UNIVERSIDADE FEDERAL DE SERGIPE (UFS)
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS (CCET)
DEPARTAMENTO DE COMPUTAÇÃO (DCOMP)

INTELIGÊNCIA ARTIFICIAL
VALUE ITERATION PARA MDP

JOSE JOAQUIM DE ANDRADE NETO - 201410011600

PROFESSOR: HENDRIK TEIXEIRA MACEDO

São Cristóvão, Sergipe
2017

1. Introdução

Este presente relatório destaca as funcionalidades e exemplos usado para a implementação do algoritmo *Value Iteration* para Redes de Decisões Markovianas (MDP). Esta implementação se baseia no mesmo algoritmo e detalhes visto em sala de aula, na unidade 3 da disciplina de Inteligência Artificial e foi escrita em Javascript.

O algoritmo de *Value Iteration* para MDP se baseia em métodos matemáticos para a aprendizagem por reforço, onde o agente possui um conjunto de ações e um universo para ele se valer das mesmas. Assim, o agente passa por um processo de aprendizagem, onde, através de utilidades e recompensas, ele se torna capaz de aprender qual ação tomar em um determinado momento, afim de conseguir a maior utilidade no final e atingir a política ótima. Um dos métodos mais usados para esse tipo de aprendizagem é com o uso da Equação de Bellman, a partir da observação de que a utilidade de um estado é a recompensa imediata do mesmo acrescida da utilidade (parcial) esperada para o próximo estado, assumindo que o agente irá tomar a decisão ótima. A utilidade do próximo estado é parcial pois é assumido que a utilidade de um estado pode não valer mais o que vale no estado atual, em um instante futuro. Para isso, a utilidade do estado seguinte é multiplicada por um fator de desconto. A equação é escrita da forma (RUSSELL; NORVIG, 2010):

$$U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a) U(s') \ .$$

Figura 1: Equação de Bellman.

Na equação, o termo *U(s)* denota a utilidade do estado atual, *R(s)* significa a recompensa do estado, γ representa o fator de desconto, e o argumento *max()* retorna a melhor ação, isto é, a ação que maximiza a utilidade do o estado atual.

Há vários exemplos reais que utilizam essa forma de aprendizagem. Os usos mais comuns estão no aprendizado de robôs a fazerem atividades específicas, como andar, segurar objetos, etc. Portanto, o exemplo usado nesse trabalho irá trabalhar com essa abordagem. O exemplo é o mesmo usado no livro Russel e Norvig (2010), no capítulo 17. Tem-se um agente, em um grid bidimensional, e ele deseja alcançar o estado positivo. Para isso, ele deve aprender a melhor forma (política) de se descolar no grid. Seu conjunto de ações é: ir para cima, ir para baixo, ir para a esquerda e ir para a direita. Há estados no grid que são inatingíveis (isto é, o agente não pode ir para ele), bem como um estado

negativo, que simboliza a pior ação para o agente. A figura 2 mostra um exemplo de um grid e a política ótima de ações.

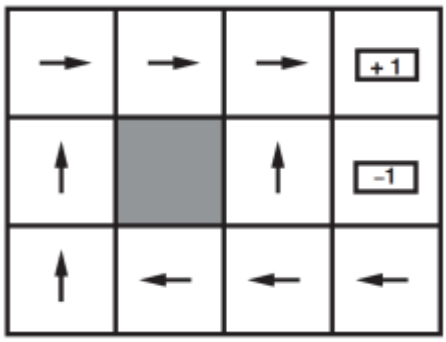

Figura 2: Exemplo de um grid (RUSSELL; NORVIG, 2010).

2. Implementação em Javascript

A política ótima varia de acordo com a probabilidade escolhida para uma ação ser feita, bem como os valores de recompensas e do estado final. Na implementação desse trabalho foram definidas quatro configurações padrões para teste do algoritmo, onde em cada uma das diferentes combinações dos parâmetros (tamanho do grid, recompensa, etc) foram testadas. Essas configurações podem ser verificadas chamando as funções *conf1()*, *conf2()*, *conf3()*, *conf4()*. Além disso, o usuário pode testar seus próprios valores na página HTML disponível. O algoritmo está descrito no procedimento *mdp()*, e é o mesmo visto na Figura 17.4 (p. 653) em Russell e Norvig (2010). Apesar da convergência do algoritmo ser garantida pela equação de Bellman, não foi possível fazer o algoritmo parar de rodar ao se atingir essa convergência, sendo necessário limitar o *loop* principal através do número de iterações. Isso aconteceu por motivos técnicos da linguagem javascript, onde o autor desse relatório não conseguiu pegar a diferença descrita na linha 275 do código devido ao método de referência de variáveis de javascript, que tornava as variáveis *gridAux* e *grid* iguais, fazendo a diferença ser sempre 0. No entanto, o limite de iterações máximo que foi colocado garante os valores ótimos para os exemplos utilizados. Os métodos *valid()*, *checkInput()* e *notValid()* são utilizados para verificar se a entrada foi digitada corretamente. As funções *partidaGeral()* e *getInput()* são utilizados a partir da página HTML. As funções *actionWest()*, *actionEast()*, *actionSouth()*, *actionNorth()* retornam a utilidade dos estados a e remontam a ação que o agente irá tomar no momento. A função *featuredPos()* verifica se o agente está em um estado especial (estado positivo, estado negativo ou estado inatingível). Por fim, o procedimento *mdp()* executa o algoritmo de *Value Iteration* e os resultados são exibidos na função *printResults()*.

3. Referências Bibliográficas

RUSSELL, S. J.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. New Jersey: Pearson Education, 2010.