DCC028 - Inteligência Artificial

Trabalho Pratico 1: Busca em Mapas

Aluno: Romeu Junio Cunha de Oliveira

Matrícula: 2012422971

Implementação

O módulo Main, invoca o método buid\_grid da classe BuildGridClass que por sua vez executa a leitura do arquivo de entrada armazenando em grid o valor das recompensas em cada estado e em terminals as posições que correspondem a um estado terminal. Logo em seguida é invocado o método run que executa o Q-learning passando como parâmetros o grid, os estados terminais, alpha, gamma e o número de iterações.

É criado em run uma instância da classe MDPClass que implementa um processo de decisão de markov e para cada entrada do grid são atribuidos seus valores: estados, estados terminais, recompensas e ações. Q é inicializado com o valor 0 para cada estado não terminal ou inacessível e atribuimos a variável episilon um valor arbitrário (entre 0 e 1) que em conjunto com a variável alpha correspondem a política de explore exploit, alpha é decrementado (tendendo a zero) de episilon assintoticamente para que a taxa de aprendizado diminua com o tempo fazendo com que o algoritmo confie mais em valores da tabela Q para realizar a escolha de uma ação, do que realizar uma ação aleatória. Agora iniciamos o algoritmo Q-learning propriamente dito, escolhendo um estado aleatório e a partir deste estado, escolhemos uma ação dentre todas com probabilidade alpha/4 ou escolhemos a ação recomendada por Q com probabilidade (1-alpha) linha 59. A ação escolhida é executada e o valor de Q é atualizado conforme a equação do algoritmo, e o próximo estado passa a ser o estado antigo em conjunto com a ação. Quando um estado terminal é atingido é atualizado os valores de episilon e alpha conforme mostra o algoritmo (linhas 40, 43). A seguir é escrito nos arquivos de saída pi.txt e q.txt o mundo substituindo as ações (representadas pelas flechas) em cada estado não terminal e os valores de Q, respectivamente.

Decisões de Projeto

A linguagem utilizada para desenvolvimento do algoritmo foi Python (versão 3.6) juntamente com uma biblioteca não nativa numpy (para facilitar a escolha aleatória com probabilidade para as ações). Em todo o código foi optado pelas estruturas de dados lista e dicionário, pela sua facilidade em acessar de elementos assim como para adicionar, remover etc.

Bibliografia

Russell, S., & Norvig, P. (1995). Artificial intelligence: a modern approach.

<https://en.wikipedia.org/wiki/Q-learning>

<http://mnemstudio.org/path-finding-q-learning-tutorial.htm>

<https://www.youtube.com/watch?v=aCEvtRtNO-M&t=102s>

<https://medium.com/@curiousily/solving-an-mdp-with-q-learning-from-scratch-deep-reinforcement-learning-for-hackers-part-1-45d1d360c120>

<https://medium.com/@m.alzantot/deep-reinforcement-learning-demysitifed-episode-2-policy-iteration-value-iteration-and-q-978f9e89ddaa>

<https://webdocs.cs.ualberta.ca/~sutton/book/the-book.html>