Relatório da 3° Avaliação de Inteligência Artificial

Brenda Aguiar, Jackson Celestino, Levi Rodrigues, Thailsson Clementino

[1]Instituto de Computação – Universidade Federal do Amazonas (UFAM)
Av. General Rodrigo Octávio, 6200, Coroado I, CEP: 69080-900

{brenda.aguiar, jackson.fernandes, lrda, thailsson.clementino}@icomp.ufam.edu.br

1 Introdução

A inteligência artificial (IA) tornou-se uma ferramenta para muitas empresas que a utilizam para automatizar tarefas, garantir mais eficiência e velocidade aos seus serviços. Porém, a IA enfrenta diversos desafios, dentre eles a confiança e a aceitação do público em relação as suas respostas. Com o *boom* na área de *Machine Learning* e a ascensão de modelos de aprendizado profundo (*deep learning*), existe ainda outro obstáculo que faz com que o desafio de entender esses modelos sejam ainda maiores: estamos lidando com modelos conhecidos como *black box* (caixa preta)[1].

O conceito de *black box* refere-se a um sistema cujo interior não pode ser desvendado, cujos elementos internos são desconhecidos e que só pode ser conhecido "por fora", por manipulações externas ou de observação externa. Isso porque algumas ferramentas baseadas em inteligência artificial como Redes Neurais Artificiais (RNA) não demonstram como funcionam, ou seja, não há como saber explicitamente como e por que chegam a determinadas conclusões [2]. De acordo com Castelvecchi [3], decifrar a *black box* tem se tornado cada vez mais difícil e urgente, pois tanto a sua complexidade quanto suas aplicações aumentaram drasticamente nos últimos anos.

Uma rede neural artificial, de acordo com Haykin [4], se assemelha ao cérebro humano em conhecimento adquirido pela rede e a partir de seu ambiente por um processo de aprendizagem. Também, por forças de conexão entre neurônios, conhecidas como pesos sinápticos, utilizados para armazenar o conhecimento adquirido. O foco deste trabalho é usar a integração entre os conceitos de conexionistas entre os neurônios das redes neurais, com as representações simbólicas para criar uma solução para o problema do trem Michalski.

2 Referencial Teorico

2.1 Sistemas Neuro Simbólicos

O aprendizado de máquina é usualmente estudado por abordagens estatísticas, experimentais, enquanto o raciocínio em IA é estudado por abordagens baseadas em lógica. Uma alternativa dentre essas duas seria desenvolver modelos integrados, híbridos, por exemplo, modelos neuro-simbólicos. A computação neuro-simbólica explora os benefícios de cada paradigma: aprendizado (neural) e raciocínio simbólico (lógico) [5].

Nesta abordagem, o conhecimento é representado por uma linguagem lógica simbólica, enquanto a computação é realizada por algoritmos conexionistas, por redes neurais. Por exemplo, a Lógica Modal Conexionista (Connectionist Modal Logic - CML), descrita por Garcez et al. [6], que considera as redes neurais como redes de mundos possíveis. A integração entre os sistemas simbólicos e conexionistas é buscada principalmente devido aos seguintes fatores:

(i) os sistemas simbólicos possuem algumas limitações no que diz respeito a manipulação de informações quantitativas e de dados imprecisos e/ou inexatos;
(ii) os sistemas simbólicos permitem uma representação de conhecimento que pode ser facilmente interpretada por um especialista da área;
(iii) os sistemas conexionistas tratam muito bem as informações quantitativas e permitem que se trabalhe com relativa facilidade com dados aproximados e/ou incorretos;
(iv) os sistemas conexionistas possuem uma representação de conhecimentos intrínseca as redes neurais que são de difícil interpretação através de métodos manuais convencionais;

Em razão dos fatores levantados acima, muitos pesquisadores buscam a união destas duas abordagens através de um modo de integração do tipo co-processamento [7]. Neste modo de integração, ambos os módulos, simbólico e conexionista, trabalham em paralelo, e os conhecimentos adquiridos sobre os problemas são transferidos de um módulo ao outro por métodos de: inserção de conhecimentos simbólicos em redes neurais [8] [9] [10] e de extração de conhecimentos simbólicos a partir de redes neurais [11] [12] [13].

A aprendizagem neuro simbólica pode ser separada em 3 partes:

- Refinamento de conhecimento na rede neural;
- Extração de conhecimento de uma rede neural treinada;
- Revisão da teoria da rede neural.

O ciclo de aprendizagem neuro-simbólica é representado pela figura 3, no qual as fases representadas podem ser descritas como: (1) inserção de conhecimento simbólico; (2) aprendizado indutivo com exemplos; (3) dedução maciçamente paralela; (4) *fine-tuning*; (5) extração de conhecimento simbólico; e (6) *feedback*.

3 Descrição do problema

No problema conhecido como "O trem de Michalski", a meta é classificar quais os trens que vão para leste e os que vão para oeste. A figura 2 ilustra todos os 10 exemplos utilizados neste trabalho dos trens que vão para o leste e os

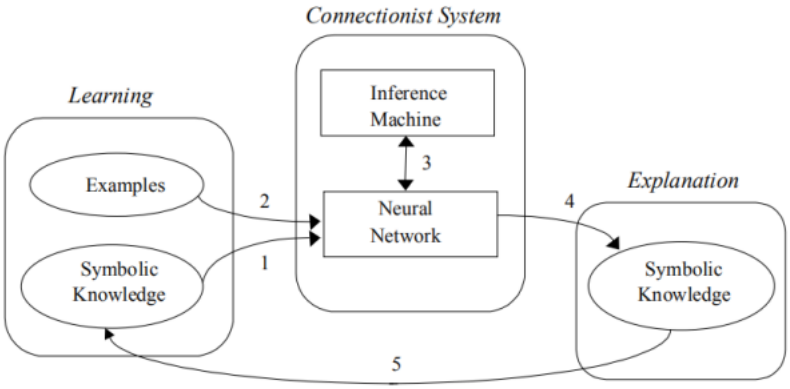

Figura 1: Ciclo de aprendizagem

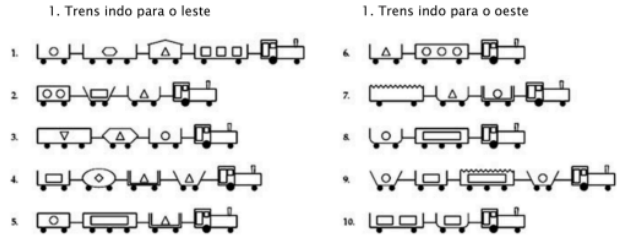

Figura 2: Exemplos dos trens de Michalski

que vão para oeste. Neste problema, cada trem é descrito por meio dos seguintes atributos:

1. Quantidade de vagões (valor entre 3 a 5);
2. Quantidade de cargas diferentes que pode levar (valor entre 1 a 4);
3. Para cada vagão de um trem:
   (a) a quantidade de eixo com rodas (valor entre 2 e 3);
   (b) o comprimento (valor curto ou longo);
   (c) o formato da carroceria do vagão, e pode ser:
      i. retângulo-fechado;
      ii. retângulo-aberto
      iii. duplo retângulo-aberto
      iv. elipse
      v. locomotiva
      vi. hexágono
      vii. topo dentado
      viii. trapézio aberto
      ix. topo triangular-fechado
   (d) Quantidade de cargas no vagão (0 a 3);
   (e) O formato da carga (círculo, hexágono, retângulo ou triângulo);

Para utilização durante o trabalho foram propostas, 10 variáveis *booleanas* (proposicionais) que descrevem se qualquer par de tipos de carga estão ou não em vagões adjacentes do trem (já que cada carro carrega um único tipo de carga). Temos as seguintes relações com respeito aos vagões de um trem, cujo valor lógico varia entre -1 (Falso) e 1 (Verdadeiro):

1. existe um retângulo próximo de um retângulo (V ou F)
2. existe um retângulo próximo de um triângulo (V ou F)
3. existe um retângulo próximo de um hexágono (V ou F)
4. existe um retângulo próximo de um círculo (V ou F)
5. existe um triângulo próximo de um triângulo (V ou F)
6. existe um triângulo próximo de um hexágono (V ou F)
7. existe um triângulo próximo de um círculo (V ou F)
8. existe um hexágono próximo de um hexágono (V ou F)
9. existe um hexágono próximo de um círculo (V ou F)
10. existe um círculo próximo de um círculo (V ou F)

Há um único atributo de classe que define a direção de um trem: leste ou oeste. Observe que para atributos com múltiplos valores deve-se assinalar valores numéricos na ordem em que surgem, por exemplo, o tipo de carga deve ser 1 para denotar círculo, 2 para hexágono, 3 para retângulo, e assim por diante. Os neurônios correspondentes devem usar função de ativação linear, i.e. $h(x) = x$.

4 Descrição do Modelo

Os modelos implementados neste trabalho baseiam-se na seção 10.6 do livro *Neural-symbolic cognitive reasoning* [5]. As implementações e resultados foram feitos em Python, utilizando a biblioteca Keras, o repositório com as implementações podem ser vistas em [14]. Esta seção está dividida em duas partes, na qual a primeira comenta sobre a implementação de uma *Perceptron multicamadas* e a segunda subseção descreve a implementação de uma rede mais profunda com 11 sub-redes que representam regras específicas.

4.1 Implementação 1

Para a primeira implementação foi utilizada uma Perceptron multicamadas com 32 neurônios de entrada, uma camada escondida com 9 neurônios e a camada de saída contendo 1 neurônio. A camada escondida utiliza a função de

ativação *ReLu*, enquanto a camada de saída utiliza a função de ativação *Sigmoid*. Portanto, a saída da rede está no intervalo $[0, 1]$, no qual 1 representa leste e 0 oeste.

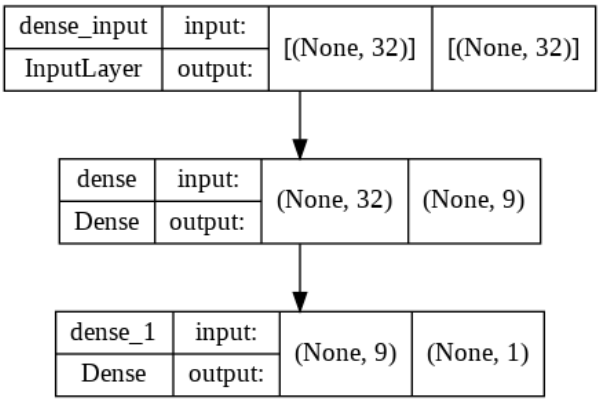

Figura 3: Arquitetura MLP

Para o treinamento do modelo foi utilizado a estratégia *leaving-one- out cross-validation* na qual dos 10 exemplos disponíveis, 1 é retirado para teste enquanto os outros 9 exemplos são utilizados em treino. Foi utilizado o optimizador SGD, com o *learning rate* 0.01, e a função de perda utilizada no treinamento foi a *Binary Cross Entropy*. Em cada um dos 10 processos de treinamento foram executados para 100 épocas e os resultados são descritos na seção 5.

4.2 Implementação 2

A meta-rede consiste em uma série de redes neurais representando proposições e uma última camada que consome as proposições para realizar a classificação. Foram modelados 11 redes, uma para cada um dos seguintes conceitos (conforme definição das páginas 136 e 137 do livro [5]) cada uma representando as seguintes regras:

- $num\_cars(t, nc)$, em que $t \in$ [1..10] e $nc \in$ [3..5].
- $num\_loads(t, nl)$ em que $t \in$ [1..10] e $nl \in$[1..4].
- $num\_wheels(t, c, w)$ em que $t \in$ [1..10] e $c \in$ [1..4] e $w \in$ [2..3].
- $length(t, c, l)$ em que $t \in$ [1..10] e $c \in$ [1..4] e $l \in$ [-1..1] (-1 denota curto e 1 longo)
- $shape(t, c, s)$ em que $t \in$ [1..10] e $c \in$ [1..4] e $s \in$ [1..10] (um número para cada forma).
- $num\_cars\_loads(t, c, ncl)$ em que $t \in$ [1..10] e $c \in$ [1..4] e $ncl \in$ [0..3].
- $load\_shape(t, c, ls)$ em que $t \in$ [1..10] e $c \in$ [1..4] e $ls \in$ [1..4].
- $next\_crc(t, c, x)$ em que $t \in$ [1..10] e $c \in$ [1..4] e $x \in$ [-1..1], em que o vagão c do trem t tem um vagão adjacente com cargas em círculo.
- $next\_hex(t, c, x)$ em que $t \in$ [1..10] e $c \in$ [1..4] e $x \in$ [-1..1], em que o vagão c do trem t tem um vagão adjacente com cargas em hexágono.
- $next\_rec(t, c, x)$ em que $t \in$ [1..10] e $c \in$ [1..4] e $x \in$ [-1..1], em que o vagão c do trem t tem um vagão adjacente com cargas em retângulo.
- $next\_tri(t, c, x)$ em que $t \in$ [1..10] e $c \in$ [1..4] e $x \in$ [-1..1], em que o vagão c do trem t tem um vagão adjacente com cargas em triângulo

A meta-rede foi treinada com 36 exemplos e validada com os 4 exemplos restantes. A última rede que representa a regra $est \in$ 0,1, que classifica se cada vagão é *eastbound(1)* ou *weastbound(0)*, foi conectada as camadas ocultas das 11 redes. Dessa forma, a meta-rede foi treinada com os métodos de: *leaving-one-out* e *cross-validation* com a relação a cada trem. Os dados com 10 instâncias foram expandidos para 40 já que cada trem pode ter até 4 vagões.

5 Resultados e Analises

5.1 Resultados da implementação 1

A Tabela 1 apresenta os resultados para o nosso primeiro modelo descrito. O modelo conseguiu acertar 7 dos 10 exemplos de testados. Nos quais os exemplos 0 e 8 são os que mais o modelo erra com convicção. Apesar de uma pequena quantidade de exemplos no treino, o modelo consegue generalizar bem o suficiente para acertar os exemplos não vistos no treino.

trem	saída contínua	saída discreta	saída esperada
0	0.01	0	1.0
1	0.70	1	1.0
2	0.95	1	1.0
3	0.85	1	1.0
4	0.87	1	1.0
5	0.26	0	0.0
6	0.41	0	0.0
7	0.07	0	0.0
8	0.99	1	0.0
9	0.78	1	0.0

Tabela 1: Resultados para o treinamento com Perceptron multicamadas 1

5.2 Resultados da implementação 2

A Tabela 2 mostra os resultados obtidos para cada teste da validação cruzada. Pelos dados da tabela, o único erro aconteceu na predição da direção do trem 7. Mesmo aumento o número de épocas, sempre havia alguma predição errada, diferenciando apenas o trem. Quando se observa o exemplo com predição incorreta, nota-se que sua predição ficou bem próxima de 0.54, o que pode ser visto como uma indecisão do modelo. Para os outros exemplos, houve bem mais certeza de qual o valor deveria ser predito.

A fim de acompanhar a evolução das métricas ao longo do treino, geraram-se figuras que mostram o desempenho do modelo ao longo das épocas. A Figura 4 mostra o processo de aprendizado de cada meta-rede durante as épocas. Como é possível notar, o erro médio absoluto de várias meta-redes de validação e teste começou a decair a partir de 200 épocas. No entanto, outras mantiveram o mesmo erro um aumento.

Já a Figura 5 mostra o processo de treino considerando-se acurácia binária. Assim como aconteceu

trem	saída contínua	saída discreta	saída esperada
0	1.00	1	1.0
1	1.00	1	1.0
2	1.00	1	1.0
3	1.00	1	1.0
4	0.83	1	1.0
5	0.37	0	0.0
6	0.00	0	0.0
7	0.54	1	0.0
8	0.00	0	0.0
9	0.00	0	0.0

Tabela 2: Resultados para o treinamento com meta-redes

com a Figura 4, várias meta-redes alcançaram acurácia máxima quando chegaram próximo a 200, enquanto outras mantiveram uma acurácia constante. Nesse caso, as quedas de acurácia aconteciam principalmente após 200 épocas, cujos valores eram instáveis para algumas meta-redes.

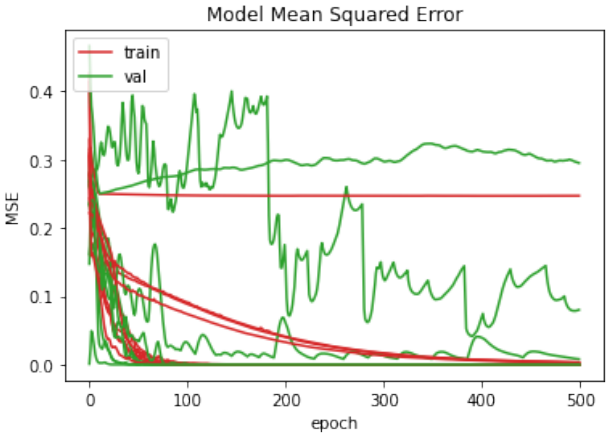

Figura 4: Erro médio absoluto para os testes

5.3 Comparação

Comparando os modelos podemos observar que o modelo com meta-redes consegue se sair melhor para os exemplos dados, o modelo consegue acertar 9 de 10 exemplos parecidos com o que foi mostrado no experimento original [5]. O melhor desempenho desse tipo de rede pode ser explicado pelo fato de a rede conseguir entender melhor a relação entre os atributos dado como entrada. Uma vez que as meta redes já atuam com regras bem definidas, a rede então atua tenta aprender a relação entre essas regras.

Outro ponto que pode ser mencionado, é que a segunda rede contém muito mais parâmetros a serem treinados do que o primeiro modelo, isso por si só contribui para esse resultado. Estudos com mais casos de teste poderiam ser conduzidos para que possivelmente temos respostas mais concretas sobre desempenho.

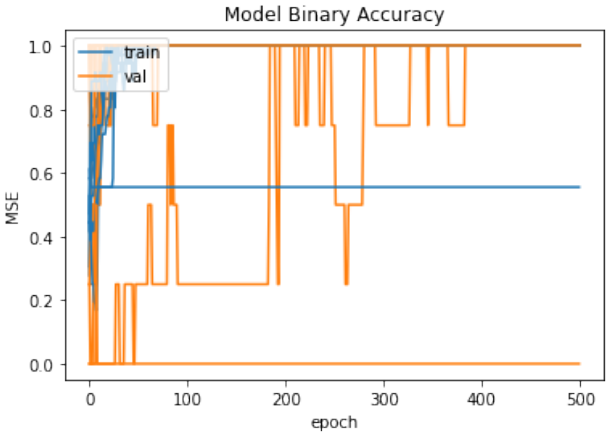

Figura 5: Acurácia binária do modelo para os testes

Referências

[1] Marketing. Black box problem: que desafio em inteligência artificial é esse? Disponível em `https://www.supero.com.br/blog/black-box-problem-um-novo-desafio-para-a-inteligencia-artificial/`, 2021. Acesso em 26 de Agosto de 2022.

[2] Philip K Maini. Mathematical models in morphogenesis. *Mathematics inspired by biology*, pages 151–189, 1999.

[3] Davide Castelvecchi. Can we open the black box of ai? Disponível em `https://www.nature.com/news/can-we-open-the-black-box-of-ai-1.20731`, 2016. Acesso em 28 de Agosto de 2022.

[4] Simon Haykin. *Redes neurais: princípios e prática*. Bookman Editora, 2001.

[5] Artur S d'Avila Garcez, Luís C Lamb, and Dov M Gabbay. Neural-symbolic learning systems. *Neural-Symbolic Cognitive Reasoning*, pages 35–54, 2009.

[6] Artur S d'Avila Garcez, Luis C Lamb, and Dov M Gabbay. Connectionist modal logic: Representing modalities in neural networks. *Theoretical Computer Science*, 371(1-2):34–53, 2007.

[7] Melanie Hilario. An overview of strategies for neurosymbolic integration. *Connectionist-Symbolic Integration: From Unified to Hybrid Approaches*, pages 13–36, 1997.

[8] Ian Cloete and Jacek M Zurada. *Knowledge-based neurocomputing*. MIT press, 2000.

[9] Fernando Santos Osório. *INSS: un système hybride neuro-symbolique pour l'apprentissage automatique constructif*. PhD thesis, Institut National Polytechnique de Grenoble-INPG, 1998.

[10] Geoffrey G Towell and Jude W Shavlik. Knowledge-based artificial neural networks. *Artificial intelligence*, 70(1-2):119–165, 1994.

[11] Adelmo Luis Cechin. *The extraction of fuzzy rules from neural networks*. Shaker, 1998.

[12] FS OSORIO, Bernard AMY, and Loïc DECLOEDT. Ruleout: Um novo método de extração incremental de regras à partir de redes neurais construtivas do tipo kbann. *Anais do V Simpósio Brasileiro de Redes Neurais*, pages 132–137.

[13] Robert Andrews, Joachim Diederich, and Alan B Tickle. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-based systems*, 8(6):373–389, 1995.

[14] Jackson Fernandes. Trabalho final de ia. `https://github.com/Jacksonfern/trabalho-final-IA`, 2022.