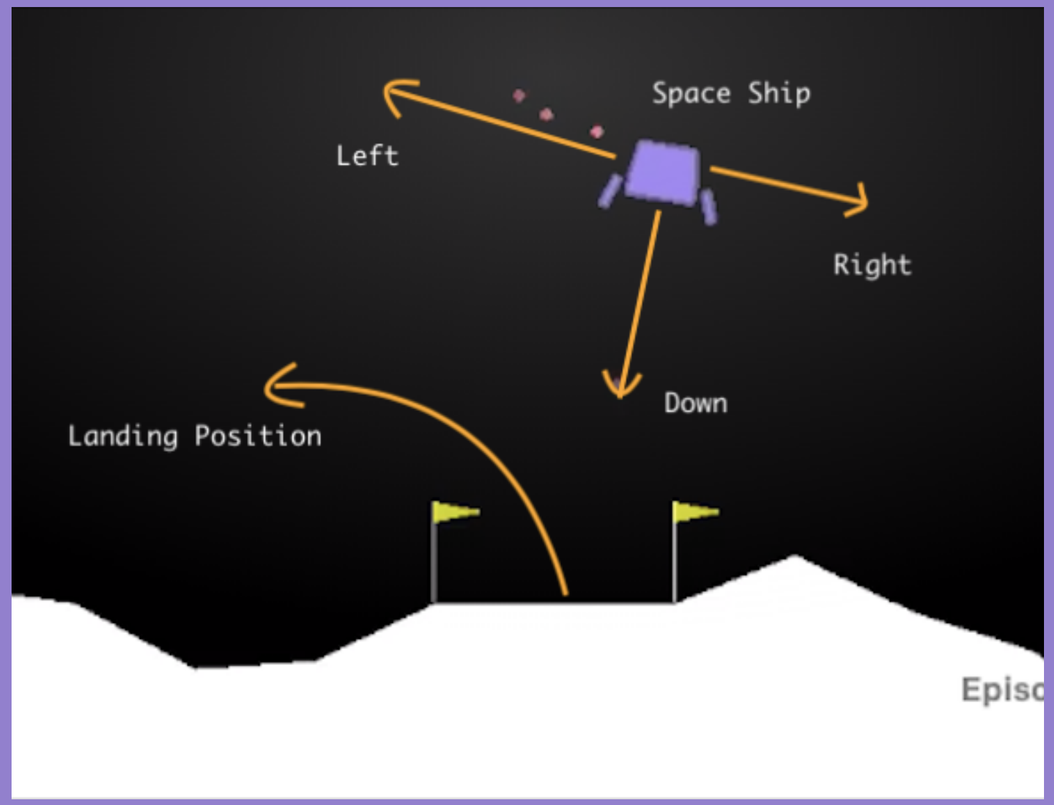
LUNAR - LANDER

Bruno Fornaro
Juan Belieni
Vanessa Berwanger Wille



Sobre

Objetivo: Ensinar um Módulo Lunar a pousar com segurança em uma plataforma de pouso fixada no ponto (0,0).

Espaço de ação:

- 0: não faça nada
- 1: motor de orientação à esquerda
- 2: motor principal de orientação para baixo
- 3: motor de orientação à direita

Espaço de observação: vetor de 8 dimensões

- Posição do agente (coordenadas x e y)
- Velocidades lineares (em x e y)
- Ângulo
- Velocidade angular
- Booleanos que representam se cada perna está em contato com o solo ou não.

Recompensas

Para cada etapa, a recompensa:

- Aumenta ao:
  - Se aproximar da plataforma de pouso
  - Se movimentar mais lento
- Diminui ao
  - Se afastar da plataforma de pouso
  - Se movimentar mais rápido
  - Inclinar o módulo de pouso
- + 10 pontos para cada perna em contato com o solo
- - 0,03 pontos a cada quadro que um motor lateral está disparando
- - 0,3 pontos a cada quadro que o motor principal está disparando
- - 100 por bater ou +100 pontos por pousar com segurança

Um episódio é considerado solução se obtiver pelo menos 200 pontos.

Término

O episódio termina se:

- o módulo de pouso cai (o corpo do módulo de pouso entra em contato com a lua);
- o módulo de pouso sai da janela de visualização (a coordenada x é maior que 1);
- o módulo de pouso não está acordado. Dos documentos do Box2D, um corpo que não está acordado é um corpo que não se move e não colide com nenhum outro corpo

Algortimos

Q-LEARNING

Q-Learning atualiza o valor Q usando o valor Q do próximo estado e a ação gananciosa depois disso.

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(r_{t+1} + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t))$$

SARSA

O algoritmo SARSA é uma modificação do Q-learning, não adotando a maximização das ações. Atualiza o valor Q usando o valor Q do próximo estado e a próxima ação da política.

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$$

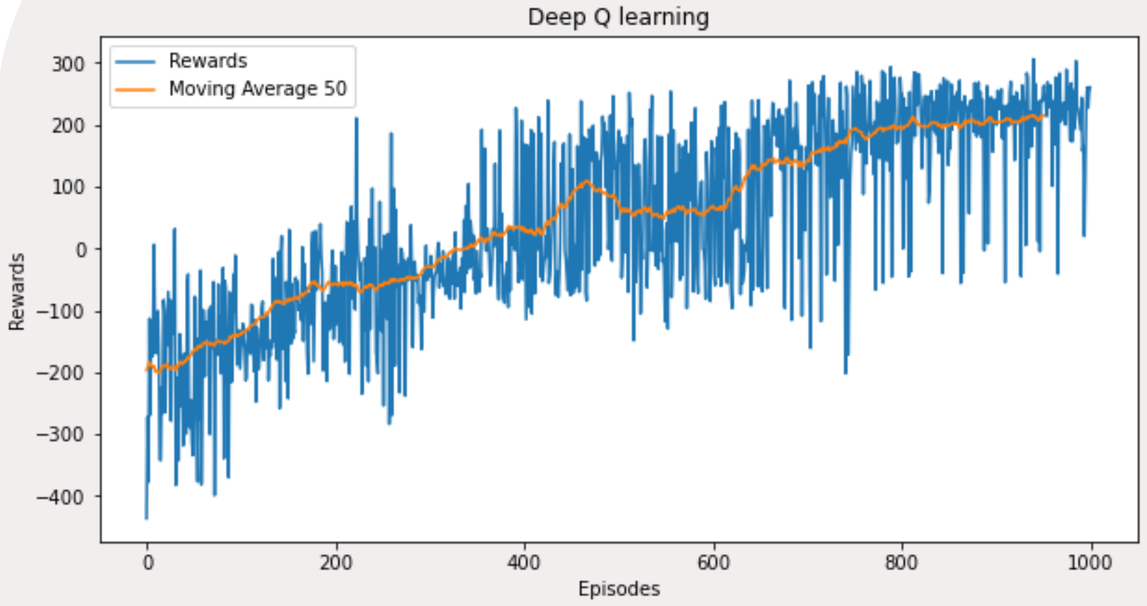
No caso do Lunar-lander a representação do estado é contínua, logo usamos uma rede neural para estimar os valores de ação do estado e para construir uma política ideal para um determinado agente, ou seja, trabalhamos com os algoritmos **Deep Q-learning** e **Deep Sarsa**.

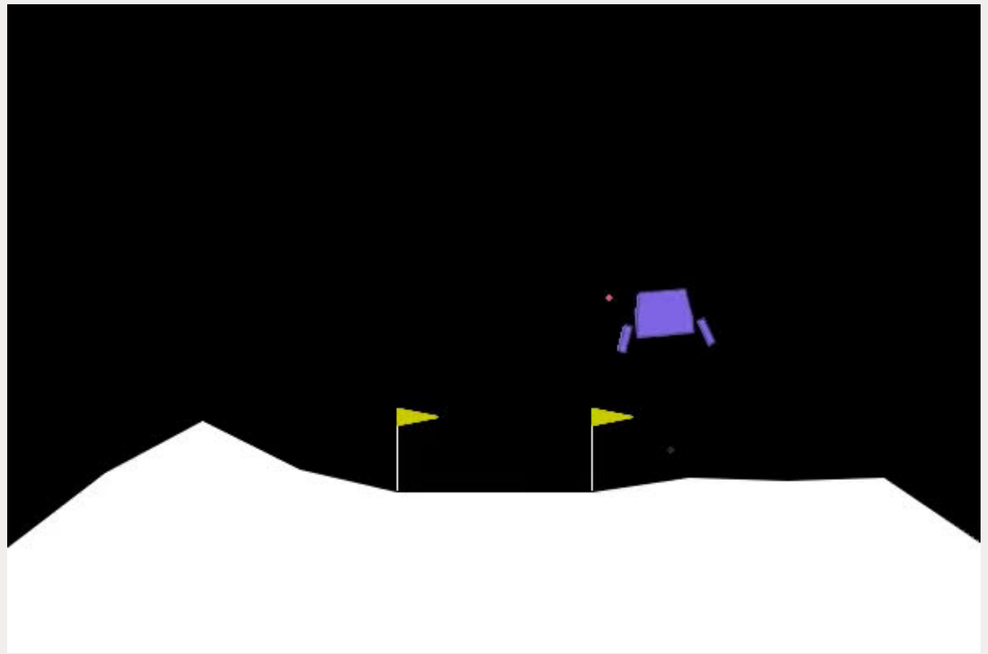
Algortimo Deep Q-learning (DQN)

- Usa uma rede neural multicamadas para estimar a tabela Q para (os valores de ação para um determinado estado).
- Usa um buffer de repetição para amostrar as informações do que aconteceu no episódio e para atualizar/treinar a rede neural.
- Usa a estratégia epsilon-greedy para exploração. Em alguns casos, escolhe a melhor ação conhecida (greedy), aquela que tem a estimativa mais alta de ser a melhor escolha e, ocasionalmente, o agente também faz escolhas aleatórias (exploração), selecionando uma ação aleatória.

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(r_{t+1} + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t))$$

Resultados


Deep Q learning
Rewards
Moving Average 50
300
200
100
0
−100
−200
−300
−400
Rewards
0
200
400
600
800
1000
Episodes



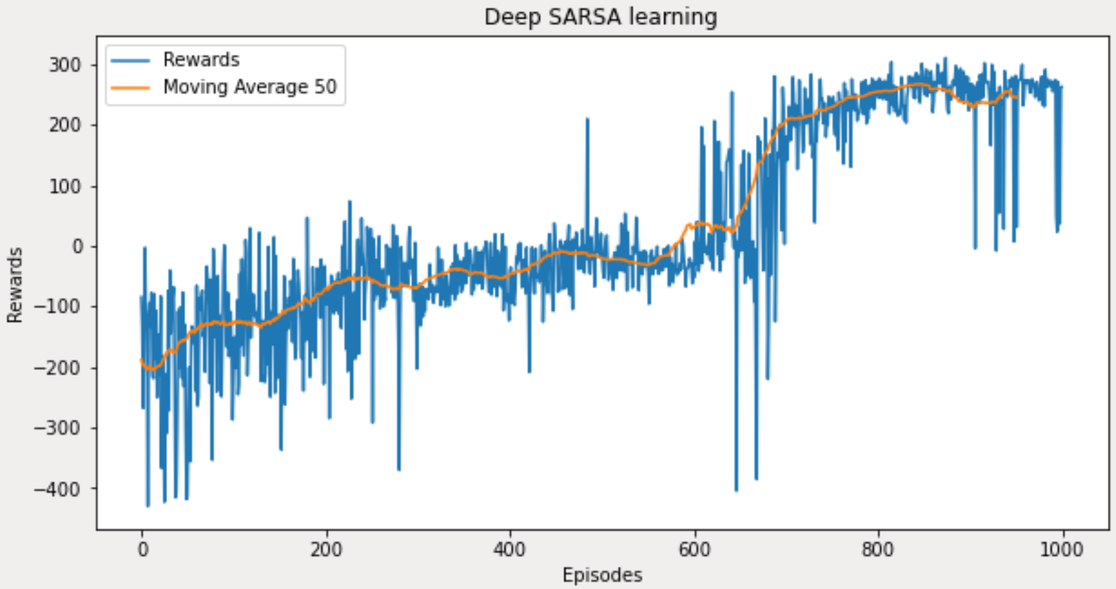
Algortimo Deep Sarsa (s, a, r, s ', a')

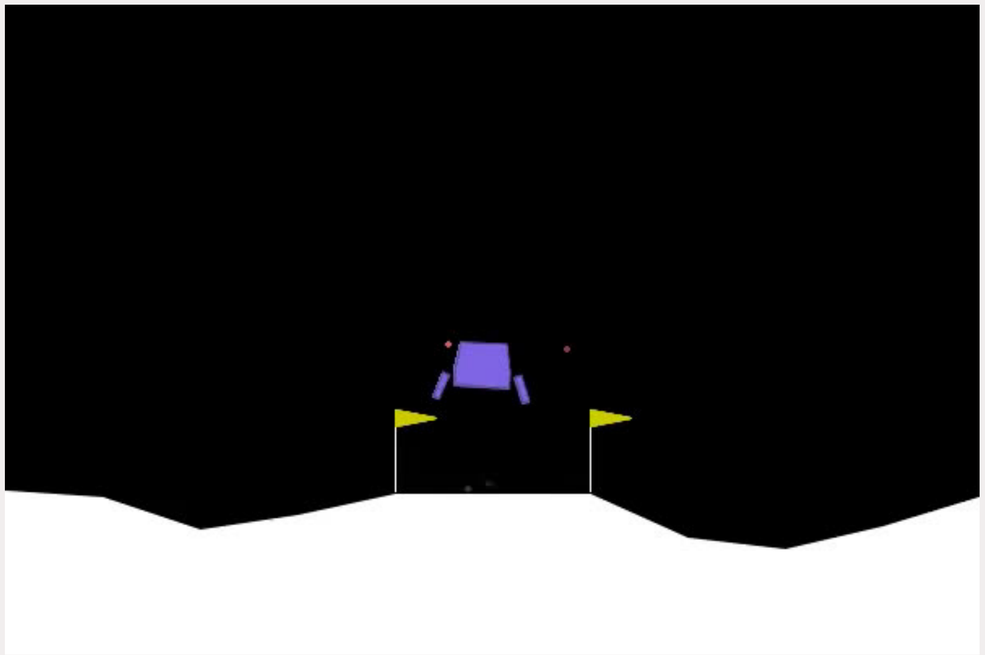
- Faz o uso das mesmas ideias do algoritmo DQN.

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$$

  - Parâmetros:
    - γ= 0.99
    - α = 5e-4
    - 1000 *steps*
    - 1000 *epochs*

Resultados


Deep SARSA learning
Rewards
Moving Average 50
300
200
100
0
−100
−200
−300
−400
Rewards
0
200
400
600
800
1000
Episodes



Ambiente aleatório

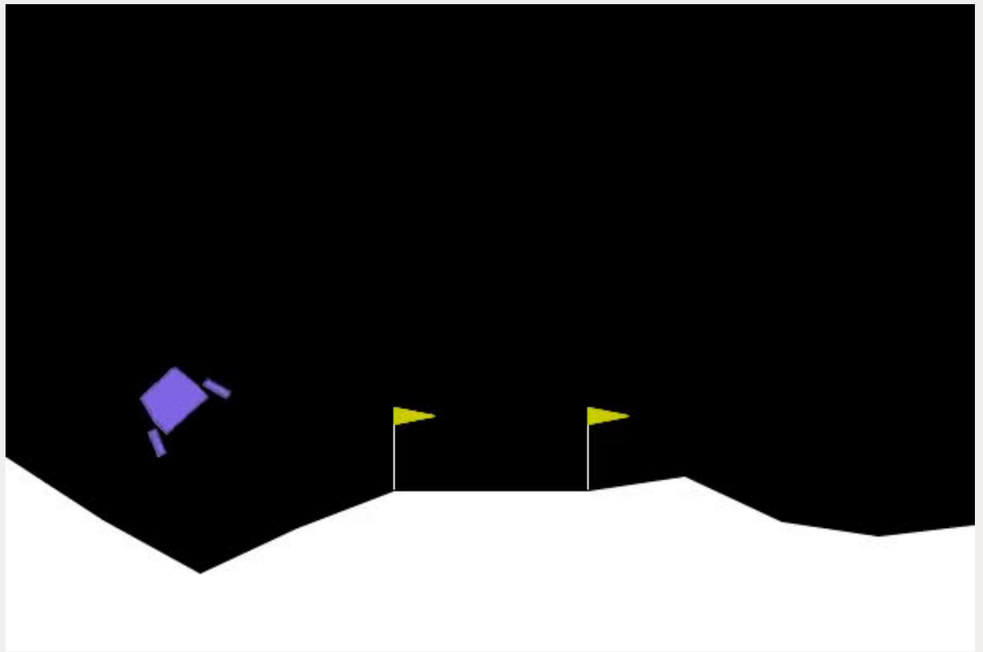
- Menos *steps* de mais *envs* *x* Mais steps de uma *env*
- Randomização dos parâmetros dos ambientes (*envs)*
  - *enable_wind*
  - *wind_power*
  - *turbulence_power*

- Tenta aprender a vencer o jogo em diferentes ambientes com um único modelo

- Modelos utilizados:
  - SARSA
  - A2C

Resultados SARSA (ruins)

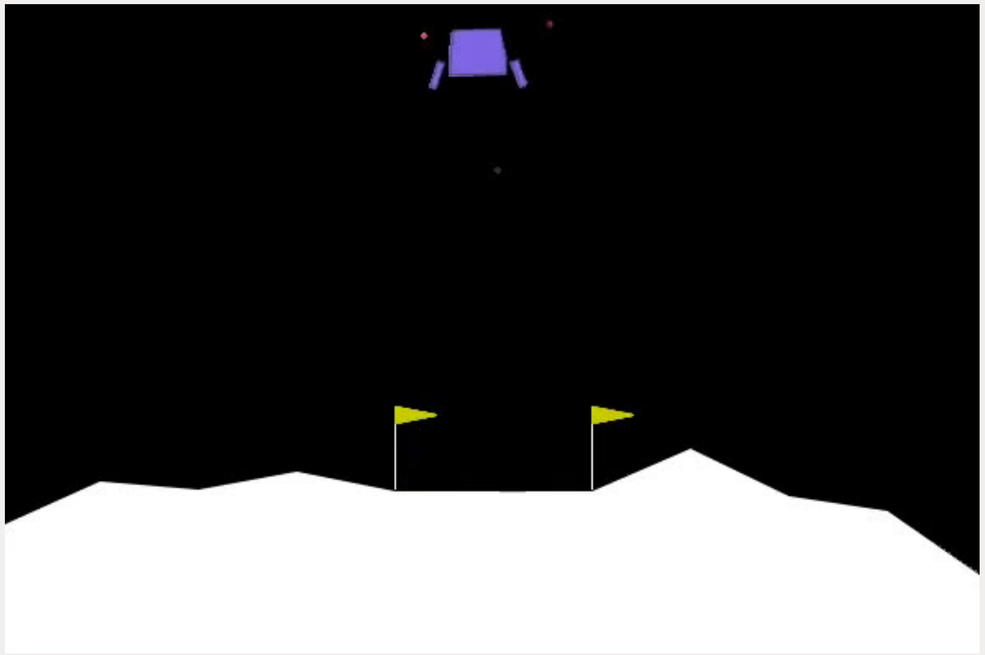
- Mais envs: melhora a estabilidade
- Valor pequeno para *steps:*
  - *Reward* inicial maior
  - Platô de *reward* baixo
  - Desliga a nave e para de aprender



Resultados SARSA (“bons”)

- Randomize = False
- Parâmetros:
  - 250 *envs*
  - 250 *steps*
  - 100.000 *epochs*

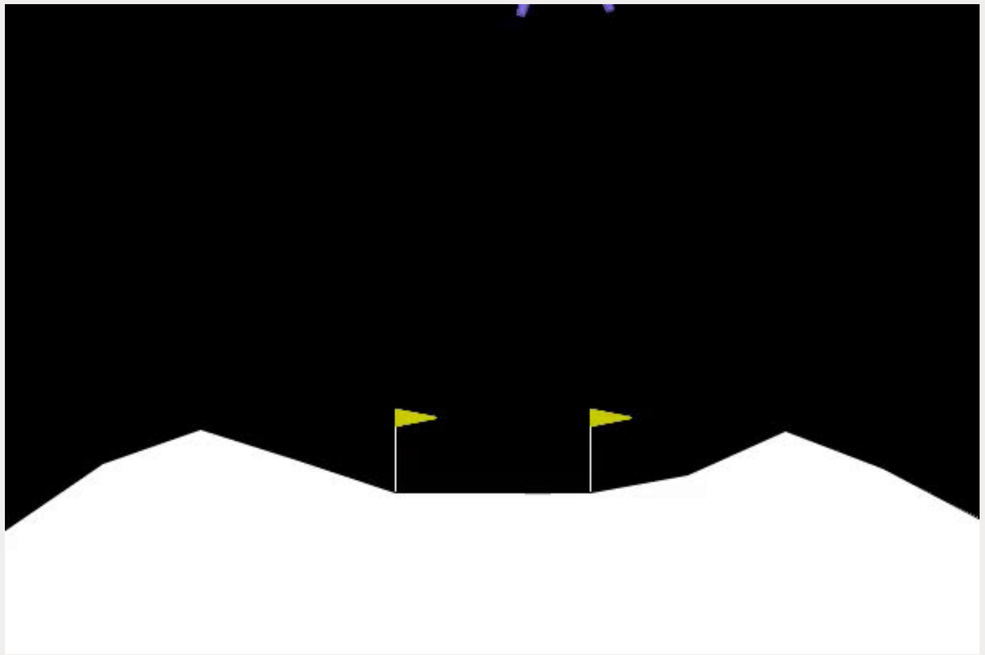
- Melhor reward (média): 125



Resultados SARSA (“bons”)

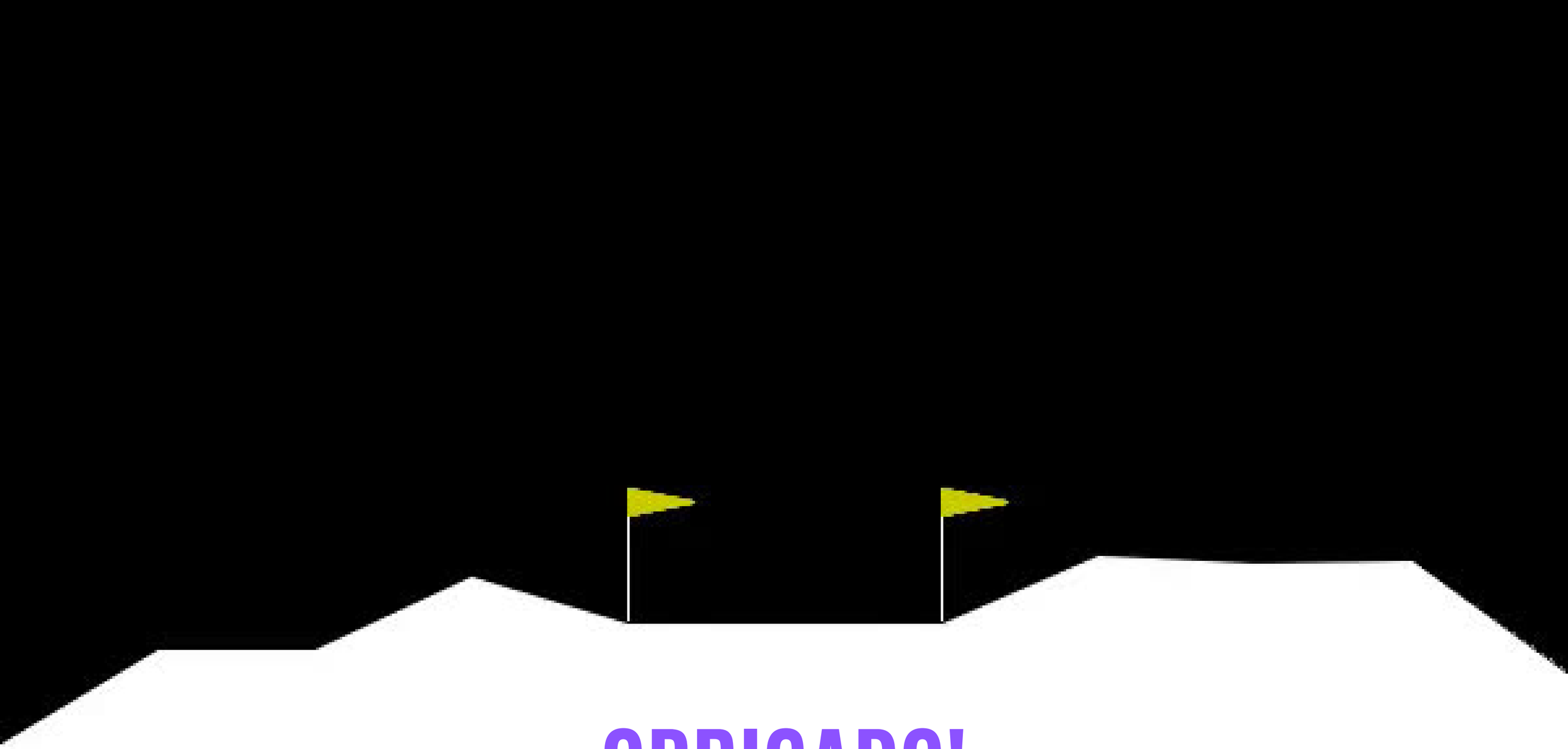
- Randomize = True
- Parâmetros:
  - 250 *envs*
  - 250 *steps*
  - 100.000 *epochs*

- Melhor reward (média): 128



Conclusões

- Sem o ambiente aleatório resolvemos o jogo sem muito esforço
- Com o ambiente aleatório:
  - A complexidade do problema aumenta muito (não conseguimos uma solução consistente)
  - A instabilidade do aprendizado aumenta muito
  - O ator aprende “estratégias” ruins (como desligar a nave) e demora para sair delas



OBRIGADO!