

Apredizado por Reforço

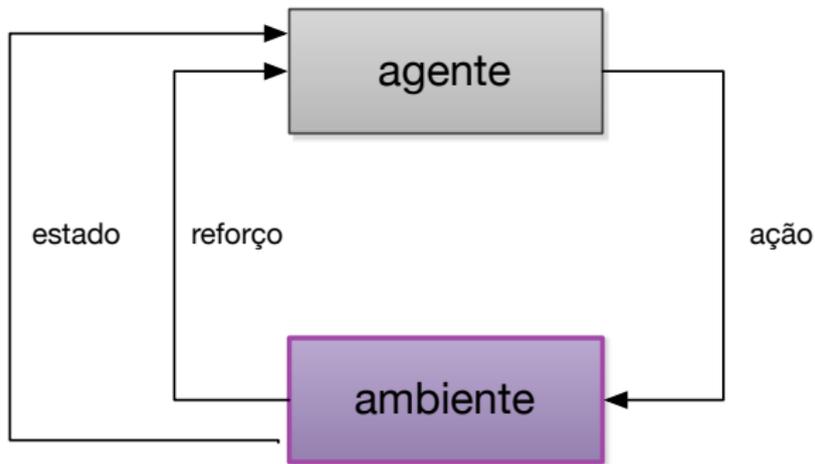
Jomi F. Hübner

Universidade Federal de Santa Catarina
Departamento de Automação e Sistemas
<http://jomi.das.ufsc.br/ia>



Visão geral

Um agente aprende a resolver uma tarefa através de repetidas interações com o ambiente, por **tentativa e erro**, recebendo (esporadicamente) reforços (punições ou recompensas) como retorno.



Processo decisório de Markov – MDP

Uma das abordagens em AR consiste em, usando técnicas estatísticas e métodos de programação dinâmica, estimar a ação de melhor resultado para cada situação (estado) do ambiente.

Definition (MDP)

Para formalização do problema, modela-se o problema como um MDP:

- um conjunto de estados do ambiente S
- um conjunto de ações A
- uma função que determina a recompensa imediata para as decisões do agente ($r : S \times A \rightarrow \mathbb{R}$), $r(s, a)$ é a recompensa por escolher a ação a no estado s
- uma função que representa as transições de estado do ambiente ($t : S \times A \times S \rightarrow \mathbb{R}$), $t(s, a, s')$ é a probabilidade do ambiente mudar do estado s para o estado s' caso a ação a seja executada

Propriedades de um MDP

$$(t : S \times A \rightarrow \Pi(S))$$

- O próximo estado do ambiente é determinado **somente** pelo estado corrente e pela ação escolhida, os estados ou ações anteriores não são necessários na função de transição
- O ambiente é
 - **indeterminístico**: a mesma ação pode levar a estados diferentes em ocasiões diferentes
 - **estacionário**: a probabilidade da uma ação levar a um estado específico não muda

Exemplo de MDP

Example

	1	2	3	4
1				
2				
3	■	■		
4	⊙			

Estados: as posições na grade ($s_{1,1}$, $s_{1,2}$, ..., $s_{4,4}$)

Ações: N, O, L, S

Recompensas: se a ação chegou na meta: 1, senão -1

$$r(s_{1,1}, N) = -1, r(s_{1,1}, O) = -1, \dots, r(s_{1,2}, N) = -1, \dots, \\ r(s_{4,2}, O) = 1$$

Transições: $t(s_{1,1}, N, s_{1,2}) = 0, t(s_{1,1}, O, s_{1,2}) = 0, \dots, \\ t(s_{1,1}, S, s_{2,1}) = 1, \dots$

Função valor – V

Definition (V)

A função valor determina um valor (utilidade) para cada estado do ambiente ($V : S \rightarrow \mathbb{R}$).

Cálculo do valor de cada estado

- **horizonte infinito**, com desconto, modelo indeterminístico

$$V^*(s) = \max_a \left(r(s, a) + \gamma \sum_{s' \in S} t(s, a, s') V^*(s') \right)$$

Equação de Bellman

Example

?	?	?	?
?	?	?	?
■	■	?	?
⊙	?	?	?

Função valor – V

Definition (V)

A função valor determina um valor (utilidade) para cada estado do ambiente ($V : S \rightarrow \mathbb{R}$).

Cálculo do valor de cada estado

- **horizonte infinito**, com desconto, modelo indeterminístico

$$V^*(s) = \max_a \left(r(s, a) + \gamma \sum_{s' \in S} t(s, a, s') V^*(s') \right)$$

Equação de Bellman

Example

-1	-1	-1	-1
-1	-1	-1	-1
■	■	-1	-1
⊙	1	-1	-1

Função valor – V

Definition (V)

A função valor determina um valor (utilidade) para cada estado do ambiente ($V : S \rightarrow \mathbb{R}$).

Cálculo do valor de cada estado

- **horizonte infinito**, com desconto, modelo indeterminístico

$$V^*(s) = \max_a \left(r(s, a) + \gamma \sum_{s' \in S} t(s, a, s') V^*(s') \right)$$

Equação de Bellman

Example

-5	-4	-3	-4
-4	-3	-2	-3
■	■	-1	-2
⊙	1	0	-1

Cálculo do valor (utilidade) dos estados

	1	2	3	4
1				
2				
3	■	■		
4	⊙			

$$\begin{aligned}
 V^*(s_{1,2}) &= \max_a (r(s_{1,2}, a) + \gamma \sum_{s'} t(s_{1,2}, a, s') V^*(s')) \\
 &= \max_{\substack{a \\ s' \in \{s_{1,1}, s_{1,3}, s_{2,2}, s_{1,2}\}}} \{ \\
 &\quad -1 + \gamma (0.8V^*(s_{1,1}) + 0.0V^*(s_{1,3}) + 0.1V^*(s_{2,2}) + 0.1V^*(s_{1,2})), \quad (a = O) \\
 &\quad -1 + \gamma (0.1V^*(s_{1,1}) + 0.0V^*(s_{1,3}) + 0.1V^*(s_{2,2}) + 0.8V^*(s_{1,2})), \quad (a = N) \\
 &\quad -1 + \gamma (0.1V^*(s_{1,1}) + 0.9V^*(s_{1,3}) + 0.05V^*(s_{2,2}) + 0.05V^*(s_{1,2})), \quad (a = L) \\
 &\quad -1 + \gamma (0.1V^*(s_{1,1}) + 0.0V^*(s_{1,3}) + 0.8V^*(s_{2,2}) + 0.1V^*(s_{1,2})) \quad (a = S) \\
 &\quad \}
 \end{aligned}$$

- há uma equação como essa para cada estado
- não são lineares (max)

Algoritmo Iteração-Valor

```
1 inicializar os valores de  $V'(s) = 0$ 
2 repeat
3    $V \leftarrow V'$ 
4    $\Delta \leftarrow 0$ 
5   foreach  $s \in S$  do
6     // Bellman update
7      $V'(s) \leftarrow \max_a (r(s, a) + \gamma \sum_{s'} t(s, a, s') V(s'))$ 
8      $\Delta \leftarrow \max(\Delta, |V'(s) - V(s)|)$ 
8 until  $\Delta$  pequeno
```

Se aplicado infinitamente, V é solução para as equações de Bellman ($V \approx V^*$)

Política de ação do agente – π

Como um agente escolhe uma ação?

- Escolher aquela que leva a um estado que dá a maior recompensa acumulada possível!

Definition (π^*)

A política ótima ($\pi^* : S \rightarrow A$) é dada por

$$\pi^*(s) = \arg \max_a \sum_{s'} t(s, a, s') V^*(s')$$

se escolhe a ação que leva ao estado de maior utilidade

Algoritmo Iteração-Política

- O algoritmo iteração valor não considera a política sendo usada pelo agente, mas todos os estados seguintes possíveis
- Considerando uma política, a equação de Bellman poderia ser simplificada:

$$V^*(s) = \max_a \left(r(s, a) + \gamma \sum_{s' \in \mathcal{S}} t(s, a, s') V^*(s') \right)$$

↓

$$V(s) = r(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} t(s, \pi(s), s') V(s')$$

removendo a não linearidade das equações!

- Porém, V será V^* só se π for π^*
- ↪ Temos que alterar também a política

Algoritmo Iteração-Política

- O algoritmo iteração valor não considera a política sendo usada pelo agente, mas todos os estados seguintes possíveis
- Considerando uma política, a equação de Bellman poderia ser simplificada:

$$V^*(s) = \max_a \left(r(s, a) + \gamma \sum_{s' \in \mathcal{S}} t(s, a, s') V^*(s') \right)$$

$$\Downarrow$$

$$V(s) = r(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} t(s, \pi(s), s') V(s')$$

removendo a não linearidade das equações!

- Porém, V será V^* só se π for π^*
- ↪ Temos que alterar também a política

Algoritmo Iteração-Política

```

1 repeat
2    $V \leftarrow \text{policy-evaluation}(\pi, V)$ ; //  $V$  se a política  $\pi$  for usada
3   modificado  $\leftarrow$  false
4   foreach  $s \in S$  do
5     // se existe uma ação melhor que a da política
6     if  $\max_a \sum_{s'} t(s, a, s') V(s') > \sum_{s'} t(s, \pi(s), s') V(s')$  then
7       // passa a usar essa ação
8        $\pi(s) = \arg \max_a \sum_{s'} t(s, a, s') V(s')$ 
9       modificado  $\leftarrow$  true
10  until not modificado

```

policy-evaluation (linha 2) pode ser calculado pelo iteração-valor usando a política π e atualização $V(s) \leftarrow r(s, \pi(s)) + \gamma \sum_{s'} t(s, \pi(s), s') V(s')$ um número k de vezes (não precisa ser o V exato)

Q-Learning

- Quanto o agente não conhece o modelo de transição de estados do mundo (t) ele não pode utilizar programação dinâmica para encontrar a política ótima.
- No Q-Learning, ao invés de considerar o valor de um estado ($V(s)$), passa-se a considerar o valor de escolher uma ação em um estado: o valor $Q(s, a)$.

Definition (Q)

$Q^*(s, a)$ denota a recompensa de escolher a ação a (não necessariamente a melhor) no estado s e depois continuar escolhendo as ações ótimas.

$$Q^*(s, a) = r(s, a) + \gamma \left(\sum_{s' \in \mathcal{S}} t(s, a, s') \max_{a'} Q^*(s', a') \right)$$

V e π no Q-Learning

Definition

No Q-Learning, os valores de V^* e π^* podem ser redefinidos da seguinte forma:

$$V^*(s) = \max_a Q^*(s, a)$$

$$\pi^*(s) = \arg \max_a Q^*(s, a)$$

A decisão do agente pode ser feita olhando somente o estado atual (sem calcular os futuros possíveis)!

O valor de Q inclui os futuros possíveis das opções de ações.

Aprender o valor de Q

O algoritmo de Q-Learning aproxima o valor Q^* a partir das experiências do agente, que tem a função de substituir a função de transição (no lugar da função t foi colocada a transição observada no ambiente).

Algoritmo para aprender o valor de Q

```
1  inicializar os valor de  $Q(s, a)$  arbitrariamente
2  for todos os episódios do
3      observar o estado inicial  $s$ 
4      repeat
5          escolher uma ação  $a$  para o estado  $s$ 
6          executar a ação  $a$ 
7          observar a recompensa  $r$  e o novo estado  $s'$ 
8           $Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$ 
           //  $\alpha$  é a taxa de aprendizado
9           $s \leftarrow s'$ 
10 until  $s$  ser um estado final
```

Propriedades do Q-Learning

- 1 Se cada ação for escolhida um número infinito de vezes em cada estado e o valor de α decair gradativamente, o valor de Q converge para Q^*
- 2 A política que o agente utiliza para escolher as ações (linha 5 do algoritmo) não interfere no aprendizado de Q^* (*off-policy*).
 - Poderia ser, por exemplo, escolher uma ação aleatoriamente.
 - Porém, normalmente se utiliza uma política que inicialmente escolhe aleatoriamente as ações e, à medida que vai aprendendo, passa a utilizar cada vez mais as decisões determinadas pela política derivada de Q .
 - Esta estratégia inicia *explorando* (tentar uma ação mesmo que ela não tenha o maior valor de Q) e termina *explorando* (escolher a ação que tem o maior valor de Q)

Exemplo de função para escolha de ações

ϵ -greedy

A escolha de uma ação para um estado é dada pela função

$$\epsilon\text{-greedy}(s) = \begin{cases} \text{random action from } A & \text{if } rv < \epsilon \\ \arg \max_a Q(s, a) & \text{otherwise} \end{cases}$$

onde a função retorna uma ação aleatória se um valor gerado aleatoriamente ($0 < rv \leq 1$) for menor que o fator de exploração ϵ ($0 \leq \epsilon \leq 1$). Caso contrário, retorna a ação com maior estimativa de recompensa.

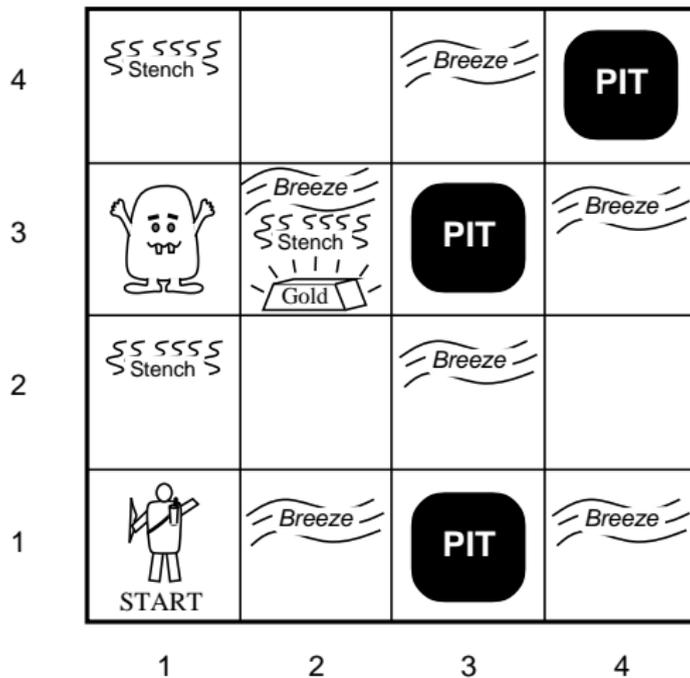
No primeiro caso, a função gera uma ação de exploração e no segundo caso uma ação de exploração.

O fator de exploração ϵ inicia com um valor alto (50%, por exemplo) e, conforme a simulação avança, diminui.

Exercício: Locação de Veículos

- João gerencia duas lojas de aluguel de veículos, uma localizada na cidade A e outra, na cidade B. Cada uma com no máximo 20 carros
 - Todos os dias clientes se dirigem a uma das lojas, havendo carro disponível na loja, João aluga um veículo por \$10, caso contrário o negócio é perdido
 - Para fazer com que veículos estejam disponíveis onde eles são procurado, veículos podem ser transportados de uma loja para outra a um custo de \$2 por veículo. No máximo, 5 carros são transferidos por dia
- ① Quais/Quantos são os estados?
 - ② Quais/Quantas são as ações?
 - ③ Qual a função de recompensa?
 - ④ Qual o tamanho do espaço de busca?

Exercício: Wumpus



Bibliografia

-  Leslie Pack Kaelbling, Michel L. Littman, and Andrew W. Moore.
Reinforcement learning: A survey.
Journal of Artificial Intelligence Reserach, 4:237–285, 1996.
-  Tom M. Mitchell.
Machine Learning.
McGraw-Hill, 1997.
-  Richard S. Sutton and Andrew G. Barto.
Reinforcement Learning: An Introduction.
Bradford, Cambridge, 1998.