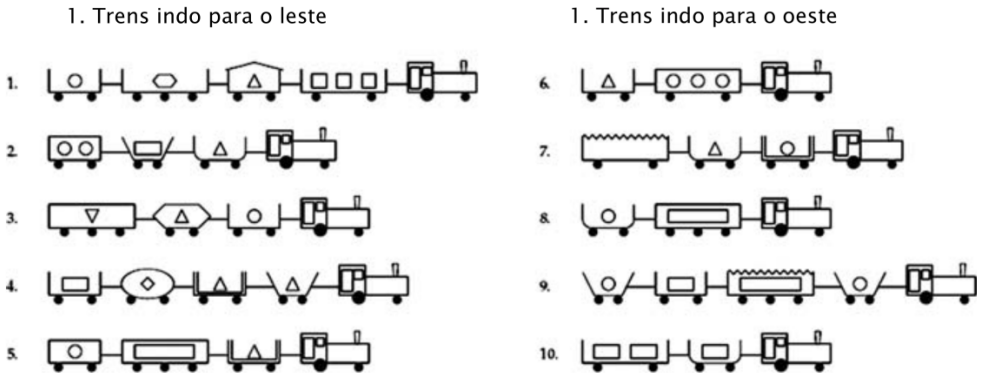
Poder Executivo
Ministério da Educação
Universidade Federal do Amazonas
Instituto de Computação - IComp

UFAM

Disciplina:	**Inteligência Artificial**	**Trabalho Final**	**Data: 18/12/24**
Professors:	Edjard Mota	Turma: EC01 & CB500	

Example do trem de Michalski. A meta é classificar quais os trens que vão para leste e os que vão para oeste.



Para cada trem temos os 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
      1.retângulo-fechado, 2. retângulo-aberto 3. duplo retângulo-aberto 4. elipse 5. 6. locomotiva. hexágono 7.topo dentado 8.trapézio aberto 9.topo triangular-fechado
   d) quantidade de cargas no vagão (0 a 3)
   e) o formato da carga (círculo, hexágono, retângulo ou triângulo)

Da Figura acima, 10 variáveis booleanos (proposicionais) 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 é **F**(0) ou 1 **V** (1).

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 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. $h(x) = x$.

Questão 1.

Antes de rodar seu experimento, verifique se o data set corresponde aos 100 trens do problem. Deppoois

a. **Agrupar trens por similaridades:** Rode um algoritmo de *clustering* para agrupar trens com caracteristicas similares independente da direção.
b. **Gerar Supostos Axiomas**: tilize os clusters encontrados e utilize o programa **`split_dataset.py`** para encontrar **similaredes entre atributos (não necessariamente igualdade)** entre os trens para propor regras logicas (Axiomas) que expliquem o padrão dos trens que vão para ambas as direções, em cada cluster. (Veja questão 3 como exemplo).
Nota: Os custers gerados não precisam agrupar por direção da viagem. Utilize como ponto de partida que os axiomas devem conter os predicados como os propostos no problema original, e **adicione mais predicados**:

1. *num_cars*(t, nc), em que $t \in [1..10]$ e $nc \in [3..5]$.
2. *num_loads*(t, nl) em que $t \in [1..10]$ e $nl \in [1..4]$.
3. *num_wheels*(t, c, w) em que $t \in [1..10]$ e $c \in [1..4]$ e $w \in [2..3]$.
4. *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)
5. *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).
6. *num_cars_loads*(t, c, ncl) em que $t \in [1..10]$ e $c \in [1..4]$ e $ncl \in [0..3]$.
7. *load_shape*(t, c, ls) em que $t \in [1..10]$ e $c \in [1..4]$ e $ls \in [1..4]$.
8. *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.
9. *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.
10. *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.
11. *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

c. Produza um documento no proprio github com estrutura similar aos tutoriais do LTNTorch. E descreva as regras como axiomas a seres testados nas questões seguintes.

Questão 2.
Implemente uma solução em LTNTorch com base nos axiomas da questão 1, e treine seu modelos para aprender a classificar os trens indo para leste (ou oeste caso contrário).
Crie um modelo LTN para classificar os trens no dataset

d. Separe 30% dos trens para fazer os teste e utilize os 70% (metade indo para o leste e metade para o oeste) para treinar seu modelo (metade indo para o leste e metade para o oeste)
e. Treine seu modelo e teste com os 30%. Mostre os graficos e tabelas acuracia similar ao problema original como mostrado no livro do Garcez

Questão 3. Compare seus resultados do seu modelo da com o do livro questão (2 no caso), e

Defina como seria possível extrair, do seu modelo. a seguinte regra genérica que classifica trens indo para o leste (east), apenas descreva como seria a extração baseado no algoritmo de extração visto em sala (e no material de classe):

1. **car(T,C)∧short(C)∧closed_top(C) → east(T) (no caso dos 10 trens)**

2. Existe alguma regra similar para os 100 trens? Se não, quantas possíveis deveria seu modelo ser capaz de extrair?

i. **Verifique se sua solução classifica, de forma aproximada, de acordo comas seguintes teorias**
   A. ***Se um trem tem um vagão curto e fechado, então ele vai para o leste, caso contrário, vai para o oeste (note que isto é uma descrição textual da regra lógica da questão 2.b)***
   B. ***Se um trem tem dois vagões, ou tem um vagão com teto irregular, então ele vai para o oeste, caso contrário, vai para o leste***
   C. ***Se um trem tiver mais de dois tipos diferentes de carga, então ele vai para o leste, caso contrário, vai para o oeste***