1 2 9 0

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

Departamento de Engenharia Informática

Fundamentos de Inteligência Artificial
2024/2025 - 2º Semestre

Trabalho Prático Nº1:
Lunar Lander

Nota: A fraude denota uma grave falta de ética e constitui um comportamento inadmissível num estudante do ensino superior e futuro profissional licenciado. Qualquer tentativa de fraude levará à anulação da componente prática tanto do facilitador como do prevaricador, independentemente de ações disciplinares adicionais a que haja lugar nos termos da legislação em vigor. Caso haja recurso a material não original, as **fontes** devem estar explicitamente indicadas. Caso use ferramentas de IA na produção deste trabalho (e.g. ChatGPT), deverá identificar de forma clara todas as partes em que a ferramenta esteve envolvida. Note que, durante a defesa, deverá demonstrar ter conhecimento profundo dos conteúdos gerados pela ferramenta, sendo esse conhecimento objeto de avaliação.

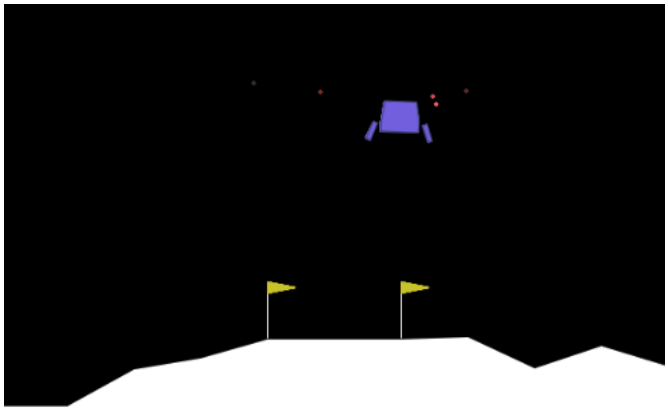

Figura 1: Lunar Lander

1 Introdução

O Lunar Lander é um ambiente de simulação frequentemente utilizado para desenvolver e testar métodos de Inteligência Artificial, tais como aprendizagem por reforço. O objetivo é controlar uma nave espacial, aterrando-a com segurança numa plataforma delimitada por duas bandeiras. A nave tem 3 motores que permitem controlar o seu movimento. O motor principal está montado sob a nave e move-a na direção em que está orientada. A nave contém ainda dois motores secundários, montados nas suas laterais, que permitem alterar a sua orientação.

2 Objectivos Genéricos

O presente trabalho tem como objetivos genéricos:

1. Aprender a desenhar um agente reativo adequado a um problema e ambiente específico.
2. Aprender a especificar e formalizar os seguintes aspetos de um agente reativo:
   (a) Perceções
   (b) Ações
   (c) Sistema de produções
   (d) Memória
3. Aprender a fazer uma descrição de alto-nível do comportamento desejado implementável através de um agente reativo.
4. Aprender a criar um agente reativo a partir de uma descrição de alto nível.

3 Lunar Lander

Este trabalho prático tem como objetivo principal a aquisição de competências relacionadas com a análise, desenvolvimento, implementação e teste de sistemas de produções para o controlo de agentes autónomos. Para tal, usar-se-á o Lunar Lander como plataforma de aprendizagem. Importa assim descrever quais os componentes fundamentais deste problema, nomeadamente o ambiente e o agente.

3.1 Ambiente

O Lunar Lander consiste num ambiente 2D, contemplando espaço e a superfície lunar, onde está a plataforma de aterragem (nas coordenadas (0,0)). A simulação começa com a nave a pairar sobre a zona central da janela com uma orientação inicial aleatória. O objectivo é desenvolver um agente que aterre a nave de forma segura, controlando os motores.

Para tal, o agente deve lidar com os seguintes desafios:

1. A nave está sujeita à força da gravidade, puxando-a continuamente para baixo.

2. O agente deve manobrar a nave, controlando a sua posição, velocidade e orientação.

3. Para que uma aterragem seja bem sucedida, devem-se verificar as seguintes condições:

   (a) Manter uma velocidade vertical baixa.

   (b) Aterrar com uma orientação aproximadamente vertical.

   (c) Aterrar na zona delimitada pelas duas bandeiras, tendo ambas as "pernas"em contacto com o solo.

A simulação passa-se em tempo discreto, sendo que a cada momento (passo de simulação), o agente recebe uma observação do ambiente, executa uma ação com base no seu controlador e transita para o próximo estado.

3.2 Agente

O agente a desenvolver deverá usar informação extraída das observações do ambiente para controlar a nave, controlo esse que será feito através das ações disponíveis. Assim, importa definir o espaço de observações e ações.

3.2.1 Espaço de Observações

O espaço de observação é representado por um vetor com 8 números reais, codificando o estado da nave no ambiente. A Tabela 1 descreve as componentes deste vetor.

3.2.2 Espaço de Ações

Cada ação é representada por um vetor com dois números reais. O primeiro valor controla o acelerador do motor principal, enquanto o segundo controla os motores secundários.

- **Motor Principal** — É ativado apenas para valores superiores a 0.5, aumentando a sua aceleração de forma linear até atingir o máximo em 1.

- **Motores secundários** — Não são ativados para valores entre -0.5 e 0.5. Para valores inferiores a -0.5, o motor direito é activado, rodando a nave para a esquerda. Para valores superiores a 0.5, o motor esquerdo é activado, rodando a nave para a direita. Assim como o motor principal, as suas acelerações aumentam linearmente até os limites de -1 (motor esquerdo) e 1 (motor direito).

Tabela 1: Espaço de observações.

Índice	Variável	Descrição
0	x	Posição horizontal da nave em relação ao centro (plataforma de aterragem). Negativa à esquerda; positiva à direita.
1	y	Posição vertical da nave em relação ao solo.
2	v_x	Velocidade horizontal da nave. Negativa quando se move para a esquerda; positiva quando se move para a direita.
3	v_y	Velocidade vertical da nave. Negativa quando desce; positiva quando sobe.
4	θ	Orientação da nave. Negativa quando está inclinada para a direita; positiva quando está inclinada para a esquerda.
5	v_θ	Velocidade angular (mudança no ângulo). Negativa quando roda no sentido horário; positiva quando roda no sentido anti-horário.
6	*left_leg_touching*	Booleano (1 ou 0): Indica se a perna esquerda está em contato com o solo.
7	*right_leg_touching*	Booleano (1 ou 0): Indica se a perna direita está em contato com o solo.

Desta forma, é possível controlar os motores de forma progressiva, permitindo ajustes suaves durante o voo ou aterragem.

4 Tarefas

O objetivo deste trabalho consiste na **modelação** de um agente reactivo para controlar o comportamento da nave através de um **sistema de produções**, bem como a **implementação** desse agente. Para tal, é-lhe fornecido um código de base, que poderá encontrar no *UCStudent*.

O código fornecido foi testado em **Python 3.12**. Recomendamos que utilize a plataforma Anaconda para criar um novo ambiente com esta versão de Python. Pode encontrar instruções de instalação do Anaconda aqui. Posteriormente, deverá instalar a framework Gymnasium com os ambientes Box2D, através do comando:

```
pip install "gymnasium[box2D]"
```

Deverá ainda usar o comando abaixo para instalar o módulo pygame, de forma a poder controlar a nave através do teclado:

```
pip install pygame
```

Após a criação do ambiente, e deverá descarregar o projeto do *UCStudent* e abri-lo no seu IDE. O código está estruturado num único ficheiro e, como se pode ver no excerto abaixo, começa por importar os módulos gymnasium, numpy e pygame e definir um conjunto de constantes que influenciam o ambiente criado . **Não deve alterar estes valores numa fase inicial do projeto, podendo apenas alterar o valor do RENDER_MODE para a experimentação**. De seguida, cria-se o ambiente de simulação (linhas 14 a 16) com as constantes previamente definidas.

```
1  import gymnasium as gym
2  import numpy as np
3  import pygame
4
5  ENABLE_WIND = False
6  WIND_POWER = 15.0
7  TURBULENCE_POWER = 0.0
8  GRAVITY = -10.0
9  RENDER_MODE = 'human'
10 #RENDER_MODE = None #seleccione esta opcão para não visualizar o
       ambiente (testes mais rápidos)
11 EPISODES = 1000
12
13 env = gym.make("LunarLander-v3", render_mode=RENDER_MODE,
14     continuous=True, gravity=GRAVITY,
15     enable_wind=ENABLE_WIND, wind_power=WIND_POWER,
16     turbulence_power=TURBULENCE_POWER)
```

De seguida, passa-se à definição das funções que asseguram a simulação, começando pela função *check_successful_landing* (linhas 19 a 39) que verifica se, aquando do término de um episódio de simulação, se a nave aterrou com sucesso. Para tal, a nave deve cumprir um conjunto de requisitos, tais como ter ambas as "pernas"em contacto com o solo, estar entre as bandeiras, ter uma velocidade vertical superior a -0.2 e estar orientada na direção aproximadamente vertical (com um desvio máximo de 20°).

```
19 def check_successful_landing(observation):
20     x = observation[0]
21     vy = observation[3]
22     theta = observation[4]
23     contact_left = observation[6]
24     contact_right = observation[7]
25
```

```
26      legs_touching = contact_left == 1 and contact_right == 1
27
28      on_landing_pad = abs(x) <= 0.2
29
30      stable_velocity = vy > -0.2
31      stable_orientation = abs(theta) < np.deg2rad(20)
32      stable = stable_velocity and stable_orientation
33
34      if legs_touching and on_landing_pad and stable:
35          print("Aterragem bem sucedida!")
36          return True
37
38      print("Aterragem falhada!")
39      return False
```

A função seguinte é a *simulate* que trata da simulação propriamente dita. Esta função recebe três argumentos: o número de passos de simulação (*steps*), a *seed* do gerador de números pseudo-aleatórios (**mantenha a None para simulações diferentes**) e a referência para uma função que implemente o agente (*policy*). A função *simulate* começa por reiniciar o ambiente, obtendo uma observação inicial. De seguida, segue-se o ciclo de simulação (linhas 43 a 49) onde, para cada passo, será chamada a função do agente (linha 44) para que, a partir da observação, produza a ação a executar. Posteriormente, é chamada a função *step* do ambiente (linha 46), que executa a ação, retornando uma nova observação do ambiente, bem como informações relativas ao término da simulação. No final do episódio chama-se a fução *check_successful_landing* (previamente descrita) para verificar se a nave aterrou com sucesso, retornando o número de passos utilizados, juntamente com essa informação.

```
41  def simulate(steps=1000,seed=None, policy = None):
42      observ, _ = env.reset(seed=seed)
43      for step in range(steps):
44          action = policy(observ)
45
46          observ, _, term, trunc, _ = env.step(action)
47
48          if term or trunc:
49              break
50
51      success = check_successful_landing(observ)
52      return step, success
```

Segue-se o local do código destinado à definição das suas percepções e ações (linhas 56 a 60). Deverá definir aqui as funções que entender serem necessárias, deixando os marcadores das linhas 56 e 59 para facilitar a análise do código.

```
56 #Perceptions
57 ##TODO: Defina as suas percecões aqui
58
59 #Actions
60 ##TODO: "Defina as suas acões aqui"
```

Seguem-se a definição de duas funções de agente. A primeira (*reactive_agent*) serve de template à função que deverá desenvolver, substituindo a linha 66 (que neste momento ignora a observação e seleciona uma ação aleatoriamente) pela implementação do seu sistema de produções. A segunda função (*keyboard_agent*) é uma função auxiliar, permitindo-lhe controlar a nave com o teclado para ganhar uma melhor compreensão do ambiente.

```
63 def reactive_agent(observation):
64     ##TODO: Implemente aqui o seu agente reativo
65     ##Substitua a linha abaixo pela sua implementacao
66     action = env.action_space.sample()
67     return action
68
69
70 def keyboard_agent(observation):
71     action = [0,0]
72     keys = pygame.key.get_pressed()
73
74     print('observacão:',observ)
75
76     if keys[pygame.K_UP]:
77         action =+ np.array([1,0])
78     if keys[pygame.K_LEFT]:
79         action =+ np.array( [0,-1])
80     if keys[pygame.K_RIGHT]:
81         action =+ np.array([0,1])
82
83     return action
```

A parte final do código permite correr um número pré-definido (*EPISODES*) de episódios de simulação, calculando métricas da taxa de sucesso (*success*), bem como a média do número de passos das simulações bem sucedidas (*steps*).

```
86 success = 0.0
87 steps = 0.0
88 for i in range(EPISODES):
89     st, su = simulate(steps=1000000, policy=reactive_agent)
90     if su:
91         steps += st
92     success += su
93
94     if su>0:
```

```
95          print('Media de passos das aterragens bem sucedidas:',
                steps/(su*(i+1))*100)
96      print('Taxa de sucesso:', success/(i+1)*100)
```

5 Metas

O presente trabalho prático encontra-se dividido em 2 metas distintas:

Meta 1 – Modelação e desenvolvimento do Sistema de Produções

Nesta meta, deverá definir um conjunto de **perceções** e **ações**, bem como modelar o comportamento do seu agente reativo através de um **sistema de produções**.

Meta 2 – Implementação do Sistema de Produções

Deve implementar as perceções e ações definidas, bem como o sistema de produções, criando assim o seu agente reativo. Deve ainda incluir uma análise da performance do seu agente, que deve abranger a taxa de sucesso, bem como o número de passos das simulações bem sucedidas. Note que dado a componente estocástica do ambiente, deverá fazer um conjunto de experiências representativo (recomenda-se 1000). Sem a presença de vento, deverá atingir uma taxa de sucesso acima de 40%.

Como desafio avançado, desenvolva um agente capaz de lidar com situações mais adversas. Comece por ativar o vento (trocando o valor da constante ENABLE_WIND na linha 5) e analise como é que o seu agente se comporta nesta situação. Faça as alterações necessárias (tanto nas perceções, ações ou sistema de produções) para criar um agente capaz de lidar com o vento e apresente uma análise comparativa com o desenvolvido anteriormente. **Note que este desafio tem uma cotação de 10% da nota.**

6 Datas e Modo de Entrega

Os grupos têm uma dimensão máxima de 3 alunos. A defesa é obrigatória, bem como a presença de todos os elementos do grupo na mesma.

A entrega da meta 1 é opcional, chama-se no entanto a atenção dos alunos para a importância de concluir atempadamente esta meta. Para efeitos de nota apenas será considerada a entrega final e a defesa.

6.1 Meta 1 – Modelação e desenvolvimento do Sistema de Produções

Material a entregar:

- Um breve documento (max. 3 páginas), em formato pdf, com a seguinte informação:
  - Identificação dos elementos do grupo (Nomes, Números de Estudante, e-mails, Turma(s) Prática(s)).
  - Definição das Perceções e Ações
  - Modelação do comportamento da nave através de um sistema de produções.
  - Outra informação que considere pertinente relativamente a esta meta.

Modo de Entrega:

Entrega eletrónica através do Inforestudante.
Data Limite: 21 de Fevereiro de 2025

6.2 Meta 2 – Implementação do Sistema de Produções

Tal como indicado anteriormente, esta entrega será a única que tem um impacto direto na nota. O relatório deve conter informação relativa a **todo** o trabalho realizado. Ou seja, o trabalho realizado no âmbito das metas 1 e 2 deve ser **inteiramente descrito**, por forma a possibilitar a avaliação.

Material a entregar:

- Script python contemplando o código alterado/implementado, que deve estar devidamente comentado.
- Um relatório (max. 10 páginas), em formato pdf, com a seguinte informação:
  - Identificação dos elementos do grupo (Nomes, Números de Estudante, e-mails, Turma(s) Prática(s)).
  - Informação pertinente relativamente à globalidade do trabalho realizado. Incluindo o sistema de produções, perceções e ações.

Num trabalho desta natureza o relatório assume um papel importante. Deve ter o cuidado de descrever detalhadamente todas as funcionalidades implementadas, dando particular destaque aos problemas e soluções encontradas. Deve ser fácil ao leitor compreender o que foi feito e ter por isso capacidade de adaptar / modificar o código.

Conforme pode depreender do enunciado, **experimentação** e **análise** são parte fundamental deste trabalho prático. Assim, deve descrever de forma sucinta mas detalhada as experiências realizadas, os resultados obtidos, analisar os resultados, e extrair conclusões. A avaliação do trabalho incidirá sobre várias componentes, sendo que **a performance da nave é uma delas**. Espera-se que sem a presença de vento, atinja uma taxa de sucesso acima de 40%.

O relatório deve conter informação relevante tanto da perspetiva do utilizador como do programador. Não deve ultrapassar as 10 páginas, formato A4. Todas as opções tomadas deverão ser devidamente justificadas e explicadas.

Modo de Entrega:

Entrega eletrónica através do Inforestudante.
Data Limite: 14 de Março de 2025

7 Bibliografia

1. **Inteligência Artificial: Fundamentos e Aplicações**
   Ernesto Costa, Anabela Simões

2. **Artificial Intelligence: A Modern Approach**
   Stuart Russel, Peter Norvig