INSPER 2^o semestre de 2024

Avaliação Final

Inteligência Artificial e Robótica Prof. Fabrício Barth

Nome:

- *Quaisquer hipóteses relevantes devem ser* **explicitamente formuladas**. *Faz parte da avaliação da prova a* **correta interpretação** *das questões. A* **clareza** *e a* **objetividade** *das respostas serão consideradas na avaliação.*
- Todas as respostas precisam ser feitas dentro dos retângulos especificados.

1. (2.0 pontos) Considere um problema de busca de custo não uniforme, com ramificação 4 e com uma estimativa de profundidade para a solução igual a 10. Cada nodo armazenado em memória ocupa 1MB e o tempo de processamento é de 1 minuto. Responda as questões abaixo.

   (a) Quero garantir a resposta ótima independentemente do tempo de processamento ou memória que será ocupada. Liste todos os algoritmos que podem ser utilizados neste caso.

   (b) A implementação será executada em um dispositivo com 15MB de memória RAM. Que algoritmos podemos utilizar para retornar uma resposta? Qual seria a ressalva para a utilização destes algoritmos?

2. (1.0 ponto) O dilema do prisioneiro (DP), dito clássico, funciona da seguinte forma: "Dois suspeitos, A e B, são presos pela polícia. A polícia tem provas insuficientes para os condenar, mas, separando os prisioneiros, oferece a ambos o mesmo acordo: se um dos prisioneiros, confessando, testemunhar contra o outro e esse outro permanecer em silêncio, o que confessou sai livre enquanto o cúmplice silencioso cumpre 10 anos de sentença. Se ambos ficarem em silêncio, a polícia só pode condená-los a 6 meses de cadeia cada um. Se ambos traírem o comparsa, cada um leva 5 anos de cadeia. Cada prisioneiro faz a sua decisão sem saber que decisão o outro vai tomar, e nenhum tem certeza da decisão do outro. A questão que o dilema propõe é: o que vai acontecer? Como o prisioneiro vai reagir?"[1]

   O enunciado clássico do dilema do prisioneiro, acima exposto, pode resumir-se, do ponto de vista individual de um dos prisioneiros, na seguinte tabela (tabela de ganhos):

[1] Texto obtido em https://pt.wikipedia.org/wiki/Dilema_do_prisioneiro

	Prisioneiro "B" nega	Prisioneiro "B" delata
Prisioneiro "A" nega	Ambos são condenados a 6 meses	"A" é condenado a 10 anos; "B" sai livre
Prisioneiro "A" delata	"A" sai livre; "B" é condenado a 10 anos	Ambos são condenados a 5 anos

Supondo que você é o prisioneiro **A**, usando o algoritmo Min-Max para escolher a sua decisão, o que você faria? Justifique a sua resposta desenhando a árvore de busca do algoritmo.

3. (1.0 ponto) Considere um agente que atua em um ambiente onde o conjunto de estados é definido por $S = s_1, s_2, s_3, s_4, s_5$, o conjunto de ações por $A = a_1, a_2, a_3$ e o espaço de estados é definido pelo grafo abaixo:

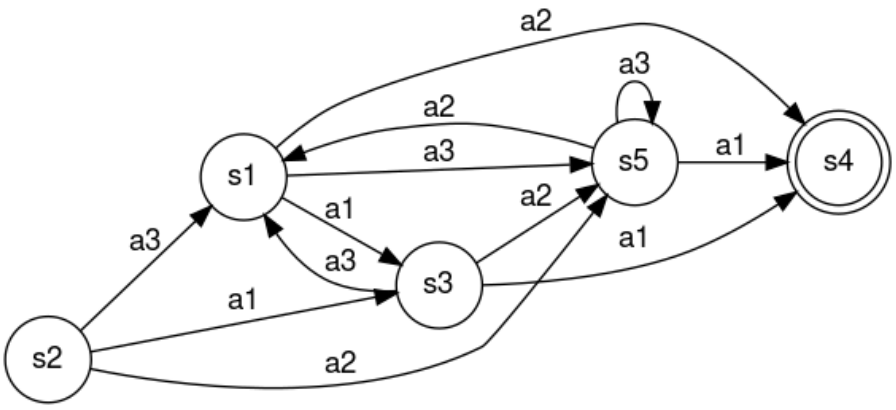


Figure 1: Espaço de estados do ambiente

Este mesmo agente foi treinado para atuar neste ambiente usando o algoritmo **Q-Learning**. Ao final do treinamento foi gerada a seguinte **Q-table**:

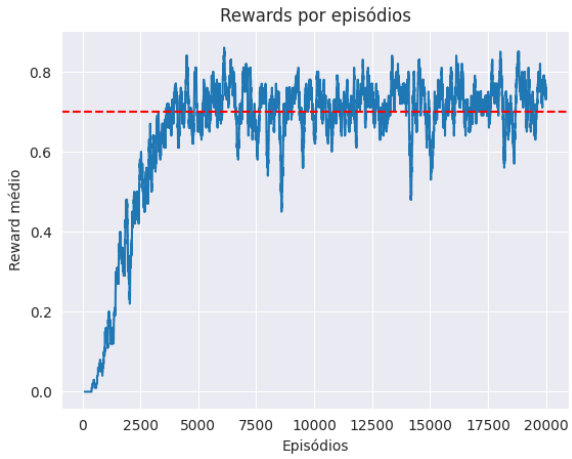
	a_1	a_2	a_3
s_1	-0.1	0.01	0.1
s_2	0.02	0.03	0.01
s_3	0.003	0.002	0.004
s_4	0	0	0
s_5	1	0	-1

Considere que o agente inicia no estado s_2 e que o estado terminal é o s_4, qual é a sequência de ações que o agente irá executar para chegar ao estado final?

4. (1.0 ponto) O ambiente *Frozen Lake* é um exemplo de ambiente não determinístico que vimos nesta disciplina. Ao treinarmos um agente para atuar neste ambiente usando a seguinte configuração:

```
env = gym.make('FrozenLake-v1', render_mode='ansi').env
qlearn = QLearning(env, alpha=0.1, gamma=0.99, epsilon=0.8, epsilon_min=0.01,
  epsilon_dec=0.999, episodes=20_000)
```

temos uma curva de aprendizado como a apresentada na figura 2a, onde a linha pontilhada significa uma meta a ser atingida.



(a) Curva de aprendizado do agente para o ambiente Frozen Lake

(b) Segunda versão da curva de aprendizado do agente para o ambiente Frozen Lake

Figure 2: Figuras relacionadas com o aprendizado do agente no ambiente Frozen Lake

Além disso, depois de testar o agente treinado usando uma rotina com 100 testes que executam 100 episódios no ambiente, temos uma **média de** 78.56 **vezes que o agente consegue chegar ao destino, com um desvio padrão de** 4.73.

Ao mudarmos as configurações para:

```
env = gym.make('FrozenLake-v1', render_mode='ansi').env
qlearn = QLearning(env, alpha=0.1, gamma=0.99, epsilon=0.8, epsilon_min=0.01,
  epsilon_dec=1, episodes=20_000)
```

temos uma curva de aprendizado como a apresentada na figura 2b. Mas ao testar o agente treinado, usando o mesmo método descrito acima, temos **uma média de** 81.99 **vezes que o agente consegue chegar ao destino, com um desvio padrão de** 3.75.

Explique o que aconteceu.

5. (2.0 pontos) Descreva o que a sua equipe utilizou na implementação do projeto 2 (o projeto do robô no labirinto) para:

   - representar os estados;
   - gerar os sucessores;
   - armazenar a informação sobre o estado meta, e;
   - implementar a heurística.

   Para cada um dos itens acima, por favor, utilize código para descrever.

6. (2.0 pontos) O ambiente **Simple Grid** é um ambiente onde o agente sabe executar 4 ações (cima, baixo, esquerda e direita) em um mapa pré-configurado com obstáculos (figura 3). Neste ambiente o estado inicial é representado pelo quadrado vermelho e o estado final é representado pelo quadrado verde. Cada estado é representado por um número inteiro. Por exemplo, para a figura 3 existem 64 estados possíveis porque o ambiente é um ambiente 8 por 8. Se o agente está na casa mais a esquerda e no topo do ambiente então o estado é representado pelo número 1. O estado da figura 3 é representado pelo número 21, pois o agente (marcado pela bola amarela) está na casa de número 21. Quando o agente chegar no objetivo (marcado como verde) o estado será representado pelo número 64.

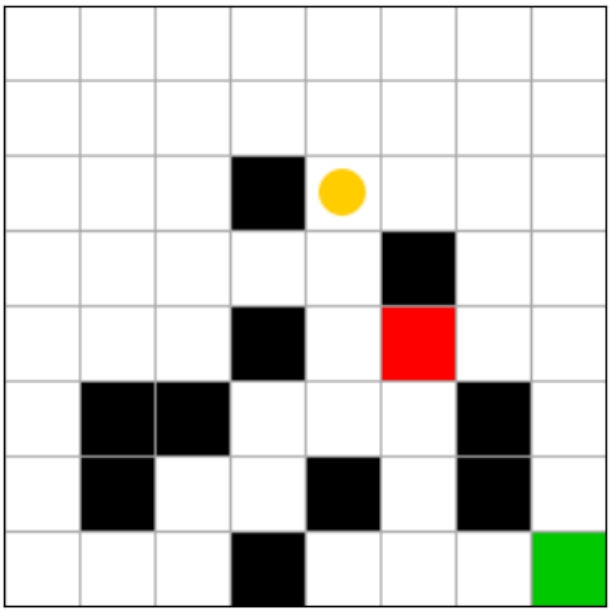


Figure 3: Exemplo de ambiente simple grid

A função de **reward** é definida por:

```python
def get_reward(self, x: int, y: int) -> float:
"""
Get the reward of a given cell.
"""
if not self.is_in_bounds(x, y):
  # if the agent tries to exit the grid, it receives a negative reward
  return -1.0
elif not self.is_free(x, y):
  # if the agent tries to walk over a wall, it receives a negative reward
  return -1.0
elif (x, y) == self.goal_xy:
  # if the agent reaches the goal, it receives a positive reward
  return 1.0
else:
  # otherwise, it receives no reward
  return 0.0
```

Código 1: Função de reward para o ambiente Simple Grid

Responda as seguintes perguntas sobre este problema:

(a) Considerando que o objetivo e os obstáculos sempre estarão no mesmo lugar, o agente consegue aprender caminhos de qualquer estado inicial para o estado objetivo usando os algoritmos Q-Learning ou Sarsa? Justifique a sua resposta.

(b) O agente é incentivado a encontrar o caminho mais rápido entre o estados inicial e final? Justifique a sua resposta.

(c) É possível utilizar algoritmos de busca em espaço de estados no desenvolvimento deste agente? Quais as modificações que deveriam ser feitas no ambiente e no agente para isto?

7. (1.0 ponto) Dado um grafo G e um nodo de origem, qual é o algoritmo de busca que descobre todos os nodos a uma distância K do nodo de origem, antes de descobrir qualquer nodo a uma distância k+1?