PESC
Programa de Engenharia de Sistemas e Computação

DeepProbLog

Programação Lógica Neuro-Probabilística

Parte 2 de 2

CPS840 – Tópicos Especiais em Inteligência Artificial
Professor: Gerson Zaverucha

Cleiton Moya de Almeida

Rio de Janeiro, 9 de setembro de 2020

Agenda

PESC
Programa de Engenharia de Sistemas e Computação

Parte 1:

- **Introdução**
  - ✓ Abordagens em IA
  - ✓ Proposta do DeepProbLog
  - ✓ Exemplo
- **Programação em Lógica**
  - ✓ Conceitos Básicos
  - ✓ Prolog
- **ProbLog**
  - ✓ Definição
  - ✓ Inferência
  - ✓ Aprendizado

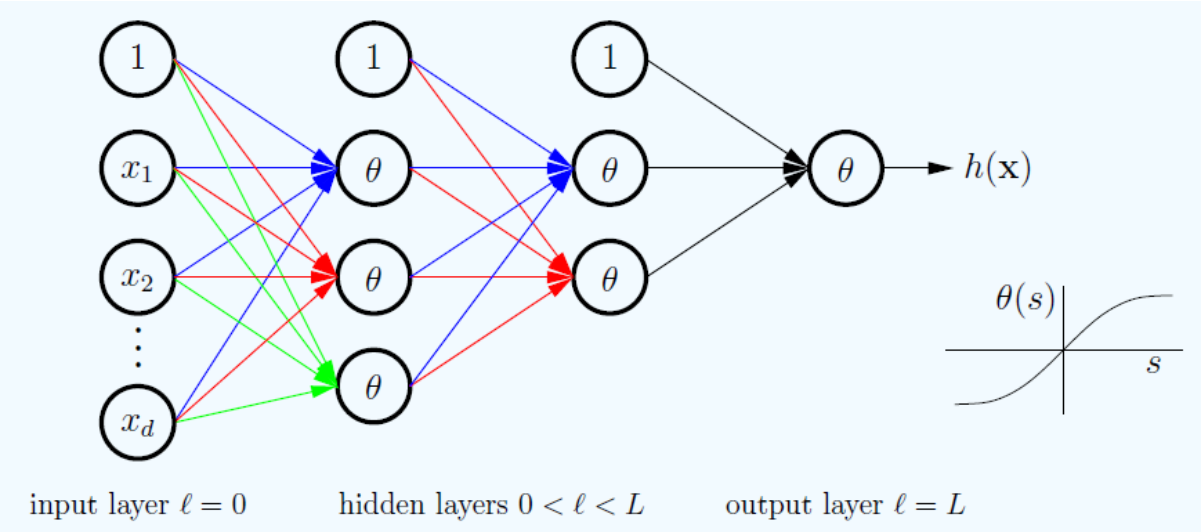
Parte 2:

- ***Deep Learning***
  - ✓ Conceitos básicos
  - ✓ *Backpropagation*
- **DeepProbLog**
  - ✓ Semântica
  - ✓ Inferência
  - ✓ Aprendizado
- **Experimentos**
  - ✓ Raciocínio Lógico e *Deep Learning*
  - ✓ Programação Indutiva
  - ✓ Programação Probabilística e *Deep Learning*
- **Trabalhos correlatos**
- **Conclusões**

Deep Learning

- Conceitos Básicos

- ✓ **Rede neural artificial (NN)**: modelo não-linear altamente hiper parametrizado (por conseguinte bastante flexível);
- ✓ Conjunto de treinamento: $\{(x_i, y_i)\}_{i=1}^{N}$, com N exemplares i.i.d.;
- ✓ $\hat{\boldsymbol{y}} = \mathcal{M}(\boldsymbol{x}|\boldsymbol{\Theta})$, onde M é uma função de mapeamento com parâmetros $\boldsymbol{\Theta}$;
- ✓ $\mathcal{L}(\hat{\boldsymbol{y}}, \boldsymbol{y})$: função de custo (*loss function*)
- ✓ Treinar o modelo significa minimizar o valor esperado $\bar{\mathcal{L}} = \frac{1}{N}\sum_i \mathcal{L}(\mathcal{M}(x_i|\boldsymbol{\Theta}), y_i)$;



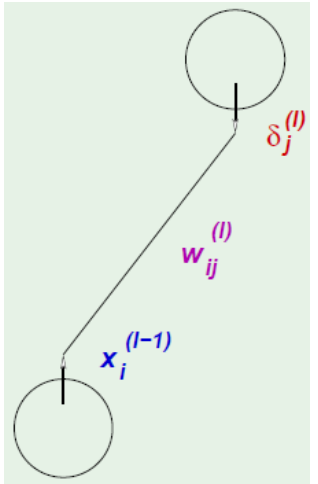
Fonte: Abu-Mostafa, Magdon-Ismail. e-Chapter 7: Neural Networks.

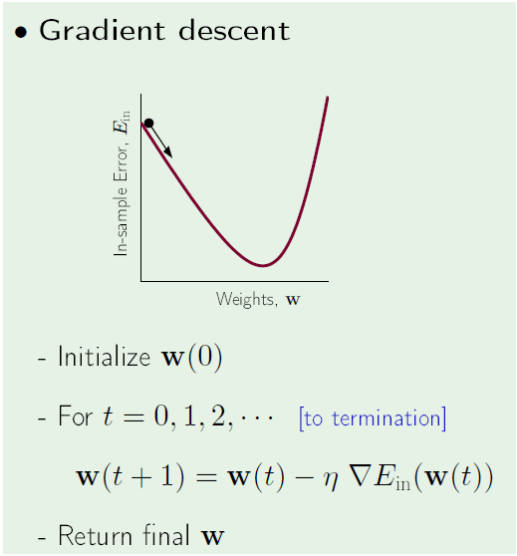
Deep Learning

- *Backpropagation*

✓ A abordagem de otimização mais utilizada em redes neurais é o algoritmo de **Gradiente Descendente;**
✓ O treinamento é feito com o algoritmo ***backpropagation***:

1: Initialize all weights $w_{ij}^{(l)}$ at random
2: for $t = 0, 1, 2, \ldots$ do
3: Pick $n \in \{1, 2, \cdots, N\}$
4: *Forward:* Compute all $x_j^{(l)}$
5: *Backward:* Compute all $\delta_j^{(l)}$
6: Update the weights: $w_{ij}^{(l)} \leftarrow w_{ij}^{(l)} - \eta\, x_i^{(l-1)} \delta_j^{(l)}$
7: Iterate to the next step until it is time to stop
8: Return the final weights $w_{ij}^{(l)}$





Fonte: Abu-Mostafa, Learning From Data. Lecture 10: Neural Networkshttp://work.caltech.edu/slides/slides10.pdf

DeepProbLog

- Semântica
  - ✓ Recordando, **ProbLog** trabalha com um conjunto de fatos probabilísticos:

```
0.2::earthquake.
0.1::burglary.
0.5::hears_alarm(mary).
```

  - ✓ ProbLog suporta também fatos probabilísticos não básicos na forma de **disjunções anotadas** (ADs):

$$p_1 :: h_1 \ ; \ \ldots \ ; \ p_n :: h_n \ :- \ b_1, \ldots, b_m.$$

Onde o somatório de $p_1, \ldots, p_n$ deve ser unitário.

  - ✓ Significado: quando ocorre <u>todos</u> b_j, implica em algum h_i, ou nenhum deles com probabilidade $(1 - \sum p_i)$ [1]
  - ✓ Exemplos:

$$\frac{1}{3} :: \texttt{color(B, green)}; \frac{1}{3} :: \texttt{color(B, red)}; \frac{1}{3} :: \texttt{color(B, blue)} :- \texttt{ball(B)}.$$

$$\underbrace{0.4 :: \texttt{earthquake(none)} \ ; \ 0.4 :: \texttt{earthquake(mild)} \ ; \ 0.2 :: \texttt{earthquake(severe)}.}$$

Conjunto de fatos

  - ✓ Auxiliam a modelagem, mas não são essenciais em ProbLog (podem ser substituídas por fatos + regras).

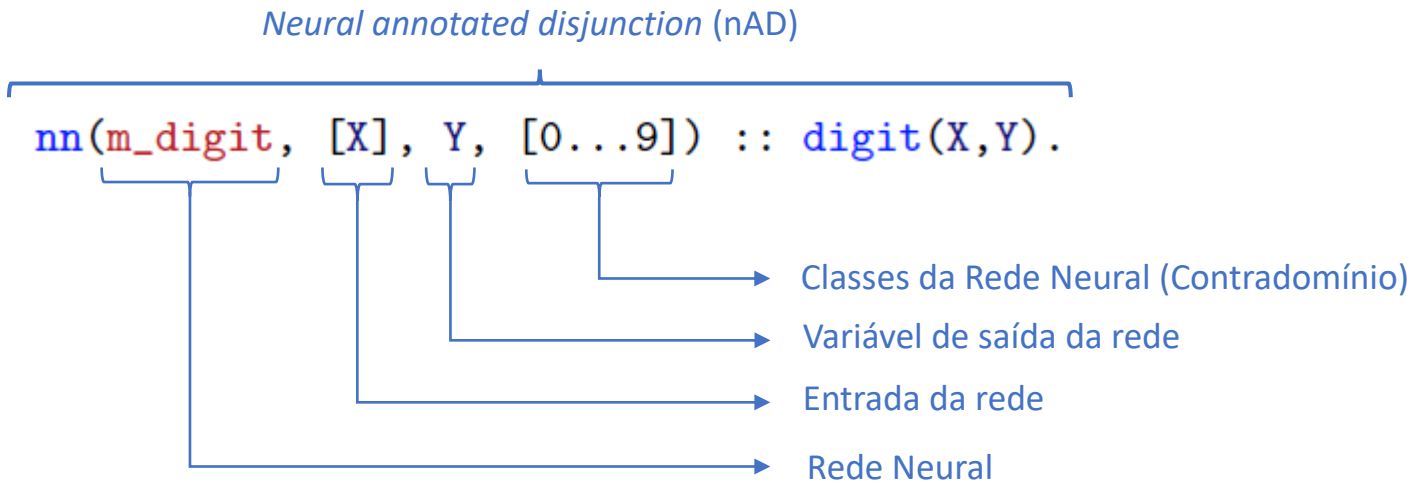
[1] L. De Raedt, A. Kimmig, Probabilistic (logic) programming concepts, Machine Learning 100 (2015) 5–47.

DeepProbLog

- Semântica

✓ Em DeepProbLog, as disjunções anotadas são estendidas de forma a **incorporar o modelo neural**:

```
nn(m_digit, [X], Y, [0...9]) :: digit(X,Y).
addition(X,Y,Z) :- digit(X,N1), digit(Y,N2), Z is N1+N2.
```



DeepProbLog

- Semântica

✓ Exemplo de programa DeepProbLog:

```
nn(m_digit, [X], Y, [0...9]) :: digit(X,Y).
addition(X,Y,Z) :- digit(X,N1), digit(Y,N2), Z is N1+N2.
```

✓ Instanciando o **nAD** a com a imagem de entrada  :

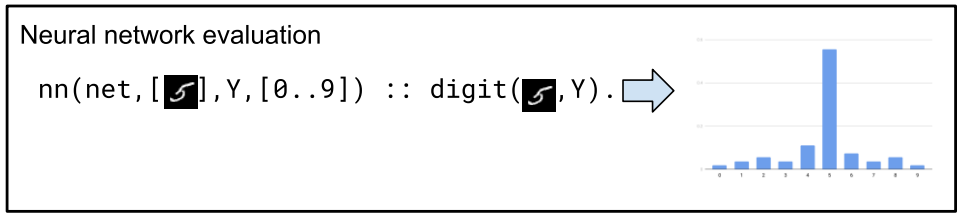
nn(m_digit, [], 0) :: digit(, 0) ; ... ; nn(m_digit, [], 9) :: digit(, 9).

Ground **nAD**

✓ Avaliando as expressões:

p_0 : digit(, 0) ; ... ; p_9 : digit(, 9).

Ground **AD**



Fonte: https://dtai.cs.kuleuven.be/stories/post/robin-manhaeve/deepproblog/

✓ $[p_0, \ldots, p_9]$: vetor de saída da rede *m_digit* quando avaliada em ;

- NN pode ser de qualquer arquitetura
- Requisito: saída precisa ser normalizada

DeepProbLog

- Semântica

✓ Além das disjunções anotadas, análogo aos fatos probabilísticos, temos também **fatos neurais:**

$$\texttt{nn}(\texttt{m}, [\texttt{X}, \texttt{Y}]) :: \texttt{similar}(\texttt{X}, \texttt{Y}).$$

✓ Neste exemplo, a rede neural m retorna o grau de similaridade das entradas X e Y (imagens).

✓ Instanciando com as entradas e :

$$\texttt{nn}(\texttt{m}, [\texttt{3}, \texttt{3}]) :: \texttt{similar}(\texttt{3}, \texttt{3}).$$

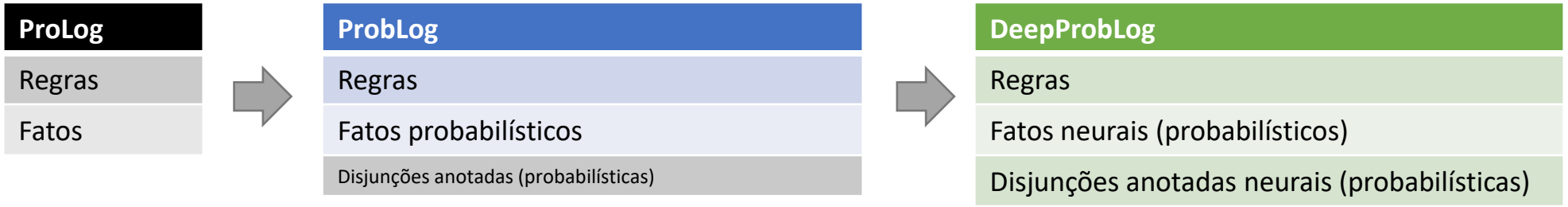
Ground neural fact

✓ A avaliação resulta em um fato probabilístico:

$$\texttt{p} :: \texttt{similar}(\texttt{3}, \texttt{3}).$$

DeepProbLog

- Semântica
  - ✓ Em resumo:



DeepProbLog

- Inferência
  - ✓ Exemplo: addition(0, 1, 1)

```
nn(m_digit, [X], Y, [0...9]) :: digit(X,Y).
addition(X,Y,Z) :- digit(X,N1), digit(Y,N2), Z is N1+N2.
```

(a) The DeepProbLog program.

```
nn(m_digit,[0],0)::digit(0,0);nn(m_digit,[0], 1)::digit(0,1).
nn(m_digit,[1],0)::digit(1,0);nn(m_digit,[1], 1)::digit(1,1).
addition(0,1,1) :- digit(0,0), digit(1,1).
addition(0,1,1) :- digit(0,1), digit(1,0).
```

(b) The ground DeepProbLog program.

```
0.8 :: digit(0,0); 0.1 :: digit(0,1).
0.2 :: digit(1,0); 0.6 :: digit(1,1).
addition(0,1,1) :- digit(0,0), digit(1,1).
addition(0,1,1) :- digit(0,1), digit(1,0).
```

(c) The ground ProbLog program.

✓ Note que: $Z = 1 \Rightarrow \begin{cases} N1, N2 \in \{0,1\} \\ (N1, N2) \in \{(0,1),\ (1,0)\} \end{cases}$

✓ À partir deste passo, igual ProbLog:

1. Cálculo da formula proposicional;
2. Compilação de SDD;
3. Criação e avaliação do AC.

DeepProbLog

- Aprendizado
  - ✓ Aprendizado de:
    - Parâmetros da rede neural;
    - Probabilidades (parâmetros probabilísticos);
  - ✓ Semelhante à abordagem **gradiente descendente** utilizado em ProbLog
  - ✓ **DeepProbLog**:
    - ✓ Saída das redes neurais são probabilidades
  - ✓ **ProbLog**:
    - ✓ Os parâmetros p_i são otimizados utilizando o gradiente (estrutura *semiring*) , o qual permite calcular $\frac{\partial P(q)}{\partial p_i}$
    - ✓ O gradiente então é usado para a atualização (gradiente descendente)

DeepProbLog

- Aprendizado
  - ✓ Exemplo:
    - Mesmo exemplo da adição de duas imagens MNIST;
    - Introdução de ruído: alguns dos *labels* são corrompidos e escolhidos aleatoriamente (distribuição uniforme);
    - Objetivo: aprender também a fração dos exemplares com ruído;

```
nn(classifier, [X], Y, [0 .. 9]) :: digit(X,Y).
t(0.2) :: noisy.

1/19 :: uniform(X,Y,0) ; ... ; 1/19 :: uniform(X,Y,18).

addition(X,Y,Z) :- noisy, uniform(X,Y,Z).
addition(X,Y,Z) :- \+noisy, digit(X,N1), digit(Y,N2), Z is N1+N2.
```

(a) The DeepProbLog program.

DeepProbLog

- Aprendizado
  - ✓ Instanciando à partir da *query* $\texttt{addition}(\texttt{a}, \texttt{b}, 1)$

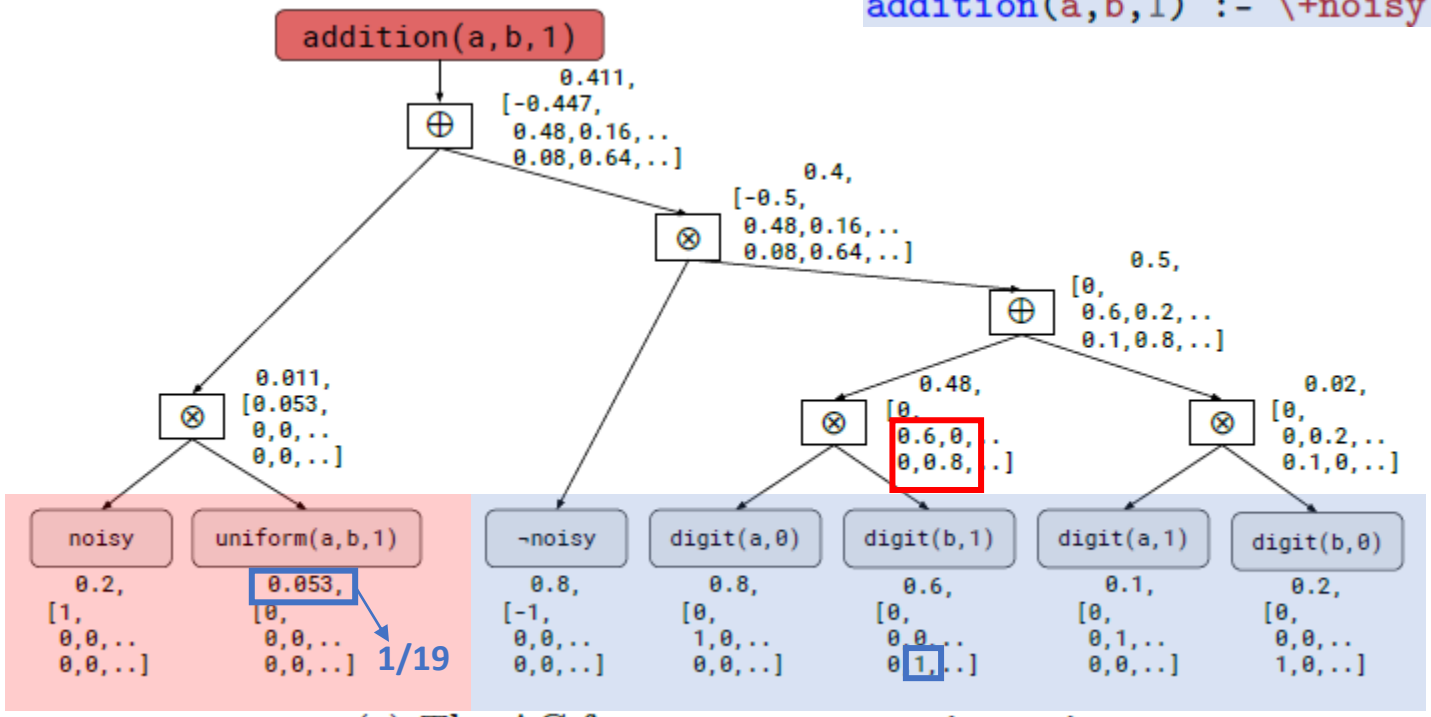
```
nn(classifier,[a],0) :: digit(a,0); nn(classifier,[a],1) :: digit(a,1).
nn(classifier,[b],0) :: digit(b,0); nn(classifier,[b],1) :: digit(b,1).
t(0.2)::noisy.

1/19::uniform(a,b,1).
addition(a,b,1) :- noisy, uniform(a,b,1).

addition(a,b,1) :- \+noisy, digit(a,0), digit(b,1).
addition(a,b,1) :- \+noisy, digit(a,1), digit(b,0).
```

(b) The ground DeepProbLog program.

DeepProbLog

- Aprendizado
  - ✓ Circuito aritmético (AC)

```
addition(a,b,1) :- noisy, uniform(a,b,1).
addition(a,b,1) :- \+noisy, digit(a,0), digit(b,1).
addition(a,b,1) :- \+noisy, digit(a,1), digit(b,0).
```



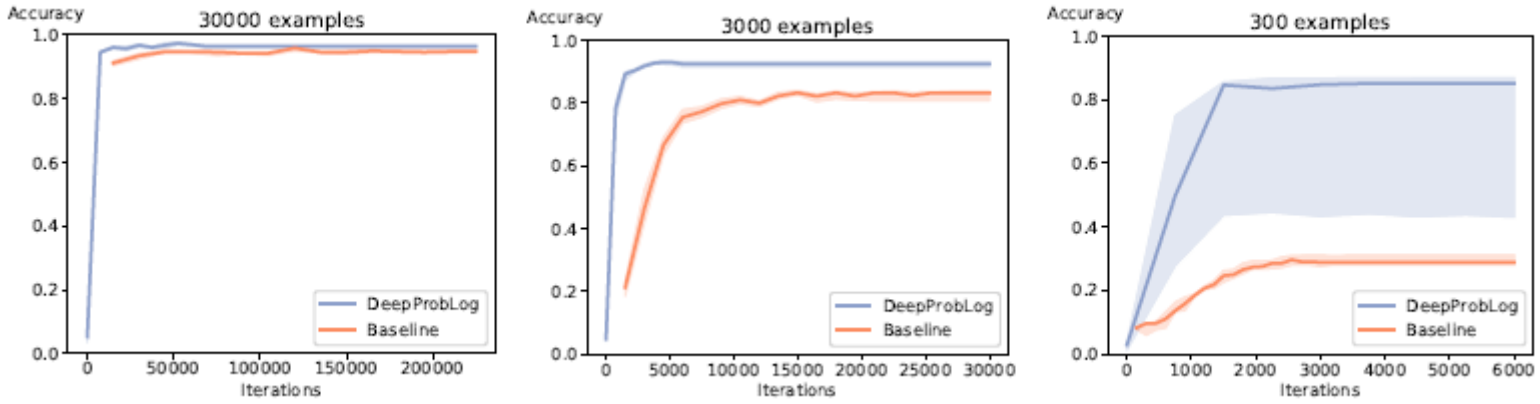
(c) The AC for query addition(a,b,1).

Legend

$$p, [\partial p/\partial p_{noisy}, \partial p/\partial p_{digit(a,0)}, \ldots, \partial p/\partial p_{digit(a,9)}, \partial p/\partial p_{digit(b,0)}, \ldots, \partial p/\partial p_{digit(b,9)}]$$

Experimentos

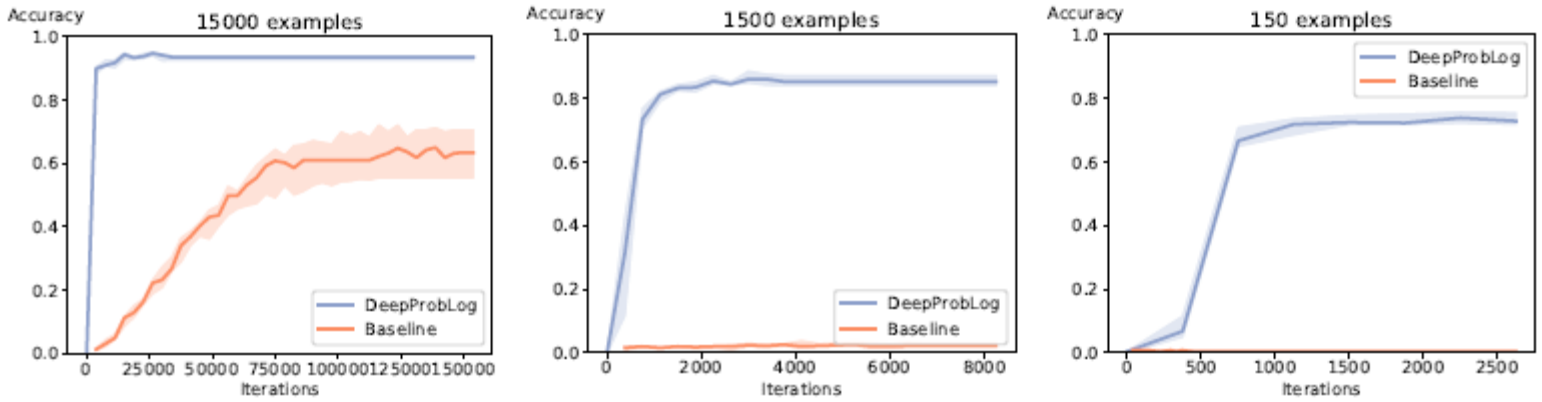
- Raciocínio Lógico + *Deep Learning*
  - **T1**: addition( ,  , 8)
    - ✓ Baseline: Rede neural convolucional (CNN)



	Number of training examples		
Model	30 000	3 000	300
Baseline	93.46 ± 0.49	78.32 ± 2.14	23.64 ± 1.75
DeepProbLog	97.20 ± 0.45	92.18 ± 1.57	67.19 ± 25.05

Experimentos

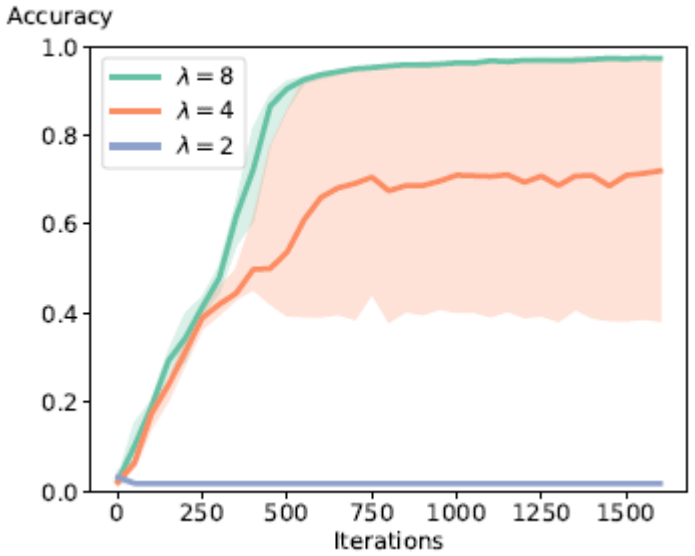
- Raciocínio Lógico + *Deep Learning*
  - **T2**: addition([3, 8], [2, 5], 63)



	Number of training examples			
Model	15 000	1 500	150	**T1** (30 000)
Baseline	60.85 ± 9.77	1.34 ± 0.53	0.80 ± 0.14	–
DeepProbLog	95.16 ± 1.70	87.21 ± 1.92	72.73 ± 3.03	93.36 ± 1.18

Experimentos

- Raciocínio Lógico + *Deep Learning*
  - **T3**: addition()
    - ✓ Necessidade de regularização a nível lógico para não ficar preso na solução trivial $0 + 0 = 0$;



Experimentos

- Raciocínio Lógico + *Deep Learning*
  - **T4**: addition( ,  , 14)
    - ✓ Tolerância ao ruído com ou sem modelagem explícita do mesmo;

	Fraction of noise					
	0.0	0.2	0.4	0.6	0.8	1.0
Baseline	93.46	87.85	82.49	52.67	8.79	5.87
DeepProbLog	97.20	95.78	94.50	92.90	46.42	0.88
DeepProbLog w/ explicit noise	96.64	95.96	95.58	94.12	73.22	2.92
Learned fraction of noise	0.000	0.212	0.415	0.618	0.803	0.985

Experimentos

- Programação Indutiva
  - ✓ *Program Sketching* [1] usando interpretado diferencial ∂4 [2]
    - **T5**: `forth_addition([4],[8],1,[1,3])`
      - ✓ Acurácia 100%, semelhantes ao ∂4;
    - **T6**: `forth_sort([8,2,4],[2,4,8])`
      - ✓ Melhor escalabilidade do DeepProbLog;

	Test length	Training length 2	3	4	5	6
∂4 [8]	8	100.0	100.0	49.22	–	–
	64	100.0	100.0	20.65	–	–
DeepProbLog	8	100.0	100.0	100.0	100.0	100.0
	64	100.0	100.0	100.0	100.0	100.0

	Training length 2	3	4	5	6
∂4 on GPU	42 s	160 s	–	–	–
∂4 on CPU	61 s	390 s	–	–	–
DeepProbLog	11 s	14 s	32 s	114 s	245 s

    - **T7**: `wap('Robert has 12 books . ... How many does he have now ?',12,3,1,10)`
      - ✓ Acurácia 96.5%, semelhantes ao ∂4;

[1] A. Solar-Lezama, Program sketching, International Journal on Software Tools for Technology Transfer 15 (2013) 475–495.
[2] M. Bošnjak, T. Rocktäschel, S. Riedel, Programming with a differentiable forth interpreter, in: Proceedings of the 34th International Conference on Machine Learning, volume 70, 2017, pp. 547–556.

Experimentos

- Programação Probabilística e Deep Learning

  - **T8**: Classificação de moedas e comparação
    - ✓ Conceito de “supervisão distante”
    - ✓ Entrada: imagem sintética de 2 moedas
      - Imagem pode mostrar cara ou coroa das moedas
      - Rótulos: {“*same*”, “*different*”}
    - ✓ Duas NN, uma para cada moeda
      - Predizem cara ou coroa
    - ✓ Duas tarefas:
      - Reconhecer e separar as duas moedas
      - Classificar cara/coroa
    - ✓ Questões:
      - Espera-se que as duas redes concordem sobre qual lado da moeda é cara e qual é coroa
      - Classificar cara/coroa

```
nn(net1, [X], Y, [heads, tails]) :: coin1(X,Y).
nn(net2, [X], Y, [heads, tails]) :: coin2(X,Y).

compare(X,X,same).
compare(X,Y,different) :- \+compare(X,Y,same).

coins(X,Comparison) :-
    coin1(X,C1),
    coin2(X,C2),
    compare(C1,C2,Comparison).
```

Experimentos

- Programação Probabilística e Deep Learning
  - **T8**: Classificação de moedas e comparação

Labeled examples	Not solved	Expected solution	Other solution
0	56%	11%	33%
5	39%	40%	21%
10	7%	92%	1%
20	4%	96%	0%
50	3%	97%	0%
100	4%	96%	0%

```
nn(net1, [X], Y, [heads, tails]) :: coin1(X,Y).
nn(net2, [X], Y, [heads, tails]) :: coin2(X,Y).

compare(X,X,same).
compare(X,Y,different) :- \+compare(X,Y,same).

coins(X,Comparison) :-
    coin1(X,C1),
    coin2(X,C2),
    compare(C1,C2,Comparison).
```

Experimentos

- Programação Probabilística e *Deep Learning*

  - **T9**: `0.8::poker([Q♡, Q♢, A♢, K♣],loss).`

    ✓ Jogo de Poker simplificado:
    - Apenas J, Q, K, A;
    - 2 jogadores com duas cartas;
    - 1 carta comunitária (não-observada);
    - Sem troca de cartas;
    - Mãos: par, trio, *straight*.

    ✓ **Entrada**: as 4 cartas distribuídas as jogadores;
    ✓ **Classes:** {*win, loss, draw*};
    ✓ **Objetivos**:
    1. Treinar a NN para reconhecer as 4 cartas;
    2. Raciocinar (probabilisticamente) sobre a carta não-observada;
    3. Aprender a distribuição da carta comunitária (não rotulada);

Experimentos

- Programação Probabilística e *Deep Learning*

✓ A fim de proporcionar convergência mais rápida, em 10% dos exemplares é adicionada a carta comunitária:

$$\texttt{poker}([\texttt{Q}\heartsuit, \texttt{Q}\diamondsuit, \texttt{A}\diamondsuit, \texttt{K}\clubsuit], \texttt{A}\diamondsuit, \texttt{loss}).$$

✓ Uma das vantagens de DeepProbLog: exemplares com diferentes graus de observabilidade;
✓ Função de custo: erro quadrático mínimo (MSE);

✓ **Resultados**:
- 10 experimentos, 6 convergiram.
- Os que convergiram foram capazes de aprender corretamente;
- Os que não convergiram: identificação incorreta das cartas;

Distribution	Jack	Queen	King	Ace
Actual	0.2	0.4	0.15	0.25
Learned	0.203 ± 0.002	0.396 ± 0.002	0.155 ± 0.003	0.246 ± 0.002

Table 8: The results for the Poker experiment (**T9**).

Trabalhos Correlatos

- Trabalhos Correlatos

  - 3 tipos de abordagens diferentes:
    - ✓ Lógica como regularização
      - Lógica incluída como um regularizador durante a otimização da NN;
    - ✓ *Templating Neural Networks*
      - Lógica utilizada como *template* para a construção de arquiteturas de NN;
    - ✓ *Neural Program Induction*
      - Objetivo: Aprender programas com base nos dados

  - Dfierenças com DeepProbLog:
    - ✓ Outras abordagens: tipicamente focam em codificar lógica nas redes neurais;
    - ✓ DeepProbLog: integra NN em um arcabouço lógico-probabilístico.

Conclusões

- **Conclusões do artigo**:
  - ✓ DeepProbLog estende ProbLog com predicados neurais;
  - ✓ Aprendizado é feito usando aProbLog para o cálculo do gradiente;
  - ✓ Experimentos demonstraram capacidade de combinar raciocínio simbólico e sub-simbólico, programação indutiva e programação lógica probabilística
  - ✓ Limitações: DeepProbLog utiliza apenas inferência exata
    - Baixa escalabilidade;
    - Impraticável para problemas maiores;
  - ✓ Trabalhos futuros: inferência aproximada.

- **Conclusões minhas**:
  - ✓ DeepProbLog é aplicável quando se consegue dividir de antemão o problema em duas partes: neural e lógico/probabilístico;
  - ✓ Trabalho não expõe com clareza a limitação atual da escalabilidade da inferência (apenas cita na conclusão)

PESC
Programa de Engenharia de Sistemas e Computação

Obrigado

Dúvidas?