CI209 - Inteligência Artificial

Prof. Aurora Pozo
DInf - UFPR
2021/1

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 de forma discreta
- 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*.

Você poderá verificar o resultado da sua implementação utilizando o seguinte código:

```
$ python3 frozen_lake.py
```

O problema: *Frozen Lake*

Neste trabalho será utilizado o ambiente Frozen Lake, do estilo *Grid World*, implementado na biblioteca Gym [2]. Neste cenário, representado pela Figura 1, você é um jogador que deve andar em um lago congelado para sair de uma célula inicial da grade e chegar em uma célula objetivo. Cada estado do ambiente é representado por um número, que indica a posição do jogador.

Existem apenas 4 ações possíveis, relacionadas às tentativas de movimento nas quatro direções. A escolha de uma ação pelo jogador não necessariamente define o movimento. Existe uma pequena probabilidade de que o jogador "escorregue" para uma direção diferente da escolhida, o que gera um ambiente não determinístico. Durante a segunda parte do trabalho você poderá usar uma

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

[2] https://gym.openai.com/envs/FrozenLake-v0/

opção para desabilitar os “escorregões”, e verificar o impacto disso no resultado do algoritmo implementado.

Sempre que o jogador escolhe uma ação, ele recebe uma recompensa imediata de valor 1 caso chegue no objetivo (G), e de valor 0 caso contrário. Quando o jogador cai na água (H), isso também implica em uma recompensa de valor igual a 0, porém o episódio acaba e o jogo é reiniciado.

Ao executar o programa *frozen_lake.py*, você poderá verificar o número de episódios (jogos) em que o jogador conseguiu atingir o objetivo durante o treinamento e também na validação. A validação consiste no uso da política que controla o jogador após o treinamento do agente.

No ambiente onde os “escorregões” estão habilitados, espera-se uma taxa de sucesso entre 70% e 80% dos episódios de validação para uma política ótima. Uma política ótima irá ter sucesso em todos os episódios de validação no ambiente onde os “escorregões” estão desabilitados.

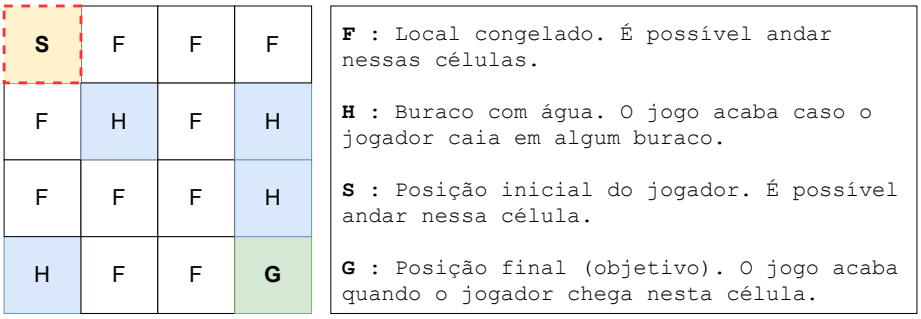

Figura 1: Ilustração do ambiente *Frozen Lake*

Divisão do trabalho

Parte 1 *Implementação do agente Q-Learning*

Você deverá implementar as funções *escolheAcao* e *atualizaAgente* da classe *QLearningAgent* no arquivo *qlearning.py*.

O método ***__init__*** é o construtor da classe que implementa o seu agente. Você poderá utiliza-la para definir as estruturas necessárias no *Q-Learning*. Perceba que alguns parâmetros do algoritmo, como *self.alpha* (α), *self.gamma* (γ) e *self.epsilon* (ϵ), são enviados como argumento do construtor.

O parâmetro ϵ deve ser utilizado para implementar uma política chamada de ϵ-greedy, que será utilizada como estratégia de exploração durante o treinamento. Esta política consiste em sortear um número entre 0 e 1. Caso este número seja maior que ϵ, o agente executa a ação considerada como ótima. Caso contrário, o agente executa uma ação aleatória.

O método ***escolheAcao*** recebe como parâmetro o id 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.env.getLegalActions***.

O método ***atualizaAgente*** é executado sempre que o agente executa uma ação. Ele é responsável por atualizar os Q-values definidos de acordo com a regra de atualização do algoritmo *Q-Learning*. Ele recebe como parâmetros o id do estado atual, o id da ação selecionada, a recompensa (*reward*) dessa ação e o id do estado que o agente se encontra após tomar a ação. As equações 1 e 2 representam as regras de atualização.

Um método chamado ***novoEpisodio*** estará disponível caso você precise. Este método é invocado sempre que um episódio acaba, e recebe como argumento a informação se o episódio era de treinamento ou validação.

$$\text{amostra} = R\left(s, a, s'\right) + \gamma \max_{a'} Q\left(s', a'\right) \tag{1}$$

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha * \text{amostra} \tag{2}$$

Parte 2 *Conclusões e Relatório*

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

Execute o agente com o parâmetro $\epsilon = 1.0$, $\epsilon = 0.5$ e $\epsilon = 0.0$. O número máximo de iterações será 1000:

```
$ python3 frozen_lake.py --epsilon 1.0 -tr 1000 -s -p
$ python3 frozen_lake.py --epsilon 0.5 -tr 1000 -s -p
$ python3 frozen_lake.py --epsilon 0.0 -tr 1000 -s -p
```

Explique em seu relatório o impacto de escolher esses valores de ϵ no *Q-Learning*. Observe que diferentes resultados serão gerados com informações sobre o desempenho do agente durante a etapa de treinamento e 1000 episódios de validação (quando o agente para de explorar e atualizar).

Também, altere o número máximo de episódios de treinamento (-tr) para diferentes valores (utilizando os mesmos valores de ϵ). Explique o impacto de aumentar o número de episódios de treinamento do agente na etapa de treinamento e validação.

```
$ python3 frozen_lake.py --epsilon 0.5 -tr 10000 -s -p
```

Você pode testar outros valores além de 10000.

Além da variação do número de episódios de treinamento, você poderá verificar os mesmos resultados dos experimentos descritos até aqui, porém em um ambiente onde o jogador não "escorrega". Para executar este ambiente, adicione o parâmetro *-ds* na invocação do *frozen_lake.py*. Por exemplo:

```
$ python3 frozen_lake.py --epsilon 0.5 -s -p -ds
```

Além de descrever os resultados observados, o seu relatório deverá abordar as seguintes questões:

- Como o parâmetro ϵ impacta o agente, e qual a sua importância para o algoritmo?
- Em quais situações vale a pena aumentar o número de episódios de treinamento? Por que?
- Dois ambientes foram considerados neste trabalho: *Frozen Lake* com o chão escorregadio (resultados das ações são incertos), e o *Frozen Lake* com o chão não escorregadio (resultados das ações são certos). Como essa diferença impacta o funcionamento do Q-Learning?

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

Bônus

Trabalhos que investigam e implementam detalhes além dos que foram especificados neste trabalho podem receber bônus. Recomenda-se a leitura de artigos e outras fontes que apresentam estratégias para melhorar o algoritmo implementado neste trabalho.

Algumas sugestões de tópicos a serem estudados:

- Estratégias para reduzir o número necessário de episódios de treinamento;
- Estratégias de exploração do ambiente;
- Estratégias que utilizam/modificam os conceitos relacionados aos parâmetros α e γ. O código base deste trabalho já considera valores padrões para esses parâmetros, mas que podem ser modificados na invocação do $frozen\_lake.py$ por meio das opções *–alpha* e *–gamma*;
- Avaliação do Q-Learning em outros ambientes (inclusive os que necessitam da versão aproximada);
- Algoritmos que encontram de forma automática bons parâmetros do Q-Learning como α, γ e ϵ.

Dicas

- A opção "-p" do $frozen\_lake.py$ permite a geração de dois gráficos referentes ao treinamento e validação, que contêm o número médio de passos a cada 10 episódios. Você pode utilizar essas informações para interpretar os seus resultados que serão discutidos no relatório.
- Utilize a biblioteca *numpy*[3] para facilitar as operações necessárias
- Caso queira alterar detalhes do uso do algoritmo implementado, modifique o arquivo $frozen\_lake.py$ para depurar a execução do programa.
- Você pode omitir o parâmetro *-s* ao invocar o $frozen\_lake.py$. Desta forma, você poderá visualizar cada ação executada pelo agente.

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.py*

[3] `https://numpy.org/`

- *login.pdf*

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