INSTITUTO TECNOLÓGICO DE AERONÁUTICA

Inteligência Artificial para Robótica Móvel

CT-213

Laboratório 11 - Programação Dinâmica

COMP 20

Sebastião Beethoven Brandão Filho

Professor:

Marcos Máximo

ITA-SJC-2019

Descrição do Laboratório

O Laboratório 11 possui como objetivo principal a implementação e teste de algoritmos de programação dinâmica associados a Processos Decisórios de Markov (MDP), no contexto da análise de "caminhos ótimos"em um grid world com uma posição de início, outra de objetivo e alguns obstáculos.

Para essa realização, então, foi feita uma divisão inicial do Laboratório em quatro partes distintas: Implementação de Avaliação de Política, Implementação de Iteração de Valor, Implementação de Iteração de Política e, por fim, Comparação entre Grid Worlds diferentes.

1. **Implementação de Avaliação de Política**

   Realizou-se a Implementação de Avaliação de Política utilizando o método policy_evaluation() presente no script dynamic_programming.py.

   Além disso, utilizou-se o script test_dynamic_programming.py para testar o método implementado. A Figura 1 apresenta o código implementando para esse algoritmo.

```
dimensions = grid_world.dimensions
value = np.copy(initial_value)
# Todo: implement policy evaluation.

gamma = grid_world.gamma

it = 1
while it <= num_iterations:
    new_value = np.zeros((dimensions[0], dimensions[1]))
    maxDif = -inf
    for i in range(dimensions[0]):
        for j in range(dimensions[1]):
            current_state = (i, j)
            value_k = 0.0
            probab = policy[i][j]
            for action in range(NUM_ACTIONS):
                r = grid_world.reward(current_state, action)
                for next_state in grid_world.get_valid_sucessors((i, j), action):
                    transition_prob = grid_world.transition_probability(current_state, action, next_state)
                    value_k += transition_prob * probab[action] * (r + gamma * value[next_state[0]][next_state[1]])
            maxDif = max(maxDif, fabs(value_k - value[i][j]))
            new_value[i][j] = value_k
    value = new_value
    if maxDif < epsilon:
        break
    it += 1

return value
```

Figura 1: Código implementando para a Avaliação de Política.

2. **Implementação de Iteração de Valor**

Realizou-se a Implementação de Iteração de Valor utilizando o método value_iteration() presente no script dynamic_programming.py.

Além disso, utilizou-se o script test_dynamic_programming.py para testar o método implementado. A Figura 2 apresenta o código implementando para esse algoritmo.

```
dimensions = grid_world.dimensions
value = np.copy(initial_value)
# Todo: implement value iteration.

gamma = grid_world.gamma

it = 1
while it <= num_iterations:
    new_value = np.zeros((dimensions[0], dimensions[1]))
    maxDif = -inf
    for i in range(dimensions[0]):
        for j in range(dimensions[1]):
            current_state = (i, j)
            value_k = np.zeros(NUM_ACTIONS)
            for action in range(NUM_ACTIONS):
                r = grid_world.reward(current_state, action)
                for next_state in grid_world.get_valid_sucessors((i, j), action):
                    transition_prob = grid_world.transition_probability(current_state, action, next_state)
                    value_k[action] += transition_prob * (r + gamma * value[next_state[0]][next_state[1]])
            maxValue = max(value_k)
            maxDif = max(maxDif, fabs(maxValue - value[i][j]))
            new_value[i][j] = maxValue
    value = new_value
    if maxDif < epsilon:
        break
    it += 1

return value
```

Figura 2: Código implementando para a Iteração de Valor.

3. **Implementação de Iteração de Política**

Realizou-se a Implementação de Iteração de Política utilizando o método policy_iteration() presente no script dynamic_programming.py.

Além disso, utilizou-se o script test_dynamic_programming.py para testar o método implementado. A Figura 3 apresenta o código implementando para esse algoritmo.

```
value = np.copy(initial_value)
policy = np.copy(initial_policy)
# Todo: implement policy iteration.

it = 1
while it <= num_iterations:
    value = policy_evaluation(grid_world, value, policy, evaluations_per_policy)
    policy = greedy_policy(grid_world, value)
    it += 1

return value, policy
```

Figura 3: Código implementando para a Iteração de Política.

4. **Comparação entre Grid Worlds diferentes**

Realizou-se uma comparação entre Grid Worlds distintos, através da modificação dos valores de probabilidade de escolha de ação correta e do gama. Para tal utilizou-se o script test_dynamic_programming.py. Foram utilizandos os seguintes dados, em cada um dos testes:

(a) CORRECT_ACTION_PROB (pc) = 1.0 e GAMMA = 1.0.

As Figuras 4, 5 e 6 apresentam os resultados obtidos para os testes associados a cada um dos algoritmos utilizados.

```
Evaluating random policy, except for the goal state, where policy always executes stop:
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 4: Resultados obtidos para o teste associado à Avaliação de Política.

```
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 5: Resultados obtidos para o teste associado à Iteração de Política.

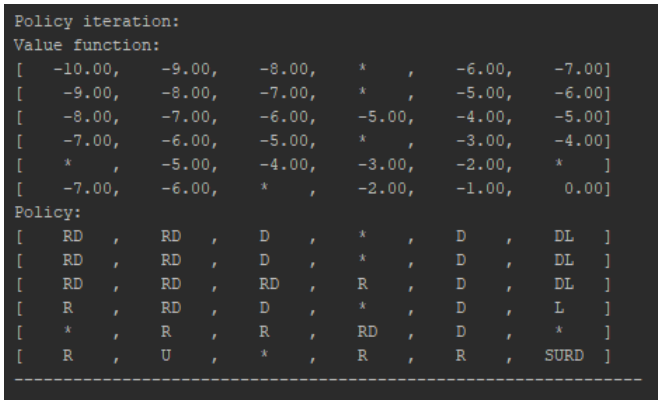

Figura 6: Resultados obtidos para o teste associado à Iteração de Política.

A partir da análise das figuras com os resultados apresentandos, pode-se verificar o cumprimento dos requisitos estabelecidos pelos algoritmos utilizados, sendo possível observar a determinação de políticas ótimas para o Grid World utilizado

(b) CORRECT_ACTION_PROB (pc) = 0.8 e GAMMA = 0.98.

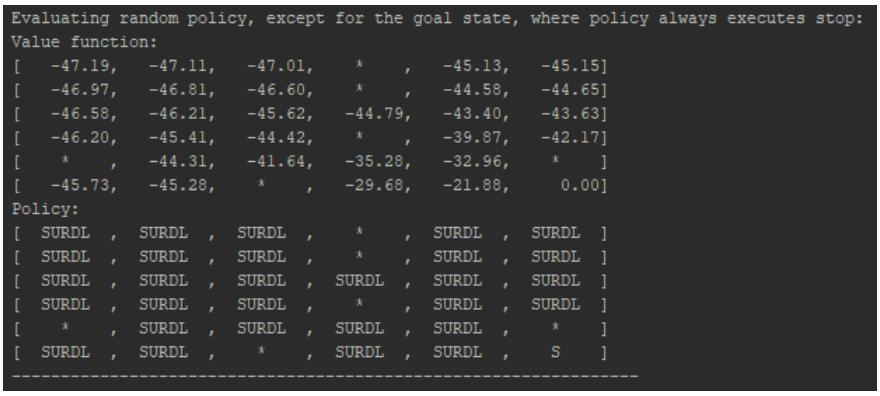

Figura 7: Resultados obtidos para o teste associado à Avaliação de Política.

```
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    ]
----------------------------------------------------------------
```

Figura 8: Resultados obtidos para o teste associado à Iteração de Política.

```
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    ]
----------------------------------------------------------------
```

Figura 9: Resultados obtidos para o teste associado à Iteração de Política.

A partir da análise das figuras com os resultados apresentandos, pode-se verificar o cumprimento dos requisitos estabelecidos pelos algoritmos utilizados, sendo possível observar a determinação de políticas ótimas para o Grid World utilizado