CC7711

Inteligência Artificial e Robótica

Prof. Dr. Flavio Tonidandel

Aprendizado por Reforço

CC7711 - Inteligência Artificial e Robótica

Aprendizado por Reforço: Para quê ?

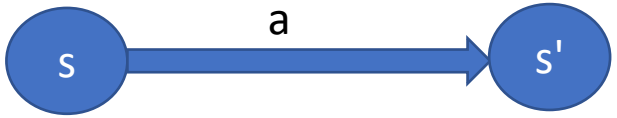
- Muitas vezes o uso de aprendizagem supervisionada é impraticável
  - Como obter exemplos de treinamento corretos para uma determinada situação? E se o ambiente for desconhecido?

- Exemplos:
  - Criança adquirindo coordenação motora
  - Robô interagindo com um ambiente para atingir objetivo(s)
  - Futebol de Robôs

- Mas como obter um sistema que aprende sem exemplos de treinamento ?!??!?

O que é aprendizagem por reforço ?

- Premissa:
  - A cada instante de tempo t, o agente está em um estado s.
  - No estado s ele executa a ação a e vai para o estado s´
  - Avalia-se o estado s´ e dá uma recompensa para o agente
  - Assim, a ação a no estado s possui um valor para o agente



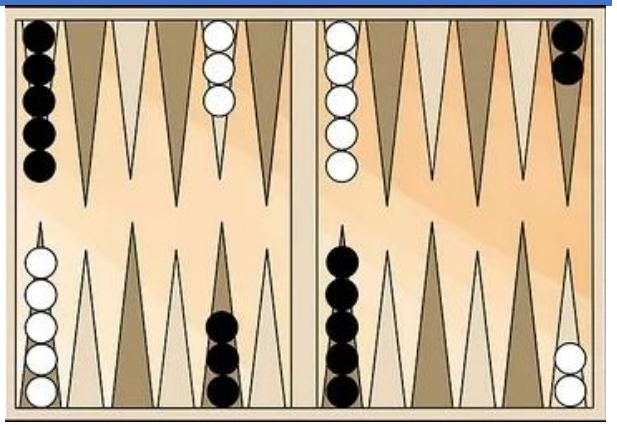
- Se escolha correta, ganha uma recompensa (ganha valor) senão recebe um castigo (perde valor)

- Aprendizagem por reforço :
  - Escolher uma *política de ações* que maximize o total de recompensas recebidas pelo agente

Premissas do Aprendizado por Reforço

- Especificar **o que** fazer, e não **como** fazer
  - Isso é feito por meio da função de recompensa
- Geralmente, encontra as melhores soluções finais
  - Baseado nas experiências atuais, não há suposições do programador
- Em suma:
- Menos tempo humano é necessário para encontrar uma boa solução
  - Não é necessário definir heurísticas, técnicas para solucionar o problema, etc.
  - Precisa apenas definir o sistema de aprendizado e deixar o sistema aprender !

Algumas aplicações



- JOGO de GAMÃO
  - Modelagem do jogo:
    - Vitória: +100
    - Derrota: – 100
    - Zero para os demais estados do jogo (*delayed reward*)
      - DELAYED REWARD -> deixa para dar recompensa no final de um processo
    - Após 1 milhão de partidas contra ele mesmo, joga tão bem quanto o melhor jogador humano

- Futebol de Robôs
  - Time Brainstormers de Futebol de Robôs (Robocup)
    - Time cujo conhecimento é obtido 100% por técnicas de aprendizagem por reforço

Markov Decision Processes

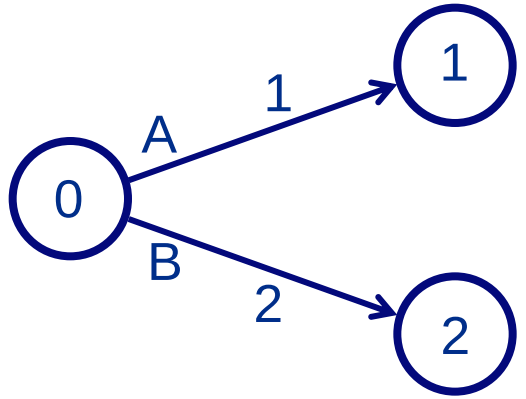
- formalmente, um MDP é dado por:
  - Um conjunto de estados, $S = \{s_1, s_2, \dots, s_n\}$
  - Um conjunto de ações, $A = \{a_1, a_2, \dots, a_m\}$
  - Uma função de Recompensa, $R: S \times A \times S \rightarrow \Re$
  - Uma função de transição de estados,
    - $T: S \times A \rightarrow S$

- Queremos aprender a politica π: S →A, ou seja, dado estados em S temos as melhores ações em A a serem aplicadas. Politica = sequência de estados e ações

- Propriedades de Markov
  - Tudo que precisa para tomar decisão está incluído no estado
  - Não há como consultar o passado (estados anteriores)

Tomando Decisões

- Com a recompensa estabelecida, o que precisamos é tomar uma decisão em cada estado:
  - Multiplas ações (A e B)
  - Cada ação tem uma recompensa associada a ela

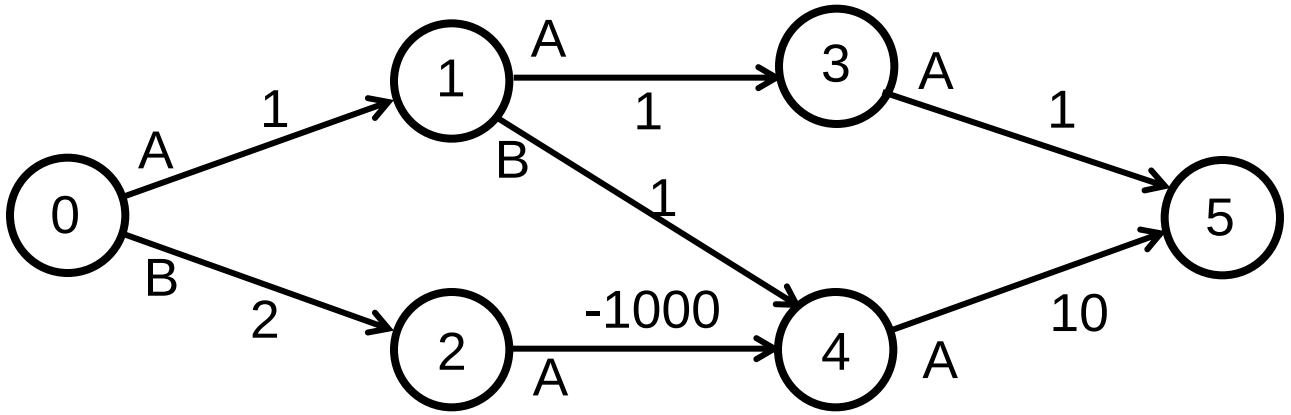


- O objetivo é maximizar a recompensa
  - Basta pegar a ação com a maior recompensa para o estado atual.

Markov Decision Processes (MDP)

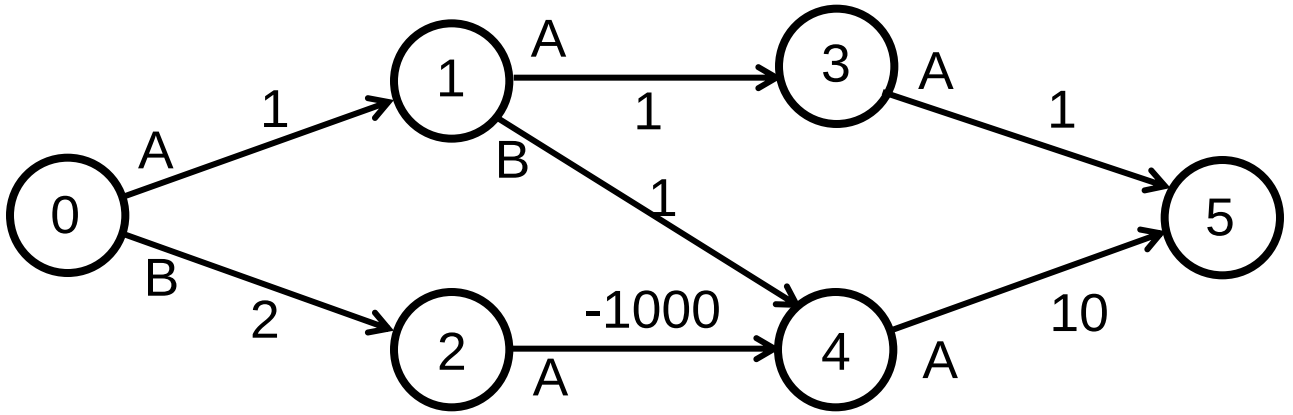
- Podemos generalizar o exemplo anterior para decisões multi-sequenciais
  - Cada decisão afeta a decisão seguinte

- Isto é formalmente modelado como Processo de Decisão de Markov (PDM ou MDP (inglês))



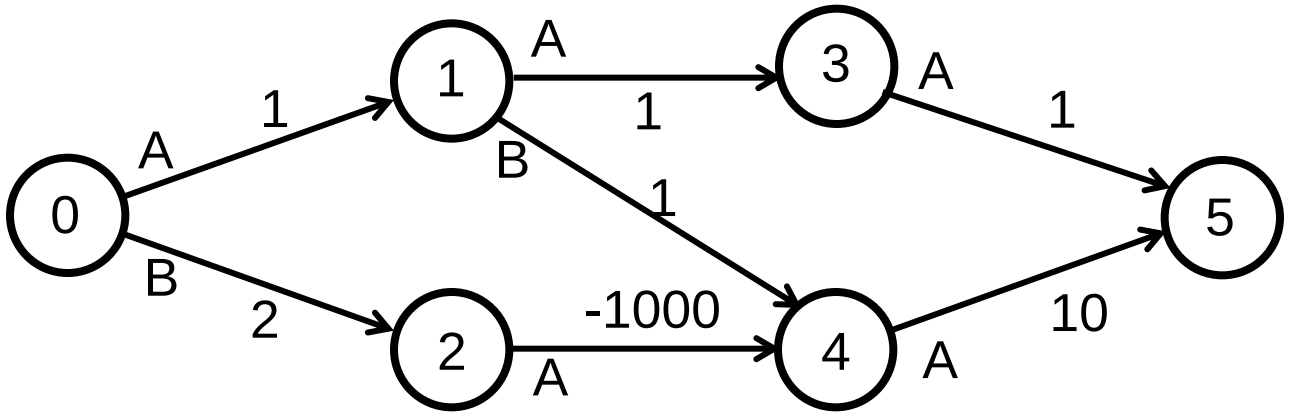
Políticas

- Existem 3 políticas para o MDP abaixo:
  1. 0 →1 →3 →5
  2. 0 →1 →4 →5
  3. 0 →2 →4 →5
- Qual é a melhor?



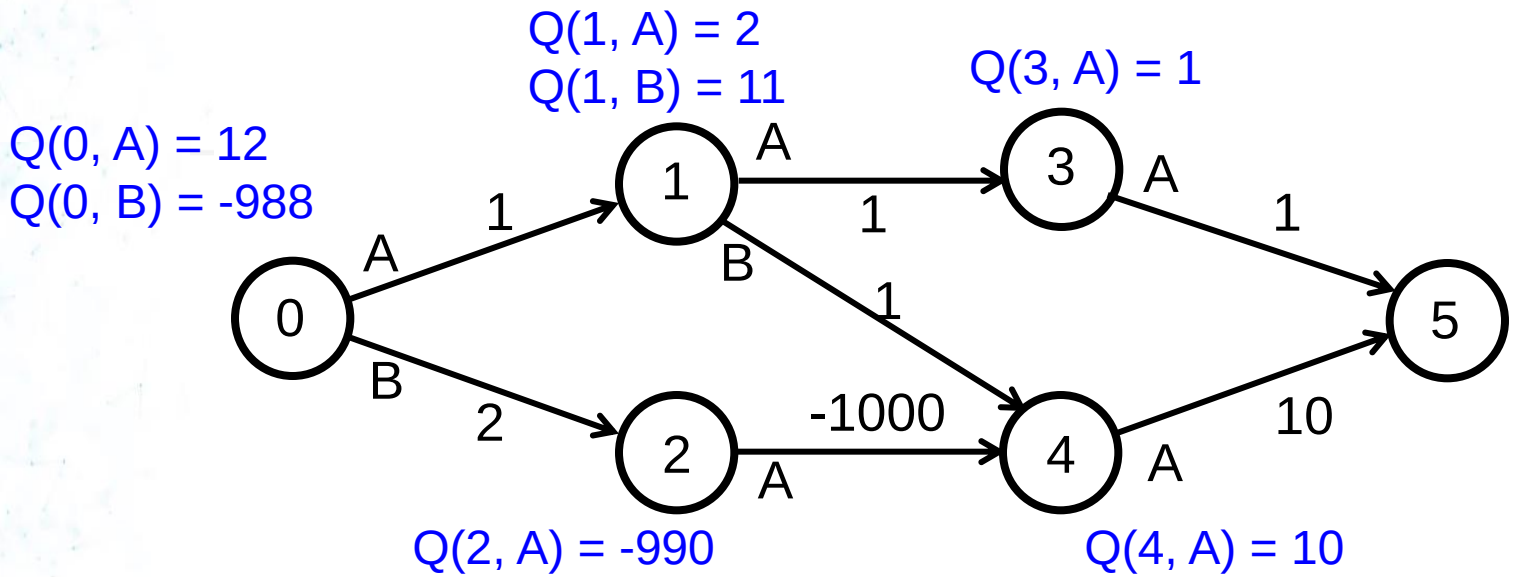
Politicas

- Ordene as politicas pelas recompensas
  1. 0 →1 →3 →5 = 1 + 1 + 1 = 3
  2. 0 →1 →4 →5 = 1 + 1 + 10 = 12
  3. 0 →2 →4 →5 = 2 – 1000 + 10 = -988



Função Estado-ação

- Podemos definir um valor sem especificar a política
  - Especificar o valor de escolher a ação **a** a partir do estado **s**
  - Isto é a função de qualidade estado-ação, Q



Funções valor

- $Q(s, a) = R(s, a, s') + \max_{a'} Q(s', a')$

s' é o próximo estado

- Forma:
  - Próxima recompensa + o melhor que posso fazer a partir do próximo estado

- Se nós temos a função valor, então achar a melhor politica é fácil:
  - $\pi(s) = \arg\max_a Q(s, a)$
  - arg max f(x) significa o argumento que torna a f(x) máxima

Política ótima

- Mas...
- Nós estamos procurando pela política ótima: $\pi^*(s)$
  - Isso significa que nenhuma política gera recompensa maior que π^*
- Política ótima define Funções valor ótimas:

$$Q^*(s,a) = R(s,a,s') + \text{argmax}_{a'} Q^*(s',a')$$

Aprendizado por Reforço

- O que acontece se nós não tivermos o MDP completo ?
  - Ou seja, precisamos aprender as recompensas associadas
  - Bem.. Sabemos sobre os estados e as ações
  - Não sabemos sobre o modelo do sistema (função de transição) ou a função de recompensa
- Podemos aprender pela experiência e executando ações para gerar tais experiências
- Este é o principal objetivo do aprendizado por reforço…

Aprendendo as Funções valor

- Nós ainda queremos aprender a função valor
  - Somos forçados a aproximá-la interativamente
  - Baseado nas experiências do mundo

- Vamos falar sobre um dos principais algoritmos:
  - Q-learning

  - Q Learning aproxima a função Q, que por sua vez, encontra a solução final sem termos o grafo completo.

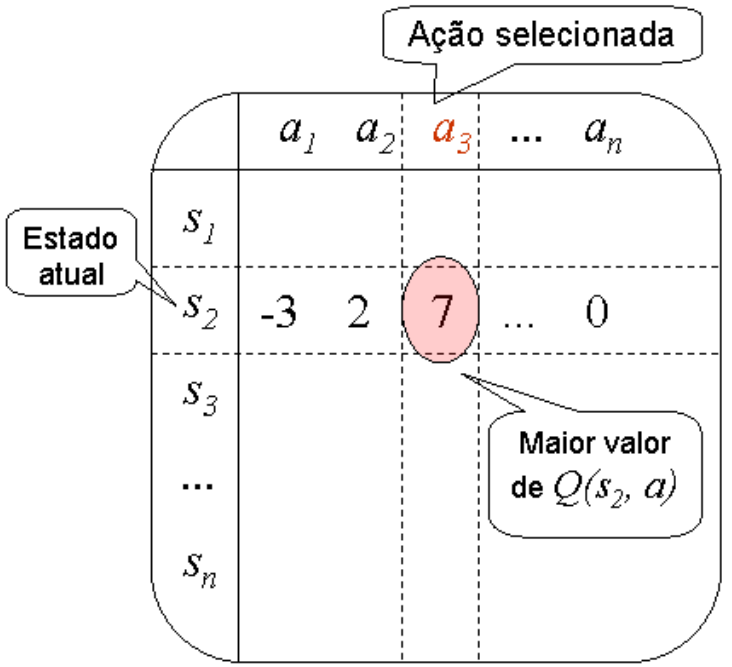
Funções valores...melhores

- Podemos introduzir um termo na função para evitar que valores altos saturem o sistema e o faça entrar em loopings.
  - Chamado de fator de desconto, γ
  - Interpretação:
    - Medida de incerteza herdada do mundo. Permite ao agente dar importância para valores no future e considerer apenas parte dos valores maximos de Q
  - $0 \leq \gamma \leq 1$
  - $Q(s, a) = R(s, a, s') + \gamma \max_{a'} Q(s', a')$

Q Learning

- **Algoritmo de aprendizagem para computar a função Q ótima (valor das ações)**
  - $\pi^*(s) = \text{argmax}_a[Q(s,a)]$

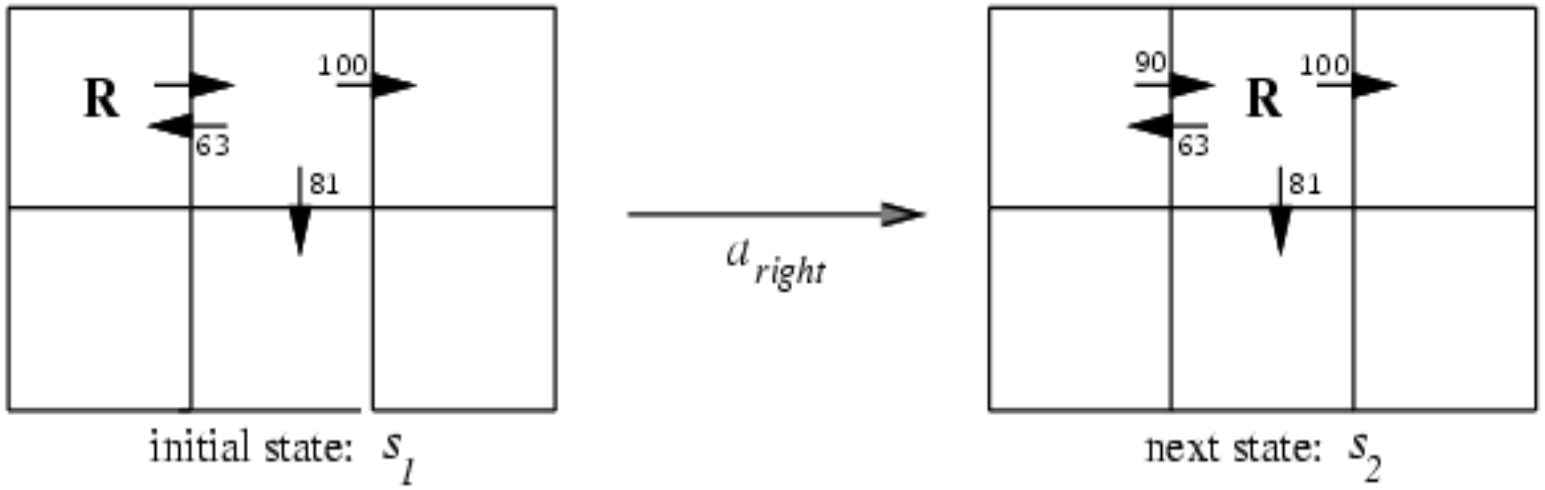
- $Q^*(s_t,a_t) = r(s_t,a_t) + \gamma \max_{a'} [Q(s_{t+1},a')]$



Esta tabela geralmente é enorme e ocupa muita memória !

Q-Learning (estados sem recompensa)

- **Atualiza-se $Q(s_t)$ após observar o estado s_{t+1}**

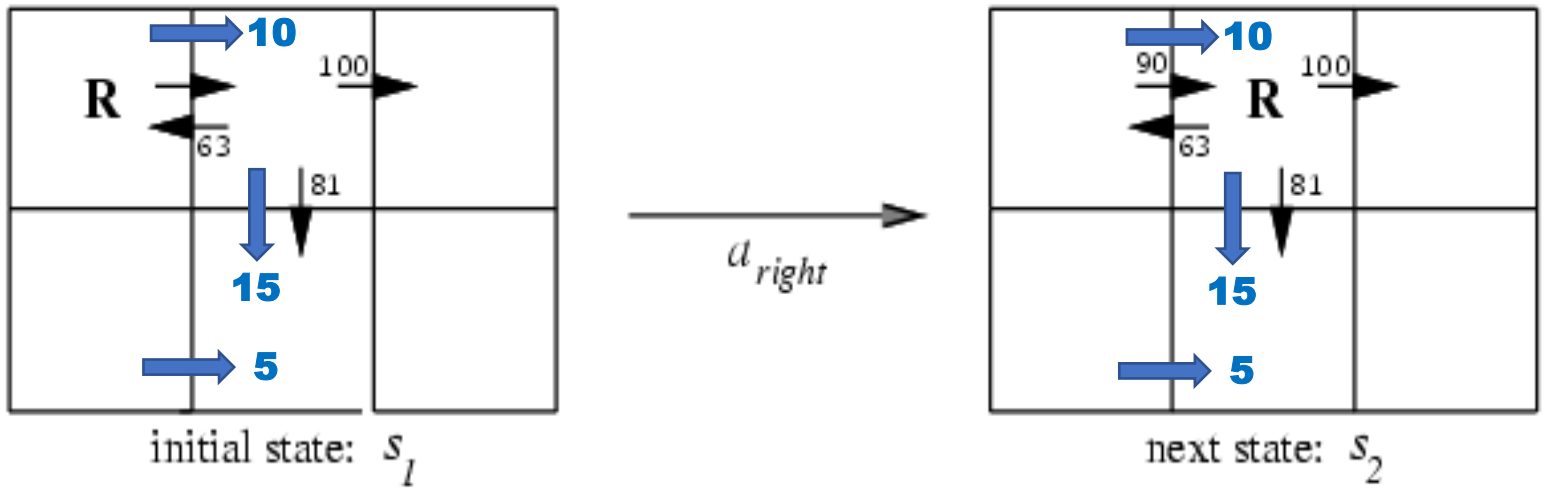


- **$Q(s_1,a_{right}) = r + \gamma \max_{a'} Q(s_2,a')$**
  $= 0 + 0.9 \max\{63,81,100\}$
  $= 90$

Q

Q-Learning (e com recompensa)

- **Atualiza-se $Q(s_t)$ após observar o estado s_{t+1} e recompensa recebida**



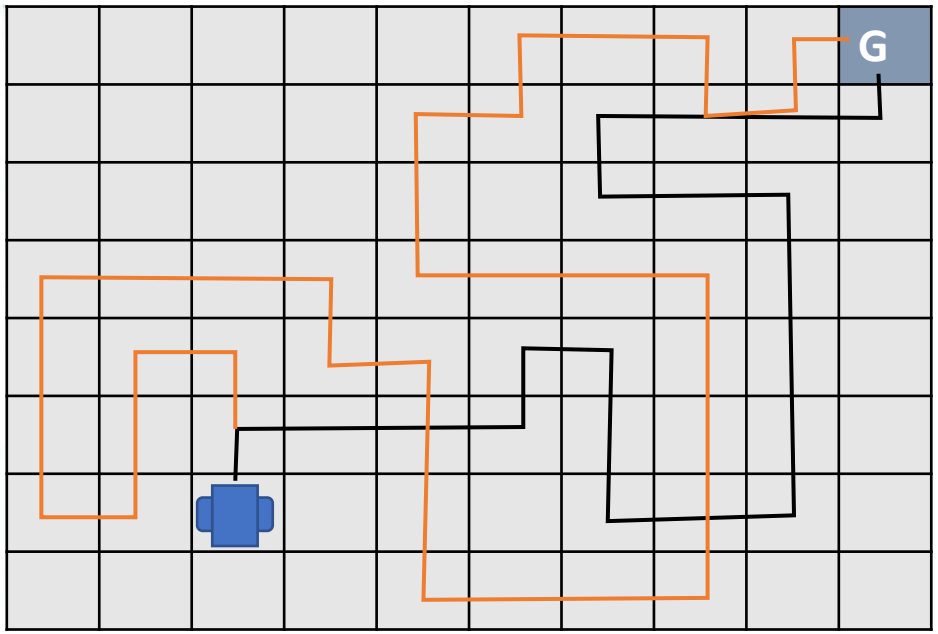
- **$Q(s_1,a_{right}) = 10 + \gamma \max_{a'} Q(s_2,a')$**
  $= 10 + 0.9 \max\{63,81,100\}$
  $= 100$

Q

Recompensa

Q-Learning

- Q-learning aproxima, iterativamente, a função valor estado-ação, Q
  - Não iremos estimar a MDP diretamente
  - Aprende a função valor e a política simultaneamente
- Mantém a estimativa de Q(s, a) em uma tabela
  - Atualiza essas estimativas conforme agrega mais experiência
  - A estimativa não depende da política de exploração



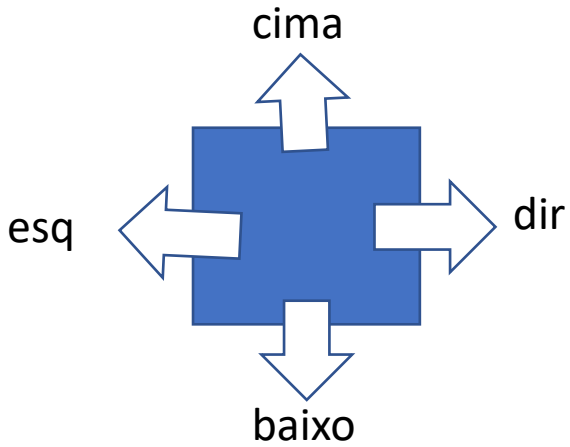


Tabela Q

80 estados x 4 ações

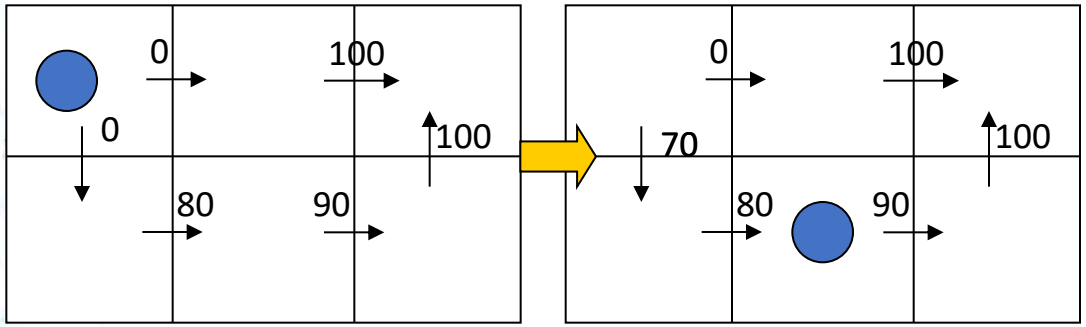
Algoritmo Q-Learning

1. Inicialize Q(s,a) para valores randomicos pequenos, $\forall s, a$
2. Observe estado, s
3. Escolha uma ação, a, e execute
4. Observe o próximo estado, s', e recompensa de s', r
5. $Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s', a'))$
6. Volte para 2

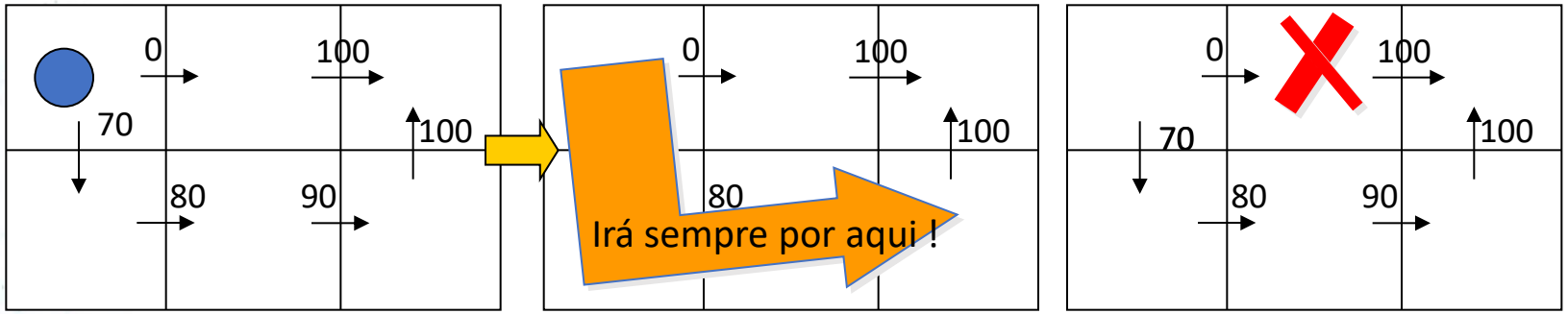
- $0 \leq \alpha \leq 1$ é a taxa de aprendizado

Um dilema !

- Se eu sempre escolher o valor Máximo para Q, eu posso cair em uma armadilha !!



Primeira interação



Dilema: Explorar ou Usufruir ?

- Usufruir
  - Escolher a ação que atualmente está com maior valor Q(s,a)
- Explorar
  - Escolher uma ação randômica, para que seu valor Q(s,a) seja atualizado
- Dilema
  - Dado que eu aprendi que Q(s,a) vale 100, vale a pena tentar executar a ação a’ se Q(s,a’) por enquanto vale 20 ?
    - Depende do ambiente, da quantidade de ações já tomadas e da quantidade de ações restantes

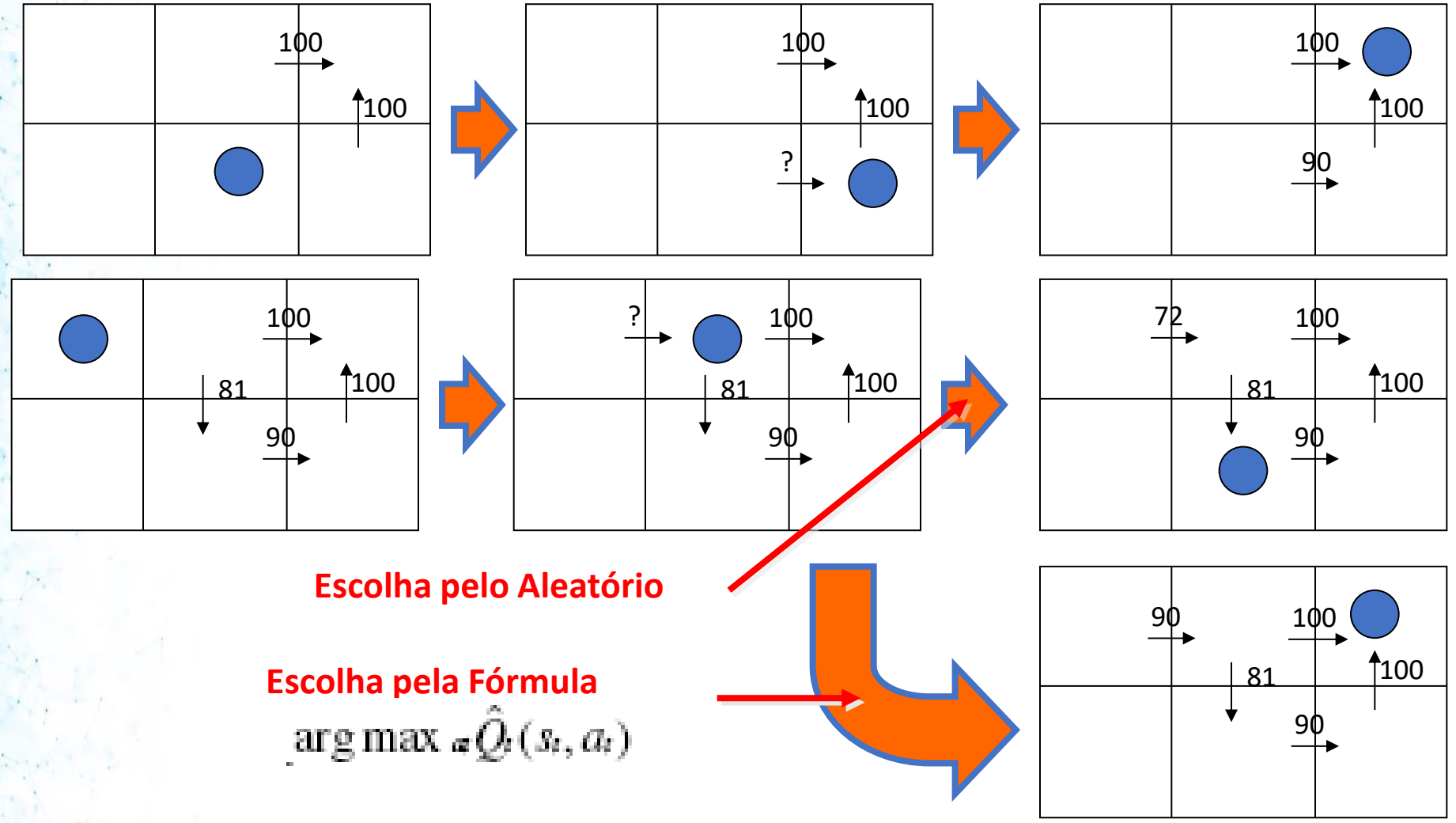
ε-Greedy

- Formula para resolver o “Dilema”:
- ε-Greedy: Exploração Aleatória
  - Dado um valor de q aleatório:

$$\pi(s_t) = \begin{cases} a_{random} & \text{se } q \leq \varepsilon, \\ \arg\max_a \hat{Q}_t(s_t, a_t) & \text{caso contrário,} \end{cases}$$

- O sistema irá escolher uma ação aleatória se q <= ε ou escolherá a ação de maior recompensa se q > ε
- Espera-se, com isso, que com muitas iterações possa-se chegar a solução ótima (política ótima)

Exemplo de Exploração



Considerações

- AR irá solucionar muitos dos seus problemas, entretanto:
  - Precisa de MUITO treinamento
  - Pegar ações aleatórias pode ser perigoso… e demorado
  - Leva muito tempo para aprender
  - Nem todos os problemas se encaixam no formato MDP
  - …
  - Claro… o algoritmo encontra a solução ótima (provado teoricamente) em infinitas iterações !!
    - Ou seja, por vezes temos que nos contentar com soluções sub-ótimas.

Bibliografia de Aprend. por Reforço

Para aprofundamento nos assuntos desta aula, segue a seguinte referência bibliográfica

- Russel & Norvig (Artificial Intelligence)
  - Capítulo 21

- Alguns slides desta aula foram baseados no slides:
- Hugo Pimentel de Santana.”Aprendizado por Reforço”. UFPE
- Bill Smart. “Reinforcement Learning: User’s Guide”.Washington University. USA. http://www.cse.wustl.edu/~wds/