Instituto Tecnológico de Aeronáutica

CT-213: Inteligência Artificial para Robótica Móvel

Lab 10: Programação Dinâmica

Bruno Benjamim Bertucci - Turma 23.2

1 Introdução

A modalidade de aprendizado de máquina denominada aprendizado por reforço (Em inglês, *Reinforcement Learning*, ou RL), consiste na otimização do comportamento de um agente para a realização de uma tarefa pelo fornecimento de sinais de recompensa, os quais ele busca maximizar. É uma tarefa significativamente mais complexa que aprendizado supervisionado, pois neste, são fornecidos dados anotados, o que torna mais fácil para o agente otimizar o seu comportamento.

Uma analogia para ilustrar a diferença entre aprendizado supervisionado e por reforço é a seguinte: Considere que o agente é um aluno, e sua tarefa é realizar provas de matemática. Em um contexto de aprendizado supervisionado, o aluno teria acesso ao gabarito da prova ao final de cada tentativa de realizá-la, e melhoraria seu desempenho a partir da comparação das suas respostas com o gabarito. Já no caso de aprendizado por reforço, o aluno teria acesso somente à sua nota final para pautar a otimização do seu desempenho. Essa analogia, embora muito generalizada, ajuda a entender alguns dos desafios de aprendizado por reforço.

A sequência de ações tomadas por um agente de RL para diferentes estados do seu ambiente é denominada política, e é denotada por $\pi(a|s)$. Dentro das formas de realizar aprendizado por reforço, existem os chamados Processos Decisórios de Markov (MDP, em ingles). Estes formam a política através dos dados representados pela tupla (S, A, p, r, γ), onde S é o conjunto finito de estados possíveis nos quais o agente pode estar, A é o conjunto finito de ações disponíveis para ele, p é a probabilidade de transição para um estado, dado o estado original do agente e a ação tomada, r é a recompensa esperada nessa transição de estados, e γ é um fator no intervalo de 0 a 1, usado no decaimento da contribuição de expectativas de recompensa para ações futuras.

Usando essas informações, o agente cria uma chamada função-valor, que é uma medida estatística do retorno esperado para cada estado ao seguir uma determinada política, e é denotada por $v_\pi(s)$. O problema, então, torna-se a otimização da função-valor e da política para a obtenção de um comportamento ótimo do agente.

2 Implementação de Programação Dinâmica

Uma classe de algoritmos destinados para a otimização de políticas e de funções-valor é a chamada programação dinâmica. Ela se baseia na equação de *Bellman* (Equação 1) para encontrar a função-valor ótima para uma determinada política π, de forma iterativa, e no uso dessa função para evoluir a política em si de forma gulosa.

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) \left(r(a|s) + \gamma \sum_{s' \in S} p(s'|s, a) v_\pi(s') \right) \tag{1}$$

2.1 Avaliação de Política

O algoritmo de avaliação de política resolve iterativamente o sistema formado pela equação de *Bellman*, uma vez que este é frequentemente complexo demais para ser resolvido analiticamente em um tempo computacional satisfatório. A equação é, então, adaptada para solução iterativa, como pode ser observado na Equação 2, onde k é um contador de iterações. Com um número suficientemente grande de iterações, v_{k+1} converge para v_π.

$$v_{k+1}(s) = \sum_{a \in A} \pi(a|s) \left(r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) v_k(s') \right) \tag{2}$$

2.2 Iteração de Valor

A implementação de iteração de valor, de forma semelhante à avaliação de política, também realiza uma convergência iterativa para a função-valor ótima, porém sem partir de uma política inicial. Para cada estado, calcula-se o valor da função para a realização de cada ação, e atualiza o valor dela para esse estado como sendo o valor máximo encontrado dentre as ações possíveis. A operação é descrita na Equação 3. Após a convergência para a função-valor ótima, encontra-se a política ótima de forma gulosa.

$$v_{k+1}(s) = max_{a \in A} \left(r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) v_k(s') \right) \tag{3}$$

2.3 Iteração de Política

A iteração de política é um algoritmo que calcula tanto a função-valor quanto a política ótima. Dado uma política e função-valor iniciais, o algoritmo, iterativamente, executa uma avaliação de política iterativa, com um número máximo de iterações reduzido (no caso dessa implementação, igual a 3), e, logo em seguida, evolui a política avaliada de forma gulosa de acordo com a Equação 4.

$$\pi'(s) = \text{greedy}(v_\pi(s)) \tag{4}$$

Todos os algoritmos implementados utilizam dois critérios de parada: o primeiro é um número máximo de iterações, igual a 10.000, e o segundo é um limite para o elemento com maior valor, em módulo, da matriz formada pela diferença entre a função-valor atual e a anterior, sendo esse limite dado por $\varepsilon = 10^{-5}$. Essa implementação de programação dinâmica foi feita para resolver problemas envolvendo mundos bi-dimensionais com um conjunto finito de estados e ações.

3 Teste dos algoritmos implementados

Foram realizados testes com os três algoritmos implementados, utilizando valores distintos para o fator de decaimento γ e a probabilidade de transição p. O problema proposto era encontrar a trajetória que um agente deveria seguir para alcançar um destino em um mundo bi-dimensional de posições discretas, com obstáculos tornando o caminho ótimo mais complexo. As ações possíveis para o agente são: permanecer parado, mover-se para cima, para baixo, para a esquerda ou para a direita, sendo estas representadas pelas letras S, U, D, L, R, respectivamente. Cada iteração executada implica uma penalização da recompensa em -1, exceto na célula objetivo, onde a recompensa é 0.

Para o teste da avaliação de política, foi criada uma política aleatória, que permite o agente executar qualquer uma das ações possíveis em todos os estados, exceto no estado designado como o destino, no qual a única ação possível é permanecer parado.

Já no teste da iteração de valor, o algoritmo é executado, iniciando com uma função-valor inicial, e retornando a função ótima. Em seguida, ela é usada para encontrar a política ótima de forma gulosa.

Finalmente, o teste de iteração de política inicia com a criação de uma política aleatória, semelhante àquela usada com a avaliação de política, retornando a política e a função-valor ótimos.

As matrizes correspondentes às políticas e funções-valores resultantes do teste realizado com $\gamma = 1$ e $p = 1$ encontram-se na Figura 1, e do teste realizado com $\gamma = 0,98$ e $p = 0,8$, na Figura 2. Na tabela que representa a política, cada carácter representa uma ação permitida pela política.

```
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    ]
```

(a) Avaliação de política aleatória.

```
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  ]
```

(b) Iteração de valor.

```
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  ]
```

(c) Iteração de política.

Figure 1: Resultados do teste com $p = 1$ e $\gamma = 1$.

Observa-se que a política aleatória fornece uma função-valor com valores muito baixos, o que se deve à grande quantidade de ações possíveis para cada estado. Observa-se, também, que os algoritmos de iteração de valor e iteração de política convergem, de fato, para uma política e função-valor ótimos, uma vez que indicam os caminhos mais rápidos do agente até o seu objetivo.

```
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    ]
```

(a) Avaliação de política aleatória.

```
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    ]
```

(b) Iteração de valor.

```
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    ]
```

(c) Iteração de política.

Figure 2: Resultados do teste com $p = 0,8$ e $\gamma = 0,98$.

Quando os parâmetros γ e p sofrem uma redução, a função-valor da avaliação de política aleatória fornece valores maiores que o teste anterior do mesmo algoritmo. Nesse caso, a redução de γ influencia significativamente nesse aumento, devido à grande quantidade de ações disponíveis para cada transição. Enquanto isso, a redução da probabilidade de transição atenua esse aumento da função-valor, pois ela leva o agente a percorrer mais iterações antes de alcançar seu destino.

Já no caso da iteração de valor e de política, ambos encontram a mesma política e função-valor, novamente. Além disso, em relação aos testes anteriores dos mesmos algoritmos, a política encontrada restringe mais as ações possíveis, e a função-valor produz valores ligeiramente menores. De forma distinta ao comportamento observado com a política aleatória, nesse teste, que lida com a política ótima, há uma quantidade menor de ações disponíveis para cada estado. Assim, a probabilidade menor de transição de estados causa uma redução da função-valor, que é parcialmente atenuada pelo valor menor de γ.