

RL aula 1



Introdução e fundamentos

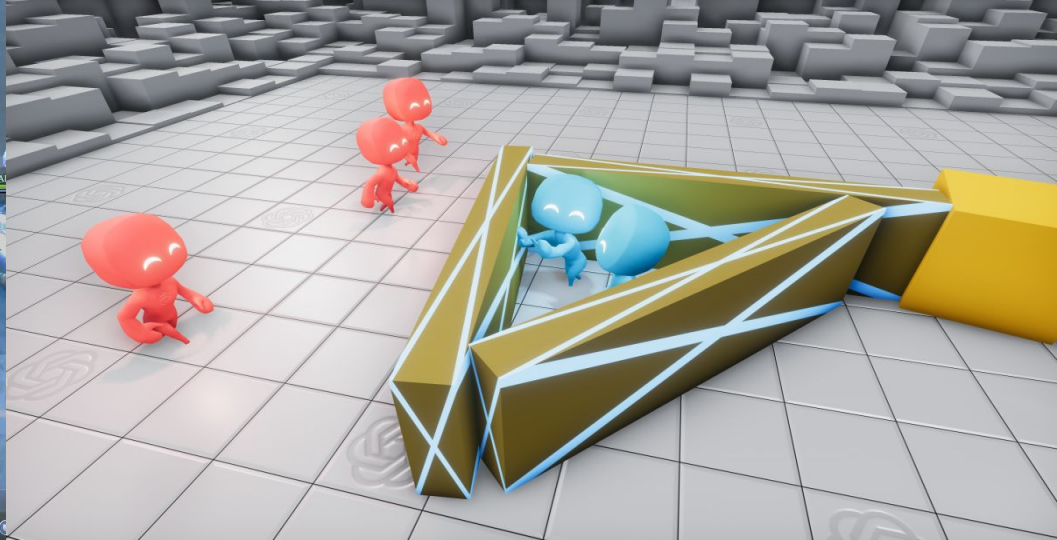
A dark blue diagonal gradient bar that starts from the bottom left and extends towards the top right, covering the lower half of the slide.

Plano de hoje

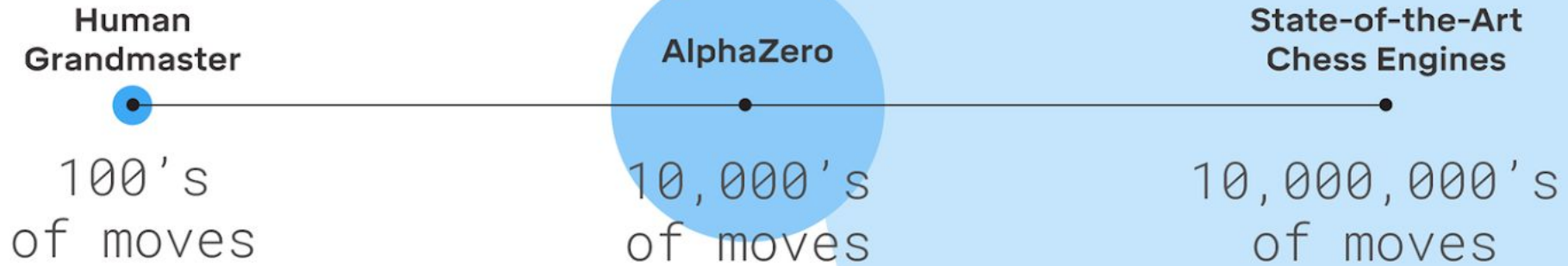
1. Por que aprendizado por reforço?
2. Definições básicas
3. Taxonomia dos algoritmos
4. Recursos úteis para estudo
5. Como tirar o máximo do curso

Por que aprendizado por reforço?

- Em certo sentido mais próximo a IA do que o Deep Learning
 - Agentes que lidam com ambientes
- Resultados muito legais atualmente
 - AlphaGo, OpenAI 5, AlphaStar
- Investimento legal e crescente de empresas
 - DeepMind 
 - Google Brain 
 - OpenAI
 - FAIR...
- Muito divertido
- Cientificamente interessante e com um potencial imenso
 - Ir além de automatizar o comportamento humano



Amount of Search per Decision



Aplicações

- Robótica
- Jogos
- Geração procedural
- Interação com usuário
- Neurociência
- Sistemas de recomendação
- Investimento

Divertido ver seu agente

[DeepMind Breakout](#)

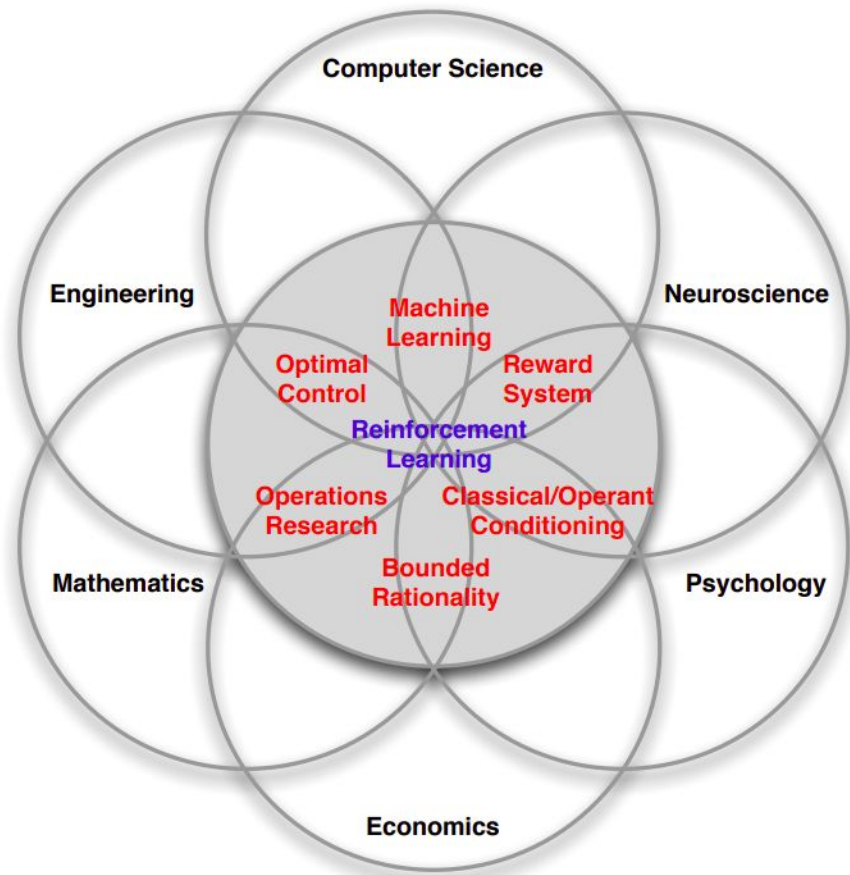
[Simulated Robot Walking](#)

[Mar/IO](#)

[OpenAI Hide and Seek](#)

Os que vamos aprender a treinar

Interação com outras áreas do conhecimento



Definições básicas

- Dois significados
 - Técnica de modelagem de problemas
 - Método de resolução de problemas
- Diferenças do resto de ML
- Agente, ambiente, observações e ações
- História e Estado
- Tipos de estado
- Value functions, policy e retorno
- Taxonomia dos agentes
- Exploration vs Exploitation

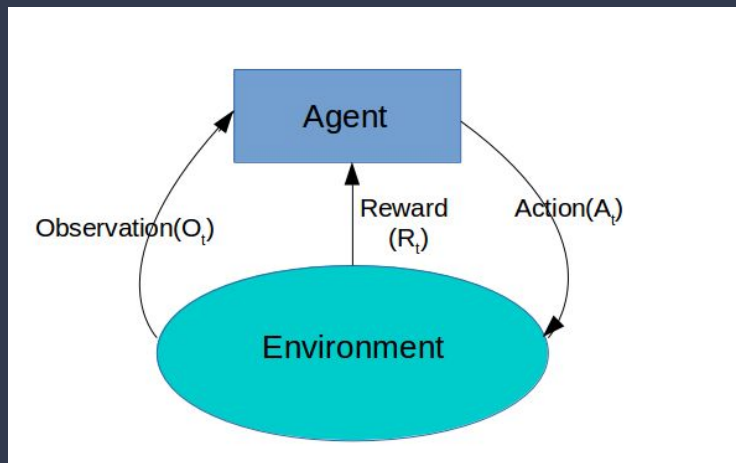
Dois significados

- Técnica de modelagem de problemas
 - Qualquer problema que possa ser modelado como um agente interagindo com um ambiente
- Método de resolução de problemas baseado em MDPs
 - DP, Monte-Carlo Control, Deep Q-learning, Policy gradients...

Diferenças do resto de ML

- Dados sequenciais e não i.i.d
- Não existe supervisor, só um reward
- Feedback não é instantâneo
- As ações de um agente afetam as observações que ele vai receber

Agente, Ambiente, observações e ações



- Em cada instante t :
 - O agente:
 - Recebe observação O_t e Reward R_t
 - Toma ação A_t
 - O ambiente
 - Recebe ação A_t
 - Comunica observação O_{t+1} e Reward R_{t+1}
- Rewards
 - Escalares
 - O feedback do ambiente
 - Indica o quão bem o agente está indo
 - O agente quer maximizar o reward cumulativo
 - **Reward Hypothesis:** Todo objetivo pode ser formulado como a maximização da esperança do reward cumulativo

História e Estado

- História é tudo que o agente experienciou até agora, ações, observações e rewards

$$H_t = O_1, R_1, A_1, \dots, A_{t-1}, O_t, R_t$$

- O estado é a informação necessária para decidir o que vai ocorrer a seguir
 - Uma função da história

$$f(H_t) = S_t$$

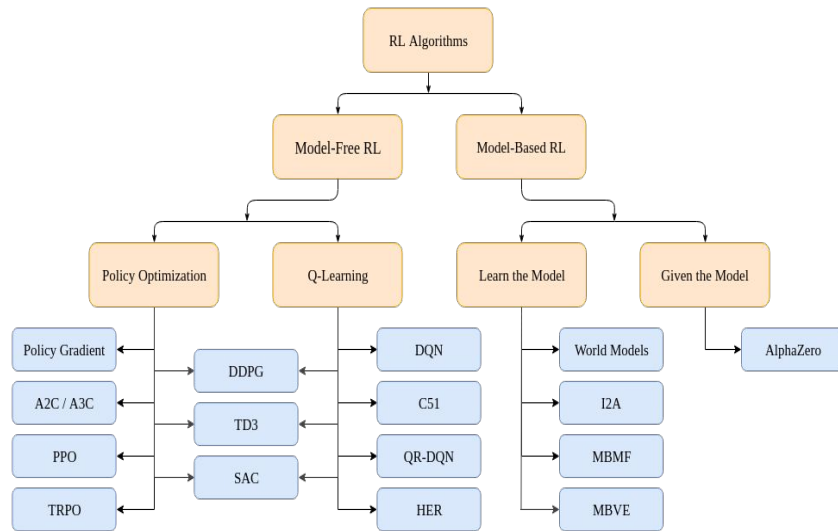
- O estado do ambiente é usado para determinar a próxima observação e reward (não precisa ser determinístico)
- O estado do agente é usado para determinar a próxima ação (também não precisa ser determinística)

Estado do Ambiente VS Estado do Agente

- Estado do ambiente S_t^e é a representação privada do ambiente
 - Em geral não é visível para o agente
 - Pode conter informação irrelevante
- Estado do agente S_t^a é a representação interna do agente
 - É a informação usada por algoritmos de RL
- Um estado é dito Markov se possui a **propriedade Markoviana**
 - $P[S_{t+1} | S_t] = P[S_{t+1} | S_1, \dots, S_t]$
 - "O futuro é independente do passado dado o presente"
 - Basicamente dado o estado não precisamos nos preocupar com a história
- O estado do ambiente é markov
- A história é markov
- MDPs: $S^e = S^a$
- POMDPs: $S^e \neq S^a$

Taxonomia dos algoritmos

- Model Based
- Value Functions
- Policy based



Value Functions, Policy e Retorno

- A policy determina como o agente deve agir, mapeando de estados para ações
 - Pode ser determinística ou probabilística
- Retorno: O reward cumulativo
 - $G_t = \sum R_{t+1+i}$
 - Reward Hypothesis
- Funções de valor
 - A função de valor de estado representa o quanto de retorno o agente espera receber do estado
 - $v_{\pi}(s) = E_{\pi}[G_t | S_t = s]$
 - A função de valor da ação representa o quanto de retorno o agente espera receber ao tomar a ação a no estado s estado
 - $q_{\pi}(s, a) = E_{\pi}[G_t | A_t = a, S_t = s]$

Policy-Based

- Aprende uma policy
- Vantagens:
 - É exatamente o nosso objetivo
 - Facilmente extensível para muitas dimensões
 - Consegue aprender policies não determinísticas facilmente
- Desvantagens:
 - Ignora informação aprendível

Value-Based

- Usam funções de valores para guiar suas decisões
- Vantagens:
 - Próxima do objetivo real
 - Relativamente bem entendida
- Desvantagens
 - Não é o objetivo real, pode gastar poder computacional em pontos irrelevantes
 - Para usar v precisamos de um modelo, para usar q precisamos computar um argmax

Model-Based

- O agente recebe um modelo ou tem que aprendê-lo
- Um modelo é:
 - Como o agente acredita que o mundo funciona
 - Composto das transições $P(S_{t+1} = s' | a_t, s_t)$
 - e dos rewards: $P(R_t = r | a_t, s_t)$
- Vantagens:
 - Aprender um modelo é 'fácil'
 - Usa os dados ao máximo, o que o torna em geral mais sample efficient
- Desvantagens
 - O objetivo envolve informações irrelevantes
 - Existe um custo computacional a mais para aprender o modelo
 - Planejamento (computar uma policy a partir de um modelo) é não trivial e computacionalmente custoso

Exploration vs Exploitation

- As decisões do agente afetam com que partes do ambiente ele interage
- Idealmente o agente deve descobrir uma boa policy a partir de sua experiência do ambiente, sem perder muito reward para isso
- Exploration descobre mais sobre o ambiente
- Exploitation usa o que você sabe para maximizar o retorno
- Ambas são extremamente importantes
- É não trivial achar um equilíbrio
 - epsilon-greedy
 - Stochastic policies
 - Bandits...

Recursos úteis

- Environments
- Bibliotecas
- Misc
 - Documentário AlphaGo na wikipedia
 - Blog da DeepMind
 - Blog da OpenAI
 - Blog do David Ha
 - How Smart Machines Think
 - Artificial Intelligence: A Guide for Thinking Humans

Environments

- OpenAI Gym
 - Most famous, standard in academia and industry
 - Unified API for many different envs
 - Implements ALE
 - Has both discrete and continuous tasks
 - Extensible
 - Maxine's minigrid
 - Maxine's miniworld
 - Little fighter
- Deepmind OpenSpiel
 - Classical multiplayer games
 - Go
 - Chess
 - Hanabi
 - Backgammon
- Bsuite
 - Analisa os comportamentos do agente

Gridworlds

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0



1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

The environment grid is a 4x4 grid with numbered cells (1-16). Cells 1 and 16 are yellow. Cells 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, and 15 contain a colored dot (red, green, or purple) and a black arrow indicating a possible action. The dots are located at (2,2) red, (3,3) green, (5,2) red, (6,3) purple, (9,3) green, (11,3) purple, (12,2) red, (14,4) green, and (15,4) red. The arrows indicate possible actions from each cell: (2,1) left, (3,1) left, (4,1) left-down, (5,1) up, (6,1) up-left, (7,1) left-down, (8,1) down, (9,1) up, (10,1) left-down, (11,1) right-down, (12,1) down, (13,1) left-down, (14,1) right, (15,1) right, and (16,1) right.

Bibliotecas

- Dopamine
 - Focada em implementar o Rainbow
- TF-agents
 - Focado em deployment
- RLlib
 - Escalável
- RLax
 - Baseada em JAX/Haiku
 - Trocadilho muito bom

Como tirar o máximo do curso

- Vir às aulas
- Fazer perguntas
- Implementar os algoritmos
 - Spinning Up Deep Reinforcement Learning
- David Silver Course
- Deep RL bootcamp
- Berkeley Course
- Sutton & Barto

Programa

1. Introdução, terminologia e environments
2. Exploration vs Exploitation: Bandits
3. Bellman Equations, backup diagrams, Exhaustive Search, DP, Value iteration and Policy Iteration
4. Monte Carlo, TD, SARSA, Q-learning
5. Function Approximation: DQN and Policy Gradient
6. Value-based Agents
7. Policy-based Agents
8. Model-based Agents
9. Classic games
10. Evolution Strategies, NEAT