CEFET/RJ - Maracanã
Bacharelado em Ciência da Computação
Inteligência Artificial
Trabalho 3: Aprendizado por Reforço
Prof. Eduardo Bezerra

1. Informações Preliminares

Realizamos em sala de aula um tutorial de utilização da ferramenta Open AI Gym (https://www.learndatasci.com/tutorials/reinforcement-q-learning-scratch-python-openai-gym/). Neste trabalho, você irá dar continuidade às atividades deste tutorial, ainda no contexto do ambiente "Taxi" utilizado no tutorial. As diversas partes desse trabalho solicitam que você implemente variações do algoritmo Q-learning, tomando como ponto de partida a implementação básica fornecida no tutorial. Você deve implementar cada uma dessas variações em uma classe separada (cujo nome é indicado nas partes abaixo). Além disso, cada uma dessas classes deve conter os seguintes métodos:

- Um construtor para permitir configurar as informações necessárias para iniciar o aprendizado do agente. Dependendo de cada caso, essas informações podem ser as seguintes:
  - α (alfa), a taxa de aprendizado;
  - ϵ (épsilon), probabilidade de realizar exploração na estratégia de exploração ϵ-greedy;
  - γ (gamma), fator de desconto;
  - k, constante usada na estratégia de exploração com função de exploração.
- método `treinar(episodes)` para realizar o treinamento do agente. Quando for invocado, esse método deve receber a quantidade de episódios por meio do parâmetro `episodes`. O valor desse parâmetro deve ser usado no lugar da quantidade fixa de episódios, que está definida como igual a 100000 (cem mil) no código fornecido no tutorial.
- método `evaluate(episodes)`. Esse método tem o propósito de avaliar o modelo que foi treinado previamente com o método `treinar`. O método `evaluate` deve encapsular a lógica fornecida no tutorial (na

seção “*Evaluating the agent*”). Quando for invocado, esse método deve receber a quantidade de episódios por meio do parâmetro `episodes`. Ao final de sua execução, esse método deve produzir a quantidade média de passos (i.e., de ações) por episódio, assim como o valor médio de recompensa recebida pelo agente por episódio.

2. Q-learning exato

Transforme o código fornecido no tutorial em um módulo de funções em Python que pode usar vários ambientes. Em particular, você deve encapsular o código do algoritmo Q-learning exato fornecido no tutorial (na seção “*Training the Agent*”) em uma classe denominada `QLearningExato`.

3. Q-learning aproximado

Crie uma classe que implemente a versão aproximada do Q-learning. Essa classe deve conter os mesmos métodos descritos na seção “Informações Preliminares”. Você deve definir uma q-função $Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \cdots + w_n f_n(s,a)$, i.e., uma função linear que corresponde à média ponderada de diversas outras funções $f_i(s,a)$ que descrevem aspectos locais do problema (*features*). Você é responsável por definir quantas funções f_i irá usar em sua q-função, assim como definir qual aspecto local cada função $f_i(s,a)$ considera. Descreva no relatório cada uma das funções $f_i(s,a)$ que você criar. Em uma classe denominada `QLearningAproximado`.

4. Q-learning com função de exploração

Vimos em sala de aula duas abordagens para fazer com que um agente equilibre os aspectos de exploração (*exploration*) e de aproveitamento (*exploitation*) do ambiente: epsilon-greedy e funções de exploração. O tutorial apresenta a implementação da abordagem epsilon-greedy. Nesta parte do trabalho, você deve implementar uma variação do algoritmo Q-learning fornecido no tutorial para utilizar uma função de exploração, em vez da

abordagem epsilon-greedy. Para isso, crie duas novas classes cujos nomes devem ser os seguintes:

- `QLearningExatoComFuncaoExploracao`
- `QLearningAproximadoComFuncaoExploracao`

Descreva no seu relatório a sua função de exploração.

5. Q-learning com decaimento da taxa de aprendizado

O algoritmo Q-learning possui diversos hiperparâmetros: α (taxa de aprendizado), γ (fator de desconto) e ϵ (probabilidade de o agente explorar uma ação do ambiente). Na implementação do Q-learning fornecida no tutorial, os valores desses hiperparâmetros são fixados inicialmente e não mudam durante o treinamento. Nesta parte do trabalho, você deve implementar variações do Q-learning que sejam capazes de ajustar os valores dos hiperparâmetros alfa, gama e épsilon ao longo dos episódios de treinamento. O texto do tutorial fornece dicas acerca de como esses ajustes podem ser feitos (veja a seção “Hyperparameters and optimizations”). Defina essas variações do algoritmo em duas classes denominadas `QLearningExatoComDecaimento` `QLearningAproximadoComDecaimento`. Essas classes devem também conter os métodos `treinar` e `avaliar`.

6. Experimentos de comparação

Nesta parte do trabalho, você deve comparar as diferentes variações do Q-learning que você implementou nas partes anteriores. Sua comparação deve gerar uma tabela similar à fornecida no tutorial (na seção “*Comparing our Q-learning agent to no Reinforcement Learning*”). Contudo, além de comparar o Q-learning com um agente aleatório, você deve apresentar os resultados de execução de diversas variações do algoritmo Q-learning, a saber;

- `QLearningExato`
- `QLearningAproximado`
- `QLearningExatoComFuncaoExploracao`
- `QLearningAproximadoComFuncaoExploracao`

- `QLearningExatoComDecaimento`
- `QLearningAproximadoComDecaimento`

Apresente em sua tabela métricas computadas ao longo de 100 episódios, para cada uma das variações acima.

7. O que deve ser entregue

Você deve produzir um relatório (em formato PDF) descrevendo de que forma implementou cada uma das variações solicitadas do Q-learning, assim como o quadro comparativo com os resultados (solicitado na Seção 6). Entregue também os seguintes arquivos:

- q_learning_exato.py (QLearningExato)
- q_learning_aprox.py (QLearningAproximado)
- q_learning_exato_fe.py (QLearningExatoComFuncaoExploracao)
- q_learning_aprox_fe.py (QLearningAproximadoComFuncaoExploracao)
- q_learning_exato_decaimento.py (QLearningExatoComDecaimento)
- q_learning_aprox_decaimento.py (QLearningAproximadoComDecaimento)
- experimentos.py: script que produz o que é solicitado na Seção 6 deste trabalho.

Você deve compactar todos os arquivos do trabalho em um arquivo denominado SEU_NOME_COMPLETO.zip. Esse deve ser o arquivo a ser submetido pelo Moodle.