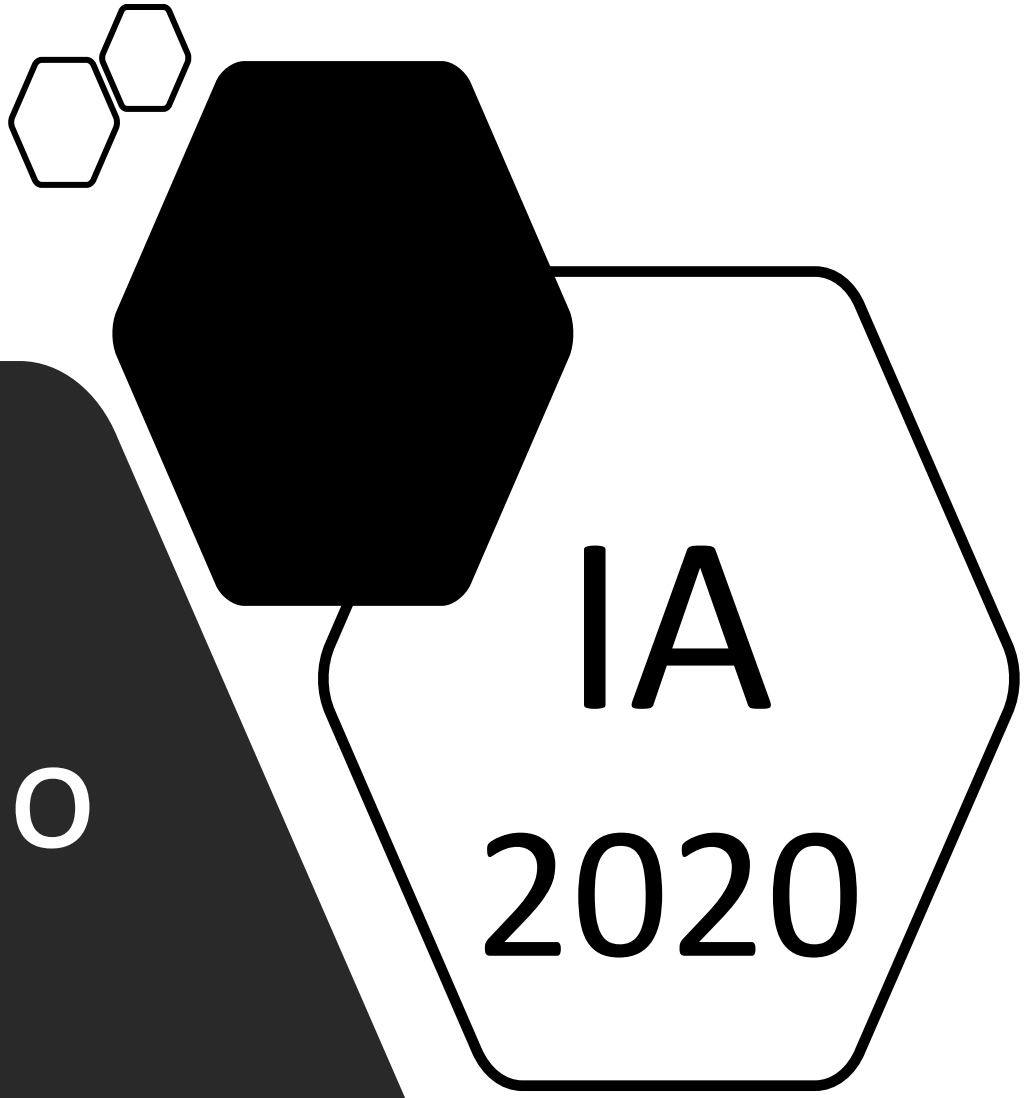
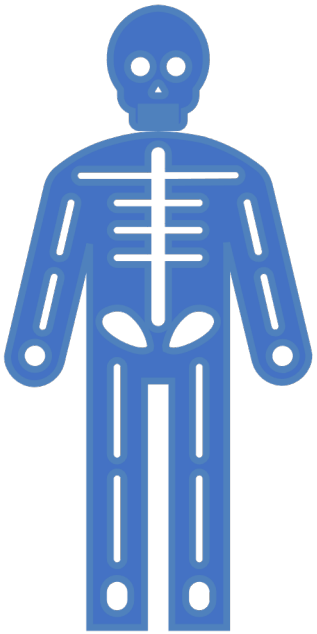
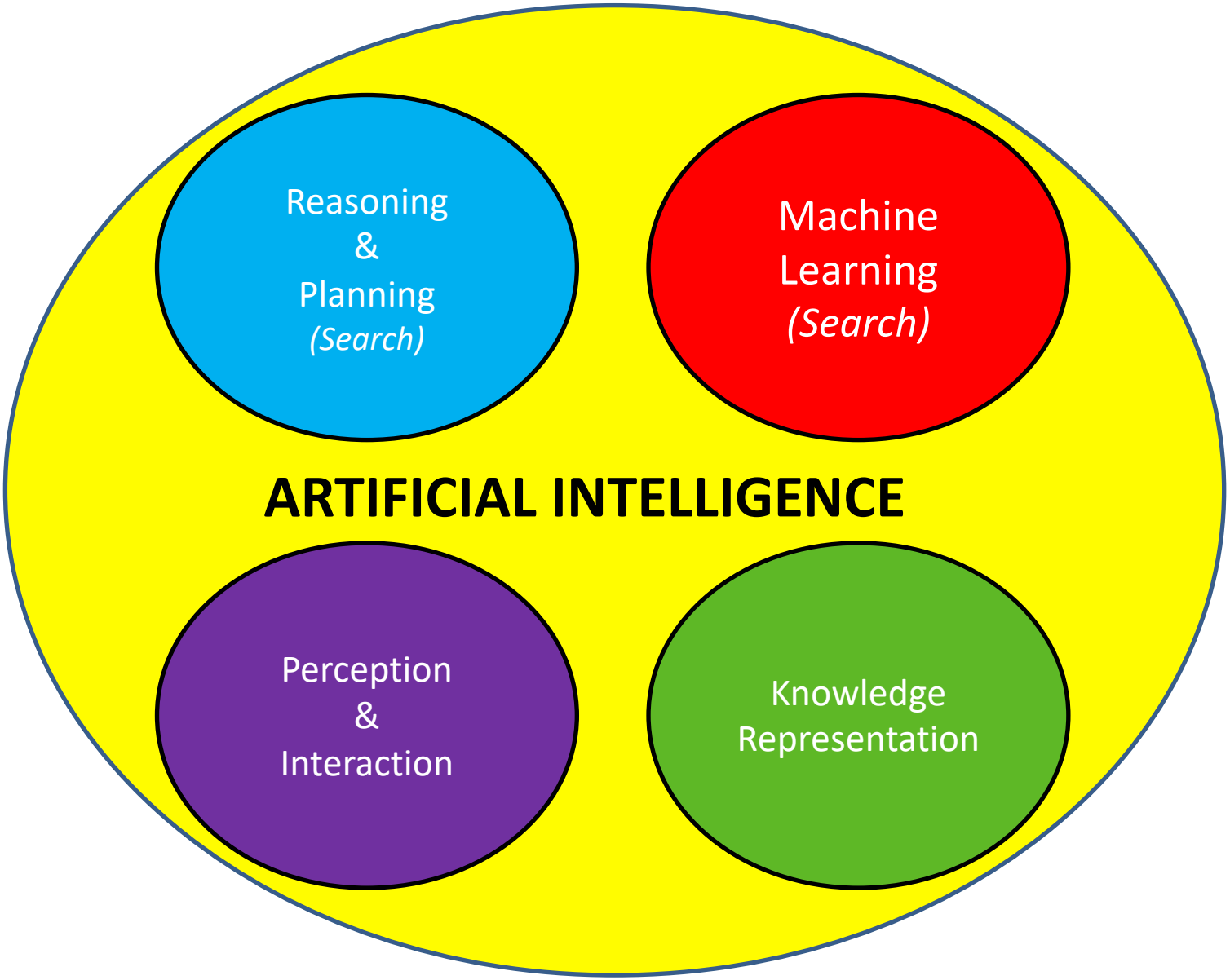
Resumo do Curso





Arquiteturas de Agentes Inteligentes


Reasoning
&
Planning
(Search)
Machine
Learning
(Search)
ARTIFICIAL INTELLIGENCE
Perception
&
Interaction
Knowledge
Representation

Buscas não informadas

- Busca em Largura
- Busca de Custo Uniforme
- Busca em Profundidade
- Busca em Profundidade Limitada
- Busca em Profundidade com Aprofundamento Iterativo
- Busca Bidirecional
- Busca com Conhecimento Incompleto

Busca Informada

BME Gulosa

$f(n) = h(n)$

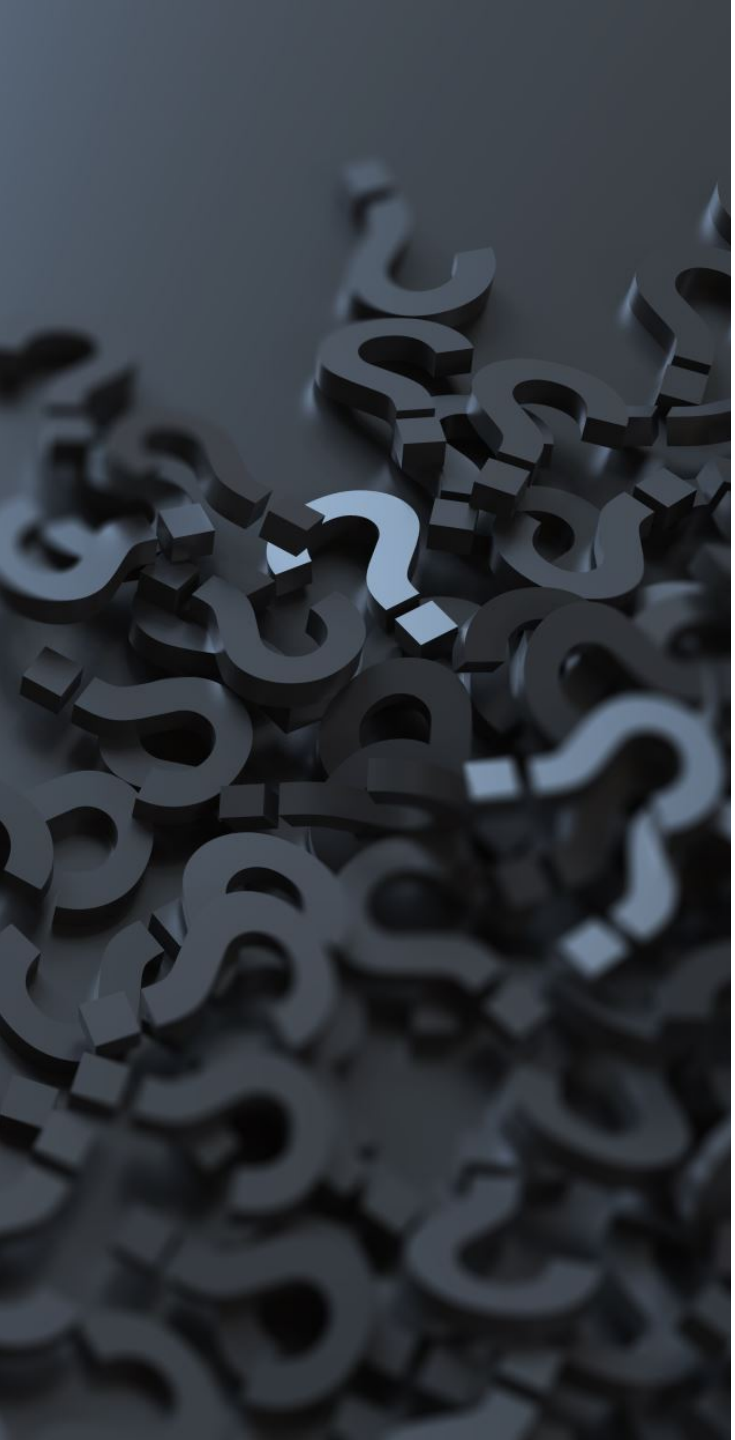
A*

$f(n) = c(n) + h(n)$

Heurísticas

Busca Local

- Hill-Climbing: Subida pela encosta mais íngreme ou Busca Local Gulosa
- Simulated Annealing: Têmpera Simulada
- Local beam search: Busca em feixe local
- Algoritmos genéticos (GA)

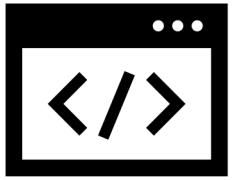


Representação de Conhecimento

- Lógica proposicional
- Lógica de primeira ordem
  - OWL
  - Inferência
  - Ontologias

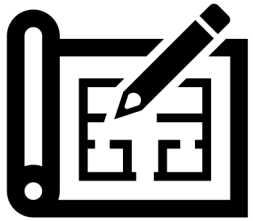
Planejamento: STRIPS & POP

- Criar uma linguagem especializada

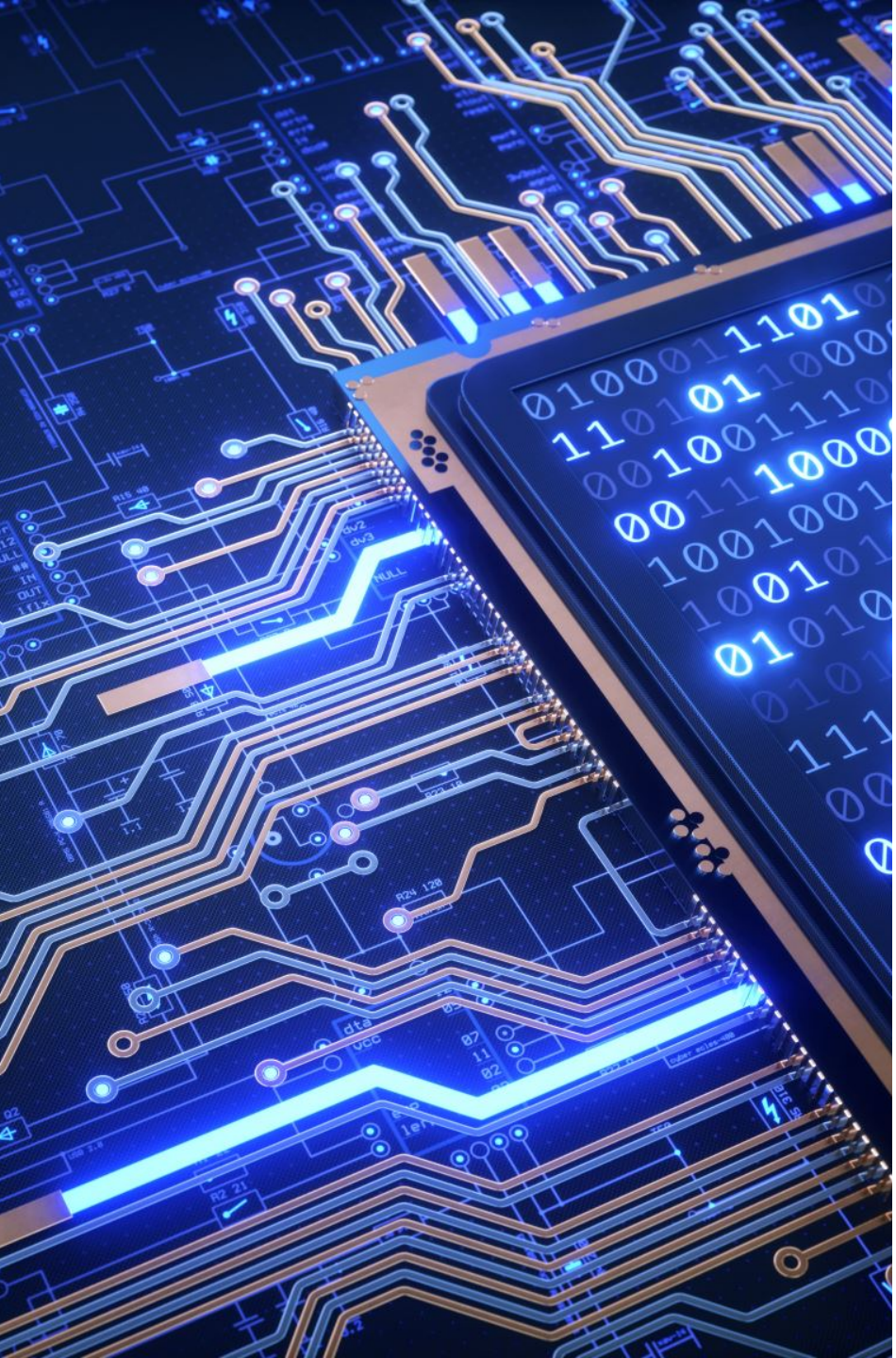


STRIPS: Stanford Research Institute Problem Solver (Fikes e Nilsson, 1971)

- Criar um algoritmo para planejar



POP: Partial Order Planning



Problemas de Satisfação de Restrições

Inteligência Artificial

PCS3438

CSP

- Um CSP consiste em:
  - um **conjunto de variáveis** que podem assumir valores dentro de um dado domínio
  - um **conjunto de restrições** que especificam propriedades da solução – valores que as variáveis podem assumir.

CSP: formulação

- **Estados:** definidos pelos ***valores*** possíveis das variáveis
- **Estado inicial:** nenhuma variável instanciada
- **Operadores:** atribuem valores (instanciação) às variáveis
  - **Uma variável por vez**
- **Teste de término:** quando todas as variáveis estão instanciadas obedecendo as restrições do problema
- **Solução:** conjunto dos valores das variáveis instanciadas
- **Custo de caminho:** número de passos de atribuição

Busca Cega com Retrocesso para CSP

- **Funcionamento**
  - estado inicial: variáveis sem atribuição
  - aplica operador: instancia **uma** variável
  - teste de parada: todas variáveis instanciadas **sem** violações
- **Retrocesso (*Backtracking*)**
  - depois de realizar uma atribuição, verifica se restrições não são violadas
  - caso haja violação $\Rightarrow$ **retrocede**
- **Análise**
  - pode ser **busca em profundidade limitada** (l = número de variáveis)
  - é completa
  - fator de ramificação máxima: max(|Di|)
  - o teste de parada é decomposto em um conjunto de restrições sobre as variáveis

Verificação Prévia

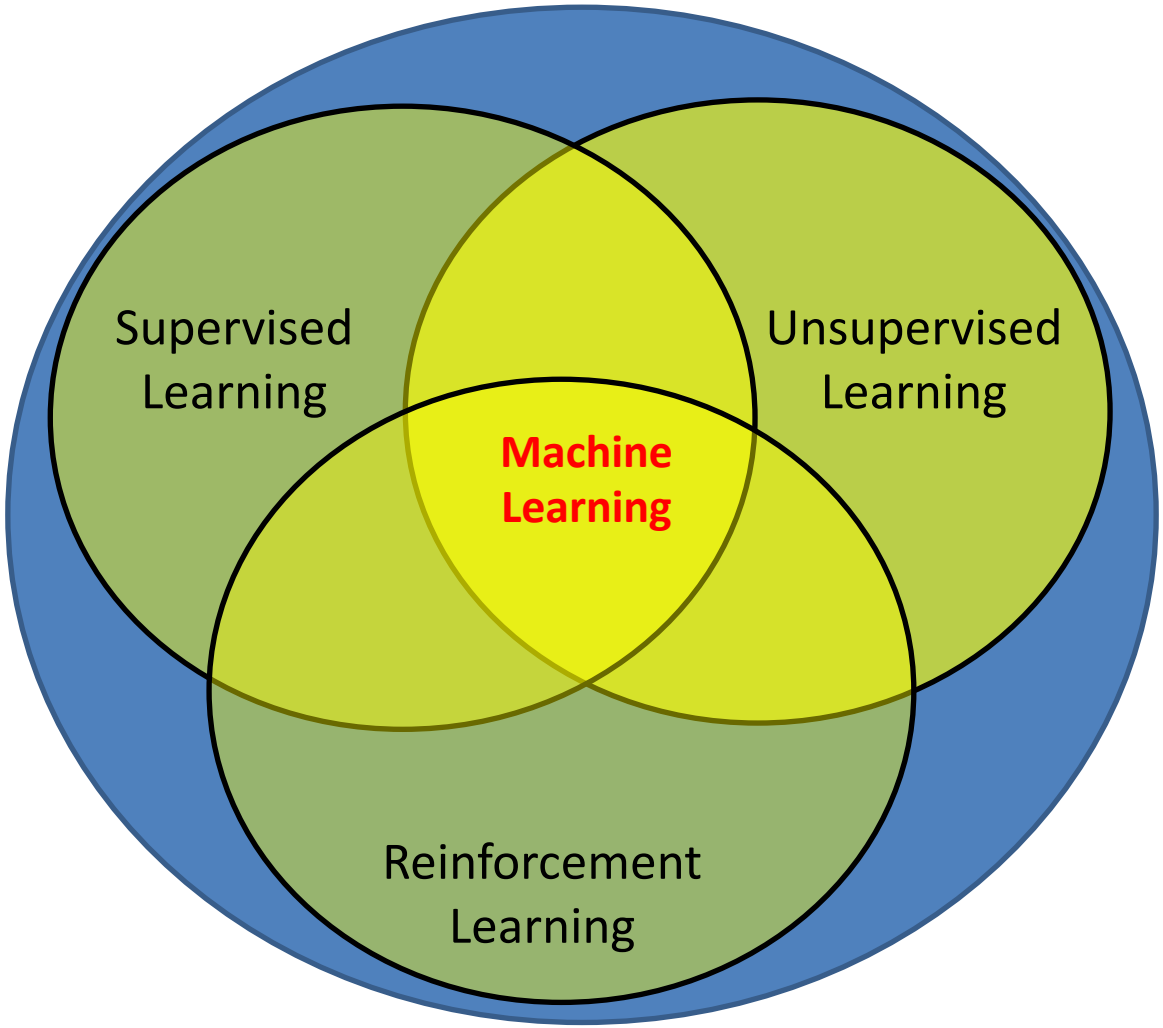
- Verificação prévia (*forward checking*)
  - **ideia**: olhar para frente para detectar situações insolúveis
- Algoritmo:
  - Após cada atribuição, **eliminar do domínio** das variáveis não instanciadas os valores incompatíveis com as atribuições feitas até agora.
  - Se um domínio torna-se vazio, retrocede imediatamente.
- É bem mais eficiente!

Propagação de restrições

- Propagação de restrições (*constraint propagation*)
  - uma consequência da verificação prévia
  - quando um valor é eliminado, isto é **propagado** para outros valores que dele dependem, podendo torná-los inconsistentes e eliminados também
    - é como uma onda que se propaga: as escolhas ficam cada vez mais restritas

Heurísticas para CSP

- Existem 3 heurísticas para isto...
  - ***Variável mais restritiva***: variável envolvida no maior número de restrições é preferida (verifica restrições)
  - ***Variável mais restringida***: variável que pode assumir menos valores é preferida (verifica os domínios das variáveis)
  - ***Valor menos restritivo***: valor que deixa mais liberdade para futuras escolhas (verifica os valores dos domínios e as restrições)



Learning Agent:

What feedback?

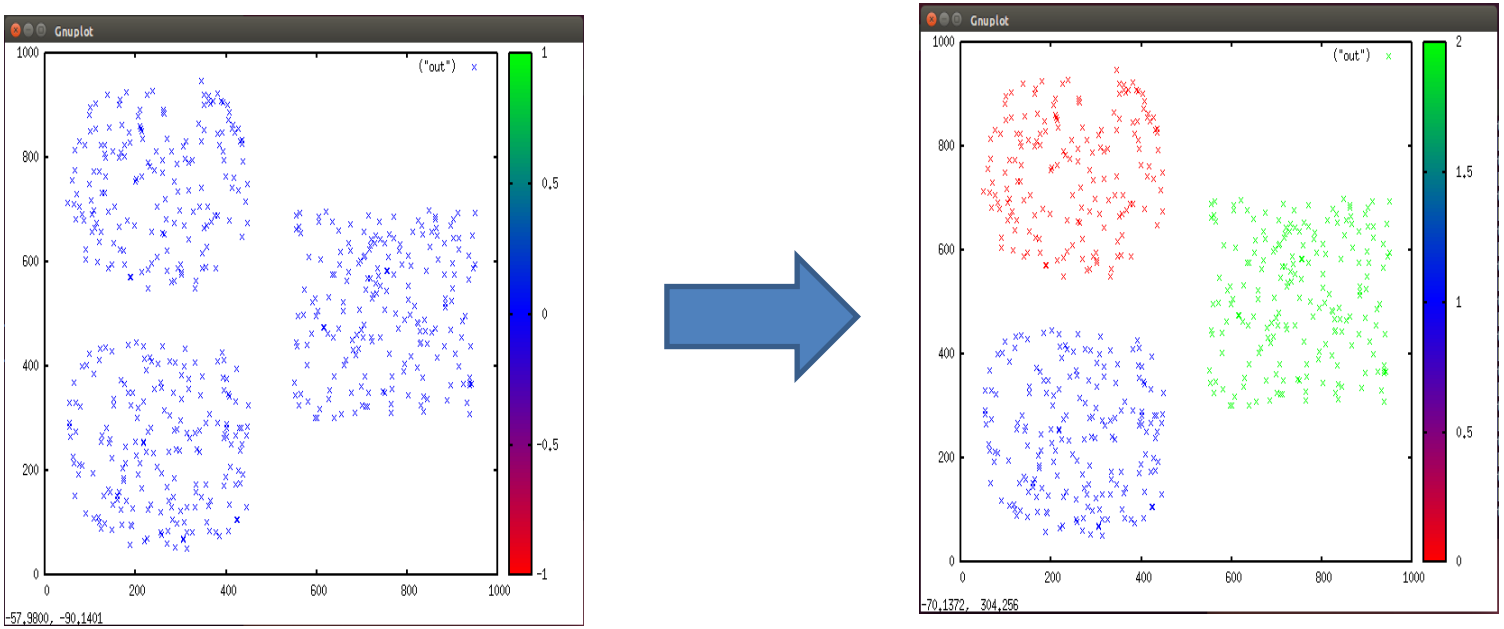
Supervised Learning

- **Supervised learning** is the machine learning task of learning a function that maps an input to an output based on example input-output pairs.

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

Unsupervised Learning

- **Unsupervised learning** is a type of machine learning that looks for previously undetected patterns in a data set with no pre-existing labels and with a minimum of human supervision.



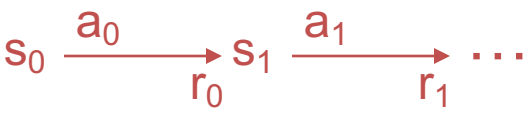
https://en.wikipedia.org/wiki/Unsupervised_learning

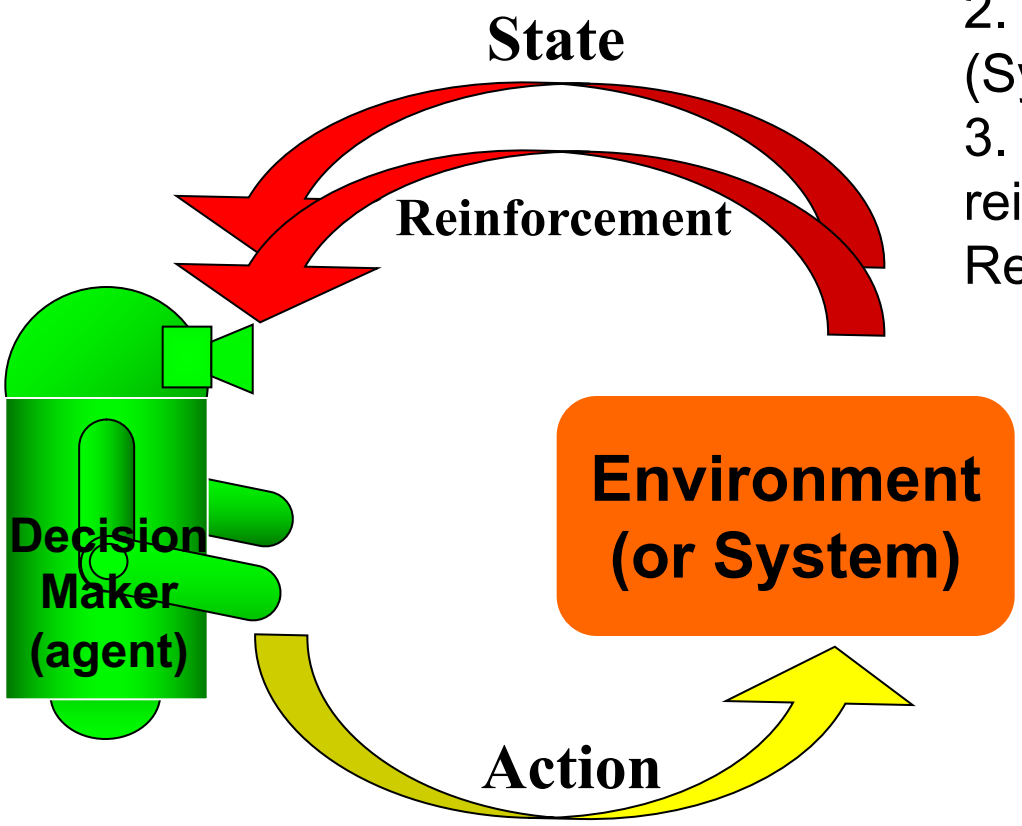
Reinforcement Learning

- RL addresses the question of how an autonomous agent that **senses** and **acts** in its environment can learn to choose optimal actions to get as **much reward** as it can over the long run.
- Applied to sequential decision problems!

Markov Decision Process

Sequential decision problem





At each time step the decision maker:

1. Observes the state of the system;
2. Chooses an action and applies it; (System evolves to a new state)
3. Observes an immediate reinforcement;

Repeat 1 – 3

This assumes discrete time.

Decisions are made at points of time referred to as ***decision epochs.***

The set of decision epochs can be **finite or infinite**:
$T = \{0,1,2,\ldots,N\}$, $N \leq \infty$.

MDP – Model Formulation

An MDP is defined as $\langle S, A, T, R, \gamma \rangle$:

- **S** is the set of possible system states (arbitrary finite set);
- **A** is the set of allowable actions (arbitrary finite set);
- **T**: $S \times A \times S \rightarrow [0,1]$ is the **transition probability function**;
- **R**: $S \times A \rightarrow \Re$ is the **reinforcement function**;
- γ: $\gamma \in [0,1]$, discount factor that balances the trade-off between immediate rewards and future rewards

Histories

- Each time a given policy is executed starting from the initial state, the stochastic nature of the environment will lead to a different environment history.

- **Each policy induces a probability distribution over histories**
  - If $h = \langle s_0, s_1, \ldots \rangle$ then

$$P(h \mid \pi) = P(s_0) \prod_{i \geq 0} p_{\pi(s_i)}(s_{i+1} \mid s_i, a_i)$$

Utility function

- **Utility function** for the agent depends on a sequence of states (environment ***history***)
- In each state **s**, the agent receives a **reinforcement r(s)**, which may be positive or negative, but must be **bounded.**
- **Utility = sum of the rewards received**

What is a Solution?

- A solution must specify what the agent should do for ***any*** state that the agent might reach
  ➔ ***policy*** π

$$\Pi: S \rightarrow A, \quad \pi(s) = a \in A$$

Optimal Policy

- Utility of a state – defined in terms of the utility of state sequences:

$$V_\pi(s) = E\left[\sum_{t=0}^{\infty} \gamma^t r(s_t) \,\middle|\, \pi, s_0 = s\right]$$

The true utility of a state, V(s), is just $V_{\pi^*}(s)$, which allows the agent to choose the action that maximizes the expected utility of the ***subsequent*** state:

$$\pi^*(s) = \arg\max_a \sum_{s'} p(s' \mid s, a) V^*(s')$$

If we know V*, then it's easy to find the optimal policy.

Value Iteration

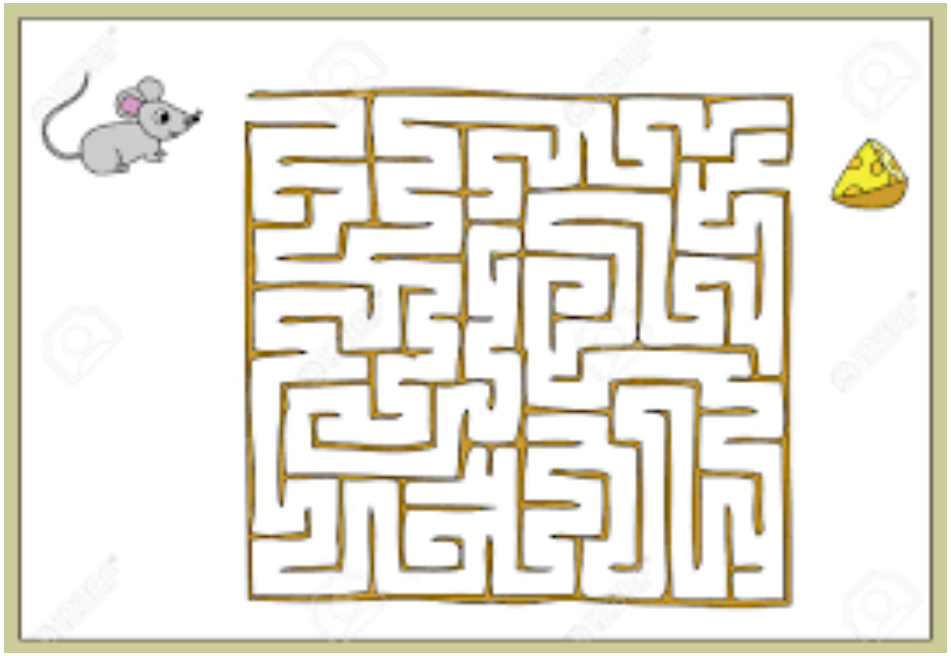
1. Initialize $V_0(s) = 0$, for all s.
2. Loop until a stop criterion is met:
   - Loop for all s:

$$V^{t+1}(s) \leftarrow r(s) + \max_a \gamma \sum_{s'} p(s' \mid s, a) V^t(s')$$

This algorithm is guaranteed to converge to V*.
The influence of **r** and **p**, which we know, drives the successive Vs to get closer and closer to V*.


Aprendizado por Reforço:
conceitos, aplicações e desafios

Reinforcement Learning



Feedback is delayed, occasional

Time really matters (sequential, non i.i.d data)

Agent's actions affect the subsequent data it receives

Trial and error learning (via experiences)

MDP and RL

- In ***Reinforcement Learning***, we would like an agent to **learn** to behave well in an **MDP** world, but **without knowing** anything about **T** or **R** when it starts out
- The goal is to learn, for each state, the action that produces the **highest expected accumulated reward**

RL solution

- **Solution**: a policy, $\pi: S \longrightarrow A$, that maximizes:

$$Q^*(s,a) = E\left[\sum_{t=0}^{\infty} \gamma^t r(s,a)\right]$$

Optimal solution: $\pi^*(s) = \arg\max_a Q^*(s,a)$

Value learning: Q-learning

- Q-learning is a **value-based method** that estimates Q*(s,a) with experiences $< s, a, s', r >$:

$$\hat{Q}(s,a) \leftarrow \hat{Q}(s,a) + \alpha \left[\left(r + \gamma \max_{a'} \hat{Q}(s',a') \right) - \hat{Q}(s,a) \right]$$

$\alpha \in [0,1]$ is the learning rate.

Q-learning Algorithm

Initialize Q(s,a) arbitrarily

Observe the current state $\boldsymbol{s}_t$

do forever (stop criterium)

select an action $\boldsymbol{\alpha}_t$ and execute it in $\boldsymbol{s}_t$

receive immediate reward $\boldsymbol{r(s_t,\alpha_t)}$

observe the new state $\boldsymbol{s}_{t+1}$

update Q(s,a) as follows:

$Q_{t+1}(s_t,a_t) \leftarrow \mathbf{(1-}\alpha\mathbf{)}\, Q_t(s_t,a_t) + \alpha\,[r(s_t,a_t) + \gamma \max_a Q_t(s_{t+1},a)]$

$\boldsymbol{s}_t \leftarrow s_{t+1}$


AlphaGo - The Movie | Full Documentary
AlphaGo
Lee Sedol

https://www.udacity.com/course/reinforcement-learning--ud600

Esperamos que tenham gostado do curso!