CI209 - Inteligência Artificial

Prof. Aurora Pozo
DInf - UFPR
2019/2

Especificação do quarto trabalho prático da disciplina [1]

Trabalho prático 4

Este trabalho é composto por duas partes:

- Implementar o algoritmo de aprendizado por reforço Q-Learning para um ambiente onde os espaços são representados por um vetor de características
- Elaborar um relatório explicando os resultados da aplicação de sua implementação

Você deverá baixar o código-base do trabalho e alterar o arquivo ***qlearning.py***

Ao final do trabalho, você deverá entender o conceito básico do funcionamento do algoritmo Q-Learning com aproximação de função.

Você poderá verificar o resultado da sua implementação utilizando os seguintes comandos:

```
$ python3 lunar_land.py
```

O problema a ser resolvido é relacionado a um ambiente presente na biblioteca Gym [2] chamado Lunar Land. O objetivo é controlar uma nave para pousar em uma região específica de um cenário 2D gerado aleatoriamente.

Este trabalho requer a instalação do pacotes **gym** e **box2d-py**:

```
$ pip3 install --user gym
$ pip3 install --user box2d-py
```

Para cada ação, o agente receberá uma recompensa de acordo com a simulação. Quando executado, o código base fornecerá métricas sobre a recompensa média de cada ação e a última recompensa para cada episódio.

[1] Elaborado por Bruno Henrique Meyer e Augusto Lopez Dantas (Prática em docência de Informática)

[2] https://gym.openai.com/envs/LunarLander-v2/

Divisão do trabalho

Parte 1 *Implementação do agente Q-Learning Aproximado*

Você deverá implementar as funções ***getAction*** e ***update*** da classe ***QLearningAgent*** no arquivo ***qlearning.py***.

A função ***getAction*** recebe como parâmetro uma representação (vetor de valores contínuos) do estado atual e deve retornar o id da ação a ser tomada de acordo com uma política ϵ-greedy. O conjunto de ações válidas para um estado é obtido através da função ***self.getLegalActions***.

Diferente do Trabalho prático 3, onde havia uma tabela chamada ***q_table*** que indicava os *Q-Values* de cada par (estado,ação), este trabalho utilizará um conjunto de redes neurais que funcionam como regressores [3].

Para isso, utilize o modelo MLPRegressor da biblioteca scikit-learn [4] que permite treinar a rede de forma incremental por meio do método ***partial_fit***. Você deverá criar uma rede que estima o Q-Value para cada ação. Cada rede deverá estimar o futuro Q-Value de qualquer estado (representado por um vetor de 8 números reais).

No cenário apresentado existem apenas 4 ações possíveis: Ativar o motor esquerdo, Ativar o motor direito, Ativar o motor principal e Não ativar nenhum motor. Ou seja, no total deverá haver 4 redes neurais que estimarão o Q-Value da respectiva ação para qualquer estado. Dessa forma, os Q-values serão obtido por meio do método ***predict*** da classe MLPRegressor de cada rede.

A regra para atualizar as redes (que abstraem a estrutura ***q_table*** do trabalho anterior) utiliza o Q-Value esperado descrito na Equação 1. Sempre que uma ação é executada, o método ***partial_fit*** deve ser chamado para informar a rede o estado anterior e o Q-Value esperado.

Utilize o parâmetro $\alpha(alpha)$ para especificar a taxa de aprendizado inicial do MLPRegressor.

$$\text{Q-Value esperado} = R(s, a, s') + \gamma \max_{a'} Q(s', a') \tag{1}$$

[3] Em aprendizado de máquina, regressores se comportam como classificadores que têm como saída valores do domínio real ao invés de classes

[4] https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html

Parte 2 *Conclusões e Relatório*

Crie um relatório de até 5 páginas reportando o desempenho do Q-Learning no problema apresentado.

Execute o simulador treinando seu agente por 1000 episódios. Salve o agente em episódios intermediários [5]:

```
$ python3 lunar_land.py -m train -ms 1000 -e 0
--save_episodes 250 500 750 1000
```

Para visualizar o comportamento do agente em cada um dos episódios salvos, execute os seguintes comandos:

```
$ python3 lunar_land.py -e 250 -m view -r
$ python3 lunar_land.py -e 500 -m view -r
$ python3 lunar_land.py -e 750 -m view -r
$ python3 lunar_land.py -e 1000 -m view -r
```

Na visualização, o valor de ϵ é 0. Você pode treinar por diferentes números de episódios se preferir. Note que o treinamento de cada episódios demanda um custo computacional relativamente alto, o que inviabiliza o treinamento por muitos episódios.

Varie os hiper-parâmetros da rede neural. Isso causa diferença significativa no processo de treinamento do agente? Explique como isso pode afetar a convergência do algoritmo.

Explique as diferenças observadas entre ambientes onde os estados são representados de forma discreta (vista no trabalho 3) comparado com a representação aproximada.

Explique as dificuldades encontradas e os principais desafios que você teve em sua implementação.

Se preferir, utilize figuras, tabelas entre outros recursos. Recomenda-se o uso do LaTeX para construir o seu relatório.

Dicas

- Leia atentamente a documentação do Scikit-Learn para entender o funcionamento do MLPRegressor

[5] Um arquivo chamado *snapshot_lunarland.pickle* será salvo com as informações de cada episódio

- Caso tenha interesse em alterar detalhes do uso do algoritmo implementado, modifique o arquivo *lunar_land.py* para depurar a execução do programa.
- O seu agente será instanciado com os parâmetros $\alpha(alpha) = 0.001$, $\epsilon(epsilon) = 0.01$ e $\gamma(gamma) = 0.999$. Se você preferir, altere o arquivo *lunar_land.py* ou modifique seu agente para testar outros valores. Nesse caso, comente no relatório as mudanças feitas e suas conclusões.
- Perceba que um dos primeiros comportamentos que o agente "aprende" é planar, característica que torna, na maior parte dos casos, a recompensa média positiva, onde a simulação atinge o limite de passos (1000). Porém, isso não é o suficiente para que o agente aprenda a pousar a nave corretamente.
- É possível que sua implementação consiga treinar o agente de forma que, a partir do episódio 250, a recompensa média normalmente fique positiva. Para isso, atente-se aos hiper-parâmetros da rede MLPRegressor.

Entrega

Você deverá entregar um arquivo compactado com o nome *login.tar.gz*, onde *login* é o nome do seu usuário no sistema do Departamento de Informática, que contenha os seguintes arquivos:

- *qlearning_aprox.py*
- *login.pdf*

A entrega será feita via Moodle. Trabalhos atrasados serão aceitos com um atraso máximo de 1 semana com desconto de nota. Após o prazo máximo o trabalho não será mais aceito.

Observações

Você deverá desenvolver e compreender todas implementações feitas no seu trabalho. Não serão admitidos quaisquer tipos de plágio.

Dúvidas

Dúvidas poderão ser retiradas durante aulas de laboratório ou por e-mail:

bhmeyer@inf.ufpr.br
aldantas@inf.ufpr.br