Neuro-Symbolic Artificial Intelligence
Chapter 8
Neuro-Symbolic Programming

Nils Holzenberger

April 8, 2024

Outline

1. End-to-end Differentiable Proving

2. Probabilistic Soft Logic

Outline

1. End-to-end Differentiable Proving
2. Probabilistic Soft Logic

End-to-end Differentiable Proving

End-to-End Differentiable Proving

Tim Rocktäschel
University of Oxford
tim.rocktaschel@cs.ox.ac.uk

Sebastian Riedel
University College London & Bloomsbury AI
s.riedel@cs.ucl.ac.uk

Rocktäschel & Riedel, *End-to-end Differentiable Proving*, NIPS 2017

The problem

Inconsistent KBs

```
country("Austria").
capitalOf("Austria", "Vienna").
city("Wien").
country("Switzerland").
neighbor("Schweiz", "Austria").
capital("Suisse", "Berne").
city("Bern").
neighboring_capitals(Cap1, Cap2) :-
    capital(Ctr1, Cap1), capital(Ctr2, Cap2),
    neighbor(Ctr1, Ctr2), city(Cap1), city(Cap2),
    country(Ctr1), country(Ctr2).
?- neighboring_capitals("Switzerland", "Austria").
```

https://github.com/mledoze/countries

The solution

- Distinct symbols represent the same entities
  `Österreich`, `Oesterreich`, `Austria`, `Autriche` → `Austria`

The solution

- Distinct symbols represent the same entities
  `Österreich`, `Oesterreich`, `Austria`, `Autriche` → `Austria`
- Soft, parametric unification $\mathtt{u}_\theta$:
  - anything can unify with anything, e.g. `Österreich` with `Austria`
  - but every unification incurs a cost
  - as we go through the SLD tree, we keep the proofs with highest scores

The solution

- Distinct symbols represent the same entities
  `Österreich`, `Oesterreich`, `Austria`, `Autriche` → `Austria`
- Soft, parametric unification u_θ:
  - anything can unify with anything, e.g. `Österreich` with `Austria`
  - but every unification incurs a cost
  - as we go through the SLD tree, we keep the proofs with highest scores
- In detail:
  - Two variables u_θ`(X, Y)` → score of 1 and `X=Y`
  - A variable and a constant u_θ`(X, c)` → score of 1 and `X=c`
  - Two constants u_θ`(a, b)` → score of $\exp(-||\theta_a - \theta_b||)$

The solution

- Distinct symbols represent the same entities
  `Österreich`, `Oesterreich`, `Austria`, `Autriche` → `Austria`
- Soft, parametric unification $\mathtt{u}_\theta$:
  - anything can unify with anything, e.g. `Österreich` with `Austria`
  - but every unification incurs a cost
  - as we go through the SLD tree, we keep the proofs with highest scores
- In detail:
  - Two variables $\mathtt{u}_\theta(\mathtt{X},\ \mathtt{Y})$ → score of 1 and $\mathtt{X}=\mathtt{Y}$
  - A variable and a constant $\mathtt{u}_\theta(\mathtt{X},\ \mathtt{c})$ → score of 1 and $\mathtt{X}=\mathtt{c}$
  - Two constants $\mathtt{u}_\theta(\mathtt{a},\ \mathtt{b})$ → score of $\exp(-||\theta_a - \theta_b||)$
- Every constant and every predicate a is represented by a high-dimensional, learnable vector θ_a
- The idea is that the vectors `Österreich`, `Oesterreich`, `Austria`, `Autriche` will end up close together

Rocktäschel & Riedel, *End-to-end Differentiable Proving*, NIPS 2017

Extensions

- Started as a PhD thesis in 2017
- Has been extended for
  - scalability (speed of inference + size of KB) [1]
  - use directly on natural language[2]
  - producing explanations[3]

[1] Minervini *et al*, *Towards Neural Theorem Proving at Scale*, NAMPI@ICML 2018

[2] Weber *et al*, *NLProlog: Reasoning with Weak Unification for Question Answering in Natural Language*, ACL 2019
Minervini *et al*, *Differentiable Reasoning on Large Knowledge Bases and Natural Language*, Knowledge Graphs for eXplainable Artificial Intelligence 2020

[3] Bianchi *et al*, *Knowledge Graph Embeddings and Explainable AI*, Knowledge Graphs for eXplainable Artificial Intelligence 2020

Outline

The problem

- For some problems, we can leverage structure, e.g. social and biological networks
- For some problems, we can leverage large amounts of data, e.g. the Web
- Structured models don't scale very well, so how do we leverage both?

The solution

Hinge-Loss Markov Random Fields and Probabilistic Soft Logic

Stephen H. Bach BACH@CS.STANFORD.EDU
Computer Science Department
Stanford University
Stanford, CA 94305, USA

Matthias Broecheler MATTHIAS@DATASTAX.COM
DataStax

Bert Huang BHUANG@VT.EDU
Computer Science Department
Virginia Tech
Blacksburg, VA 24061, USA

Lise Getoor GETOOR@SOE.UCSC.EDU
Computer Science Department
University of California, Santa Cruz
Santa Cruz, CA 95064, USA

Editor: Luc De Raedt

Bach *et al*, *Hinge-Loss Markov Random Fields and Probabilistic Soft Logic*, J. Mach. Learn. Res. 2017

The solution

- Rewrite Prolog-like rules into CNF, interpret them as objective functions
- Relax the resulting SAT problem using soft logic
- Use convex optimization to find the truth values (in [0,1]) for each grounded formula

Example

Knowledge base:

```
a(X) <- b(X).
a(c1).
b(c2).
```

Groundings + truth values:

`a(c1)`	$x_1 = 1$
`a(c2)`	$x_2 \in [0, 1]$
`b(c1)`	$x_3 \in [0, 1]$
`b(c2)`	$x_4 = 1$

Example

Knowledge base:

```
a(X) <- b(X).
a(c1).
b(c2).
```

Groundings + truth values:

`a(c1)`	$x_1 = 1$
`a(c2)`	$x_2 \in [0,1]$
`b(c1)`	$x_3 \in [0,1]$
`b(c2)`	$x_4 = 1$

Turning the rule into an objective:

- `a(c1) <- b(c1)`
- $a(c1) \vee \neg b(c1)$
- $\min\{1, x_1 + (1 - x_3)\}$ using Łukasiewicz logic

Full objective:

$$\operatorname*{argmax}_{x_1,x_2,x_3,x_4} \min\{1, x_1 + (1 - x_3)\} + \min\{1, x_2 + (1 - x_4)\}$$

Example

Knowledge base:

```
a(X) <- b(X).
a(c1).
b(c2).
```

Groundings + truth values:

`a(c1)`	$x_1 = 1$
`a(c2)`	$x_2 \in [0,1]$
`b(c1)`	$x_3 \in [0,1]$
`b(c2)`	$x_4 = 1$

Turning the rule into an objective:

- `a(c1) <- b(c1)`
- $a(c1) \vee \neg b(c1)$
- $\min\{1, x_1 + (1 - x_3)\}$ using Łukasiewicz logic

Full objective:

$$\operatorname*{argmax}_{x_1,x_2,x_3,x_4} \min\{1, x_1 + (1 - x_3)\} + \min\{1, x_2 + (1 - x_4)\}$$

Actual objective: $\operatorname*{argmax}_{x_2,x_3} \min\{1, 2 - x_3\} + \min\{1, x_2\}$

Example

Knowledge base:

```
a(X) <- b(X).
a(c1).
b(c2).
```

Groundings + truth values:

`a(c1)`	$x_1 = 1$
`a(c2)`	$x_2 \in [0,1]$
`b(c1)`	$x_3 \in [0,1]$
`b(c2)`	$x_4 = 1$

Turning the rule into an objective:

- `a(c1) <- b(c1)`
- $a(c1) \vee \neg b(c1)$
- $\min\{1, x_1 + (1 - x_3)\}$ using Łukasiewicz logic

Full objective:

$$\underset{x_1,x_2,x_3,x_4}{\operatorname{argmax}} \min\{1, x_1 + (1 - x_3)\} + \min\{1, x_2 + (1 - x_4)\}$$

Actual objective: $\underset{x_2,x_3}{\operatorname{argmax}} \min\{1, 2 - x_3\} + \min\{1, x_2\}$

→ $x_2 = 1$ and the value of x_3 can be anywhere between 0 and 1.

Extensions

The package is called *Probabilistic Soft Logic (PSL)*

- It is well documented
  - Website[4]
  - Talks and tutorials
  - Wikipedia page
- It has been extended for scalability etc[5]

[4] https://psl.linqs.org/

[5] Magliacane *et al*, *foxPSL: A Fast, Optimized and eXtended PSL implementation*, Int. J. Approx. Reason. 2015

Extensions

It has been used in lots of applications

- Drug-drug interaction [6]
- Entity resolution [7]
- Recommender systems [8]
- Stance prediction in online debates [9]
- Knowledge graph inference [10]

[6] Sridhar *et al*, *A probabilistic approach for collective similarity-based drug-drug interaction prediction*, Bioinform. 2016

[7] Bhattacharya & Getoor, *Collective entity resolution in relational data*, ACM Trans. Knowl. Discov. Data 2007

[8] Kouki *et al*, *HyPER: A Flexible and Extensible Probabilistic Framework for Hybrid Recommender Systems*, RecSys 2015

[9] Sridhar *et al*, *Joint Models of Disagreement and Stance in Online Debate*, ACL 2015

[10] Pujara *et al*, *Knowledge Graph Identification*, ISWC 2013