

# Minigrid – Door Key – Uma análise de modelos (DQN x PPO)

Giancarlo Vanoni Ruggiero, Luciano Felix Dias, Tales Ivalque

Engenharia da Computação, INSPER

Prof. Fabrício Barth

**Abstract**—Neste projeto foi desenvolvido e validado um agente capaz de atuar no ambiente Door Key. O método de treinamento empregado foi ... Os resultados obtidos foram ...

**keywords**—*Reinforcement Learning, DQN, PPO, Minigrid, Door Key*

## 1. Introdução

O projeto tem como objetivo comparar o desempenho do ambiente single-agent *Minigrid – Doorkey* utilizando os algoritmos DQN (*Deep Q-Network*) e PPO (*Proximal Policy Optimization*). Neste ambiente, há uma chave que o agente deve adquirir para destrancar uma porta e chegar ao objetivo no quadrado verde.

## 2. Ambiente

**Table 1.** Direções do espaço de observação

ID	Vetor direção
0	(1, 0)
1	(0, 1)
2	(-1, 0)
3	(0, -1)

O espaço de observação é descrito por um dicionário contendo a direção do agente *direction*, a imagem observável *image* e o espaço de missão *mission*. Sendo: *direction* um atributo discreto de valor como descrito na tabela 1; *image* sendo uma matriz RGB quadrada de valores inteiros contidos no intervalo [0, 255] do tamanho do campo de visão do agente, e sendo este parametrizável como numero inteiro ímpar maior que 2 por padrão 7; e *mission* sendo o universo de ação do agente como uma matriz retangular de tamanho arbitrário contendo trincas seguindo a seguinte estrutura, respectivamente:

**Table 2.** Objetos do espaço de observação

ID	Nome	Objeto
0	UNSEEN	Não visível
1	EMPTY	Vazio
2	WALL	Parede
3	FLOOR	Chão
4	DOOR	Porta
5	KEY	Chave
6	BALL	Bola
7	BOX	Caixa
8	GOAL	Objetivo
9	LAVA	Lava
10	AGENT	Agente

- **ID do objeto** contido no latrilho atual, descrito como na tabela 2;
- **ID da cor** do latrilho atual, descrito como na tabela 3;
- **ID do estado** do objeto condito no latrilho atual, descrito como na tabela 4.

O espaço de ação é discreto com ações de movimento e poucas interações com elementos do ambiente. O espaço de ação é implementado conforme especificado na tabela 5.

**Table 3.** Cores do espaço de observação

ID	Nome	Vetor RGB
0	RED	(255, 0, 0)
1	GREEN	(0, 255, 0)
2	BLUE	(0, 0, 255)
3	PURPLE	(112, 39, 195)
4	YELLOW	(255, 255, 0)
5	GREY	(100, 100, 100)

**Table 4.** Estados do espaço de observação

ID	Nome	Estado
0	OPEN	Aberto
1	CLOSED	Fechado
2	LOCKED	Trancado

A função de recompensa é definida no domínio contínuo do intervalo [0, 1] e proporcional a razão entre o número de passos e o número máximo de passos permitido para o agente atingir o estado final com sucesso, caso contrário a recompensa é anulada. Esta razão é definida com maior rigor na equação 1.

$$\text{RECOMPENSA} = \begin{cases} 1 - 0.9 \cdot \frac{\# \text{PASSOS}}{\max(\# \text{PASSOS})}, & \text{se sucesso} \\ 0, & \text{caso contrário} \end{cases} \quad (1)$$

## 3. Método

Para conseguir comparar bem o desempenho dos dois algoritmos no ambiente, será analiado o crescimento da recompensa de um agente treinado em cada um desses algoritmos, utilizando a MlpPolicy, e 200.000 passos de aprendizado. O ambiente utilizado dentro do Minigrid - Doorkey será o "MiniGrid-DoorKey-6x6-v0", que já oferece uma complexidade razoável para o agente. Mas, para obter uma comparação satisfatória, é necessário primeiramente alterar a função recompensa de modo a ficar mais nítido o aprendizado de cada agente.

### Atenção

Para treinamento será utilizado a biblioteca *stable baselines*.

Inicialmente, para conseguir utilizar a biblioteca *stable baselines*, foi necessário utilizar do ImgObsWrapper, que é um Wrapper específico do ambiente, que transforma a observação do enviroment em um

**Table 5.** Espaço de ação

Número	Nome	Ação
0	LEFT	Vira para a esquerd
1	RIGHT	Vira para a direita
2	FORWARD	Move para a frente
3	PICKUP	Pega um objeto
4	DROP	Não utilizado
5	TOGGLE	Alternar/ativar um objeto
6	DONE	Não utilizado

formato de Imagem, próprio para interpretação da biblioteca. Junto a isso, foi necessário utilizar dois outros Wrappers que alterassem a recompensa entregue ao agente para que o treinamento pudesse ser realizado e analisado. Desses, um é o PositionBonusWrapper, que também é específico do ambiente Minigrid e permite a recompensa da exploração do agente, o outro é um Wrapper original feito pela equipe de forma para recompensar as ações intermediárias antes de chegar ao objetivo final.

O Wrapper original implementado tem o seguinte funcionamento:

```
1 from minigrid.core.world_object import Door, Key, Goal
2 from gymnasium import Wrapper
3
4
5 class CustomRewardWrapper(Wrapper):
6     def __init__(self, env):
7         super().__init__(env)
8         self.key_was_picked = False
9         self.door_was_unlocked = False
10
11
12     def step(self, action):
13         obs, _, terminated, truncated, info = self.env.
14         step(action)
15
16         # Custom reward logic
17         # Check if the agent picked up a key
18         if action == self.unwrapped.actions.pickup and
19         self.unwrapped.carrying and isinstance(self.
20         unwrapped.carrying, Key) and not self.
21         key_was_picked:
22             reward = 0.5 # Assign a reward for picking
23             up the key
24             self.key_was_picked = True
25
26         # Check if the agent opened a door
27         if action == self.unwrapped.actions.toggle:
28             front_cell = self.unwrapped.grid.get(*self.
29             unwrapped.front_pos)
30             if front_cell and isinstance(front_cell,
31             Door):
32                 if front_cell.is_locked and self.
33                 unwrapped.carrying and isinstance(self.unwrapped.
34                 carrying, Key) and not self.door_was_unlocked:
35                     reward = 0.5 # Assign a reward for
36                     unlocking a door
37                     self.door_was_unlocked = True
38                     if not front_cell.is_locked:
39                         reward = -0.25
40
41         # Check if the agent reached the goal
42         if terminated and self.unwrapped.agent_pos == (
43         self.unwrapped.width - 2, self.unwrapped.height -
44         2):
45             reward = 2 # Assign a reward for reaching
46             the goal
47         else:
48             reward = -0.1 # Small penalty for each
49             step
50
51         return obs, reward, terminated, truncated, info
```

Code 1. Código do Wrapper

Agora, a função recompensa é definida no domínio contínuo do intervalo [-0.25, 3], levando em conta as ações pegar a chave e destrancar a porta, além de punir o número de passos do agente, isso na sua primeira parcela. O PositionBonusWrapper provê uma segunda parcela inversamente proporcional à raiz do número de vezes que uma determinada posição foi visitada. O cálculo da recompensa final pode ser visto nas equações 2 e 3.

$$\text{NEW REWARD} = \begin{cases} 0.5, & \text{Pegar a chave ou destrancar a porta} \\ -0.25, & \text{Caso feche uma porta} \\ 2, & \text{Ao chegar ao destino final} \end{cases} \quad (2)$$

$$\text{NEW REWARD} += \frac{1}{\sqrt{\# \text{VISITAS}}} \quad (3)$$

Após a configuração dos rewards, vem a etapa de treinamento dos agentes. O setup é feito da seguinte maneira para o agente DQN:

```
1 from minigrid.wrappers import ImgObsWrapper,
2   PositionBonus
3 from stable_baselines3 import DQN
4 from stable_baselines3.common.evaluation import
5   evaluate_policy
6 from stable_baselines3.common.logger import configure
7 from minigrid.features_extractor import
8   MinigridFeaturesExtractor
9 import gymnasium as gym
10
11
12 from CustomRewardWrapper import CustomRewardWrapper
13
14 results_path = "./results/dqn"
15
16 new_logger = configure(results_path, ["stdout", "csv",
17   "tensorboard"])
18
19 policy_kwargs = dict(
20     features_extractor_class=MinigridFeaturesExtractor,
21     features_extractor_kwargs=dict(features_dim=128),
22 )
23
24 env = gym.make("MiniGrid-DoorKey-6x6-v0", render_mode="
25   rgb_array")
26 env = ImgObsWrapper(env)
27 env = CustomRewardWrapper(env)
28 env = PositionBonus(env)
29
30 model = DQN("MlpPolicy",
31     env,
32     policy_kwargs=policy_kwargs,
33     verbose=0,
34     exploration_fraction=0.001,
35     exploration_final_eps=0.5)
36
37 model.set_logger(new_logger)
38 model.learn(500_000, progress_bar=True)
39 model.save(results_path + "/dqn_minigrid")
40
41 mean_reward, std_reward = evaluate_policy(model, model.
42   get_env(), n_eval_episodes=10)
43 print(f'Mean reward: {mean_reward} +/- {std_reward:.2f}')
```

Code 2. Modelo DQN

E para o modelo PPO:

```
1 from minigrid.wrappers import ImgObsWrapper
2 from stable_baselines3 import PPO
3 from stable_baselines3.common.evaluation import
4   evaluate_policy
5 from stable_baselines3.common.logger import configure
6 from minigrid.wrappers import PositionBonus
7 from MinigridFeaturesExtractor import
8   MinigridFeaturesExtractor
9 import gymnasium as gym
10
11
12 from CustomRewardWrapper import CustomRewardWrapper
13
14 results_path = "./results/po"
15
16 new_logger = configure(results_path, ["stdout", "csv",
17   "tensorboard"])
18
19 policy_kwargs = dict(
20     features_extractor_class=MinigridFeaturesExtractor,
21     features_extractor_kwargs=dict(features_dim=128),
22 )
23
24 env = gym.make("MiniGrid-DoorKey-6x6-v0", render_mode="
25   rgb_array")
26 env = ImgObsWrapper(env)
27 env = CustomRewardWrapper(env)
28 env = PositionBonus(env)
29
30 model = PPO("MlpPolicy",
31     env=env,
32     policy_kwargs=policy_kwargs,
33     verbose=0,
34 )
35
36 model.set_logger(new_logger)
37 model.learn(200_000, progress_bar=True)
```

```

34 model.save(results_path + "/ppo_minigrid")
35
36
37 mean_reward, std_reward = evaluate_policy(model, model.
    get_env(), n_eval_episodes=10)
38 print(f'Mean reward: {mean_reward} +/- {std_reward:.2f}
    ')

```

Code 3. Modelo PPO

#### 4. Resultados

O objetivo desta parte do relatório é descrever os resultados obtidos. Durante a disciplina nós vimos as principais técnicas para mostrar o aprendizado do agente e o desempenho do mesmo para realizar a tarefa. Esta seção precisa mostrar dados quantitativos que descrevem isto: a curva de aprendizagem e as validações feitas depois do treinamento.

Para apresentar os resultados obtidos com certeza a equipe terá que fazer uso de figuras, como a apresenta em 1.

Além de tabelas, como a apresentada em 6.

Table 6. Exemplo de tabela

Column 1	Column 2
Data 1	Data 2
Data 3	Data 4

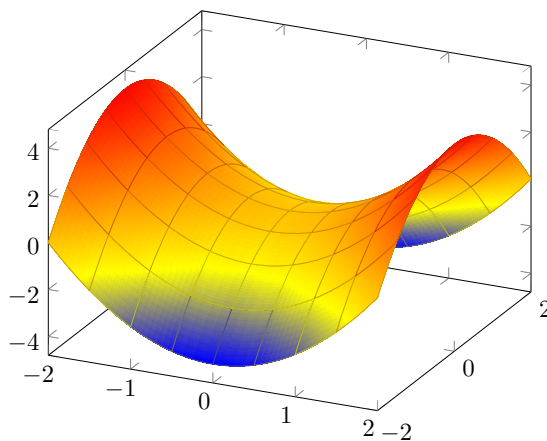


Figure 1. Exemplo de figura obtido a partir de *PGFPlots - A LaTeX package to create plots*. [Online]. Available: <https://pgfplots.sourceforge.net/>.

#### 5. Considerações finais

O objetivo deste trabalho foi ...