Relational Knowledge Extraction from Neural Networks

Manoel V. M. França
manoel.franca@city.ac.uk

Artur S. d'Avila Garcez
a.garcez@city.ac.uk

Dept. of Computer Science
City University London
United Kingdom

Gerson Zaverucha
gerson@cos.ufrj.br

PESC/COPPE
Univ. Fed. do Rio de Janeiro
Brazil

CoCo@NIPS15 – 12 December 2015

Contents

Representation precedes Learning

We need a language for describing the alternative algorithms that a network of neurons may be implementing…

Computer Science Logic + Neural Computation

GOAL of NSI: Learning from experience and reasoning about what has been learned in a computationally efficient way

Les Valiant

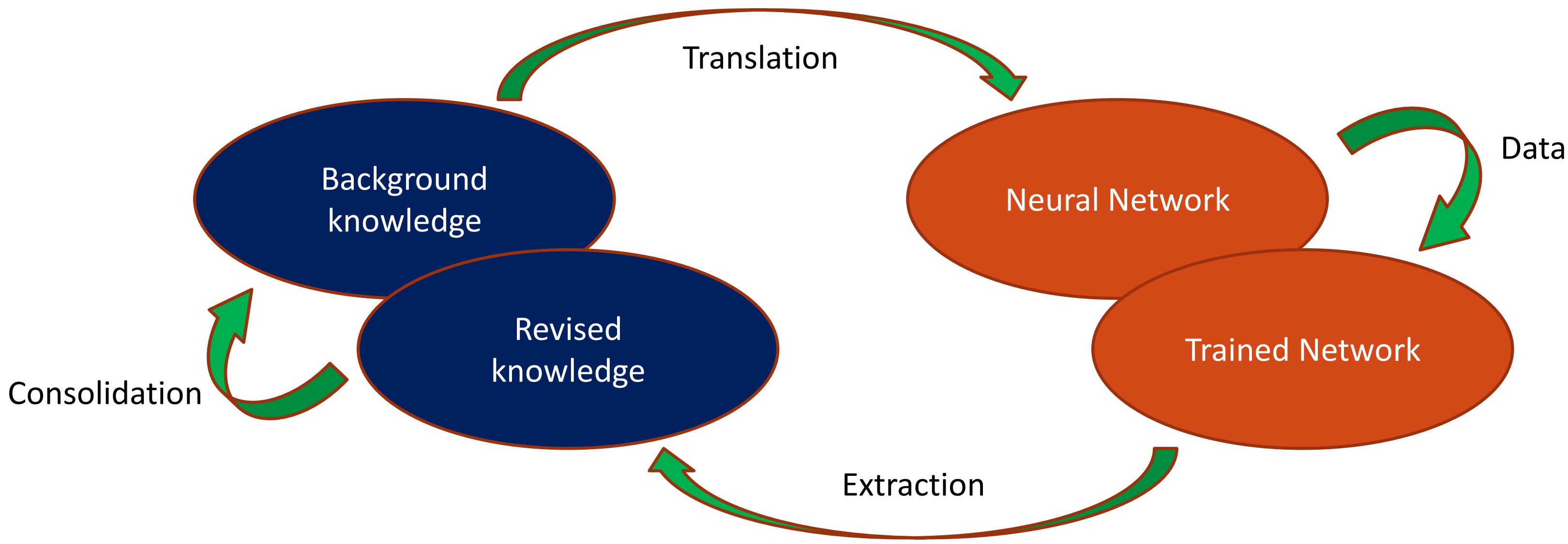
Motivation

- Relational learning using simple neural nets that can be trained efficiently e.g. using backpropagation
- Why relational learning? Learning a first-order logic theory from (real-valued) data: R(x,y) implies C(x) or ¬C(y)
- Either by searching for candidate hypotheses at first-order logic level or through propositionalization
- Propositionalization enables the use of any state-of-the-art attribute-value learner
- But results in a loss of the neat first-order knowledge representation which breaks down reasoning
- We seek to reconcile efficient propositionalization, learning with backpropagation, and first-order logic reasoning

Efficient Relational Learning using CILP++

- We have extended the CILP neural-symbolic system to solve ILP problems (França et. al., Mach. Learn. 94(1):81-104, Jan 2014):
  - background knowledge provided in the form of first-order logic clauses is inserted into a neural net which can be trained by backpropagation;
  - a revised first-order logic knowledge-base is extracted from the trained network using a variation of the TREPAN rule extraction algorithm.
- In this paper, we investigate empirically the relational models extracted from CILP++
- CILP++ performs efficient relational learning through:
  - Bottom Clause Propositionalization (BCP)
  - Neural network training with backpropagation
  - Relational knowledge extraction using TREPAN (Craven and Shavlik, NIPS-8, 1995)

Neural-Symbolic Integration



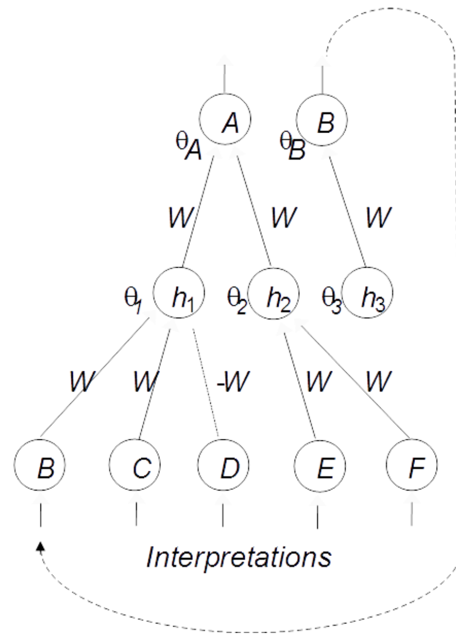
CILP++ system (Connectionist Inductive Logic Programming): download it from http://sourceforge.net/projects/cilppp/

Connectionist ILP

r_1: $A \leftarrow B, C, \sim D;$

r_2 : $A \leftarrow E, F;$

r_3 : $B \leftarrow$

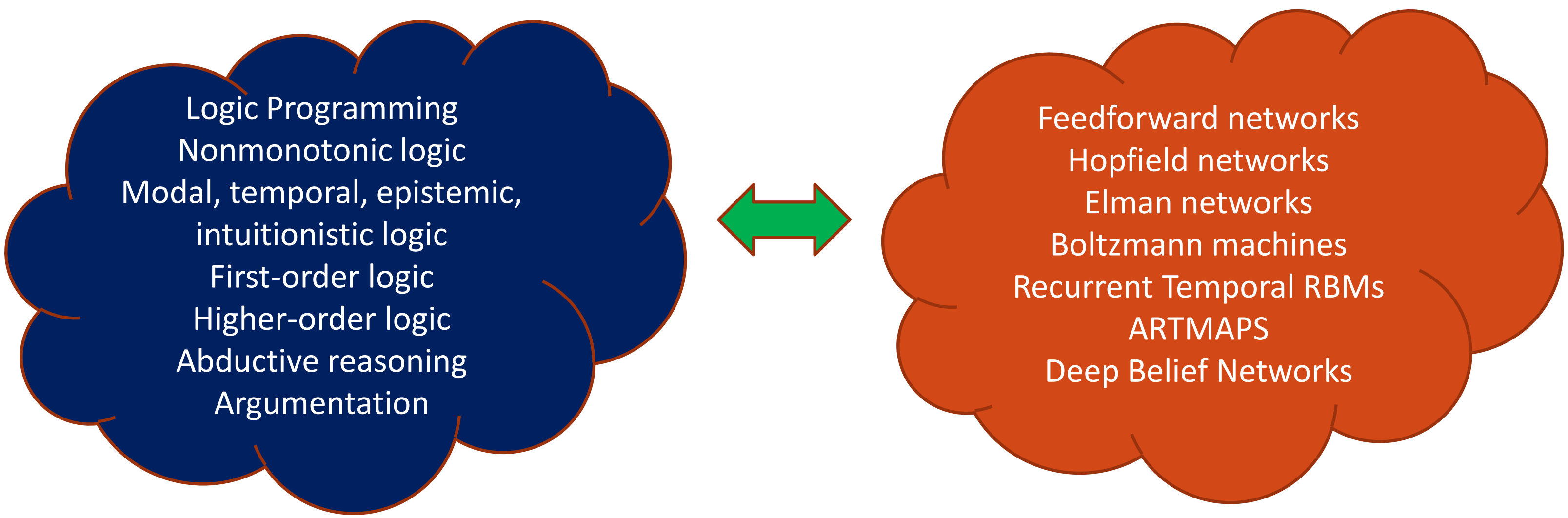


THEOREM 1: For any logic program P there exists a neural network N such that N computes P

(due to Hoelldobler and Kalinke and extended by Garcez and Zaverucha to allow use with Backpropagation)

Neural-Symbolic Systems

Neural-symbolic methodology: translation algorithms to and from symbolic and connectionist models



In search of robustness and explanations; “combining the logical nature of reasoning and the statistical nature of learning”, L. Valiant

A small example: Family Relationship

BCP: generates a most specific (bottom) clause for each (positive or negative) example; converts each bottom clause into a vector

Background Knowledge:

mother(mom1, daughter1)

wife(daughter1, husband1)

wife(daughter2, husband2)

Positive Examples:

motherInLaw(mom1, husband1)

Negative Examples:

motherInLaw(daughter1, husband2)

Using the Progol bottom clause algorithm on each example (Muggleton, New Generation Computing, 1995):

$\perp_{+} = [\mathit{motherInLaw}(A, B) \leftarrow \mathit{mother}(A, C), \mathit{wife}(C, B)]$ is generated

$\perp_{-} = [\mathit{motherInLaw}(A, B) \leftarrow \mathit{wife}(A, C)]$ is generated

Unifications made during bottom clause generation are stored into *hash* tables

BCP example (cont.)

Background Knowledge:

mother(mom1, daughter1)
wife(daughter1, husband1)
wife(daughter2, husband2)

Positive Examples:

motherInLaw(mom1, husband1)

Negative Examples:

motherInLaw(daughter1, husband2)

$\bot_+ = [\mathit{motherInLaw}(A, B) \leftarrow \mathit{mother}(A, C), \mathit{wife}(C, B)]$

$\bot_- = [\mathit{motherInLaw}(A, B) \leftarrow \mathit{wife}(A, C)]$

- From $\bot_+$, hash_+:

Key	Value
mom1	A
husband1	B
daughter1	C

- From $\bot_-$, hash_-:

Key	Value
daughter1	A
husband2	B
husband1	C

- F = {mother(A, C), wife(C, B), wife(A, C)}
- $v_+ = (1,1,0)$ $V_- = (0,0,1)$

Shortcomings of BCP, first version

- Extracted first-order clauses are likely not to follow proper variable chaining
  - Each first-order literal from the bottom clauses is treated as a feature, potentially causing considerable information loss
- The hash tables can get very large and contain redundant features
  - All distinct literals from each bottom clause that is generated are stored in the hash tables
- Feature selection (mRMR, IEEE PAMI 27(8), 2005) can make the information loss worse
  - Literals that are essential for preserving a bottom clause structure might be filtered out

BCP with semi-propositionalization

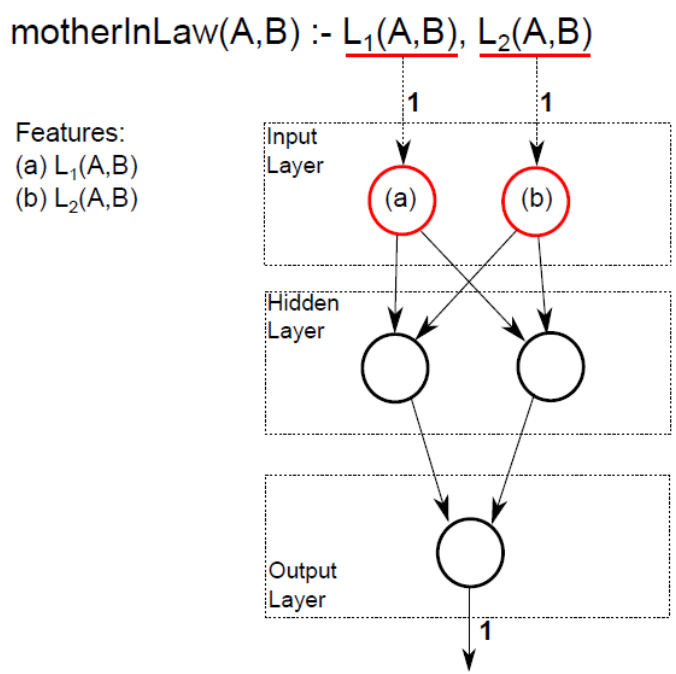
- The LINUS ILP system was extended to use the concept of semi-propositionalization (Lavrac and Flach, 2001)
- Semi-propositionalized rules L_i won't break up sets of literals that share local variables (e.g. C below)
- Semi-propositionalization creates better first-order features for CILP++

Bottom clause:

$$motherInLaw(A,B) :- parent(A,C), wife(C,B),$$
$$wife(A,D), brother(B,D)$$

First-order features:

$$\boldsymbol{L_1(A,B) :- parent(A,C), wife(C,B)}$$
$$\boldsymbol{L_2(A,B) :- wife(A,D), brother(B,D)}$$



Knowledge Extraction from CILP++

- BCP with semi-propositionalization permits relational knowledge extraction from CILP++
  - Semi-propositionalization generates independent features, in a relational sense
  - There's no information loss caused by rupture of linkages of bottom clauses
- We use TREPAN to extract a decision tree, having features such as $L_1(A,B)$ in its nodes, from a trained CILP++ neural network
- The decision tree is then converted into logical clauses in the usual way

(First-order) TREPAN

1. The CILP++ network is used as an oracle to generate a set of examples by querying;
2. The examples are used to produce tree nodes following a greedy information gain heuristic;
3. Nodes are iteratively re-evaluated after each node insertion for relevance checking;
4. The generated decision tree is converted into a set of disjunctive rules;
5. Feature descriptions from BCP are used to convert the features back to first-order literals.

East-West Trains example

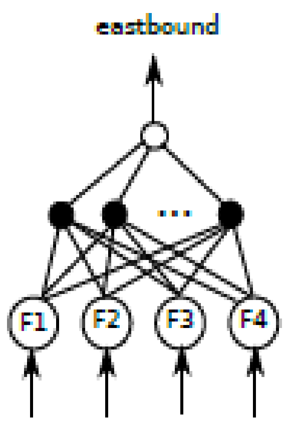
Bottom clauses

eastbound(A) :-
has_car(A,B), long(B), wheels(B,2).
eastbound(A) :-
has_car(A,B), has_car(A,C), long(C), double(B).
eastbound(A) :-
has_car(A,B), has_car(A,C), long(B), shape(C,u_shaped).

Features

F1(A) :- has_car(A,B), long(B), wheels(B,2).
F2(A) :- has_car(A,B), double(B).
F3(A) :- has_car(A,B), long(B).
F4(A) :- has_car(A,C), shape(C, u_shaped).

Neural network



Generated theory

eastbound(A) :- F1(A).
eastbound(A) :- F2(A).

Semi-propositionalization clauses

F1(A) :- has_car(A,B), short(B), wheels(B,1).
F2(A) :- has_car(A,B), short(B), closed(B), wheels(B, 2), jagged(B).

Leave-one-out cross validation

90% average rule accuracy on the test set

95% average rule fidelity to the network

Experimental results (network accuracy)

SYSTEM	METRICS	DATASETS			
		Mutagenesis	UW-CSE	Alzheimer-amine	Cora
Aleph	*ACC*	80.85%(±10.51)	85.11%(±7.11)	78.71%(±5.25)	−−
	AUC	0.80(0.02)	0.81(±0.07)	0.79(±0.09)	−−
	RUN	721	875	5465	−−
MLN	*ACC*	62.11%(±0.022)	75.01%(±0.028)	50.01%(±0.002)	70.10%(±0.191)
	AUC	0.67(±0.022)	0.76(±0.12)	0.51(±0.02)	0.809(±0.001)
	RUN	4341	1184	9811	97148
CILP++	*ACC*	91.70%(±5.84)	77.17%(±9.01)	79.91%(±2.12)	68.91%(±2.12)
	AUC	0.82(±0.02)	0.77(±0.07)	0.80(±0.01)	0.79(±0.02)
	RUN	851	742	3822	11435

- CILP++ has accuracy and AUC measure comparable with both Aleph and MLNs, while having considerably better runtimes

Experimental results (Extracted rules)

SYSTEM	METRICS	DATASETS			
		Mutagenesis	UW-CSE	Alzheimer-anime	Cora
Aleph	*ACC*	80.85%(±10.51)	85.11%(±7.11)	78.71%(±5.25)	−−
CILP++	*ACC*	77.72(±2.12)	81.98(±3.11)	78.70%(±6.12)	67.98%(±5.29)
	FID	0.89	0.90	0.88	0.82

- Rule accuracy levels comparable with Aleph have been obtained after extraction
- Best fidelity (of rules to network) measures seem reasonable
- This indicates that CILP++ can learn relational knowledge efficiently

Conclusion and Future Work

- CILP++ is able to learn relational knowledge efficiently improving on Aleph or MLNs on at least one dataset.
- We are currently investigating the use of macro-operators (Alphonse, 2004) in CILP++; this produces 100% average test set accuracy on the trains dataset
- CILP++ is available at: http://sourceforge.net/projects/cilppp
- CILP++ learning the Mutagenesis benchmark is available at: http://sourceforge.net/projects/cilppp/files/Windows/CILP%2B%2B_mutaExamples.zip/download

Old subject… new developments…

Neural-Symbolic Learning and Reasoning: Dagstuhl seminar 14381, Wadern, Germany, September 2014

AAAI Spring Symposium on Knowledge Representation and Reasoning: Integrating symbolic and neural approaches, Stanford University, March 2015

Neural-Symbolic Learning and Reasoning workshop (NeSy15 at IJCAI), Buenos Aires, July 2015 http://www.neural-symbolic.org/NeSy15/

Cognitive Computation: Integrating neural and symbolic approaches, NIPS 2015, Montreal, December 2015 http://www.neural-symbolic.org/CoCo2015/

Neural-Symbolic Learning and Reasoning workshop (NeSy16), The New School, New York (60 years of the Dartmouth conference), July 2016

JLC learning and reasoning corner, A. d'Avila Garcez and L. Valiant (eds.)

JAIR special track on deep learning and symbolic reasoning (TBC)

NeSy association: www.neural-symbolic.org

Neural-Symbolic Learning and Reasoning
(Dagstuhl seminar 14381, September 2014)

Programming analogy: the need for low-level and high-level languages

Knowledge representation: computer science logic, planning, actions, time, modalities, preferences, defeasibility, relations

Learning: semi-supervised, levels of abstraction, modularity

Consolidation: knowledge extraction and transfer learning

Killer app: big data + descriptions / explanations

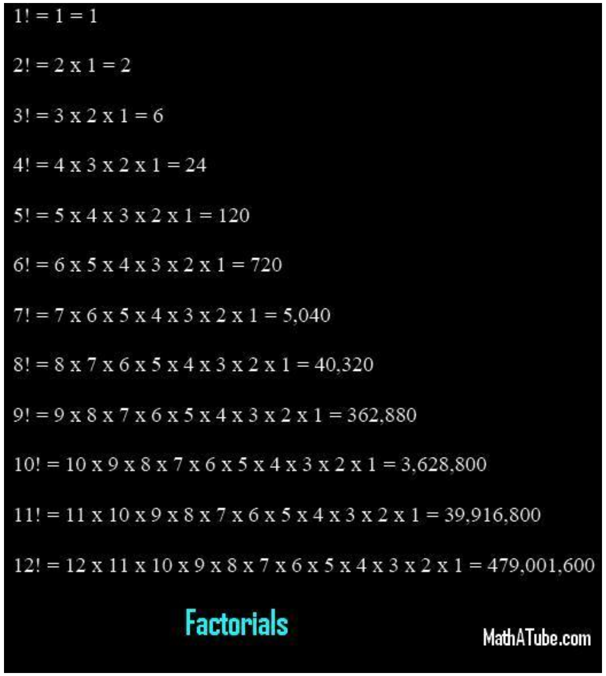
Challenges: c.f. Davis and Marcus CACM article on Commonsense Reasoning (September 2015)

Challenge:
Goal-directed Reasoning and Learning

```
factorial(0,1).
factorial(N,F) :- N>0, N1 is N-1, factorial(N1,F1),
                  F is N * F1.


?- factorial(3,W).
W=6
```

Tightly coupled and loosely coupled neural-symbolic systems…



Thank you!

Machine Learning group at City University London

PESC 40 anos
COPPE • UFRJ
Programa de Engenharia de Sistemas e Computação

Manoel V. M. França
manoel.franca@city.ac.uk

Artur S. d'Avila Garcez
a.garcez@city.ac.uk

Gerson Zaverucha
gerson@cos.ufrj.br