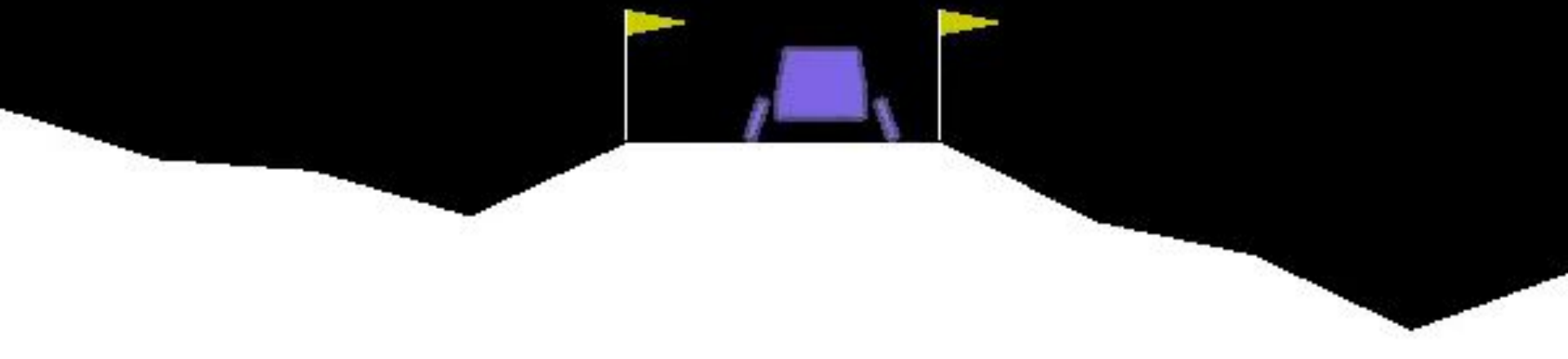
Deep Q-Networks (DQN)

GRUPO TURING

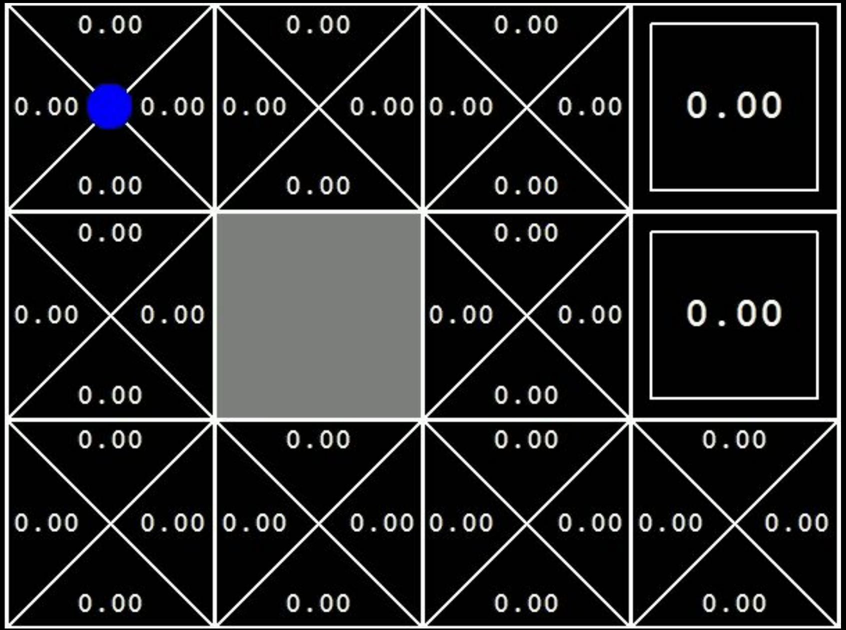
Episódio 0000

Roteiro

- Recap de Q-Learning tabular
- Limitações de Q-Learning tabular
- Deep Q-Networks (DQNs) e aproximação da função de valor
- Intervalo
- Prática (Lunar Lander)



Recap: Q-Valores



$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots$$

$$q_\pi(s, a) = E_\pi\Big[\, G_t \;\Big|\; S_t = s,\, A_t = a \,\Big]$$

Recap: Q-Learning

$$q_\pi(s, a) = E_\pi\left[G_t \,\middle|\, S_t = s, A_t = a \right]$$

Retorno:

$$\begin{aligned} G_t &= R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots \\ &= R_t + \gamma \cdot \left(R_{t+1} + \gamma^2 R_{t+2} + \cdots \right) \\ &= R_t + \gamma G_{t+1} \end{aligned}$$

Valor de bootstrap: $Q_{\text{bootstrap}}(s, a) = r + \gamma \max_{a'} Q_*(s', a')$

Recap: Q-Learning

$$Q_{\text{bootstrap}}(s,a) = \underbrace{r}_{\text{recompensa}} + \underbrace{\gamma}_{\substack{\text{fator de}\\ \text{desconto}}} \cdot \underbrace{\max_{a'} Q_*(s',a')}_{\substack{\text{estimativa do}\\ \text{valor futuro ótimo}}}$$

- Atualização das estimativas Q_*:

$$Q_*(s,a) \leftarrow \underbrace{Q_*(s,a)}_{\substack{\text{valor}\\ \text{antigo}}} + \underbrace{\alpha}_{\substack{\text{taxa de}\\ \text{aprendizado}}} \cdot \overbrace{\left(\underbrace{r + \gamma \max_{a'} Q_*(s',a')}_{Q_{\text{bootstrap}}(s,a)} - Q_*(s,a)\right)}^{\text{erro}}$$

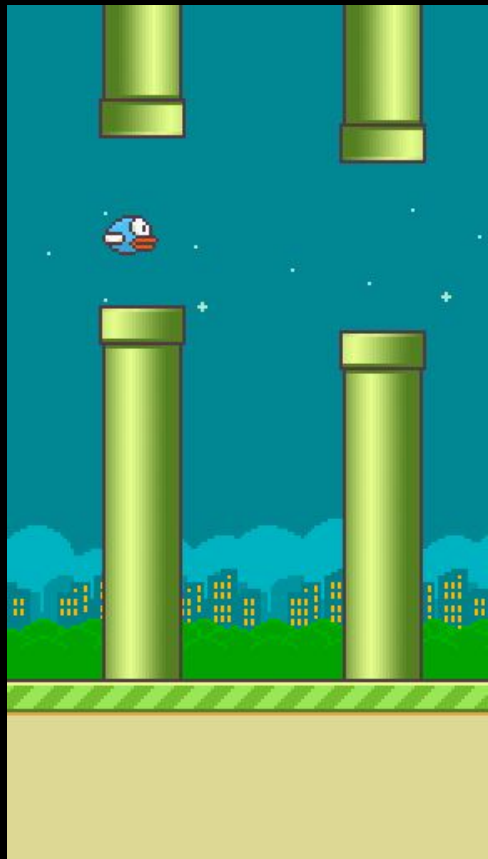
Recap: Q-Learning Tabular

- Os Q-valores Q(s,a) são armazenados numa tabela, com uma célula para cada par (s,a)

Q	a0	a1	a2
s0	1	**15**	2
s1	**24**	5	16
s2	10	**62**	-7
s3	10	15	**35**

Limitações da Abordagem Tabular

- Em tarefas com espaços de estados grandes:
  - Estado = imagem preto e branco 10x10
    Cada pixel: 256 possibilidades
    Total: $256^{10 \times 10} = 6{,}7 \times 10^{240}$
  - Estado contínuo:
    Quantidade “infinita” de estado (é necessário discretizar)



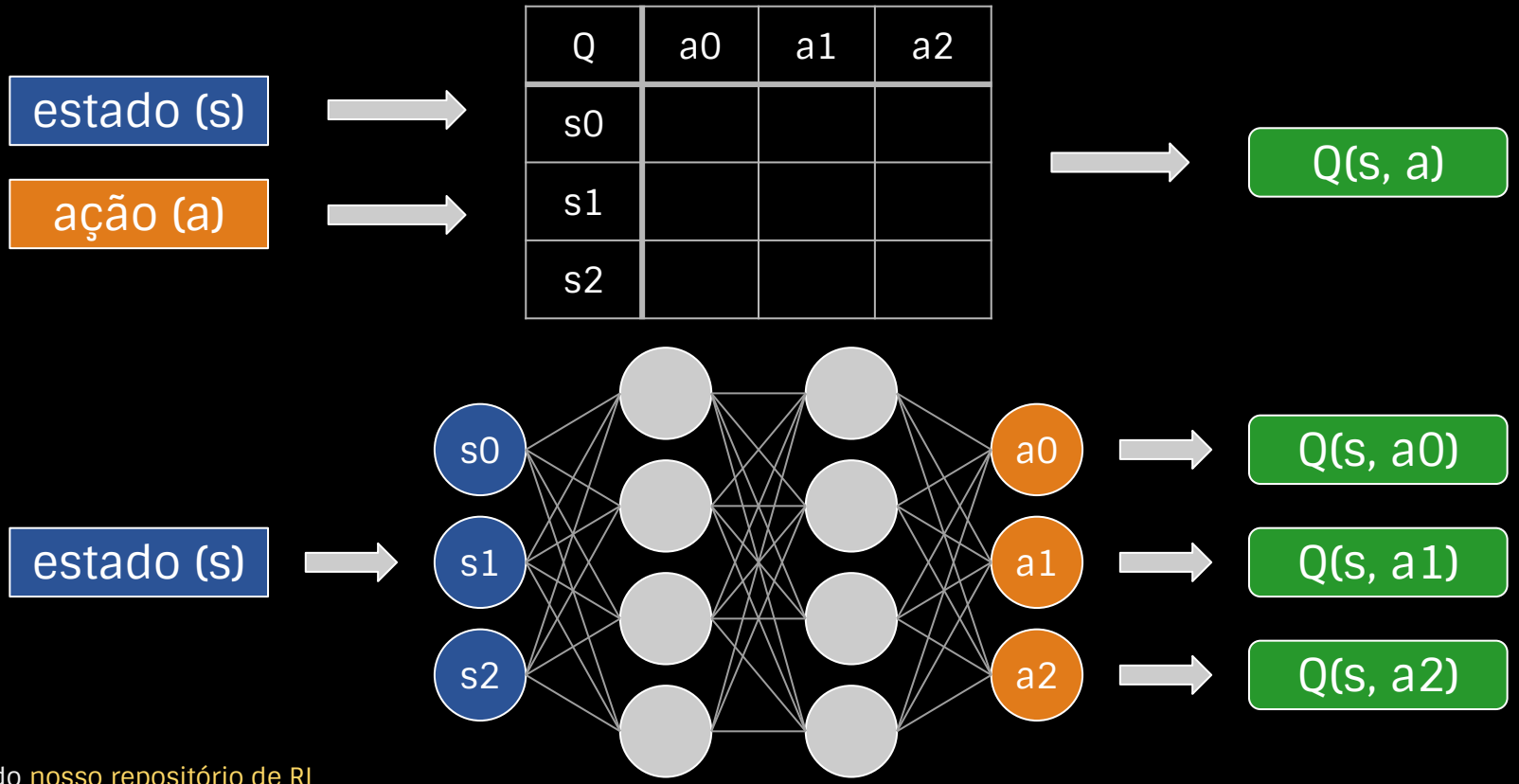
Limitações da Abordagem Tabular

- Em tarefas com espaços de estados grandes, a abordagem tabular demanda:

1. uma tabela imensa
2. uma quantidade absurda de experiências

Q	a0	a1	a2
s0	1	15	2
s1	24	5	16
s2	10	62	-7
s3	10	15	35

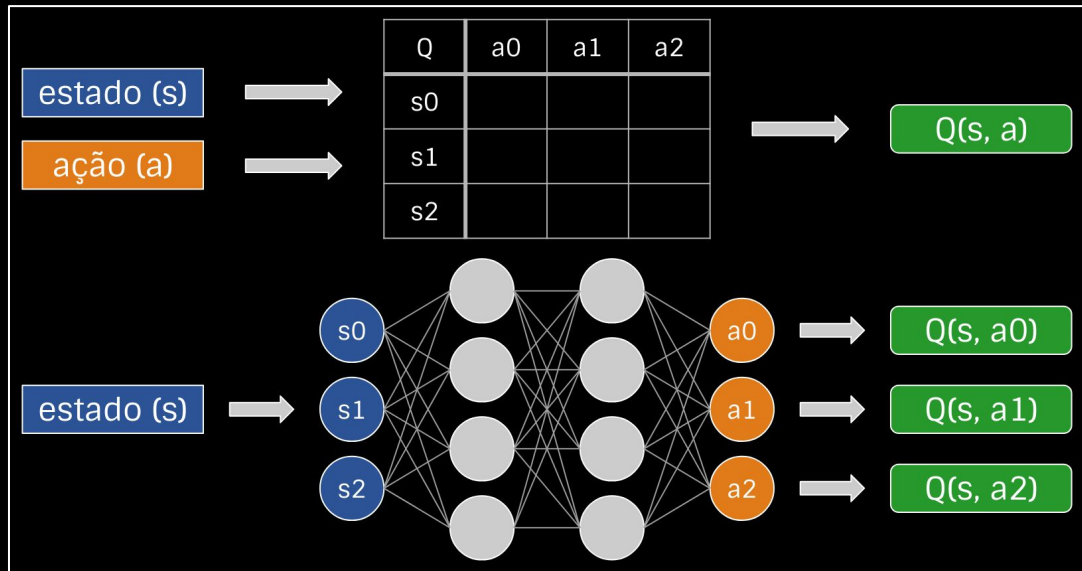
Deep Q-Networks (DQN)



Adaptado do nosso repositório de RL

Deep Q-Networks

- Uma saída por ação:
  - Intuição
    - as primeiras camadas pré-processam o estado (igual para todas as ações)
    - as últimas camadas calculam o valor de cada ação



Deep Q-Networks

- Treinamento com gradient descent
  - Q-Learning como minimização de erro:

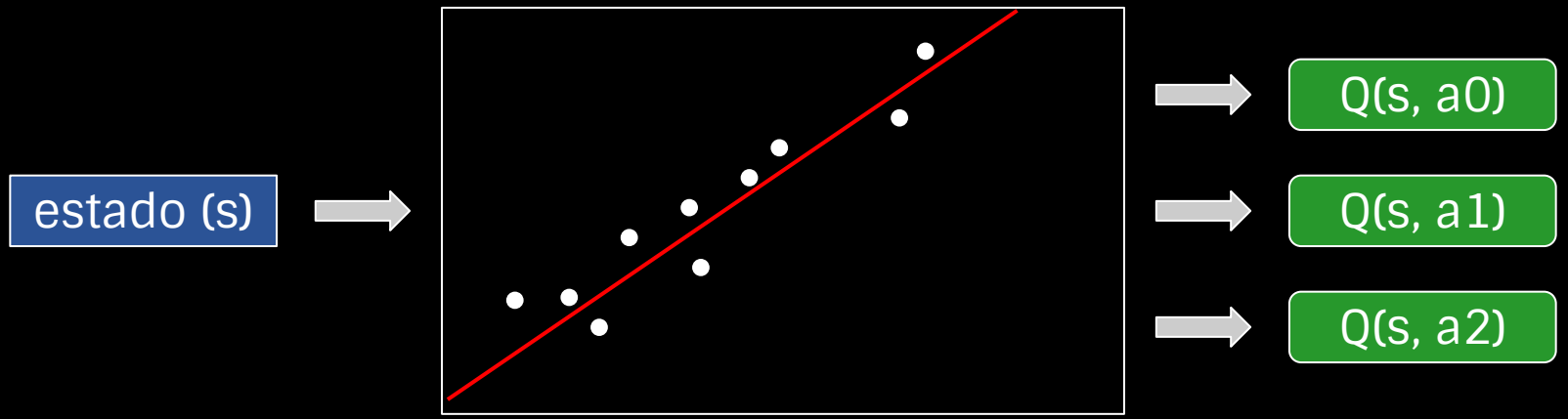
$$Q_*(s,a) \leftarrow \underbrace{Q_*(s,a)}_{\text{valor antigo}} + \underbrace{\alpha}_{\text{taxa de aprendizado}} \cdot \overbrace{\left(\underbrace{r + \gamma \max_{a'} Q_*(s',a')}_{Q_{\text{bootstrap}}(s,a)} - Q_*(s,a)\right)}^{\text{erro}}$$

$$-\nabla J(w)$$

  - Passando para redes neurais:
    DQN corresponde a um passo de gradient descent em direção a $Q_{\text{bootstrap}}(s, a)$
  - Gradient descent: $w \leftarrow w - \alpha \cdot \nabla J(w)$

Aproximação da Função de Valor

- Técnicas que substituem a tabela de Q-Learning por algum estimador, como uma rede neural (DQN) ou uma regressão linear



Experience Replay

- Treinamento linear

t	s	a	r	s’	terminal
3	S_3	A_3	R_3	S_4	não

  - o agente pode “se esquecer” das experiências antigas

- Treinamento com experience replay
  - o agente relembra as experiências antigas
  - as experiências são menos correlacionadas:
    - melhor para a rede neural

replay buffer

t	s	a	r	s’	terminal
1	S_1	A_1	R_1	S_2	não
2	S_2	A_2	R_2	S_3	não
3	S_3	A_3	R_3	S_4	não
4	S_4	A_4	R_4	S_5	não
5	S_5	A_5	R_5	S_6	não
⋮	⋮	⋮	⋮	⋮	⋮

Algoritmo de DQN

Parâmetros: parâmetros $\alpha, \gamma \in (0, 1]$, ε pequeno > 0.
Inicialize a *memória de replay* D e a *rede neural* $Q(s,a)$
Loop para cada episódio:
    Inicialize s_t
    Loop para cada instante t do episódio:
        Escolha a_t usando uma política ε-gulosa
        Tome a ação a_t, observe r_t, s_{t+1}
        Guarde a transição (s_t, a_t, r_t, s_{t+1}, terminal) em D
        Amostre um conjunto de transições de D
        Calcule $Q_{bootstrap}$ para cada transição amostrada:

$$Q_{bootstrap}(s_j, a_j) = \begin{cases} r_j & \text{se } s_j \text{ for terminal} \\ r_j + \gamma \max_{a'} Q(s_{j+1}, a') & \text{caso contrário} \end{cases}$$

        Treine a rede neural utilizando $Q_{bootstrap}$ como target
    até que s_{t+1} estado terminal

Adaptado do nosso repositório de RL

Intervalo (aprox. 15 min)

Em seguida: parte prática

Repositório da aula: github.com/GrupoTuring/Aula-Aberta-DQN

Recursos:

- Repositório de RL
- Turing Talks: post de DQN com Flappy Bird
- Spinning Up (OpenAI)