# Reinforcement Learning

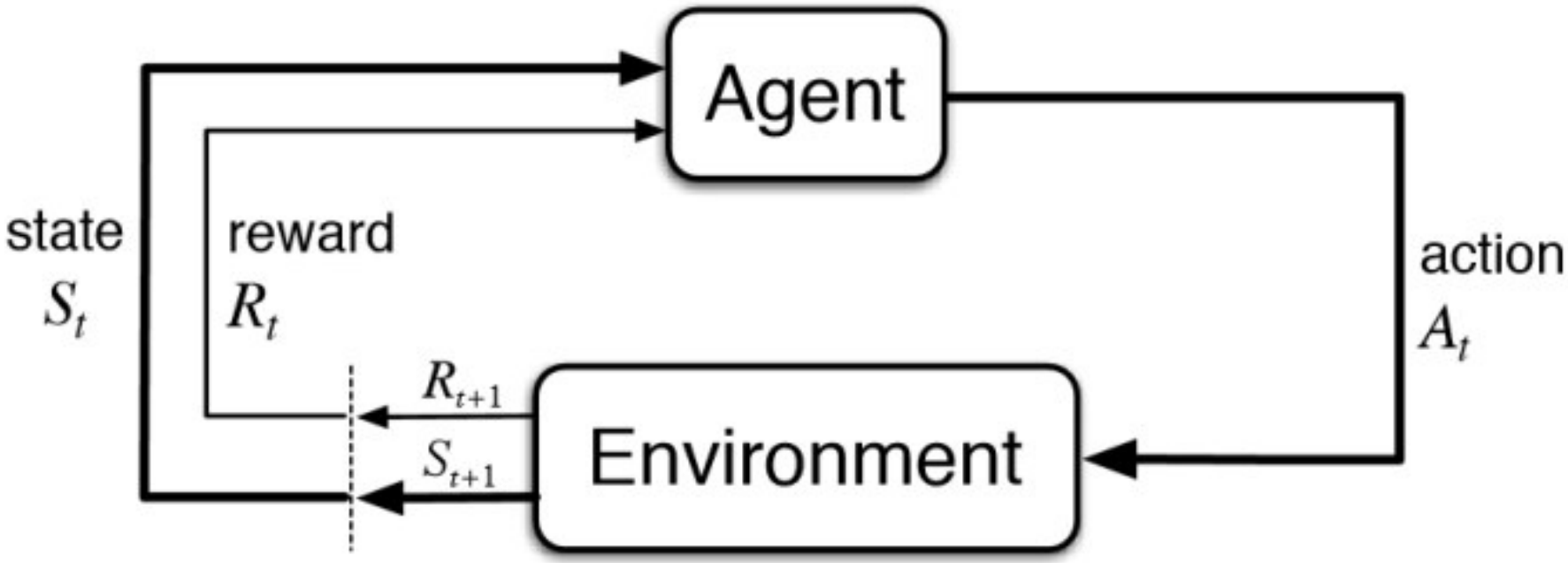
Prof. Maiquel de Brito[1]

[1]CAC – UFSC
Blumenau – SC – Brazil

# Reinforcement Learning

https://www.youtube.com/watch?v=JA96Fba-WHk

# Reinforcement Learning



# Reinforcement Learning

## Otimização

atingir o melhor resultado possível

## Consequências postergadas (delayed consequences)

Decisões atuais têm impacto (somente) no futuro

ex.: estudar no início do semestre, poupar para aposentadoria

## Exploração

O agente descobre como o mundo "funciona" explorando-o

## Generalização

Identificar novas situações a partir do aprendizado anterior

## RL vs. NN: delayed consequences, exploration

# Estados Markovianos ou Markov Property

Um estado é dito **markoviano** se o resultado de uma ação executada a partir dele depende apenas do estado atual e não de todos os estados anteriores

No caso de sistemas estocásticos: se as probabilidades de transição para estados futuros dependem apenas do estado atual

## Markov Decision Process - MDP

Definição de uma sequência de ações em um ambiente

1. completamente observável

   conhece-se todos os estados possíveis e as ações que levam a eles

2. estocástico

   não há garantias – apenas probabilidades – de atingir um estado a partir de uma ação

3. markoviano

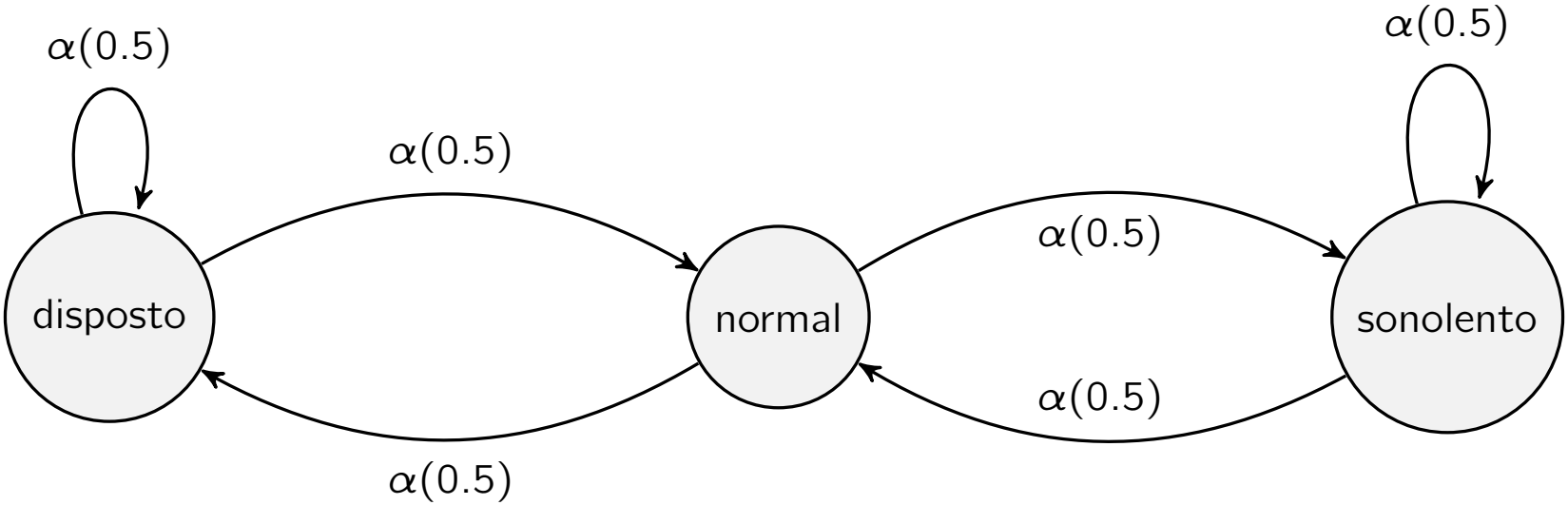
   o resultado de uma ação depende apenas do estado atual e não de todos os estados anteriores

# Markov Decision Process - MDP

Formalmente, um MDP é caracterizado por:

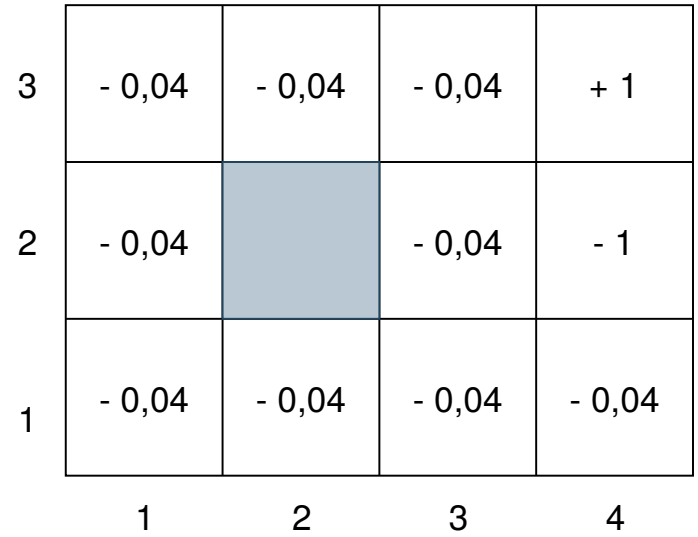
- Um conjunto $A$ de ações a serem executadas;
- Um conjunto $S$ de estados em que o agente pode se encontrar;
- Uma função de recompensa $R : S \to \mathbb{R}$, que define uma recompensa associada a cada estado;
- Um modelo de transição dado pela função $P : S \times A \times S \to \mathbb{R}$ s.t. $\forall_{p \in P} : p = P(s'|s, a)$

# MDP - Exemplo




- $A = \{\alpha\}$;
- $S = \{normal, sonolento, disposto\}$;
- $R(s) = \{r|(s, r) \in \{(disposto, 4), (normal, 0), (sonolento, -8)\}\}$
- $P(disposto, \alpha, disposto) = 0,5$
  $P(disposto, \alpha, normal) = 0,5$
  $P(disposto, \alpha, sonolento) = 0$
  $P(normal, \alpha, disposto) = 0,5$
  $P(normal, \alpha, normal) = 0$
  $P(normal, \alpha, sonolento) = 0,5$
  $P(sonolento, \alpha, disposto) = 0$
  $P(sonolento, \alpha, normal) = 0,5$
  $P(sonolento, \alpha, sonolento) = 0,5$

# MDP - Exemplo




- $A = \{Up, Down, Left, Right\}$
- $S =$
  $\{(1,1),(1,2),(1,3),(2,1),(2,2),(2,3),(3,1),(3,2),(3,3),(4,1),(4,2),(4,3)\}$
- $R(s) = \begin{cases} -0.04 & if\ s \neq (4,2) \wedge s \neq (4,3) \\ -1 & if\ s = (4,2) \\ +1 & if\ s = (4,3) \end{cases}$
- $P((1,1), Up, (1,2)) = 0.8$
- $P((1,1), Up, (1,1)) = 0.1$
- $P((2,1), Up, (1,2)) = 0.1$

$\vdots$

# MDP

exercícios...

# Utilidades de histórias em MDP - Recompensas aditivas

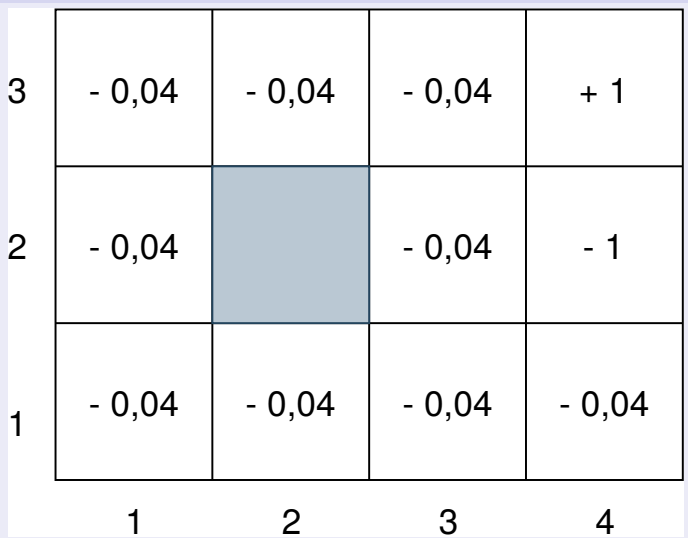
**História**: sequência de estados alcançados pelo agente

A utilidade de um MDP está relacionada às recopensas que o agente recebe ao longo do processo

Utilidade por **recompensa aditiva**: soma de todas as recompensas obtidas.

## Exemplo - Grid World

$$
\begin{aligned}
h =& [(1,1),(1,2),(1,3),(2,3),\\
& (3,3),(4,3))]\\
U_h =& (-0.04)+(-0.04)+\\
& (-0.04)+(-0.04)+\\
& (-0.04)+1=0.8
\end{aligned}
$$



# Utilidades de histórias em MDP - Recompensas descontadas

Utilidade por **recompensa descontada**:

fator de desconto $\gamma$ modela a preferência por recompensas imediatas
$\gamma \rightsquigarrow 0$: recompensas futuras são insignificantes
$\gamma = 1$: recompensas descontadas iguais às aditivas

$$U_h([s_0, s_1, s_2, \cdots, s_n]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots \gamma^n R(S_n)$$

ou

$$U_h([s_0, s_1, s_2, \cdots, s_n]) = \sum_{k=0}^{n} \gamma^k R(S_n)$$

# Recompensas descontadas - Exemplo

Sejam:

- $S = \{a, b, c, d, e\}$
- $R(a) = 1$; $R(b) = 3$; $R(c) = 5$; $R(d) = 8$; $R(e) = 10$
- $h_1 = [a, c, e]$ e $h_1 = [a, b, c, d]$

Recompensa aditiva

$$U_{h_1} = 1 + 5 + 10 = 16$$
$$U_{h_2} = 1 + 3 + 5 + 8 = 17$$

Recompensa descontada (com $\gamma = 0.8$)

$$U_{h_1} = 0.8^0 \times 1 + 0.8^1 \times 5 + 0.8^2 \times 10 = 11.4$$
$$U_{h_2} = 0.8^0 \times 1 + 0.8^1 \times 3 + 0.8^2 \times 5 + 0.8^3 \times 8 = 10.696$$

# Utilidades de estados em MDP

A seleção de uma ação requer que o agente conheça as utilidades dos estados que podem ser alcançados

A utilidade de um estado é afetada pela utilidade de seus vizinhos

## Equação de Bellmann

$$U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a) U(s')$$

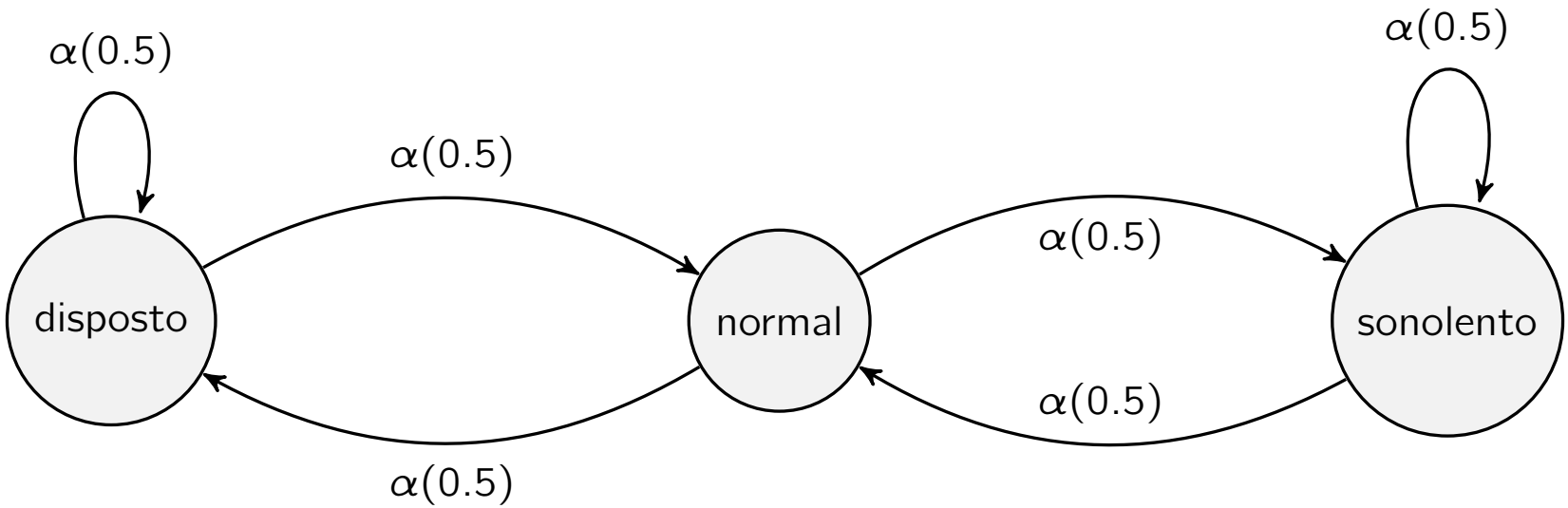
# Algoritmo Iteração-Valor

inputs : um MDP(um conjunto $S$ de estados; um conjunto $A$ de ações; um modelo de transição $P(s'|s,a)$; um conjunto $R$ de recompensas); fator de desconto $\gamma$

output: conjunto $U$ de utilidades

iniciar $U'$ com zeros
repeat
    $U \leftarrow U'$
    foreach $s \in S$ do
        $U'(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a)U(s')$
        $\delta \leftarrow max(\delta, |U'(s) - U(s)|)$
    end
until $\delta$ pequeno;
return $U$

# Equação de Bellman - Exemplo de aplicação




| k | U(disposto) | U(normal) | U(sonolento) |
|---|---|---|---|
| 0 | 4 | 0 | -8 |
| 1 | $4 + 0,5 \times [0,5 \times 4 + 0,5 \times 0] = 5$ | $0 + 0,5 \times [0,5 \times 4 + 0,5 \times -8] = -1$ | $-8 + 0,5[0,5 \times -8 + 0,5 \times 0] = -10$ |
| 2 | $4 + 0,5 \times [0,5 \times 5 + 0,5 \times -1] = 5$ | $0 + 0,5 \times [0,5 \times 5 + 0,5 \times -10] = -1,25$ | $-8 + 0,5 \times [0,5 \times -1 + 0,5 \times -10] = -10,75$ |
| 3 | $4 + 0,5 \times [0,5 \times 5 + 0,5 \times -1,25] = 4,9375$ | $0 + 0,5 \times [0,5 \times 5 + 0,5 \times -10,75] = -1,4375$ | $-8 + 0,5 \times [0,5 \times -1,25 + 0,5 \times -10,75] = -11$ |
| 4 | $4 + 0,5 \times [0,5 \times 4,9375 + 0,5 \times -1,4375] = 4,875$ | $0 + 0,5 \times [0,5 \times 4,9375 + 0,5 \times -11] = -1,5156$ | $-8 + 0,5 \times [0,5 \times -1,4375 + 0,5 \times -11] = -11,1094$ |
| 5 | $4 + 0,5 \times [0,5 \times 4,875 + 0,5 \times -1,5156] = 4,8398$ | $0 + 0,5 \times [0,5 \times 4,875 + 0,5 \times -11,1094] = 1,5586$ | $-8 + 0,5 \times [0,5 \times -1,5156 + 0,5 \times -11,1094] = 11,1562$ |

# Utilidades em MDP – Exercício

1. Para o cenário *grid world*, calcular a utilidade de cada um dos estados considerando horizontes de 1, 2, 3, 4 e 5 passos

2. Analisar a implementação do algoritmo iteração-valor disponivel no Moodle

3. Modificar parâmetros da implementação do algoritmo iteração-valor e analisar os resultados

4. Para o cenário *grid world*, calcular a utilidade de cada um dos estados considerando horizonte infinito

# Políticas

**MDP:** Markov Decision Process

Decision: como agir em um ambiente estocástico?

A solução para um MDP é uma sequência de ações

É necessário definir uma **política ($\pi$)**

i.e. ações para todos os possíveis estados

pois uma ação pode levar a um estado diferente do pretendido e o agente precisa saber o que fazer neste caso

# Políticas

$\pi : S \to A$

$\pi(s) \in A$: ação $a \in A$ recomendada pela política $\pi$ no estado $s$

Política ótima ($\pi^*$): política que produz a maior utilidade para o agente

# Políticas - *Policy evaluation*

Utilidade de $\pi$ ($U^{\pi}$): considera a utilidade *esperada* em cada estado
recompensas imprevisíveis

Utilidade de um estado $s$ sob uma política $\pi$:

$$U^{\pi}(s) = R(s) + \gamma \sum_{s'} P(s'|s, \pi(s)) U(s') \tag{1}$$

Política ótima: ($\pi^*$):

$$\pi^*(s) = \underset{a \in A(s)}{\operatorname{argmax}} \sum_{s'} P(s'|s, a) U(s') \tag{2}$$

Discussão: diferenças entre a Equação 1 e a Equação de Bellman

# Políticas - Exercício

Encontrar a política ótima do cenário *Grid World* considerando um horizonte de 5 passos.

– considerar a política inicial composta por ações aleatórias em todos os estados

# Algoritmo Iteração-Política

**inputs :** um MDP(um conjunto $S$ de estados; um conjunto $A$ de ações; um modelo de transição $P(s'|s,a)$; um conjunto $R$ de recompensas)

**output:** uma política $\pi$

**repeat**
  $U \leftarrow$*policy_evaluation*$(\pi, U, mdp)$
  $unchanged \leftarrow$ *true*
  **foreach** $s \in S$ **do**
    **if** $\max_{a \in A(s)} \sum_{s'} P(s'|s,a)U(s') > \sum_{s'} P(s'|s,\pi(s)U(s')$ **then**
      $\pi(s) \leftarrow \operatorname{argmax}_{a \in A(s)} \sum_{s'} P(s'|s,a)U(s')$
      $unchanged \leftarrow$ *false*
    **end**
  **end**
**until** *unchanged*;
**return** $\pi$

# Aprendizado por reforço

MDP: agente conhece o modelo do ambiente

- modelo de transição, recompensas etc
- é possível calcular $\pi^*$

Aprendizado por reforço: agente aprende $\pi^*$ enquanto atua

- descobrindo modelo de transição, recompensas etc

# Métodos de Monte Carlo - *First-Time Monte Carlo*

**inputs :** conjunto $S$ de estados, política $\pi$

**output:** conjnto $U^\pi$ de utilidades dos estados sob a execução de $\pi$

inicializar $N(s) \leftarrow 0$, $U(s) \leftarrow 0\ \forall s \in S$

**repeat**

  gerar um episódio $i = \{(s_1^i, a_1^i, r_1^i), \cdots, (s_n^i, a_n^i, r_n^i)\}$

  s.t $s_k^i, a_k^i, r_k^i$ são o estado, ação e recompensa observados no $k$ºpasso do $i$ºepisódio

  $U_t^i(s) \leftarrow r_t^i + \gamma r_{t+1}^i + \gamma^2 r_{t+2}^i + \cdots + \gamma^{n-1} r_{t+n}^i$

  **foreach** estado $s \in S$ visitado em $i$ **do**

    **for** primeiro tempo $t$ que $s$ é visitado **do**

      $N(s) = N(s) + 1$

      $U(s) = U(s) + U_t^i$

      estimar $U^\pi(s) = U(s)/N(s)$

    **end**

  **end**

**until** forever;

**return** $U^\pi$

# Métodos de Monte Carlo - *Every-Time Monte Carlo*

**inputs :** conjunto $S$ de estados, política $\pi$

**output:** conjnto $U^\pi$ de utilidades dos estados sob a execução de $\pi$

inicializar $N(s) \leftarrow 0$, $U(s) \leftarrow 0\ \forall s \in S$

**repeat**

  gerar um episódio $i = \{(s_1^i, a_1^i, r_1^i), \cdots, (s_n^i, a_n^i, r_n^i)\}$

  s.d $s_k^i, a_k^i, r_k^i$ are the state, action and reward observed in the $k^{th}$ step of the $i^{th}$ episode

  $U_t^i(s) \leftarrow r_t^i + \gamma r_{t+1}^i + \gamma^2 r_{t+2}^i + \cdots + \gamma^{n-1} r_{t+n}^i$

  **foreach** estado $s \in S$ visitado em $i$ **do**

    **for** todo tempo $t$ que $s$ é visitado **do**

      $N(s) = N(s) + 1$

      $U(s) = U(s) + U_t^i$

      estimar $U^\pi(s) = U(s)/N(s)$

    **end**

  **end**

**until** forever;

**return** $U^\pi$

# Aprendizado por reforço passivo

Agente conhece $\pi$ i.e. não decide quais ações executar
mas não conhece

- o modelo de transição

  (não sabe quais estados são alcançados através das ações)
- recompensas

O agente aprende o valor da política $\pi$

# Estimação de utilidade direta

A utilidade de um estado é o valor esperado das recompensas obtidas a partir dele em sucessivas execuções da política $\pi$.

$$U^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(S_t)\right] \tag{3}$$

# Programação dinâmica adaptativa

A utilidade de um estado é resultado da equação de Bellman, calculada durante a execução da política $\pi$.

$$U^{\pi}(s) = R(s) + \gamma \sum_{s'} P(s'|s, \pi(s)) U^{\pi}(s') \tag{4}$$

# Aprendizado por reforço ativo

Agente não conhece $\pi$, desconhecendo, assim

- o modelo de transição

  (não sabe quais estados são alcançados através das ações)
- recompensas

O agente aprende a própria política $\pi$

# Q-Learning

Aprende-se a utilidade $Q(s, a)$ de executar a ação $a$ no estado $s$

em vez de aprender a utilidade $U^{\pi}(s)$

porque não se conhece a política $\pi$

Política ótima (a ser aprendida pelo agente):

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}\, Q(s, a) \tag{5}$$

# Q-Learning

A utilidade de um estado é o resultado da execução da melhor ação a partir dele:

$$U(s) = \max_a Q(s, a) \tag{6}$$

O resultado de uma ação depende dos estados subsequentes:

$$Q(s, a) = R(s) + \gamma \sum_{s'} P(s'|s, a) \max_{a'} Q(s', a') \tag{7}$$

# Q-Learning - Algoritmo

**Algorithm 1:** Algoritmo Q-Learning

```
1  Inicializar Q(s,a) aleatoriamente
2  for todos os episódios do
3      repeat
4          observar o estado s
5          escolher uma ação a no estado s
6          executar a
7          observar o novo estado s'
8          observar a recompensa R(s') obtida
9          Q(s, a) = Q(s, a) + α(R(s') + γ max_a' Q(s', a') − Q(s, a))
10         s ← s'
11     until s ser um estado terminal;
12 end
```

# Q-Learning

$$Q(s,a) = Q(s,a) + \alpha(R(s') + \gamma \max_{a'} Q(s',a') - Q(s,a)) \quad (8)$$

- $\alpha$: captura o quanto a transição para o estado $s'$ é relevante;
- $\gamma$: captura a importância da execução da melhor ação no próximo estado ($s'$)
- $max_{a'} Q(s',a') - Q(s,a)$: captura a vantagem obtida na mudança do estado $s$ para $s'$

# Q-Learning - $\epsilon$-greedy

$$\epsilon\text{-greedy}(s, \epsilon, x) = \begin{cases} random(A) & if\ x < \epsilon \\ \text{argmax}_a\, Q(s, a) & otherwhise \end{cases} \tag{9}$$

$$\text{s.t. } s \in S; 0 \leq \epsilon \leq 1; 0 \leq x \leq 1,$$

$x < \epsilon$: exploração
$x \geq \epsilon$: exploitação