Relatório do Lab 11 de CT-213

Trabalho 11 – Processo Decisório de Markov (*MDP*) de um *grid world* com Programação Dinâmica

Aprendizado por Reforço (*RL*) com modelo de *MDP* conhecido

Aluno:

Bruno Costa Alves Freire

Turma:

T 22.4

Professor:

Marcos Ricardo Omena de Albuquerque Máximo

Data:

09/06/2019

Instituto Tecnológico de Aeronáutica – ITA
Departamento de Computação

1. Implementando a solução do MDP

O Processo Decisório de Markov de um *grid world* 4 conectado pode ser resolvido utilizando-se programação dinâmica. A solução deste problema consiste em determinar políticas ótimas por meio de duas estratégias: a iteração de política, e a iteração de valor.

Na iteração de política, vamos avaliando uma política iterativamente a partir da *equação de expectativa de Bellman*, e em seguida buscamos uma política que tenha aquela função valor por meio de uma estratégia gulosa. Ao executar esses passos iterativamente, convergimos para a política ótima.

Na iteração de valor, nos focamos em buscar a função valor ótima a partir da *equação de otimalidade de Bellman*, e em seguida construímos a política ótima a partir dessa função valor utilizando um algoritmo guloso.

A implementação de ambos os métodos foi feita de maneira síncrona, isto é, a cada iteração da avaliação, utilizamos apenas os valores antigos da função valor para construir a nova função valor. A estratégia assíncrona, chamada *in-place*, seria utilizar os valores recém atualizados durante a iteração para construir os próximos.

No arquivo `dynamic_programming.py` temos a implementação dos métodos `policy_evaluation`, `policy_iteration` e `value_iteration`. Além disso, há o método `greedy_policy` já implementado, que dá conta de obter a política associada a uma dada função valor. A implementação dos três primeiros métodos é testada no *script* `test_dynamic_programming.py`, cuja saída produzida consta no arquivo `test_output.txt`.

No nosso *grid world*, o agente pode executar 5 ações, **S**TOP, **U**P, **R**IGHT, **D**OWN e **L**EFT. A ação STOP é sempre executada com perfeição, e o agente não se move, contudo, as demais ações têm uma probabilidade p_c de serem executadas corretamente, caso contrário uma das 4 ações restantes é executada com distribuição de probabilidade uniforme.

Note que essa possibilidade de erro na tomada da ação faz com que a probabilidade do agente ficar parado aumente, uma vez que é a única ação que sempre é executada perfeitamente.

Para testar a implementação, consideramos dois cenários, onde temos diferentes possibilidades para a probabilidade do agente tomar a ação correta prevista pela política, p_c, e para o fator de desconto, γ, que mede o quanto o agente valoriza recompensas mais imediatas em detrimento de um maior retorno futuro.

Para os parâmetros p_c e γ, ambos iguais a 1, temos a saída:

```
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    ]
-----------------------------------------------------------------
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  ]
-----------------------------------------------------------------
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,    -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  ]
-----------------------------------------------------------------
```

Para p_c = 80% e γ = 0,98, temos a saída:

```
Evaluating random policy, except for the goal state, where
policy always executes stop:
Value function:
[  -100.46,   -98.43,   -99.76,    *    ,  -101.94,  -102.14]
[   -97.33,   -95.11,   -95.93,    *    ,   -96.58,   -97.19]
[   -96.38,   -93.58,   -92.80,   -94.25,   -90.53,   -92.81]
[   -98.13,   -94.72,   -92.04,    *    ,   -82.85,   -90.42]
[    *    ,   -99.95,   -88.28,   -72.31,   -66.76,    *    ]
[  -128.81,  -116.94,    *    ,   -60.48,   -43.41,     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    ]
------------------------------------------------------------
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    ]
------------------------------------------------------------
```

Comparando o resultado da avaliação da política aleatória em ambos os casos, podemos ver que a política foi menos mal avaliada quando a probabilidade do agente tomar a ação correta era menor. Isso faz sentido pois, como explicado anteriormente, essa possibilidade de erro faz com que a probabilidade do agente ficar parado aumente, logo, ele pelo menos tem uma tendência menor de se aventurar em caminhos que o distanciam do objetivo.

Ao comparar o resultado da iteração de valor, primeiro observando a avaliação da política ótima, podemos notar uma certa anisotropia nos valores dos estados quando introduzimos a possibilidade de erro. Alguns estados que seriam avaliados igualmente quando a ação era executada perfeitamente agora possuem avaliações piores. Uma observação que se pode extrair acerca disso é que, ficaram mais mal avaliados os estados em que o agente dispõe de mais formas de se afastar do objetivo, isto é, de tomar uma ação não ótima. Por conta da sua probabilidade de erro, ter mais possibilidades para tomar uma ação subótima faz com que, estatisticamente, o agente obtenha retorno menor a partir daquele estado.

Agora observando a política obtida gulosamente a partir da função valor, podemos notar que, enquanto no cenário ideal tínhamos uma política relativamente estocástica, isto é, alguns estados podiam escolher entre mais de uma ação sem perder a otimalidade, quando introduzimos possibilidade de erro, a maioria dos estados agora tem uma ação bem definida. Isso ocorre porque a simetria que existia no cenário ideal se perdeu, pois a introdução da possibilidade de erro do agente criou uma anisotropia na função valor dos estados, como discutido no parágrafo anterior.

Podemos observar, por fim, que a política ótima e a função valor ótima obtidas por meio da iteração de valor e da iteração de política são idênticas, em ambos os casos (com p_c e γ iguais a 80% e 0.98, ou ambos iguais a 1). Isso era de se esperar, dado que ambos os algoritmos convergem para a política ótima do MDP.

Agora, comparando o resultado da iteração de política entre os dois cenários, podemos tirar exatamente as mesmas conclusões que no caso da iteração de valor, pois os resultados são exatamente os mesmos.