

Insper

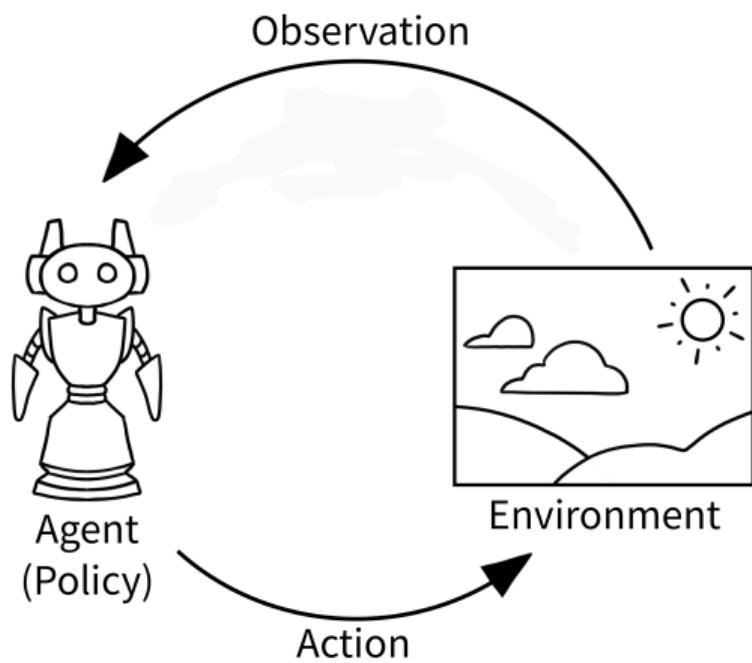
# Introdução ao Aprendizado por Reforço

Fevereiro 2025

Aprendizagem por Reforço  
Fabricio Barth

# **Inteligência Artificial: Definições e principais conceitos**

# Agente Autônomo



“Agentes autônomos são sistemas computacionais que habitam um **ambiente dinâmico complexo**, **sentem** e **agem** autonomamente neste ambiente, e ao fazer isso realizam um conjunto de **objetivos** ou **tarefas** para os quais são projetados.” Pattie Maes

# IBM Deep Blue versus Kasparov (1997)



- ▶ **ambiente:** tabuleiro de xadrez;
- ▶ **tarefa:** jogar e vencer uma partida de xadrez;
- ▶ **ações:** mover peças de xadrez;
- ▶ **implementação:** algoritmo Min-Max + heurísticas + base de conhecimento + hardware dedicado.

# Reconhecimento de dígitos manuscritos



O banco de dados MNIST é um conjunto de 70.000 imagens de dígitos manuscritos de 0 a 9 criado em 1998.

- ▶ **environment:** uma imagem de um dígito manuscrito;
- ▶ **task:** classificar uma imagem sem erro;
- ▶ **actions:** classificar a imagem;
- ▶ **implementation:** existem muitas implementações. No entanto, a melhor abordagem são modelos de rede neural, mais especificamente CNN.

O gold standard para este problema é um erro de teste igual a 0,23% (2012).

# Robô em um labirinto



- ▶ **ambiente:** labirinto;
- ▶ **tarefa:** encontrar a saída do labirinto;
- ▶ **ações:** mover-se para frente, para trás, para a esquerda ou para a direita;
- ▶ **implementação:** algoritmo de busca A\*.

# Veículo autônomo

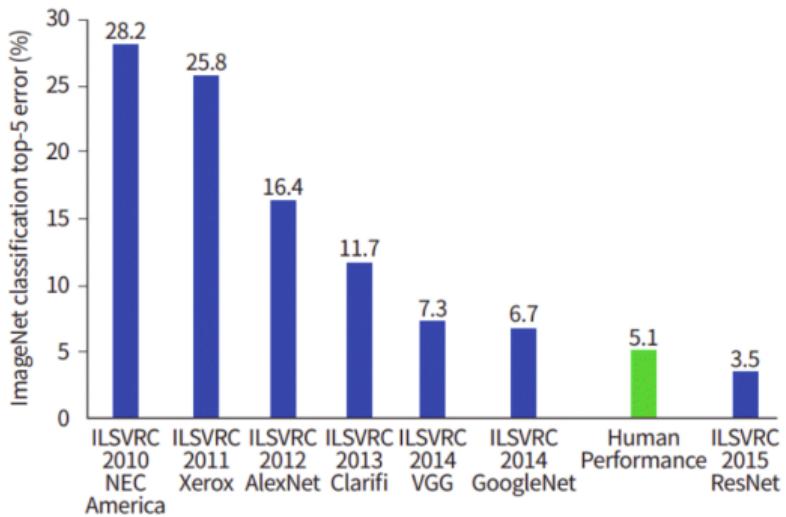


Este é o carro da equipe de Stanford, que venceu a competição DARPA em 2005.

- ▶ **environment:** uma estrada no deserto;
- ▶ **task:** dirigir por um deserto e chegar a um ponto específico;
- ▶ **actions:** acelerar, frear, virar à esquerda, virar à direita;
- ▶ **implementation:** uma implementação bem complexa com diferentes sensores e atuadores.

A equipe de Stanford foi a primeira equipe a vencer esta competição (2005).

# Dataset ImageNet (2009)



Este conjunto de dados tem mais de 14 milhões de imagens anotadas de acordo com a taxonomia do WordNet.

Este dataset tem sido usado no ImageNet Large Scale Visual Recognition Challenge (ILSVRC) desde 2010. Esta competição é uma referência para tarefas de **classificação de imagens e reconhecimento de objetos**.

# AlphaGO jogando GO (2016)

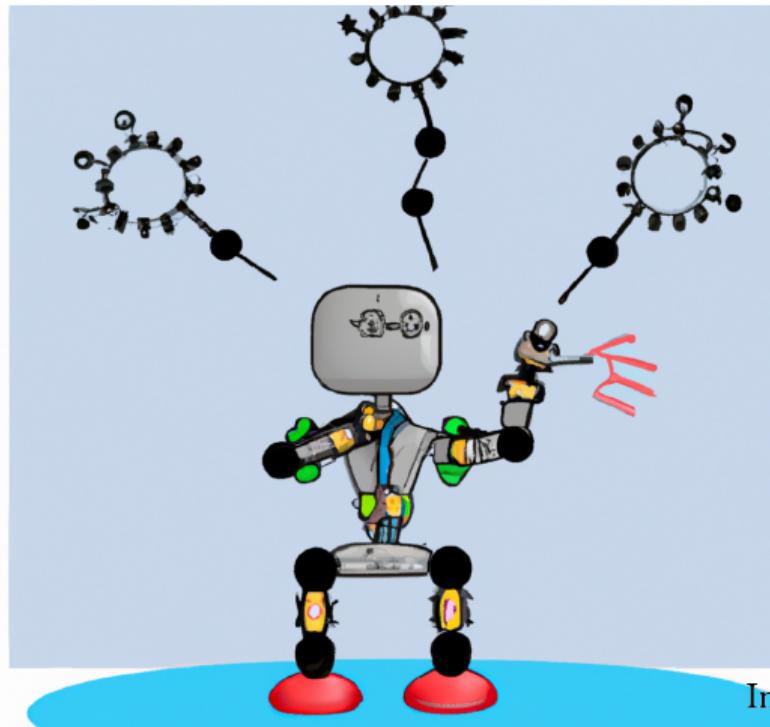


- ▶ **environment**: tabuleiro de GO;
- ▶ **task**: jogar e vencer uma partida de GO;
- ▶ **actions**: mover peças de GO;
- ▶ **implementation**: Aprendizagem por Reforço.

# Modelos Generativos (DALL-E e chatGPT)

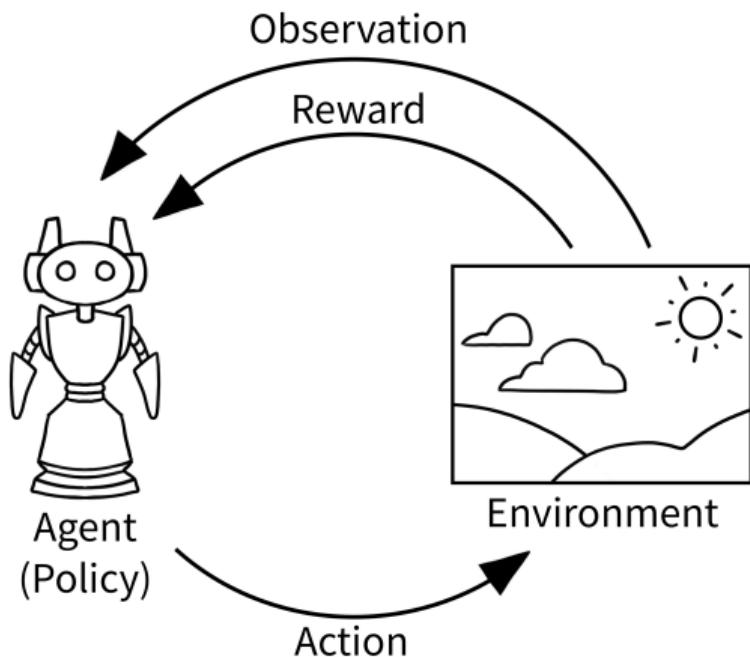
**DALL-E** entrada: “*an autonomous robot solving a problem*”

- ▶ **environment**: uma descrição textual;
- ▶ **task**: gerar imagens relacionadas ao texto informado;
- ▶ **actions**: gerar imagens;
- ▶ **implementation**: uma Deep Neural Network com uma arquitetura bem complexa.



# **Aprendizado por Reforço: definição e principais conceitos**

# Aprendizagem por Reforço ou Reinforcement Learning (RL)



- ▶ RL é uma abordagem de IA onde um agente aprende a tomar ações sequenciais em um ambiente.
- ▶ O aprendizado deste agente acontece através de uma *loop* de interação com o ambiente.
- ▶ O objetivo do agente é aprender a melhor sequência de ações para maximizar uma recompensa.
- ▶ Para cada estado  $s_t$ , o agente escolhe uma ação  $a_t$ , que muda o ambiente para um novo estado  $s_{t+1}$  e gera uma recompensa  $r_t$ .

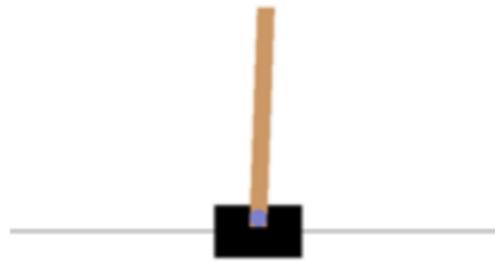
# Aprendizagem por Reforço: conceitos

- ▶ A tupla  $(s_t, a_t, r_t)$  é chamada de **experiência**. Onde  $t$  denota o instante de tempo onde esta experiência acontece.
- ▶ O loop entre o agente e o ambiente termina quando o agente atinge um estado terminal ou depois de um número máximo de iterações  $T$ <sup>1</sup>. O loop entre o agente e o ambiente é chamado de **episódio**.
- ▶ Um agente tipicamente precisa de muitos episódios para aprender uma boa política, variando de centenas de episódios até milhões de episódios.
- ▶ Uma **trajetória** é uma sequência de estados, ações e recompensas:  
 $\tau = (s_0, a_0, r_0), (s_1, a_1, r_1), \dots, (s_T, a_T, r_T)$ .

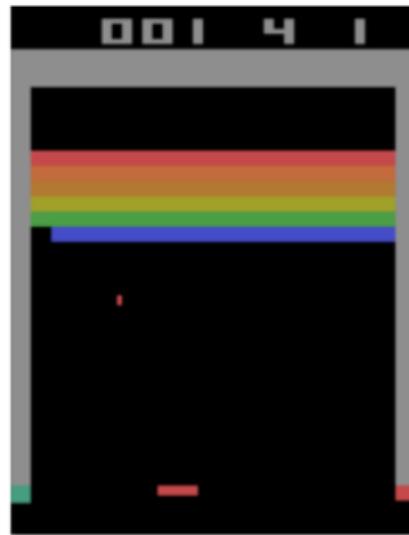
---

<sup>1</sup>Teoricamente, existe a possibilidade de loop infinito.

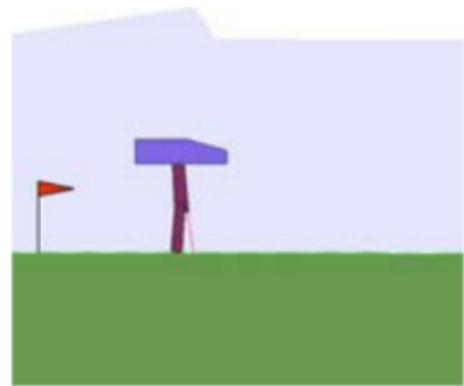
## Exemplos de ambientes



(a) CartPole



(b) Atari Breakout



(c) BipedalWalker

## Ambiente CartPole

- ▶ **Objetivo:** Manter o pêndulo em pé por 200 passos de tempo (*time steps*).
- ▶ **Estado:** Um vetor de tamanho 4 que representa: posição do carrinho, velocidade do carrinho, ângulo do pêndulo, velocidade angular do pêndulo. Por exemplo,  $s = [-0.034, 0.032, -0.031, 0.036]$ .
- ▶ **Ação:** Um valor inteiro, sendo 0 para mover o carrinho uma distância fixa para a esquerda, ou 1 para mover o carrinho uma distância fixa para a direita.
- ▶ **Reward:** +1 para todo passo de tempo que o pêndulo permanece em pé.
- ▶ **Finalização:** quando o pêndulo cai (mais de 12 graus da vertical), ou quando o carrinho sai da tela, ou quando o tempo máximo de 200 passos é atingido.

# Atari Breakout

- ▶ **Objetivo:** maximizar o score do jogo.
- ▶ **Estado:** Uma imagem RGB com resolução  $160 \times 210$  pixels, ou seja, o que vemos na tela do jogo.
- ▶ **Ação:** Um valor inteiro de 0 a 3 que mapeia as ações do controle do jogo: {no-action, launch the ball, move right, move left}.
- ▶ **Reward:** A diferença do score do jogo entre os estados consecutivos.
- ▶ **Finalização:** Quando todas as vidas do jogo são perdidas.

## Bipedal Walker

- ▶ **Objetivo:** Andar para a direita sem cair.
- ▶ **Estado:** Um vetor de tamanho 24 que representa: [ângulo do casco, velocidade angular do casco, velocidade em x, velocidade em y, ângulo da junta do quadril 1, velocidade da junta do quadril 1, ângulo da junta do joelho 1, velocidade da junta do joelho 1, contato com o solo da perna 1, ângulo da junta do quadril 2, velocidade da junta do quadril 2, ângulo da junta do joelho 2, velocidade da junta do joelho 2, contato com o solo da perna 2, ..., 10 leituras do lidar].

Por exemplo,

$$s = [2.745e^{-03}, 1.180e^{-05}, -1.539e^{-03}, -1.600e^{-02}, \dots, 7.091e^{-01}, 8.859e^{-01}, 1.000e+00, 1.000e+00].$$

- ▶ **Ação:** Um vetor de quatro números de ponto flutuante no intervalo [-1.0, 1.0] que representa: [torque e velocidade do quadril 1, torque e velocidade do joelho 1, torque e velocidade do quadril 2, torque e velocidade do joelho 2]. Por exemplo,  $a = [0.097, 0.430, 0.205, 0.089]$ .
- ▶ **Reward:** Recompensa positiva para avançar para a direita, até um máximo de +300. -100 se o robô cair. Além disso, há uma pequena recompensa negativa (custo de movimento) a cada passo de tempo, proporcional ao torque absoluto aplicado.
- ▶ **Finalização:** Quando o robô toca o solo com o corpo, ou quando o robô alcança a meta à direita, ou após o número máximo de passos de tempo de 1600.

## Algumas formalizações e simplificações importantes

- ▶  $s_t \in \mathcal{S}$  é o estado,  $\mathcal{S}$  é o espaço de estados.
- ▶  $a_t \in \mathcal{A}$  é a ação,  $\mathcal{A}$  é o espaço de ações.
- ▶  $r_t = \mathcal{R}(s_t, a_t, s_{t+1})$  é a recompensa,  $\mathcal{R}$  é a função de recompensa.
- ▶ A probabilidade de uma transição de estado  $s_t$  para  $s_{t+1}$  depende de todos os estados e ações que ocorreram até o momento em um episódio:  
 $s_{t+1} \sim \mathcal{P}(s_{t+1} | (s_0, a_0), \dots, (s_{t-1}, a_{t-1}), (s_t, a_t))$ . Isto é muito complexo para modelar, principalmente se o episódio tem um número grande de passos de tempo.

- ▶ Para fazer a transição de ambiente mais prática, transformamos em um *Markov Decision Process* (MDP) adicionando a suposição de que a transição para o próximo estado  $s_{t+1}$  depende apenas do estado anterior  $s_t$  e a ação  $a_t$ . Isto é conhecido como a propriedade de Markov.
- ▶  $s_{t+1} \sim \mathcal{P}(s_{t+1}|s_t, a_t)$
- ▶ **Consequências práticas:** a representação do estado precisa ter toda a informação necessária para tomar uma decisão<sup>2</sup>.

---

<sup>2</sup>O agente pode fazer uso de representações internas do estado que são diferentes da representação fornecida pelo ambiente

# Problema de RL formulado como MDP

- ▶  $\mathcal{S}$  é o espaço de estados.
- ▶  $\mathcal{A}$  é o espaço de ações.
- ▶  $\mathcal{P}(s_{t+1}|s_t, a_t)$  é a função de transição de estado do ambiente.
- ▶  $\mathcal{R}(s_t, a_t, s_{t+1})$  é a função de recompensa do ambiente.

**Importante:**

- ▶ Agentes não tem acesso a função de transição de estado  $\mathcal{P}$  e a função de recompensa  $\mathcal{R}$  do ambiente.
- ▶ A única forma que o agente tem para pegar informações sobre estas funções é através da interação com o ambiente,  $(s_t, a_t, s_{t+1})$ .

- ▶ Dado uma trajetória  $\tau = (s_0, a_0, r_0), (s_1, a_1, r_1), \dots, (s_T, a_T, r_T)$ , o retorno desta trajetória é definido por:
  - ▶  $R(\tau) = r_0 + \gamma r_1 + \gamma^2 r_2 + \dots + \gamma^T r_T = \sum_{t=0}^T \gamma^t r_t$
  - ▶ que é o desconto da soma das recompensas em uma trajetória, onde  $\gamma \in [0, 1]$  é o fator de desconto.
- ▶ A função objetivo  $J(\tau)$  é a expectativa dos retornos sobre todas as trajetórias:
  - ▶  $J(\tau) = \mathbb{E}_{\tau \sim \pi}[R(\tau)]$
  - ▶ O objetivo  $J(\tau)$  é o retorno médio ao longo de vários episódios.

# Loop genérico para treinamento

```
1: Given an env (environment) and an agent:  
2: for episode = 0, ..., MAX_EPISODE do  
3:     state = env.reset()  
4:     agent.reset()  
5:     for t = 0, ..., T do  
6:         action = agent.act(state)  
7:         state, reward = env.step(action)  
8:         agent.update(action, state, reward)  
9:         if env.done() then  
10:             break  
11:         end if  
12:     end for  
13: end for
```

## Funções que podem ser aprendidas pelo agente

- ▶ Uma **política**  $\pi$  que mapeia um estado  $s$  em uma ação  $a$ :  $a \sim \pi(s)$ . Idealmente, a política  $\pi$  é a melhor política que maximiza a recompensa esperada.
- ▶ Uma **função valor**,  $V^\pi(s)$  ou  $Q^\pi(s, a)$ , para estimar uma expectativa de retorno  $\mathbb{E}_\tau[R(\tau)]$ .  $V^\pi(s)$  é a expectativa do retorno começando no estado  $s$  e seguindo a política  $\pi$ .  $Q^\pi(s, a)$  é a expectativa do retorno começando no estado  $s$ , tomando a ação  $a$  e seguindo a política  $\pi$ .
- ▶ Um **modelo do ambiente**,  $P(s'|s, a)$ . Algoritmos desta família podem aprender um modelo do ambiente ou usar um modelo do ambiente para, por exemplo, facilitar o aprendizado de uma política.

# Algoritmos de Aprendizado por Reforço

- ▶ **Policy-based:** Reinforce
- ▶ **Value-based:** Q-Learning, SARSA, Deep Q-Network (DQN), Double DQN.
- ▶ **Model based:** Monte Carlo Tree Search (MCTS)
- ▶ **Policy-based + Value-based:** Actor-Critic, Proximal Policy Optimization (PPO), Trust Region Policy Optimization (TRPO)

## **Exemplos de projetos e aplicações**

## Exemplos (1/2)

- ▶ Playing Atari with Deep Reinforcement Learning, 2013.
- ▶ Human-Level control through deep reinforcement learning, 2015. → *Overpassed human results*
- ▶ A General reinforcement learning algorithm that masters chess, shogi, and Go through self-play, 2018.

## Exemplos (2/2)

- ▶ **Veículos autônomos:** há alguns componentes em veículos autônomos que são otimizados através de RL.
- ▶ **Automatização industrial:** os data centers do Google estão usando RL para reduzir o gasto de energia.
- ▶ **Negociação e finanças:** há muitas empresas dizendo que estão usando RL para criar robôs para *trading*.
- ▶ **Processamento de linguagem natural (NLP):** LLMs estão usando RL para melhorar seu treinamento.
- ▶ **Recomendação:** há muitos artigos sobre RL em sistemas de recomendação.
- ▶ **Path Planning:** há muitos artigos sobre RL em planejamento de trajetórias para sistemas multi-agentes.

# **Diferenças entre Aprendizado Supervisionado, Não Supervisionado e por Reforço**

# Diferenças

- ▶ **Aprendizado supervisionado:**
  - ▶ o principal objetivo é criar um modelo preditivo.
  - ▶ Toda a construção do modelo é baseada em conjuntos de dados de treinamento e teste e ambos os conjuntos de dados devem ter um atributo de **label**.
  - ▶ Todo o processo de treinamento é em modo **batch**.
- ▶ **Aprendizado não supervisionado:**
  - ▶ o principal objetivo é resumir dados. Normalmente, através de modelos de agrupamento ou regras.
  - ▶ Toda a construção do modelo é baseada em conjuntos de dados de treinamento sem um atributo de **label**.
- ▶ **Aprendizado por Reforço:**
  - ▶ O principal objetivo é aprender a interagir com um ambiente.
  - ▶ Todo o processo de treinamento é **interativo**.
  - ▶ Não há dados de treinamento. No entanto, há um **ambiente**.

# Características do Aprendizado por Reforço

- ▶ Falta de um oráculo: a única informação que um agente tem depois de executar uma ação  $a$  em um estado  $s$  é a recompensa. Ninguém fala para o agente qual é a melhor ação a ser tomada. Ele apenas recebe uma indicação de que a ação foi boa ou ruim.
- ▶ *Sparsity of Feedback*: em muitos ambientes, a recompensa pode ser esparsa. Isto é, o agente pode receber uma recompensa positiva ou negativa apenas após um grande número de ações.

A combinação destas duas características torna o aprendizado por reforço menos efetivo se comparado com uma abordagem supervisionada<sup>3</sup>.

---

<sup>3</sup>No contexto do aprendizado por reforço uma experiência entrega menos informação.

- ▶ Geração dos dados: a qualidade do algoritmo não afeta somente o aprendizado, mas também afeta a geração dos dados. Se o agente não está aprendendo bem, ele pode não explorar o ambiente corretamente.

## **Referências e próximas atividades**

## Principal Referência

- ▶ Capítulo 1 do livro Laura Graesser and Wah Loon Keng. 2019. Foundations of Deep Reinforcement Learning: Theory and Practice in Python (1st. ed.). Addison-Wesley Professional.

## Próximas atividades

- ▶ Ferramentas e ambientes para aprendizagem por reforço.
- ▶ Estimando funções valor  $Q^\pi(s, a)$  com o algoritmo Q-Learning.