Scalable Rule Learning in Probabilistic Knowledge Bases

Arcchit Jain, Tal Friedman, Ondřej Kuželka, Guy Van den Broeck, Luc De Raedt

Submitted to the conference of **Automatic Knowledge Base Construction** 2019 (https://openreview.net/forum?id=Hkyl-5667)

November 26, 2018

DTAI
KU Leuven

Background: Probablistic Databases (PDBs)

researcher	paper	P
bob	plp	0.9
carl	plp	0.6
greg	plp	0.7
ian	db	0.9
harry	db	0.8

Table 1: *author/2*

researcher	university	P
edwin	harvard	1.0
fred	harvard	0.9
alice	mit	0.6
dave	mit	0.7

Table 2: *location/2*

researcher	researcher	P
alice	edwin	0.2
alice	fred	0.3
bob	carl	0.4
bob	greg	0.5
bob	harry	0.6
bob	ian	0.7
carl	greg	0.8
carl	harry	0.9
carl	ian	0.8
dave	edwin	0.7
dave	fred	0.6
edwin	fred	0.5
greg	harry	0.4
greg	ian	0.3
ian	ian	0.2

Table 3: *coauthor/2*

Background: Rules and queries

Probablistic Rules

ProbLog rules of the following form:

$0.9 :: coauthor(A, B) :- \text{author}(A, C), \text{author}(B, C).$

$0.1 :: coauthor(A, B) :- \text{location}(A, C), \text{location}(B, C).$

Queries

We mainly work with Union of Conjunctive Queries that do not have negation.

For example:

$Q = author(A, C), author(B, C), p1(D) \vee location(A, E), location(B, E), p2(F).$

Background: Lifted Inference

- To do exact probabilistic inference for a query in a database setting, a sequence of database operations like join, projection, union, selection and difference are executed.
- Such sequences are called **query plans**.

- **Lifted inference** constructs (extensional) query plans by decomposing the query into simpler queries.

  Example: For a query $Q = Q_1 \vee Q2$ such that $Q_1 \perp Q_2$,
  $$\mathrm{P}(Q) = 1 - (1 - \mathrm{P}(Q_1)) * (1 - \mathrm{P}(Q_2))$$

- Every query for which a correct query plan exists is called a **safe query**.
- For every safe query, the complexity of evaluating it on a PDB is PTIME.

Problem Specification

Given:

PDB $\mathcal{D}$, a target relation *target*, and loss function $\mathcal{L}$

To Find:

A set of probabilistic rules called $H = \{h_1, h_2, \ldots, h_n\}$ with *target* in the head of each rule such that $\mathcal{L}(H)$ is minimum over all the *target* tuples w.r.t. $\mathcal{D}$.

Structure Learning: Generating Candidate Rules

- We use **AMIE+** for generating deterministic rules as candidates.
- AMIE+ uses a language bias to mine rules from a vast search space.

Example:
java -jar amie_plus.jar -minhc 1e-05 -minpca 1e-05 -htr '⟨coauthor⟩' -bexr '⟨coauthor⟩' -oute Data/test_amie.tsv

?a ⟨location⟩ ?f ?b ⟨location⟩ ?f => ?a ⟨coauthor⟩ ?b
?a ⟨author⟩ ?f ?b ⟨author⟩ ?f => ?a ⟨coauthor⟩ ?b

- AMIE+ only uses deterministic tuples to generate deterministic rules.
- SafeLearner only select top k rules with the highest confidence as our candidates.

Parameter Learning

- To predict probability of a tuple, SafeLearner represents the rules H into a UCQ Q and feed it to **SafeSample** - a lifted inference engine based on $Lift_R^0$ algorithm developed at UCLA
- SafeLearner uses **cross entropy** for the loss function $\mathcal{L}$ to learn the rule probabilities.

$$CE = \sum_{\langle t_i, p_i \rangle \in E} \left(p_i \log q_i + (1 - p_i) \log (1 - q_i) \right); \qquad \mathcal{L} = 1 - \frac{CE}{|E|}$$

where E is the table of *target* (coauthor) and p_i, and
q_i are the actual and predicted probabilities of i^{th} tuple respectively.

- Cross entropy is used as it is equal to the expectation of likelihood of the rules.

Parameter Learning

- When estimating the loss of a set of rules H (represented as a UCQ Q), we need to split the possible tuples into three categories:
  1. E tuples contained in the training set
  2. E_2 tuples contained in the answer of Q but not in the training set, and
  3. E_3 the other tuples.

- SafeLearner initializes the probabilistic weights of the rules and sets them equal to their supports/confidences estimated from the training data.

- We perform **Stochastic Gradient Descent** to optimize rule weights.

SafeLearner - Algorithm

Algorithm 1 SafeLearner

Input: PDB $\mathcal{D}$, *target*, loss $\mathcal{L}$
E := Set of all *target* tuples in $\mathcal{D}$
H := Set of all the type consistent and significant (deterministic) rules from AMIE+ using $\mathcal{D}$ with *target* in head
Initialize probability p_{h_i} for each rule h_i in H
Embed rule probabilities $p_h = \{p_{h_1}, p_{h_2}, \ldots, p_{h_n}\}$ in H
Sample *target* tuples in E_2 and E_3 and compute their respective sampling weights w_2 and w_3
Q := UCQ corresponding all the probabilistic rules in H
for all i in range(0, MaxIterations) **do**
    Randomly select an example e from all the target examples $E \cup E_2 \cup E_3$
    y := predicted probability of e w.r.t. Q as a function of p_h
    if $e \in E$ **then**
        x := actual probability of e from E
        Compute $\mathcal{L}$ for e using x, y with sampling weight = 1
    else if $e \in E_2$ **then**
        Compute $\mathcal{L}$ for e using $x := 0$ and y with sampling weight := w_2
    else if $e \in E_3$ **then**
        Compute $\mathcal{L}$ for e using $x := 0$ and y with sampling weight := w_3
    end if
    Get gradient of $\mathcal{L}$ at current p_h
    Update p_h
end for
Remove rules with insignificant rule weights from H
return H =0

Experiments

We empirically address the following two crucial questions experimentally:

1. How does SafeLearner **compare** against related baselines?
   Dataset: NELL Sports Dataset (850th iteration) (Same as ProbFOIL^{+})

2. How well does SafeLearner **scale-up**?

Datasets	# of Relations	# of Tuples
NELL Sports Dataset (1115th iteration)	426	233k
Yago 2.4	33	948k

How does SafeLearner compare against related baselines?

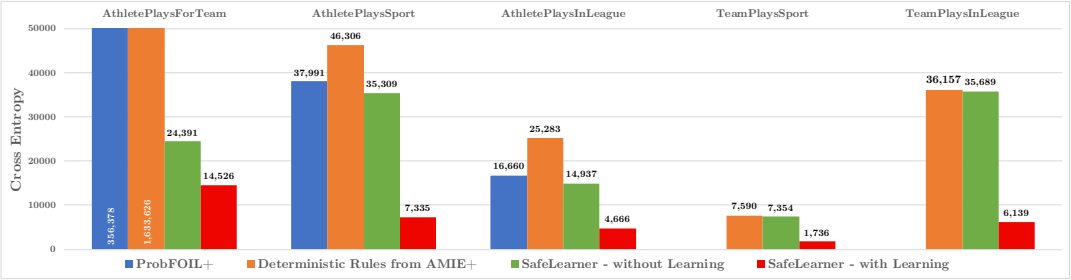

Figure 1: SafeLearner has better Cross Entropy

How does SafeLearner compare against related baselines?

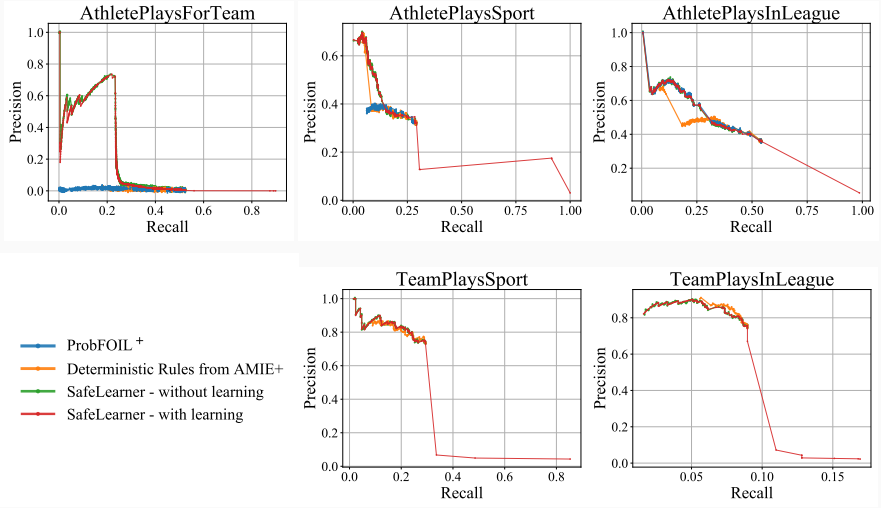

Figure 2: Precision-Recall Curves for 5 relations in NELL Sports Dataset

How well does SafeLearner scale-up?

Dataset	*target* relation	# of *target* tuples	# of rules learned	Learning time
NELL	AthletePlaysForTeam	1687	3	1 Hour 11 Minutes
NELL	AthletePlaysSport	1959	5	2 Hour 8 Minutes
NELL	AthletePlaysInLeague	1310	5	1 Hour 34 Minutes
NELL	TeamPlaysSport	355	5	5 Hours 16 Minutes
NELL	TeamPlaysInLeague	1354	5	3 Hours 1 Minutes
Yago	IsCitizenOf	14554	4	2 Hours 25 Minutes

Table 4: SafeLearner is able to scale upto NELL (1115) and YAGO 2.4 (standard subset)

Key contributions

- The paper accomplishes probabilistic rule learning using a novel inference setting.
- Unlike ProbFOIL$^+$, SafeLearner scales well on the full database of **NELL** with 233k tuples as well as on the standard subset of **Yago** 2.4 with 948k tuples.
- SafeLearner is faster than ProbFOIL$^+$ because :
  1) it disintegrates longer complex queries to smaller simpler ones,
  2) it caches the structure of queries before doing inference, and
  3) it uses lifted inference to infer on those simple queries.
- No declarative bias is required in SafeLearner from the user, unlike ProbFOIL $^+$.

Future Works

- SafeLearner cannot learn compex rules that translate to **unsafe** queries.
- It does not use rules within the background theory.
- It can not learn rules with negations.
- It can not learn rules on **numeric** data.
- Probabilistic rule learning could be done in the **open-world setting**.

Questions?

Thank you for your attention.