



Laboratório 11 – Programação Dinâmica
Inteligência Artificial para Robótica Móvel – CT-213

Aluno: Caio Graça Gomes

Professor: Marcos Ricardo Omena de Albuquerque Maximo

Introdução:

Nesse laboratório, teve-se por objetivo implementar três algoritmos de programação dinâmica para solução de um Processo Decisório de Markov (*MDP*). Os algoritmos implementados serão os de avaliação de política (*policy evaluation*), iteração de valor (*value iteration*) e iteração de política (*policy iteration*). Deve-se avaliar e encontrar políticas ótimas em uma ambiente de *grid world*.

Metodologia e descrição em alto nível do algoritmo utilizado:

O problema consistiu em um *grid world* com 5 ações possíveis: STOP, UP, LEFT, RIGHT e DOWN, além de obstáculos e limites do *grid* como barreiras. A ação STOP é sempre executada com probabilidade 1, enquanto as outras ações têm uma certa probabilidade de serem executadas corretamente. Se uma ação não é executada corretamente, o resultado das demais ações acontece com igual probabilidade. O MDP tem fator de desconto γ (que será adotado como 1 em uma parte do experimento, e 0,98 como outra parte) e a recompensa é -1 para cada instante que o agente passa em uma célula que não é a objetivo. Há uma única célula objetivo no *grid*, onde o agente recebe recompensa 0.

O algoritmo *policy evaluation* se baseia na equação de Bellman de expectativa, será adotada uma solução iterativa (Gauss-Jacobi) do sistema linear associado às equações de Bellman.

$$v_{\pi}(s) = \sum_{a \in A} \pi(a|s)r(s,a) + \gamma \sum_{a \in A} \sum_{s' \in S} \pi(a|s)p(s'|s,a)v_{\pi}(s')$$

O algoritmo de iteração de valor se baseia em iterar diretamente de acordo com a equação da otimalidade de Bellman:

$$v_*(s) = \max_{a \in A} (r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a)v_{\pi}(s'))$$

Já o algoritmo de iteração de política alterna entre avaliação de política (3 vezes, nesse laboratório) e aprimoramento de política. Uma política gulosa *greedy* é obtida ao fazer:

$$\pi'(s) = \operatorname{argmax}_{a \in A} (r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a)v_{\pi}(s'))$$

Quando existir mais de uma ação ótima, é dada a mesma probabilidade para cada uma ser executada.

Os três algoritmos foram testados em *test_dynamic_programming.py*, sob duas diferentes circunstâncias: primeiramente fez-se `CORRECT_ACTION_PROBABILITY = 1.00` e `GAMMA = 1.00`, e então `CORRECT_ACTION_PROBABILITY = 0.80` e `GAMMA = 0.98`. Ao término, foram comparados seus desempenhos.

Resultados do test_dynamic_programming.py:

Figura 1: *Policy Evaluation* na primeira condição.

```
Value function:
[ -384.09, -382.73, -381.19, * , -339.93, -339.93]
[ -380.45, -377.91, -374.65, * , -334.92, -334.93]
[ -374.34, -368.82, -359.85, -344.88, -324.92, -324.93]
[ -368.76, -358.18, -346.03, * , -289.95, -309.94]
[ * , -344.12, -315.05, -250.02, -229.99, * ]
[ -359.12, -354.12, * , -200.01, -145.00, 0.00]
Policy:
[ SURDL , SURDL , SURDL , * , SURDL , SURDL ]
[ SURDL , SURDL , SURDL , * , SURDL , SURDL ]
[ SURDL , SURDL , SURDL , SURDL , SURDL , SURDL ]
[ SURDL , SURDL , SURDL , * , SURDL , SURDL ]
[ * , SURDL , SURDL , SURDL , SURDL , * ]
[ SURDL , SURDL , * , SURDL , SURDL , S ]
```

Figura 3: *Policy Iteration* na primeira condição.

```
Policy iteration:
Value function:
[ -10.00, -9.00, -8.00, * , -6.00, -7.00]
[ -9.00, -8.00, -7.00, * , -5.00, -6.00]
[ -8.00, -7.00, -6.00, * , -4.00, -5.00]
[ -7.00, -6.00, -5.00, * , -3.00, -4.00]
[ * , -5.00, -4.00, -3.00, -2.00, * ]
[ -7.00, -6.00, * , -2.00, -1.00, 0.00]
Policy:
[ RD , RD , D , * , D , DL ]
[ RD , RD , D , * , D , DL ]
[ RD , RD , RD , R , D , DL ]
[ R , RD , D , * , D , L ]
[ * , R , R , RD , D , * ]
[ R , U , * , R , R , SURD ]
```

Figura 5: *Value Iteration* na segunda condição.

Figura 2: *Value Iteration* na primeira condição.

```
Value iteration:
Value function:
[ -10.00, -9.00, -8.00, * , -6.00, -7.00]
[ -9.00, -8.00, -7.00, * , -5.00, -6.00]
[ -8.00, -7.00, -6.00, -5.00, -4.00, -5.00]
[ -7.00, -6.00, -5.00, * , -3.00, -4.00]
[ * , -5.00, -4.00, -3.00, -2.00, * ]
[ -7.00, -6.00, * , -2.00, -1.00, 0.00]
Policy:
[ RD , RD , D , * , D , DL ]
[ RD , RD , D , * , D , DL ]
[ RD , RD , RD , R , D , DL ]
[ R , RD , D , * , D , L ]
[ * , R , R , RD , D , * ]
[ R , U , * , R , R , SURD ]
```

Figura 4: *Policy Evaluation* na segunda condição.

```
Value function:
[ -47.19, -47.11, -47.01, * , -45.13, -45.15]
[ -46.97, -46.81, -46.60, * , -44.58, -44.65]
[ -46.58, -46.21, -45.62, -44.79, -43.40, -43.63]
[ -46.20, -45.41, -44.42, * , -39.87, -42.17]
[ * , -44.31, -41.64, -35.28, -32.96, * ]
[ -45.73, -45.28, * , -29.68, -21.88, 0.00]
Policy:
[ SURDL , SURDL , SURDL , * , SURDL , SURDL ]
[ SURDL , SURDL , SURDL , * , SURDL , SURDL ]
[ SURDL , SURDL , SURDL , SURDL , SURDL , SURDL ]
[ SURDL , SURDL , SURDL , * , SURDL , SURDL ]
[ * , SURDL , SURDL , SURDL , SURDL , * ]
[ SURDL , SURDL , * , SURDL , SURDL , S ]
```

Figura 6: *Policy Iteration* na segunda condição.

```

-----
Value iteration:
Value function:
[ -11.65, -10.78, -9.86, * , -7.79, -8.53]
[ -10.72, -9.78, -8.78, * , -6.67, -7.52]
[ -9.72, -8.70, -7.59, -6.61, -5.44, -6.42]
[ -8.70, -7.58, -6.43, * , -4.09, -5.30]
[ * , -6.43, -5.17, -3.87, -2.76, * ]
[ -8.63, -7.58, * , -2.69, -1.40, 0.00]
Policy:
[ D , D , D , * , D , D ]
[ D , D , D , * , D , D ]
[ RD , D , D , R , D , D ]
[ R , RD , D , * , D , L ]
[ * , R , R , D , D , * ]
[ R , U , * , R , R , S ]
-----

```

```

-----
Policy iteration:
Value function:
[ -11.65, -10.78, -9.86, * , -7.79, -8.53]
[ -10.72, -9.78, -8.78, * , -6.67, -7.52]
[ -9.72, -8.70, -7.59, -6.61, -5.44, -6.42]
[ -8.70, -7.58, -6.43, * , -4.09, -5.30]
[ * , -6.43, -5.17, -3.87, -2.76, * ]
[ -8.63, -7.58, * , -2.69, -1.40, 0.00]
Policy:
[ D , D , D , * , D , D ]
[ D , D , D , * , D , D ]
[ RD , D , D , R , D , D ]
[ R , RD , D , * , D , L ]
[ * , R , R , D , D , * ]
[ R , U , * , R , R , S ]
-----

```

Conforme esperado, os algoritmos de *policy iteration* e *value iteration* convergiram para um mesmo valor, no caso, encontraram os valores e políticas ótimos. Percebe-se ainda, um desempenho não muito razoável do algoritmo de *policy evaluation*, que melhorou seu desempenho (aproximou-se dos valores do *policy* e *value iteration*) ao fazer o gamma ser 0.98.