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

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").
nearby(Cap1, Cap2) :-
capital(Ctr1, Cap1), capital(Ctr2, Cap2),
nearby(Ctr1, Ctr2), city(Cap1), city(Cap2),
country(Ctr1), country(Ctr2).

?- nearby("Switzerland", "Austria").

https://github.com/mledoze/countries
The solution

- Distinct symbols represent the same entities
  Österreich, Oesterreich, Austria, Autriche $\rightarrow$ Austria
The solution

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

- Soft, parametric unification $u_\theta$:
  - 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 $u_\theta$:
  - 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 $u_\theta(X, Y) \rightarrow$ score of 1 and $X=Y$
  - A variable and a constant $u_\theta(X, c) \rightarrow$ score of 1 and $X=c$
  - Two constants $u_\theta(a, b) \rightarrow$ score of $\exp(-||\theta_a - \theta_b||)$
The solution

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

- Soft, parametric unification $u_\theta$:
  - 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 $u_\theta(X, Y) \rightarrow$ score of 1 and $X=Y$
  - A variable and a constant $u_\theta(X, c) \rightarrow$ score of 1 and $X=c$
  - Two constants $u_\theta(a, b) \rightarrow$ score of $\exp(-||\theta_a - \theta_b||)$

- Every constant and every predicate $a$ is represented by a high-dimensional, learnable vector $\theta_a$

- 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


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

1. End-to-end Differentiable Proving

2. Probabilistic Soft Logic
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  
*Computer Science Department*  
*Stanford University*  
*Stanford, CA 94305, USA*

Matthias Broecheler  
*DataStax*

Bert Huang  
*Computer Science Department*  
*Virginia Tech*  
*Blacksburg, VA 24061, USA*

Lise Getoor  
*Computer Science Department*  
*University of California, Santa Cruz*  
*Santa Cruz, CA 95064, USA*

Editor: Luc De Raedt

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:

\[ a(X) \leftarrow b(X). \]
\[ a(c1). \]
\[ b(c2). \]

Groundings + truth values:

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

Knowledge base:

\[ a(X) \leftarrow b(X). \]
\[ a(c1). \]
\[ b(c2). \]

Groundings + truth values:

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

Turning the rule into an objective:

- \[ a(c1) \leftarrow b(c1) \]
- \[ a(c1) \lor \neg b(c1) \]
- \[ \min\{1, x_1 + (1 - x_3)\} \text{ using Łukasiewicz logic} \]

Full objective:

\[ \text{argmax } \min\{1, x_1 + (1 - x_3)\} + \min\{1, x_2 + (1 - x_4)\} \]
\[ x_1, x_2, x_3, x_4 \]
Example

Knowledge base:

\[ a(X) \leftarrow b(X). \]
\[ a(c1). \]
\[ b(c2). \]

Turning the rule into an objective:

\[ a(c1) \leftarrow b(c1) \]
\[ a(c1) \lor \neg b(c1) \]
\[ \min\{1, x_1 + (1 - x_3)\} \text{ using Łukasiewicz logic} \]

Full objective:

\[ \arg\max_{x_1, x_2, x_3, x_4} \min\{1, x_1 + (1 - x_3)\} + \min\{1, x_2 + (1 - x_4)\} \]

Actual objective: \[ \arg\max_{x_2, x_3} \min\{1, 2 - x_3\} + \min\{1, x_2\} \]
Example

Knowledge base:

\[ a(X) \leftarrow b(X). \]
\[ a(c_1). \]
\[ b(c_2). \]

Groundings + truth values:

- \[ a(c_1) \quad x_1 = 1 \]
- \[ a(c_2) \quad x_2 \in [0, 1] \]
- \[ b(c_1) \quad x_3 \in [0, 1] \]
- \[ b(c_2) \quad x_4 = 1 \]

Turning the rule into an objective:

- \[ a(c_1) \leftarrow b(c_1) \]
- \[ a(c_1) \lor \neg b(c_1) \]
- \[ \min\{1, x_1 + (1 - x_3)\} \text{ using } \Lukasiewicz \text{ logic} \]

Full objective:

\[ \arg \max_{x_1, x_2, x_3, x_4} \min\{1, x_1 + (1 - x_3)\} + \min\{1, x_2 + (1 - x_4)\} \]

Actual objective: \[ \arg \max_{x_2, x_3} \min\{1, 2 - x_3\} + \min\{1, x_2\} \]

\[ \rightarrow x_2 = 1 \text{ and the value of } x_3 \text{ can be anywhere between 0 and 1.} \]
Extensions

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

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

---

4 https://psl.linqs.org/

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 \(^10\)

---


\(^10\) Pujara et al, *Knowledge Graph Identification*, ISWC 2013