).`
- `modeb(1,parent(+person,-person)).`
- `modeb(1,male(+person)).`

Metagol

sfrutta la meta-rule come bias sintattico:

- `metarule(preCondInverse,[X,Y,Z],[X,A,B],[Y,A],[Z,B,A]).`

Caratteristiche

- Leggibilità
- F.I.R.S.T

Orientamento: Family Relations

- son/daughter
- grandparent
- ancestor
- idempotence

Coverage				
Class Name	Class %	Method %	Line %	Branch %
GolemInducer	100%	94,4%	97,2%	100%
ProgolInducer	100%	94,1%	96,3%	100%
MetagolInducer	100%	85,9%	94,2%	100%

La CLI è dotata di tre sottocomandi per permettere all'utente di scegliere l'algoritmo e i parametri per avviare l'induzione.

Running...

① run ad algoritmo singolo:

▷ Golem:

```
ilp golem -s SAMPLING_FACTOR -p PATH -n PATH -b PATH
```

▷ Progol:

```
ilp progol -p PATH -n PATH -d PATH -t PATH -b PATH
```

▷ Metagol:

```
ilp metagol -s DEPTH_INDUCTION -p PATH -n PATH -m PATH  
-b PATH
```

② run ad algoritmi multipli: concatenando più run sopra descritti.

Punti Fermi di ILP

- apprendimento facilmente ispezionabile
- soluzione human-readable
- eterogeneità nella manipolazione della conoscenza

Contributo

- ricognizione dello stato dell'arte
- implementazione fruibile in contesti pervasivi
- integrazione Logic Programming - Functional Programming
- supporto multi-piattaforma

Sezioni Architetture

- estensione del framework ad altri algoritmi d'induzione
- affiancamento di un sistema sub-simbolico

Sezioni Implementative

- estensione degli algoritmi verso domini diversi
- incremento delle percentuali di coverage
- miglioramento del processo di testing
- rinforzo della UI