# Aprendizado por Reforço

Elaine C. R. Cândido

Pontifícia Universidade Católica de Minas Gerais

Março 2021

# Programa do Curso

1. Introdução
   - Contextualização do problema
   - Agentes e seus componentes
   - Aplicações de aprendizado por reforço
2. Fundamentos matemáticos
   - Processos de decisão Markovianos (MDPs - Markov Decision Process)
   - Política, estado, recompensa
   - Equação de Bellman
3. Q-learning
   - Introdução e fundamentos
   - Implementação com Python

# Pré-requisitos

- Lógica de programação
- Programação básica em Python
- Nível do curso: intermediário

# Visão Geral

1. Introdução ao Aprendizado por Reforço
   - Contextualização da Área
2. Exemplos de Aprendizado por Reforço
   - Cartpole
   - Robôs
   - Atari
   - Starcraft
   - O problema
3. Processos de Decisão de Markov (MDPs)
4. Equação de Bellman

# Créditos

Aulas baseadas no curso de *Reinforcement Learning* do professor David Silver, pesquisador na DeepMind e professor at University College London. Para curso completo, consulte: `https://www.davidsilver.uk/`. E também no Livro *Reinforcement Learning Algorithms with Python*

# Contextualização

- Como e porquê pessoas tomam decisões e tentam otimizar a utilidade de cada uma?

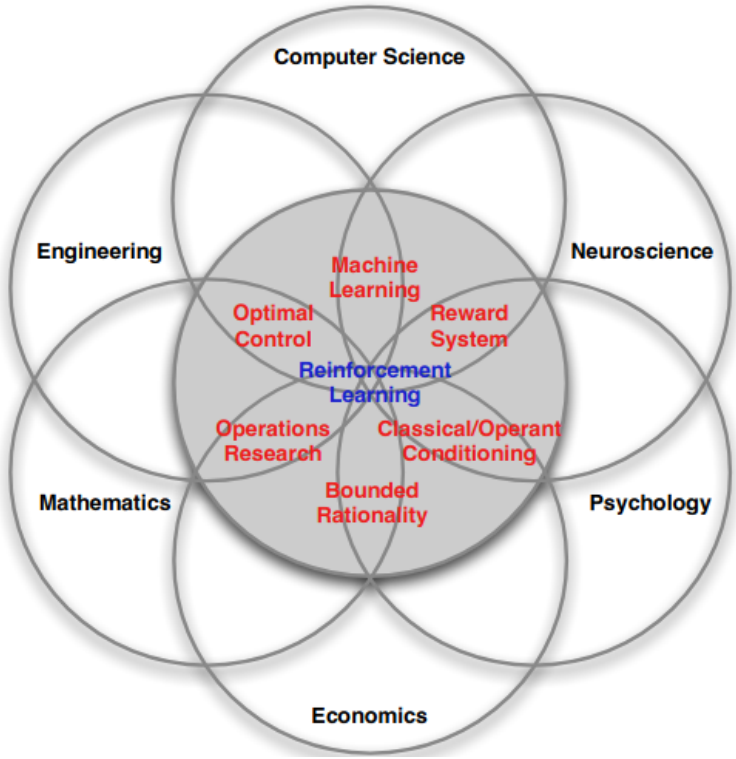
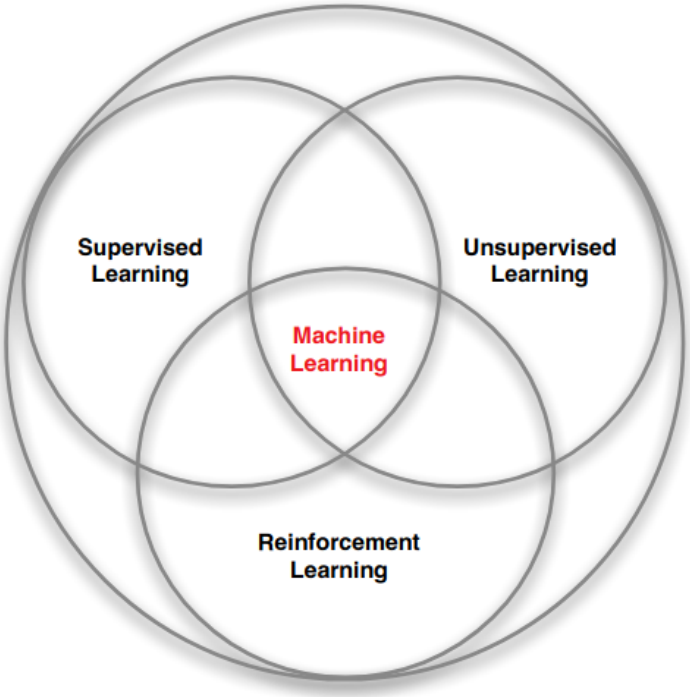


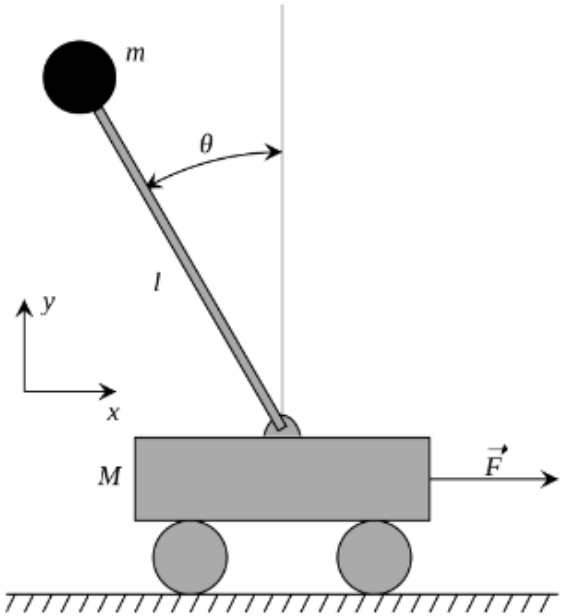

Figure: As diversas faces do Aprendizado por Reforço

# Machine Learning (ML)




- Ações sequenciais
- Não há supervisão (um sinal de recompensa)
- Feedback pode ser atrasado
- O tempo importa
- Ação do agente afeta os próximos dados que o mesmo pode receber

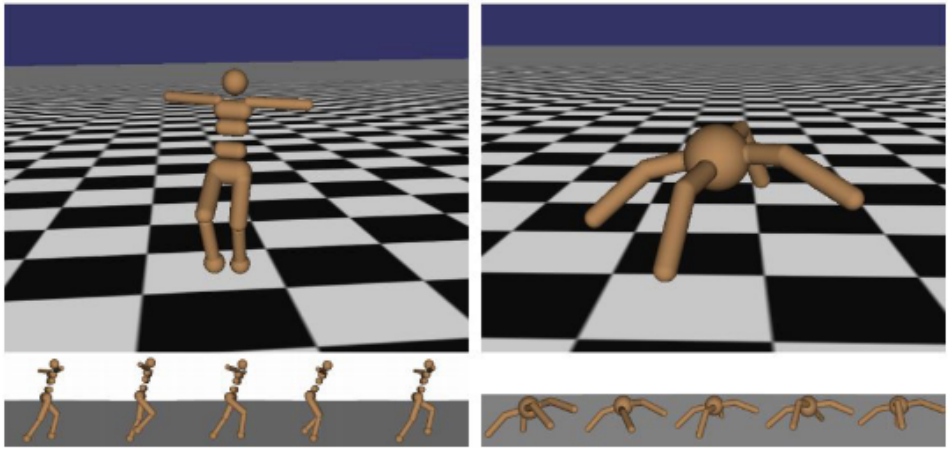
# Cartpole




**Objective**: Balance a pole on top of a movable cart

**State:** angle, angular speed, position, horizontal velocity
**Action:** horizontal force applied on the cart
**Reward:** 1 at each time step if the pole is upright

# Locomoção de Robôs



**Objective**: Make the robot move forward

**State:** Angle and position of the joints
**Action:** Torques applied on joints
**Reward:** 1 at each time step upright + forward movement

# Jogos de Atari

**Objective**: Complete the game with the highest score

**State:** Raw pixel inputs of the game state
**Action:** Game controls e.g. Left, Right, Up, Down
**Reward:** Score increase/decrease at each time step

# Starcraft II

- Tomada de decisão em tempo real
- Espaço de $10^{26}$ ações possíveis a cada espaço de tempo
- Planejamento de longo prazo
- Ambiente dinâmico

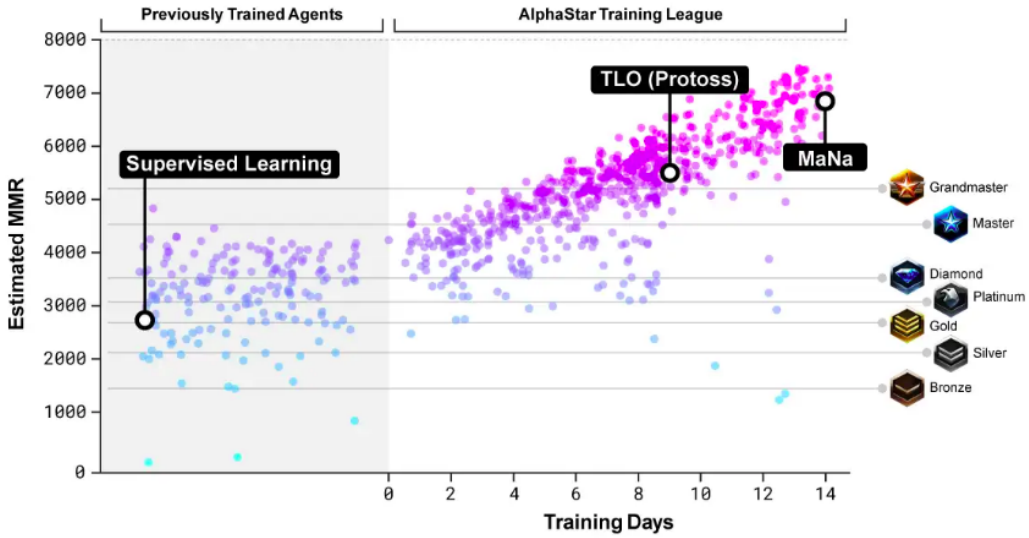



Figure: https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii

# Conceitos

- Área de aprendizado de máquina que lida com decisões sequenciais
- Constituído por um agente tomador de decisões (toma ações) e um mundo em que o agente interage, chamado ambiente.

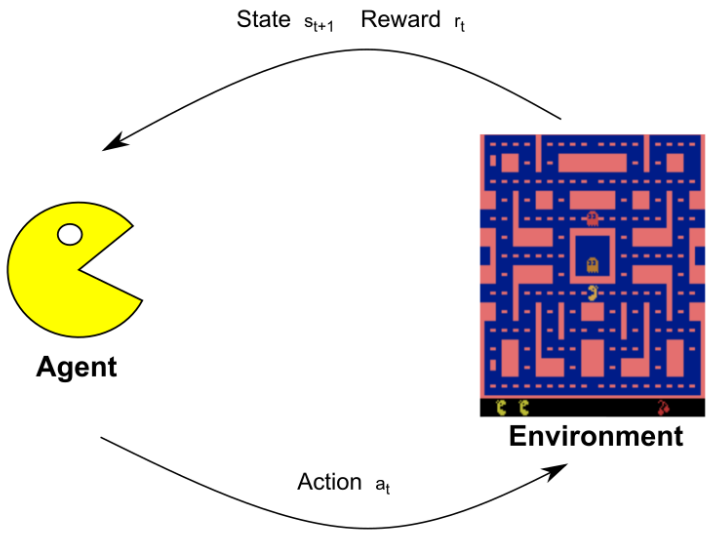



Figure: Agente e Ambiente

- A meta do agente é maximizar o total de recompensas acumuladas
- Ele busca otimizar o resultado a cada ação

# Conceitos

- Uma recompensa $R_t$ é um sinal de feedback escalar
- Indica quão bem o agente está no espaço de tempo $t$

Aprendizado por reforço é baseado na hipótese de recompensa

**Definição (Hipótese de Recompensa)**

Todas as metas podem ser descritas pela maximização da esperança da recompensa acumulada.

# Exemplos de recompensas

- Acrobacias de voo de helicoptero
  - + voar na trajetória correta
  - - bater ou cair
- Fazer locomoção de humanoide
  - + movimentos para frente
  - - cair
- Jogar atari melhor do que humanos
  - + aumentar pontuação
  - - diminuir pontuação

# Tomada de decisão sequencial

- Meta: maximar a recompensa total futura
- As ações podem ter consequências de longo prazo
- Recompensa pode ser atrasada
- Talvez seja melhor sacrificar recompensas imediatas para ganhar recompensas de longo prazo
- Exemplos
  - Investimento financeiro
  - Reabastecer helicóptero

# O agente e o ambiente

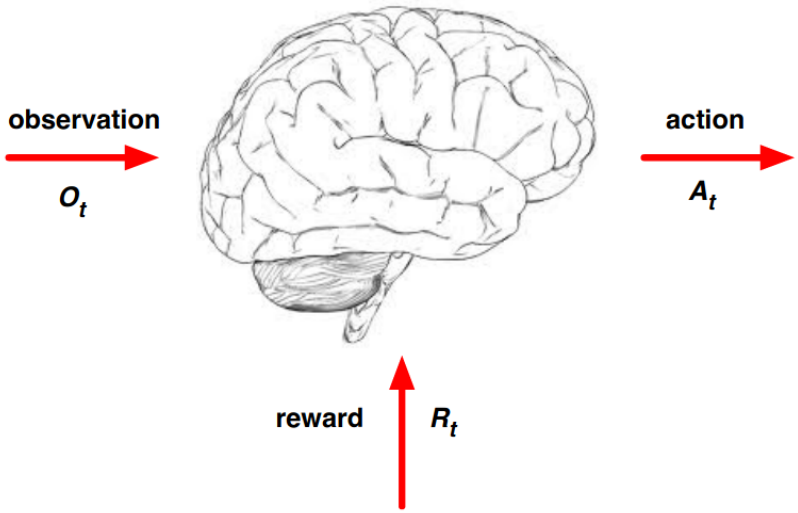



Figure: O agente e o ambiente

# Histórico e estado

- Um histórico é um sequência de observações, ações e recompensas

$$H_t = O_1, R_1, A_1, ..., A_{t1}, O_t, R_t$$

- Por exemplo: todas as variáveis até o tempo $t$
- O que acontece no próximo instante depende do histórico:
  - O agente seleciona ações
  - O ambiente seleciona observações/recompensas
- Estado é a informação usada para determinar o que acontece no próximo intervalo de tempo

$$S_t = f(H_t)$$

# Estado de Informação

- Um estado de informação é conhecido como estado de Markov, ele contém toda informação útil do histórico

**Definição**

Um estado $S_t$ é Markov se, e somente se:

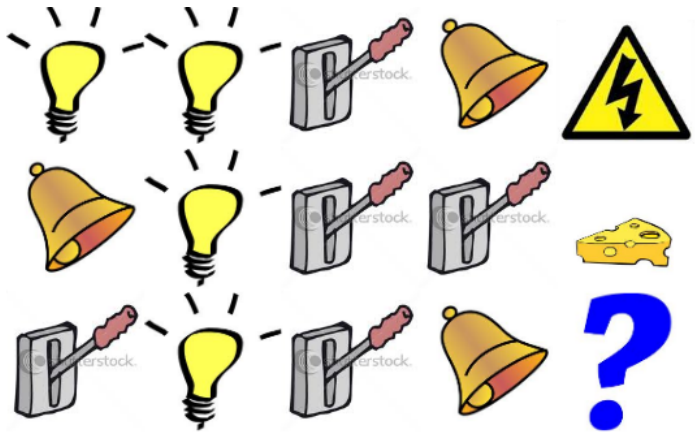
$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1, ..., S_t]$$

- O futuro é independente do passado dado o presente

$$H_{1:t} \rightarrow S_t \rightarrow H_{t+1:\infty}$$

# Exemplo do Rato

- O que escolheria de acordo com as últimas experiências?

# Componentes de um agente

- Um agente pode conter um ou mais destes componentes
  - Política: função de comportamento do agente
  - Função valor: quão bom é o estado e/ou ação
  - Modelo: representação do ambiente do agente

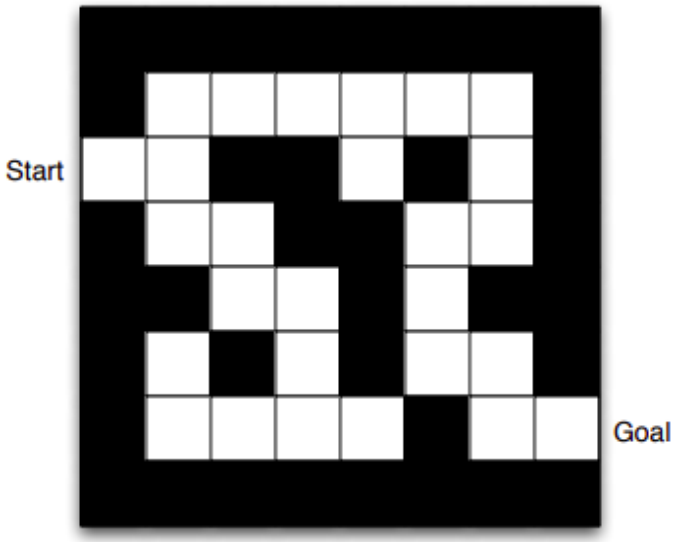
# Componentes de um agente

- Política
  - É o comportamento do agente
  - É um mapeamento do estado para ação, por exemplo.
  - Determinístico: $a = \pi(s)$
  - Estocástico: $\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$
- Função valor
  - a predição do valor de recompensa futuro
  - Usado para avaliar quão bom/ruim cada estado é
  - Assim, seleciona ações, por exemplo.

$$v_\pi(s) = \mathbb{E}_\pi[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + ...|S_t = s]$$

- Modelo
  - Prediz o que o ambiente fará no próximo espaço de tempo

# Exemplo




- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent's location

# Exemplo de política

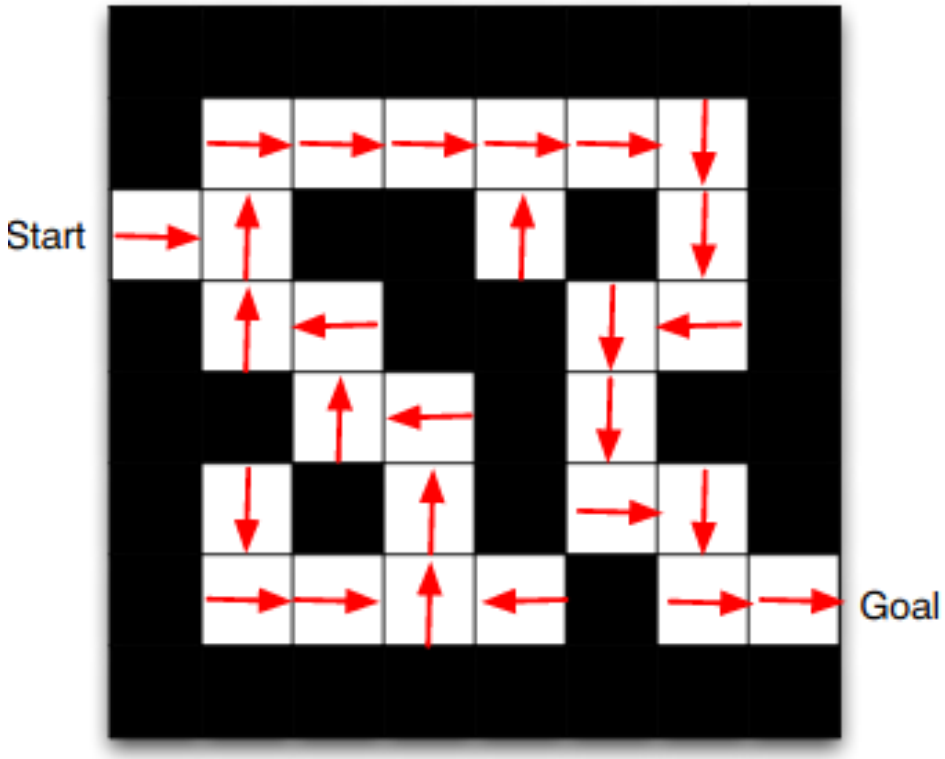



Figure: Exemplo de política

# Exemplo de Função Valor

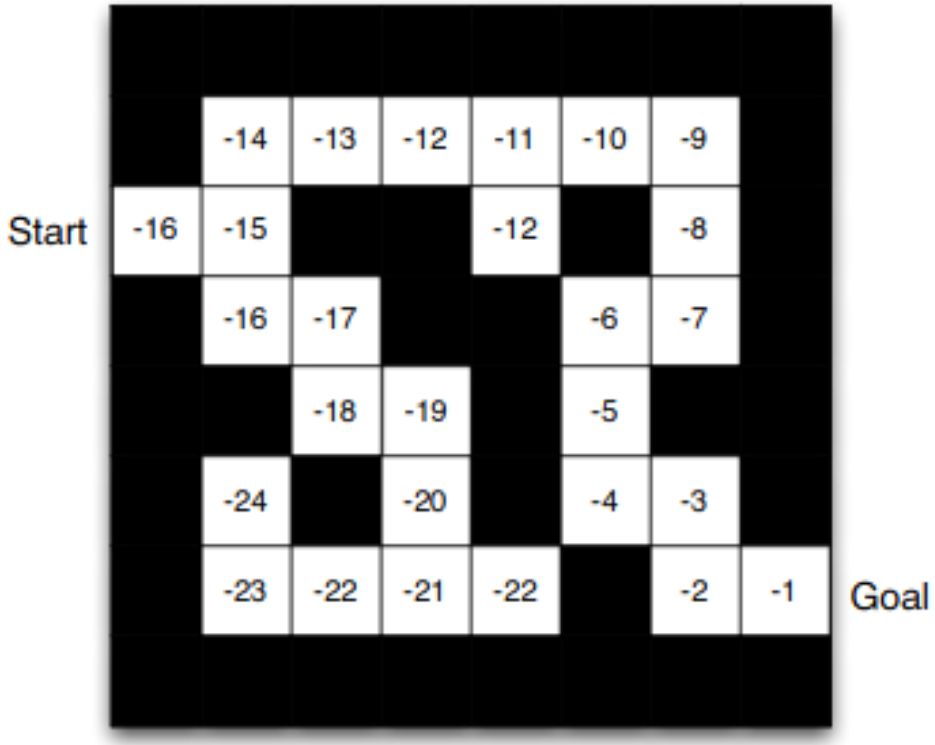
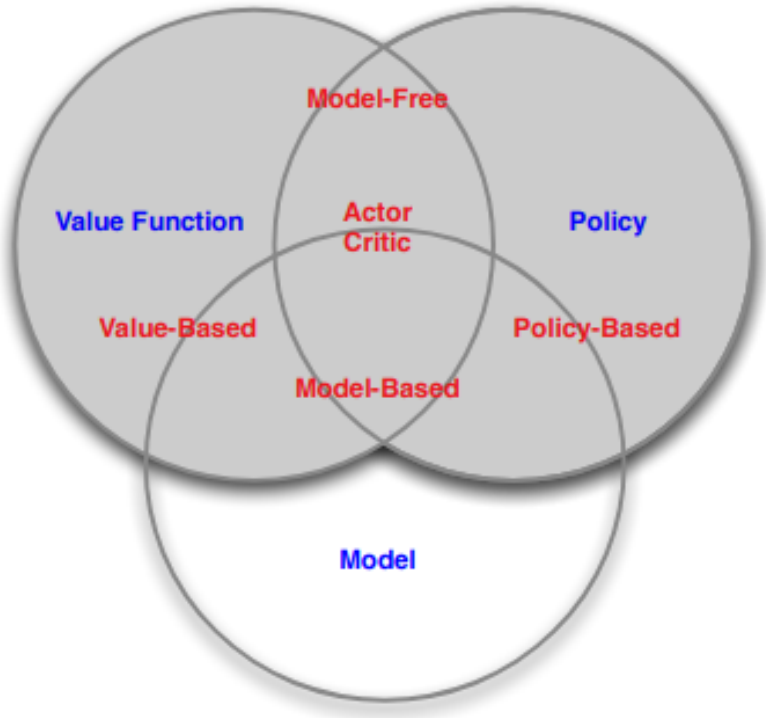



Figure: Exemplo função valor

# Categorização de agentes

- Baseado em valor
  - Sem política implícita
  - Função Valor
- Baseado em política
  - Política
  - Sem função Valor
- Actor Critic
  - Política
  - Função Valor



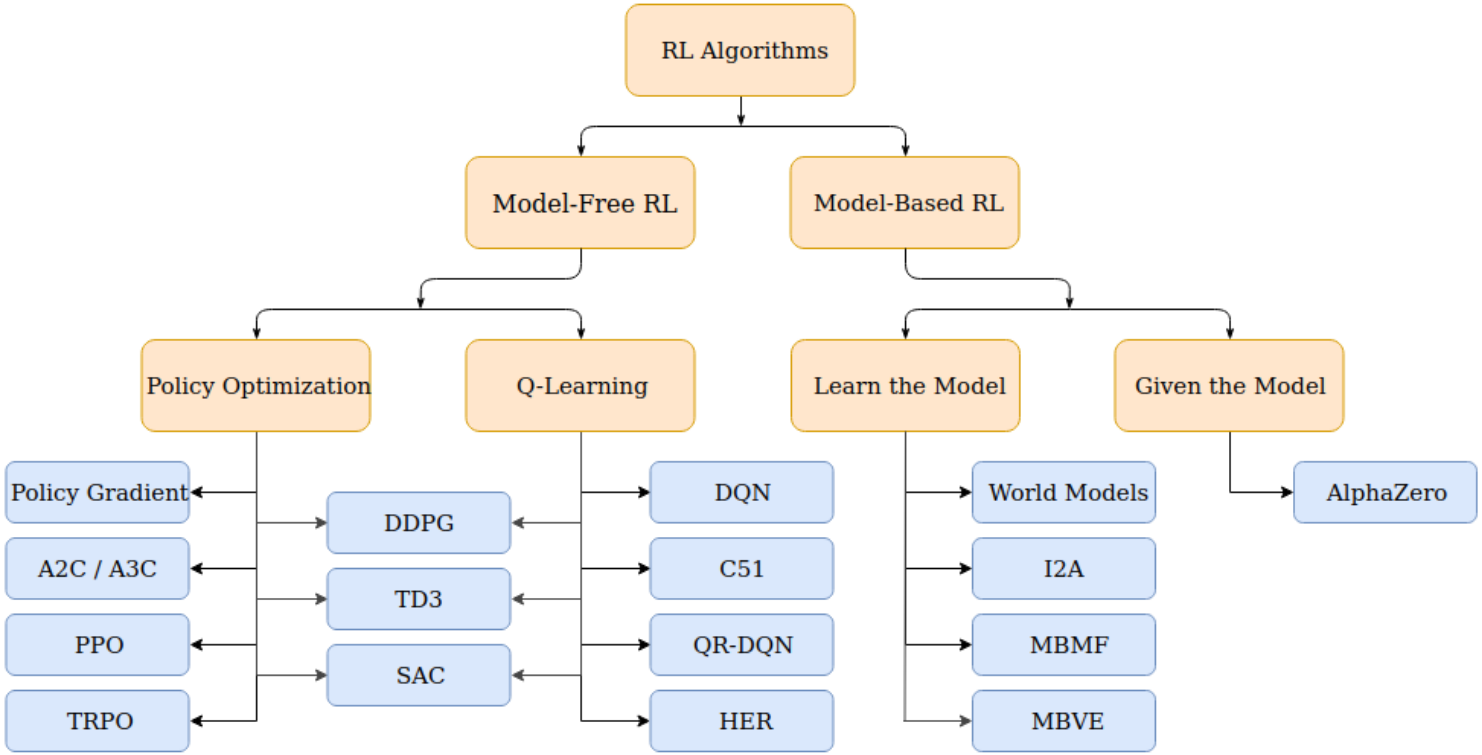



Figure: Lista de algoritmos da Taxonomia
https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html#links-to-algorithms-in-taxonomy

# Introdução - Processos de Decisão de Markov (MDPs)

- Formalmente descreve um ambiente de aprendizado por reforço
- Ambiente totalmente observável
- Quase todos problemas de aprendizado por reforço podem ser caracterizados como MDPs

# Propriedade de Markov

**Definição**

Um estado $S_t$ é Markov se, e somente se:

$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1, ..., S_t]$$

- O futuro é independente do passado dado o presente

# Processo de Markov

Processo randômico sem memória, como uma sequência de estados randômicos $S_1, S_2, \ldots$ com a propriedade de Markov

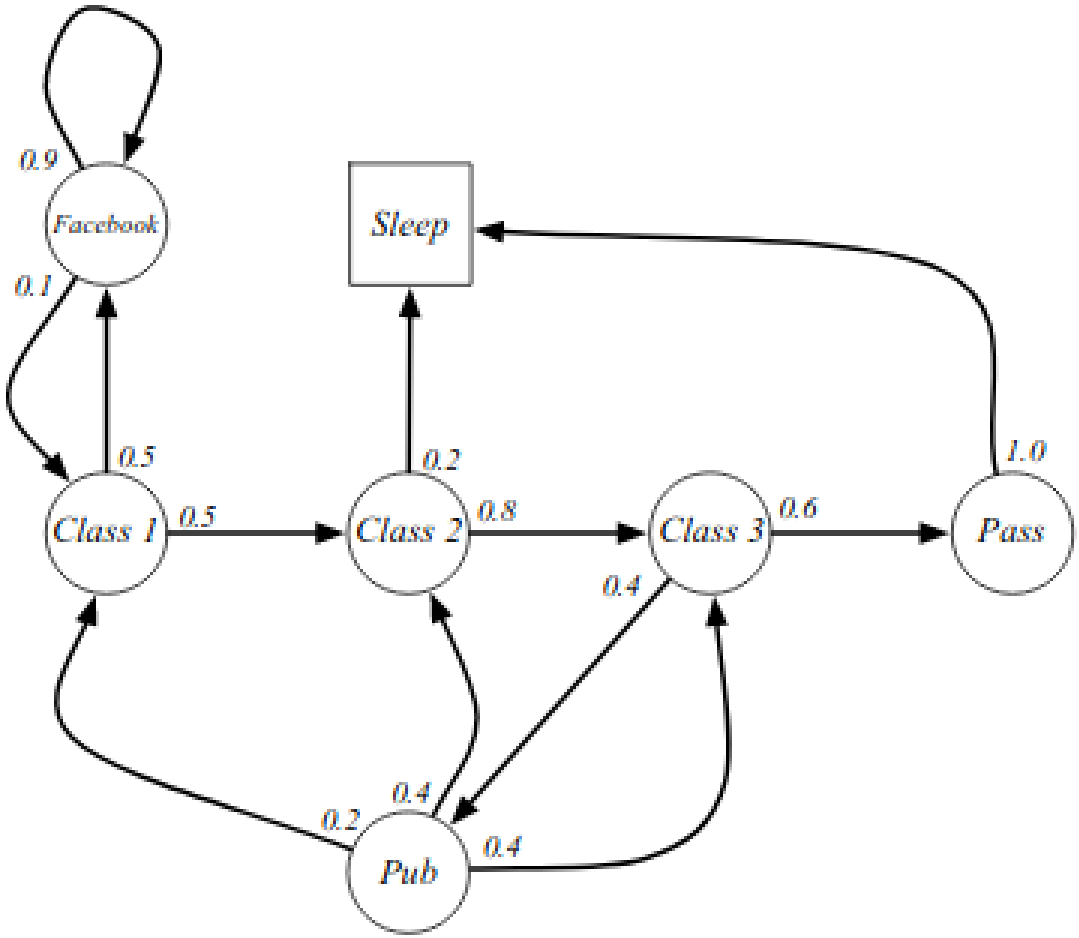
## Definição

Um processo de Markov ou uma cadeia de Markov é uma tupla $\langle S, P \rangle$

- $S$ é um conjunto finito de estados
- $P$ é uma matriz de probabilidades de transição de estado

$$P_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s]$$

# Processo de Markov



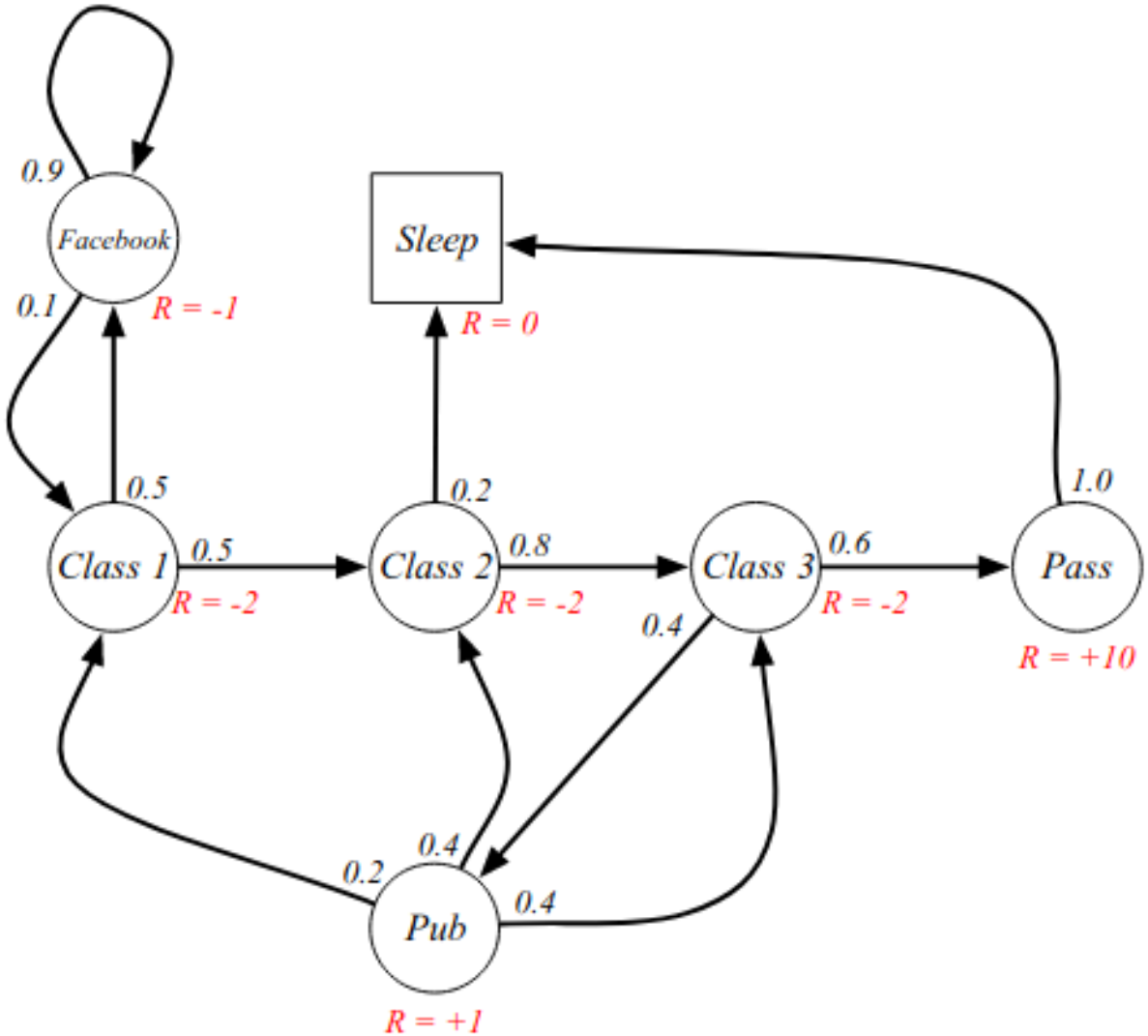
# Processo de Recompensa de Markov (MRP)

## Definição

Um processo de Markov ou uma cadeia de Markov é uma tupla $\langle S, P, R, \gamma \rangle$

- $S$ é um conjunto finito de estados
- $P$ é uma matriz de probabilidades de transição de estado
$$P_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s]$$
- $R$ é uma função de recompensa,
$$R_s = \mathbb{E}[R_{t+1} | S_t = s]$$
- $\gamma$ é um fator de desconto $\gamma \in [0, 1]$

# MRP do Estudante



# Retorno

## Definição

O retorno $G_t$ é a recompensa total descontada no espaço de temp $t$.

$$G_t = R_{t+1} + \gamma R_{t+2} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

# Função Valor no MRP

Representa o valor de recompensa de longo prazo a partir de S

**Definição**

É o retorno esperado iniciando do estado $s$.

$$v(s) = \mathbb{E}[G_t | S_t = s]$$

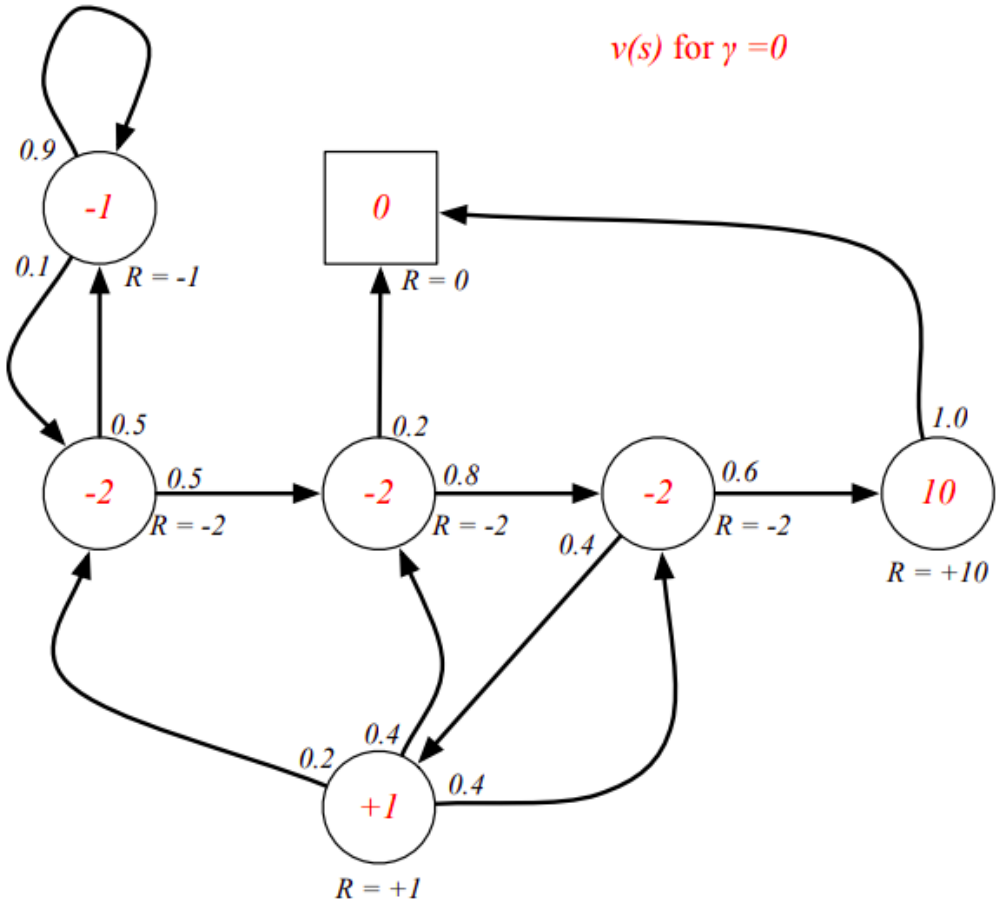
# MRP do Estudante

Starting from $S_1 = \text{C1}$ with $\gamma = \frac{1}{2}$

$$G_1 = R_2 + \gamma R_3 + ... + \gamma^{T-2} R_T$$

| | | | |
|---|---|---|---|
| C1 C2 C3 Pass Sleep | $v_1 = -2 - 2 * \frac{1}{2} - 2 * \frac{1}{4} + 10 * \frac{1}{8}$ | $=$ | $-2.25$ |
| C1 FB FB C1 C2 Sleep | $v_1 = -2 - 1 * \frac{1}{2} - 1 * \frac{1}{4} - 2 * \frac{1}{8} - 2 * \frac{1}{16}$ | $=$ | $-3.125$ |
| C1 C2 C3 Pub C2 C3 Pass Sleep | $v_1 = -2 - 2 * \frac{1}{2} - 2 * \frac{1}{4} + 1 * \frac{1}{8} - 2 * \frac{1}{16} \dots$ | $=$ | $-3.41$ |
| C1 FB FB C1 C2 C3 Pub C1 ... FB FB FB C1 C2 C3 Pub C2 Sleep | $v_1 = -2 - 1 * \frac{1}{2} - 1 * \frac{1}{4} - 2 * \frac{1}{8} - 2 * \frac{1}{16} \dots$ | $=$ | $-3.20$ |

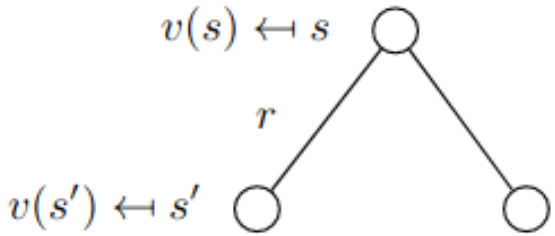
# MRP do Estudante com Função Valor



# Equação de Bellman

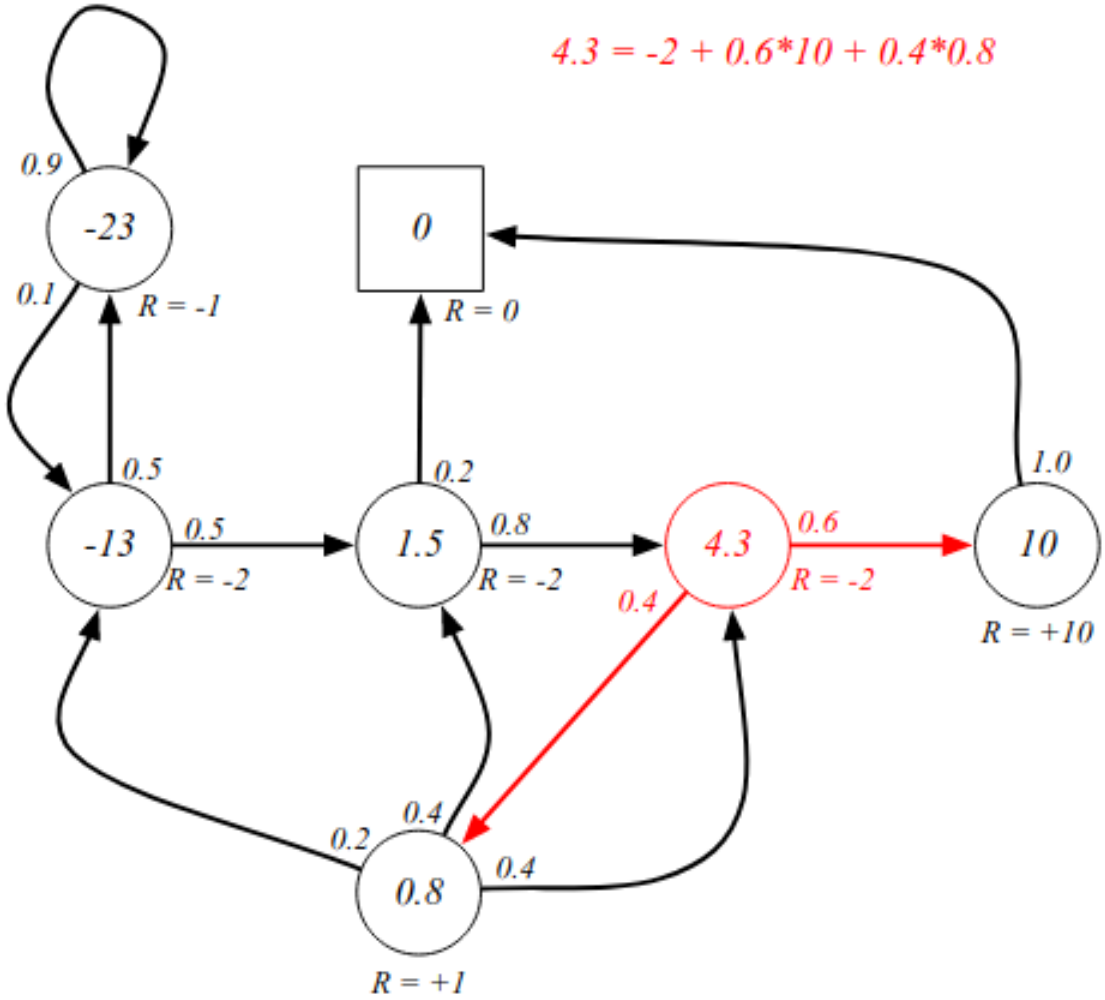
- A função valor por ser decomposta em duas partes:
  - Recompensa imediata $R_{t+1}$
  - Valor descontado a partir do estado sucessor

$$v(s) = \mathbb{E}\left[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s\right]$$




$$v(s) = \mathcal{R}_s + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'} v(s')$$

# Bellman aplicado ao Estudante MRP



# Processo de Decisão Markoviano (MDP)

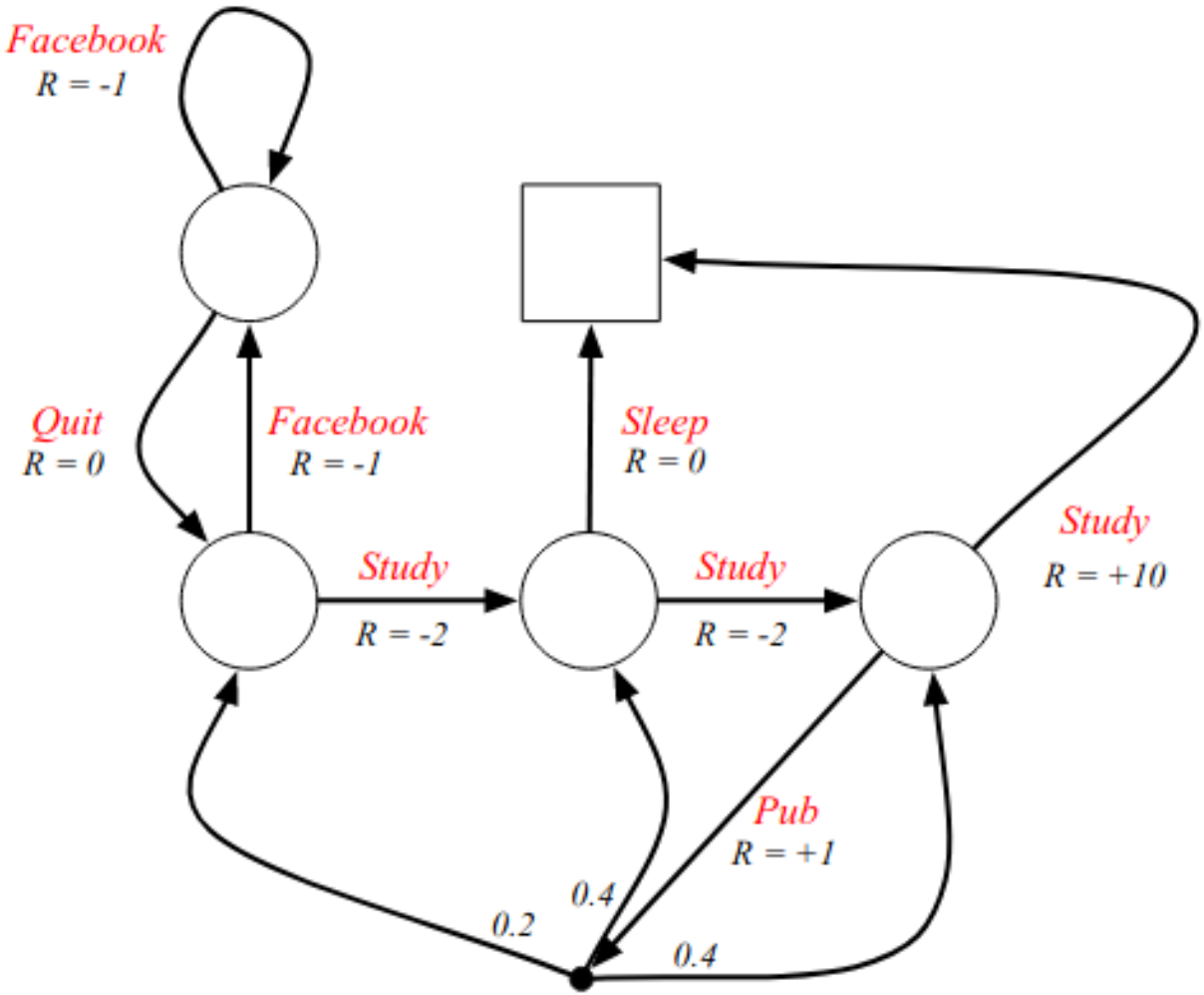
Um Processo de Decisão Markoviano (MDP) é um processo de recompensa de Markov com decisões. É um ambiente no qual todos os estados são Markov.

## Definição

Um processo de Decisão de Markov é uma tupla $\langle S, A, P, R, \gamma \rangle$

- $A$ é um conjunto finito de ações
- $S$ é um conjunto finito de estados
- $P$ é uma matriz de probabilidades de transição de estado
$$P^{a}_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$$
- $R$ é uma função de recompensa
$$R^{a}_{s} = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$$
- $\gamma$ é um fator de desconto $\gamma \in [0, 1]$

# Estudante como MDP



# Função Valor no MDP

- Política:

$$\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$$

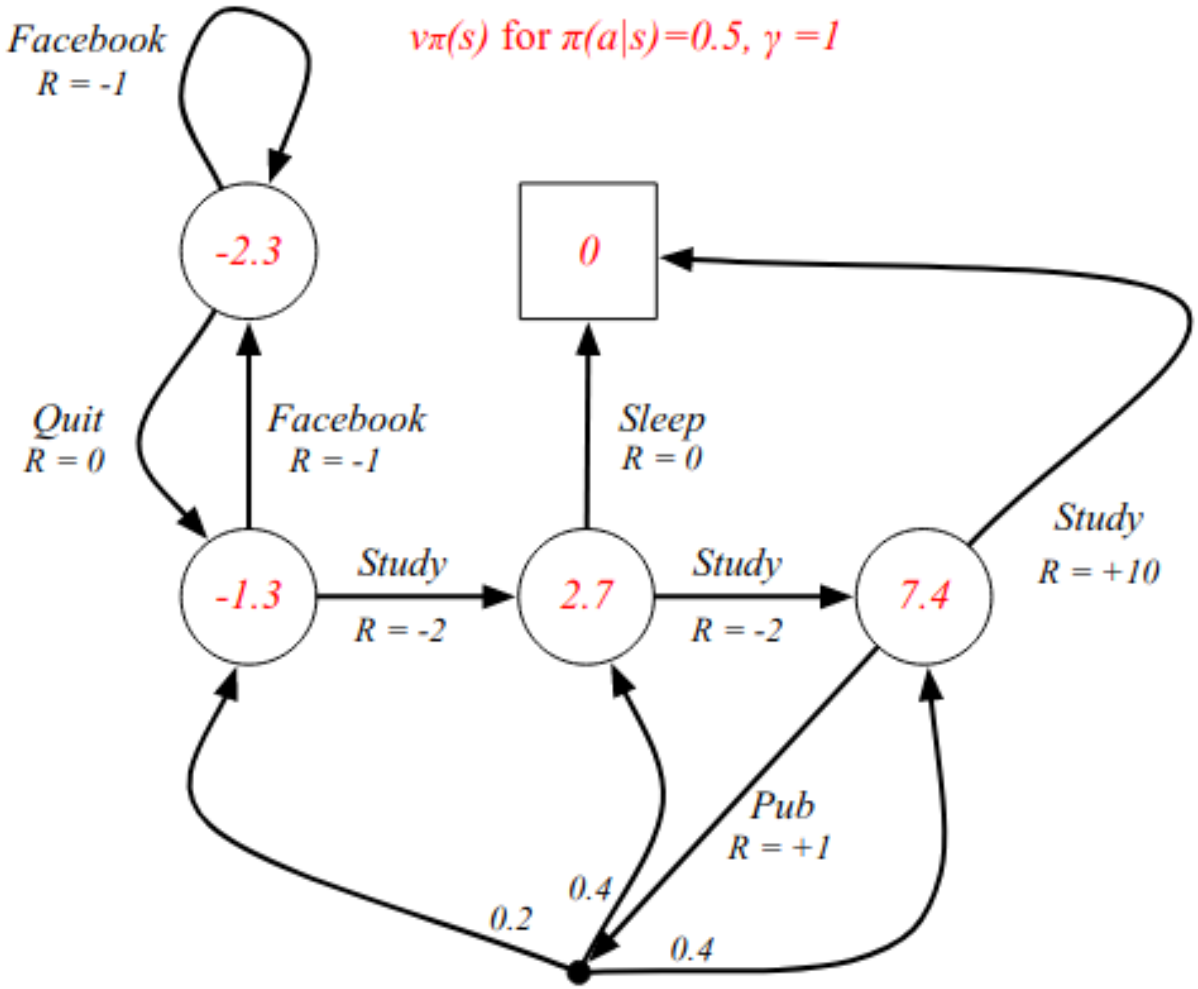
- Função valor-estado:

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$$

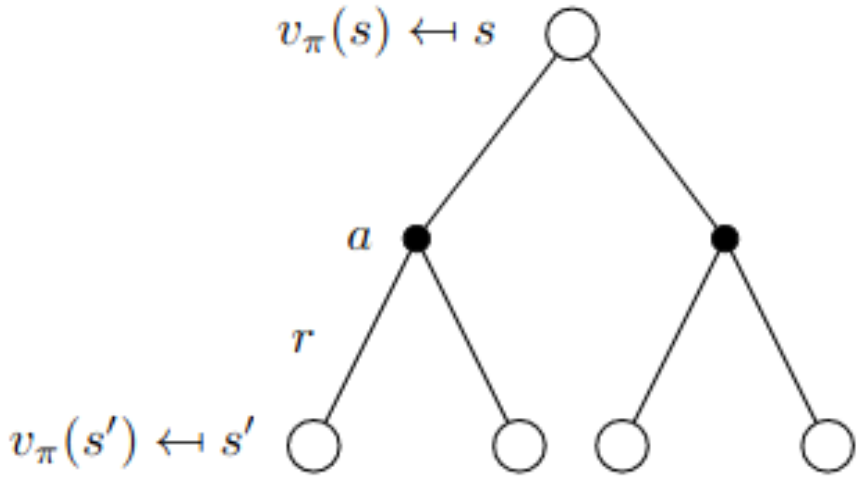
- Função valor-ação:

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$$

# Estudante como MDP com valor-estado

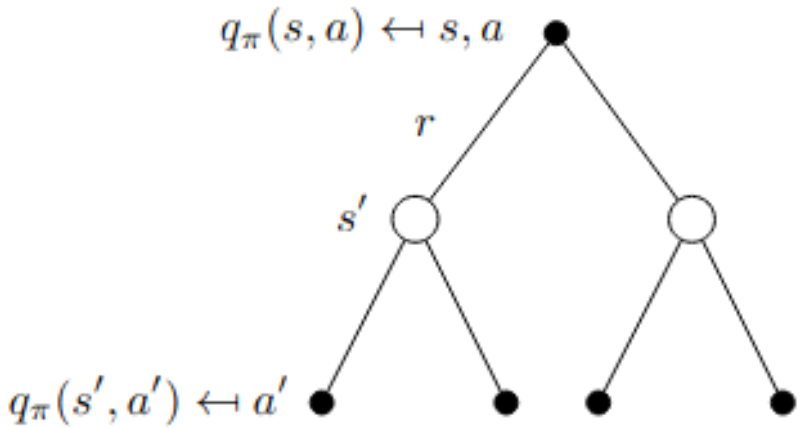


# Esperança da Função de Bellman



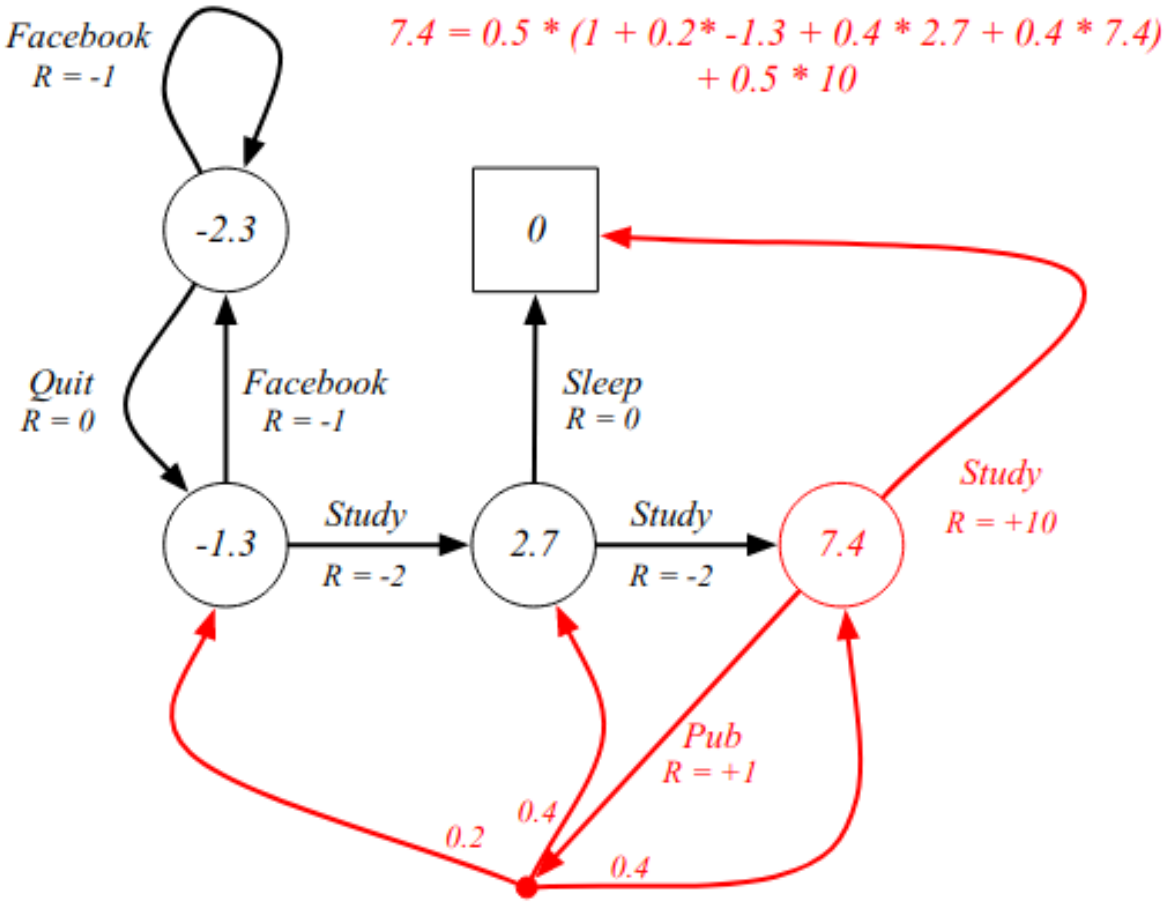

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left( \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$

# Esperança da Função de Bellman



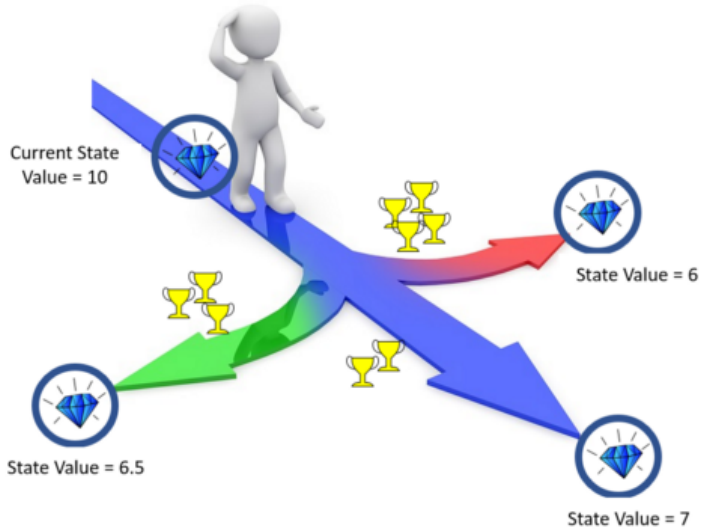

$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \sum_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s',a')$$

# Esperança da Função de Bellman no MDP Estudante



# Aprendizado por reforço é isso?

- Temos que otimizar essas funções



# Função Valor Ótima

- Função valor:
  - Especifica a melhor performance possível do MDP
  - Busca o valor máximo de todas políticas
  - Busca o valor máximo de todas as ações que podem ser tomadas
  - MDP é resolvido quando sabemos o valor ótimo

$$v_\star(s) = \max_\pi v_\pi(s)$$
$$q_\star(s,a) = \max_\pi q_\pi(s,a)$$

- Política:

$$\pi_*(a|s) = \begin{cases} 1 & \text{if } a = \underset{a \in \mathcal{A}}{\text{argmax}}\ q_*(s,a) \\ 0 & \textit{otherwise} \end{cases}$$

  - Se sabemos $q_\star$, imediatamente temos a política ótima

# Equação de Bellman e Valores Ótimos

$$v_*(s) = \max_a \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$

$$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \max_{a'} q_*(s',a')$$

# Solução para Equação Ótima de Bellman

- Equação ótima de Bellman é não linear
- Métodos que consideram soluções iterativas
  - Iteração de valor
  - Iteração de política
  - Q-learning
  - Sarsa

# Referências


- Livros
  - Ravichandiran, S., 2018. Hands-on Reinforcement Learning with Python: Master Reinforcement and Deep Reinforcement Learning Using OpenAI Gym and TensorFlow. Packt Publishing Ltd.
  - Lonza, A., 2019. Reinforcement Learning Algorithms with Python: Learn, understand, and develop smart algorithms for addressing AI challenges. Packt Publishing Ltd.
  - Richard S. Sutton and Andrew G. Barto, Reinforcement Learning: An Introduction `https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf`
- Vídeos
  - Lecture 14 - Deep Reinforcement Learning. Disponível em: `https://www.youtube.com/watch?v=lvoHnicueoE`
  - RL Course by David Silver - Lecture 1: Introduction to Reinforcement Learning. Disponível em: `https://www.youtube.com/watch?v=2pWv7GOvuf0`
- Train a Deep Q Network with TF-Agents `https://www.tensorflow.org/agents/tutorials/1_dqn_tutorial`

# Referências


- Artigos
  - Comprehensive Review of Deep Reinforcement Learning Methods and Applications in Economics. Disponível em: https://arxiv.org/abs/2004.01509
  - Markov Decision Processes: Concepts and Algorithms https://www.cs.vu.nl/~annette/SIKS2009/material/SIKS-RLIntro.pdf
  - Bellman, Richard Ernest, The Theory of Dynamic Programming. Santa Monica, CA: RAND Corporation, 1954. https://www.rand.org/pubs/papers/P550.html