Neuro-Symbolic Artificial Intelligence
Chapter 7
Statistical Machine Learning with ProbLog

Nils Holzenberger

April 2, 2024

Outline

Outline

Sampling programs...?

```
1(X,Y):- f(X,Y).
0.8::1(X,Y):- f(X,Z), 1(Z,Y).
0.3::f(john,mary).
0.7::f(mary,pedro).
0.4::f(mary,tom).
0.6::f(pedro,tom).
```

```
1(X,Y):- f(X,Y).
1(X,Y):- f(X,Z), 1(Z,Y).
f(mary,tom).
f(pedro,tom).
```

Probability of sample:

$1 \times .8 \times (1 - .3) \times (1 - .7) \times .4 \times .6 \approx .04$

Sampling programs...?

```
1(X,Y):- f(X,Y).
0.8::1(X,Y):- f(X,Z), 1(Z,Y).
0.3::f(john,mary).
0.7::f(mary,pedro).
0.4::f(mary,tom).
0.6::f(pedro,tom).
```

Most probable program:

```
1(X,Y):- f(X,Y).
1(X,Y):- f(X,Z), 1(Z,Y).
f(mary,pedro).
f(pedro,tom).
```

Probability of sample:

$1 \times .8 \times (1 - .3) \times .7 \times (1 - .4) \times .6 \approx .14$

Sampling programs...?

```
l(X,Y):- f(X,Y).
0.8::l(X,Y):- f(X,Z), l(Z,Y).
0.3::f(john,mary).
0.7::f(mary,pedro).
0.4::f(mary,tom).
0.6::f(pedro,tom).
```

Probability of this sample?

```
l(X,Y):- f(X,Z), l(Z,Y).
f(mary,pedro).
f(pedro,tom).
```

Probability of sample: $0 \times \ldots = 0$

Sampling programs

`T = [(w_1, c_1), (w_2, c_2) ... ]` a ProbLog program

Pseudo-code to sample Prolog programs from `T`:

```
def sample(T):
    import random
    output = []
    for w, c in T:
        theta = random.rand()
        if w>theta:
                output.append(c)
    return output
```

You can do this with `problog sample model.pl`: it will sample Prolog programs and evaluate the result of the queries in `model.pl`.

`evidence(a).` *vs* `1::a.`

See `fruit.pl`

```
.5::happy(X) :- X=bob; X=alice.
happy(bob) :- fruit.
happy(alice) :- cake.
.3::cake ; .7::fruit.
```

Negative rules

```
.5::happy.
.3::happy :- fun_class.
.7::\+happy :- raining.
.9::\+happy :- school_test.
.1::fun_class.
.05::school_test.
.2::raining.
```

Draw a sample:

```
happy.
happy :- fun_class.
\+happy :- raining.
raining.
```

A sample is a Prolog program; but what is `\+happy :- raining.` in Prolog??

Negative rules

```
.5::happy.
.3::happy :- fun_class.
.7::\+happy :- raining.
.9::\+happy :- school_test.
.1::fun_class.
.05::school_test.
.2::raining.
```

Under the hood, ProbLog adds a new rule to each sample:

```
happy :- happy_p, \+ happy_n.
```

Previous sample becomes:

```
happy_p.
happy_p :- fun_class.
happy_n :- raining.
raining.
happy :- happy_p, \+ happy_n.
```

Now it's a valid Prolog program.

SLD tree

R1 `1(X,Y):- f(X,Y).`

R2 `0.8::1(X,Y):- f(X,Z), 1(Z,Y).`

R3 `0.5::f(john,mary).`

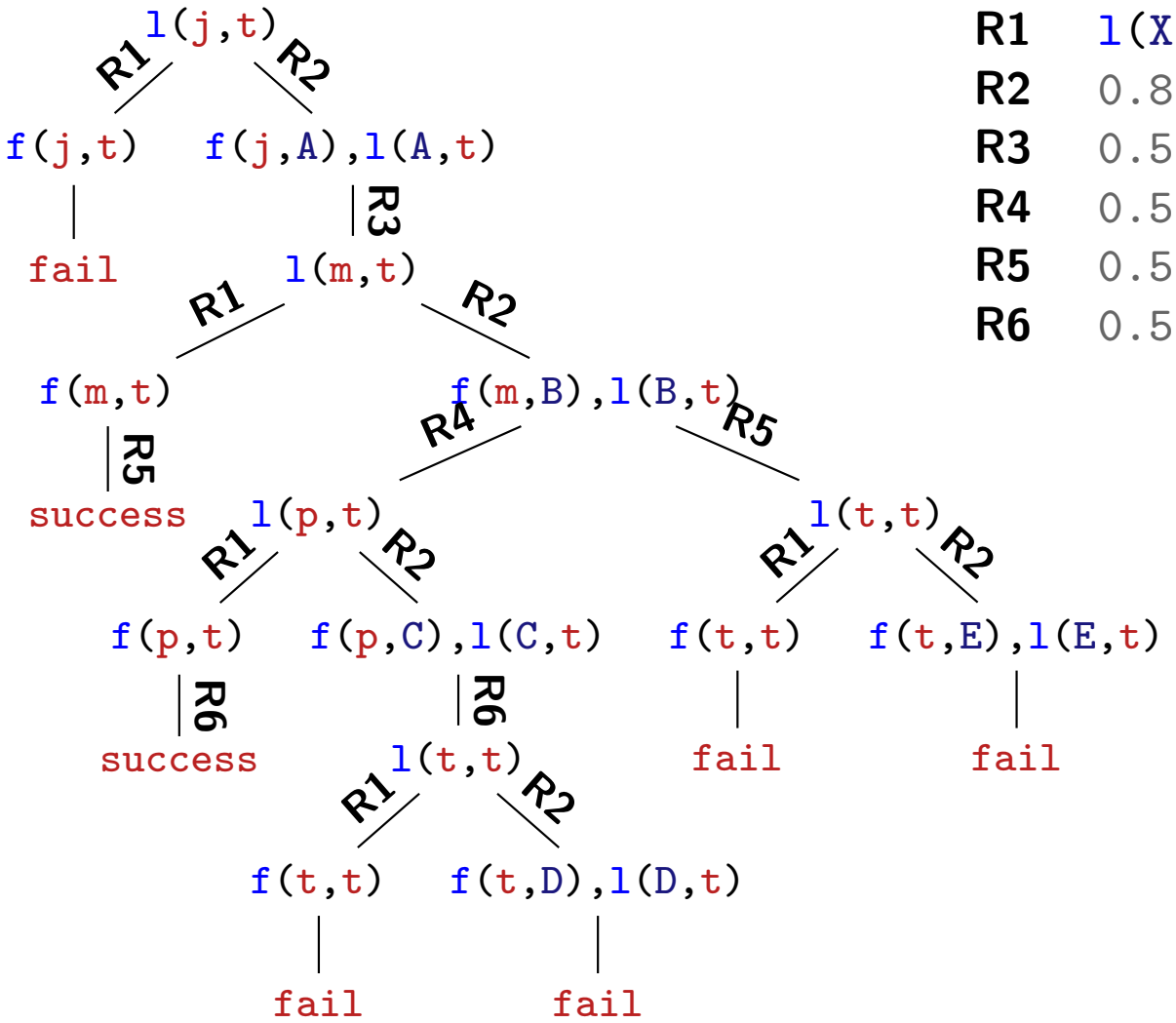
R4 `0.5::f(mary,pedro).`

R5 `0.5::f(mary,tom).`

R6 `0.5::f(pedro,tom).`

SLD tree

Query: `l(john,tom)`



R1 `l(X,Y):- f(X,Y).`
R2 `0.8::l(X,Y):- f(X,Z), l(Z,Y).`
R3 `0.5::f(john,mary).`
R4 `0.5::f(mary,pedro).`
R5 `0.5::f(mary,tom).`
R6 `0.5::f(pedro,tom).`

$Q = (R1 \wedge R2 \wedge R3 \wedge R5) \vee (R1 \wedge R2 \wedge R3 \wedge R4 \wedge R6)$

Run the query `l(j,t)` with the Prolog `trace`. The Prolog solver will traverse this tree depth-first.

De Raedt *et al*, *ProbLog: A Probabilistic Prolog (...)*, IJCAI 2007

Outline

Poker dice

Poker dice

8 languages

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

Poker dice are dice which, instead of having number pips, have representations of playing cards upon them. Poker dice have six sides, one each of an Ace, King, Queen, Jack, 10, and 9, and are used to form a poker hand.

Each variety of poker dice varies slightly in regard to suits, though the ace of spades is almost universally represented. 9♣ and 10♦ are frequently found, while face cards are traditionally represented not by suit, but instead by color: red for kings, green for queens and blue for jacks. Manufacturers have not standardized the colors of the face sides. The game can also be played with ordinary dice.

A set of poker dice owned by a member of the Royal Indian Army Service Corps during the Second World War

As a game [edit]

https://en.wikipedia.org/wiki/Poker_dice

Poker dice

- Roll 5 dice
- How many possible outcomes ? $6^5 = 7776$

Name	Example	Number of combinations	Probability (%)
One pair	5 5 3 4 1	3600	46.30
Two pair	6 6 3 3 1	1800	23.15
Three of a kind	4 4 5 1 4	1200	15.43
Straight	2 3 4 5 6	240	3.09
Full house	1 1 1 5 5	300	3.86
Four of a kind	3 3 3 3 2	150	1.93
Five of a kind	6 6 6 6 6	6	0.08
Nothing	1 2 3 4 6	480	6.17

See `poker-dice.pl`

https://en.wikipedia.org/wiki/Poker_dice

Poker dice – sampling

See `poker-dice.pl`

Run `problog sample poker-dice.pl -N 5`

Run `problog sample poker-dice.pl --estimate -N 78` with multiple copies

Poker dice – Biased dice

My opponent uses biased dice, can I estimate the bias?

```
problog lfi \
    poker-dice.pl poker-dice-samples.pl \
    -v -O learned-model.pl
```

How does ProbLog determine the probabilities of the dice? → *Maximum likelihood estimation*

Example: probabilities in Natural Language Processing

Step 1. Express the quantities of interest as random variables.

eg spam classification:

Experiment = I receive an email

X = the email I receive (it's a string)

Y = 1 if the email is spam, 0 otherwise

Example: probabilities in Natural Language Processing

X = the email I receive (it's a string)

Y = 1 if the email is spam, 0 otherwise

$p(y\|x)$	Given that I received email x, is it spam?
$p(y)$	How probable is it that an email I receive should be spam?
$p(x)$	How probable is it that I should receive email x?
$p(x\|y)$	How probable is it that I should receive email x, assuming that it's spam/not spam?

Poker dice

My opponent uses biased dice, can I estimate the bias?

- What is the experiment?
  rolling 5 dice
- What are the random variables?
  the result of each die: X_1, X_2, X_3, X_4, X_5 with $X_i \in \{1,2,3,4,5,6\}$
- What probability do I want to estimate?
  $p(X_i = j)$ for $i \in \{1,2,3,4,5\}$ and $j \in \{1,2,3,4,5,6\}$.
- What are the parameters of the probability distribution?
  θ_i, the probability that a die lands on i (this distribution is the same for all dice, that's an assumption we're making)

Probability of an observation

- We never observe the probability distribution that we want to estimate
- But we do observe *samples* from that probability distribution
- And we can compute the probability of the samples *as a function of the parameters of our model*
  - Probability of observing the roll $[1,3,5,1,2]$ is $p(X_1=1)p(X_2=3)p(X_3=5)p(X_4=1)p(X_5=2)=\theta_1\theta_3\theta_5\theta_1\theta_2=\theta_1^2\theta_2\theta_3\theta_5$
  - Probability of observing the roll $[3,5,1,5,1]$ is $\theta_1^2\theta_3\theta_5^2$
  - Probability of observing the roll $[4,5,4,6,3]$ is $\theta_3\theta_4^2\theta_5\theta_6$

Maximum likelihood

- We have a set of samples from our target distribution:
  $S = \{o_1, o_2, ..., o_n\}$ where $\forall i,\ o_i = [x_{i,1}, x_{i,2}, x_{i,3}, x_{i,4}, x_{i,5}]$

- We can express the probability of drawing S from the distribution, given our parameters θ:

$$p(S|\theta) = p(o_1, o_2, ..., o_n|\theta) = \prod_{i=1}^{n} p(o_i|\theta)$$

$p(S|\theta)$ is the *likelihood* of S

what assumption are we making here? that the samples are independent

$$p(o_i) = p(x_{i,1}, x_{i,2}, x_{i,3}, x_{i,4}, x_{i,5}|\theta) = \theta_{x_{i,1}}\theta_{x_{i,2}}\theta_{x_{i,3}}\theta_{x_{i,4}}\theta_{x_{i,5}}$$

$$p(S|\theta) = \prod_{i=1}^{n} \theta_{x_{i,1}}\theta_{x_{i,2}}\theta_{x_{i,3}}\theta_{x_{i,4}}\theta_{x_{i,5}} = \prod_{i=1}^{n} f_i(\theta)$$

Maximum likelihood

- We have a set of samples from our target distribution:
  $S = \{o_1, o_2, ..., o_n\}$ where $\forall i,\ o_i = [x_{i,1}, x_{i,2}, x_{i,3}, x_{i,4}, x_{i,5}]$

The likelihood of S is $p(S|\theta) = \prod_{i=1}^{n} f_i(\theta)$

- Surely the parameters that best explain the samples from the distribution must be close to the real distribution, so let's find

$$\theta^\star = \underset{\theta}{\operatorname{argmax}}\, p(S|\theta)$$

- This is the *maximum likelihood estimation* of θ

Maximum likelihood estimation in ProbLog

When you run `problog lfi model.pl samples.pl` what happens?

- `samples.pl` contains the samples from earlier: observations of outcomes of experiments
- `model.pl` contains a probabilistic model that can be used to compute the probability of the observations – where some probabilities are learnable parameters

Maximum likelihood estimation in ProbLog

When you run `problog lfi model.pl samples.pl` what happens?

- Learnable parameters are initialized randomly
- ProbLog tries to maximize the probability of the samples, by modifying the values of the learnable parameters. How? Using *expectation maximization*:
  - For each sample, ProbLog computes the most probable program that explains the sample
  - For each clause, ProbLog counts how many times it appeared in the programs derived above, and sets the probability of the clause s the number of times it was used divided by the number of the times it could have been used
- There are other ways of maximizing likelihood, like gradient descent

Outline

cute

I want to build an image recognizer that can recognize images that are cute

I have 100 images labeled with whether or not they are cute

Manual inspection of the images shows the following:

```
cute(Image) :- dog(Image, Dog),
        small(Image, Dog), fluffy(Image, Dog).
cute(Image) :- cat(Image, Cat), fluffy(Image, Cat).
```

An image is cute if it has a small fluffy dog on it, or it has a fluffy cat on it

So really I just need to build 4 simpler image recognizers: `dog`, `cat`, `small`, `fluffy`.

Trade secrets

I want to build a system that can read a legal case and tell me whether or not someone has done trade secret misappropriation.

Legal experts sent me this:

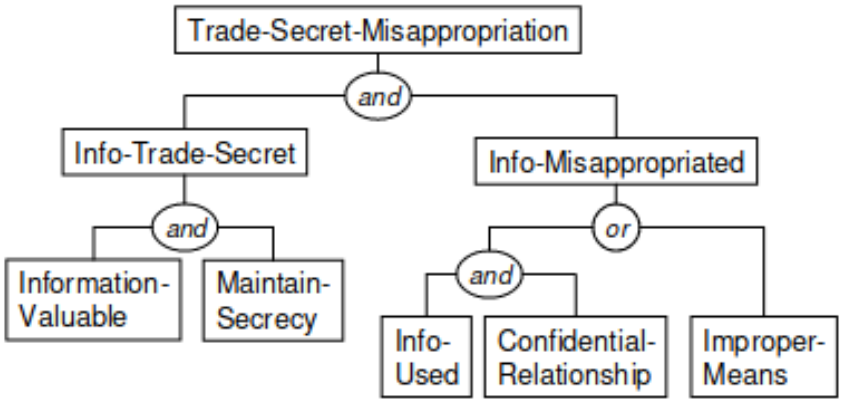


Figure 1: Logical structure for Trade Secrets law

"Trade secret" means information, [...] that:

(i) derives independent economic value, [...] from not being generally known to, and not being readily ascertainable by proper means [...] and

(ii) is the subject of efforts that are reasonable under the circumstances to maintain its secrecy.

One [...] is liable [for trade secret misappropriation if]

(a) he discovered the secret by improper means, or

(b) his disclosure or use constitutes a breach of confidence [...]

Brüninghaus & Ashley, *Predicting Outcomes of Case-based Legal Arguments*, ICAIL 2003

Trade secrets

I want to build a system that can read a legal case and tell me whether or not someone has done trade secret misappropriation.

```
trade_secret_misappropriation(X) :-
        info_trade_secret(X),
        info_misappropriated(X).
info_trade_secret(X) :-
        information_valuable(X),
        maintain_secrecy(X).
info_misappropriated(X) :-
        info_used(X),
        confidential_relationship(X).
info_misappropriated(X) :- improper_means(X).
```

→ I need 5 simple binary classifiers.

Deep ProbLog

- This is the idea behind Deep ProbLog: certain predicates are best modeled with neural networks
  - `dog(Image, Dog)`: whether the input `Image` contains a dog in the region `Dog`
  - `confidential_relationship(X)`: whether a confidential relationship existed between the parties in case `X`
- Neural networks are attached to predicates:
  - `nn(rnn, [X])::confidential_relationship(X).`
  - `nn(cnn, [Image, Dog])::dog(Image,Dog) :- between(1,9,Dog).`
- Parameters of the neural network are learned with the same *maximum likelihood estimation*

Digit addition

Simple problem:

- Input is a pair of black-and-white images, with digits on them
- Output is the sum of the two digits

Manhaeve *et al*, *DeepProbLog: Neural Probabilistic Logic Programming*, NeurIPS 2018
MNIST database `http://yann.lecun.com/exdb/mnist/`

Digit addition

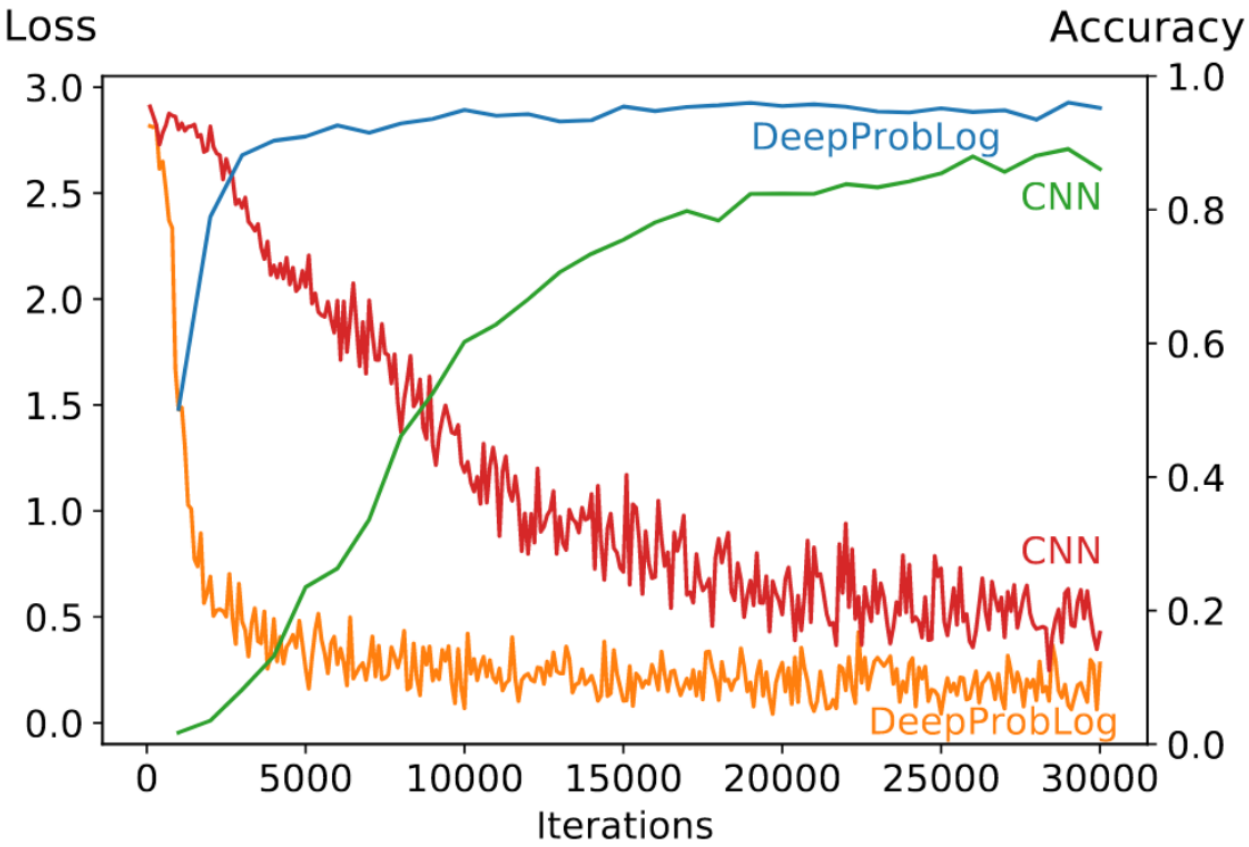
- One way to do it is to learn a classifier mapping pairs of images to their sum: `classifier(Image1, Image2, Sum)`
- Another way is to decompose the problem and learn to classify each digit separately:

```
addition(Image1, Image2, Sum) :-
        digit(Image1, Digit1), digit(Image2, Digit2),
        Sum is Digit1+Digit2.
nn(cnn, [Img], Dgt, [0,1,2,3,4,5,6,7,8,9])::digit(Img, Dgt).
```

Manhaeve *et al*, *DeepProbLog: Neural Probabilistic Logic Programming*, NeurIPS 2018

Digit addition

Does it work?



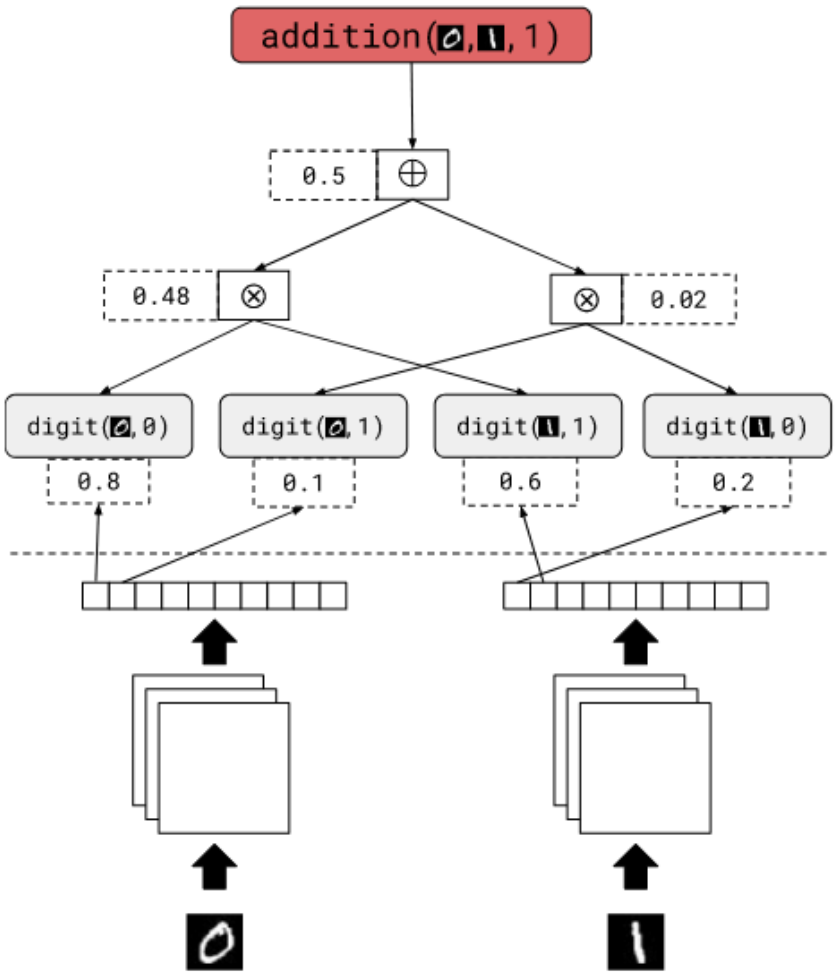
- Final accuracy
- Convergence speed
- Sample efficiency

Manhaeve *et al*, *DeepProbLog: Neural Probabilistic Logic Programming*, NeurIPS 2018

How it works

- Same as for ProbLog, write down the SLD tree, then the BDD
- Get a set of independent proofs to compute the correct answer's probability i.e. the likelihood of the data
- Do gradient descent

Example



(d) The AC for query addition(0, 1, 1).

- Get all ways of proving the observation, and maximize its probability
- This also encourages some wrong classifications
- With enough training data the right classification becomes a much better solution than wrong ones

Manhaeve *et al*, *Neural probabilistic logic programming in DeepProbLog*, Artif. Intell. 2021

Lab session

- Coin flipping
- Parameter learning in ProbLog

Next week

- Lecture (1:30pm-2:45pm)
  - Brief intro to other neuro-symbolic libraries
  - Review session prior to the exam
- Exam (3:15pm-4:15pm)
  - No documents allowed
  - No switched on devices allowed
  - Exams of past 3 years available on website