Aprendizagem por Reforço

Capítulo 23

Secções 23.1-23.3

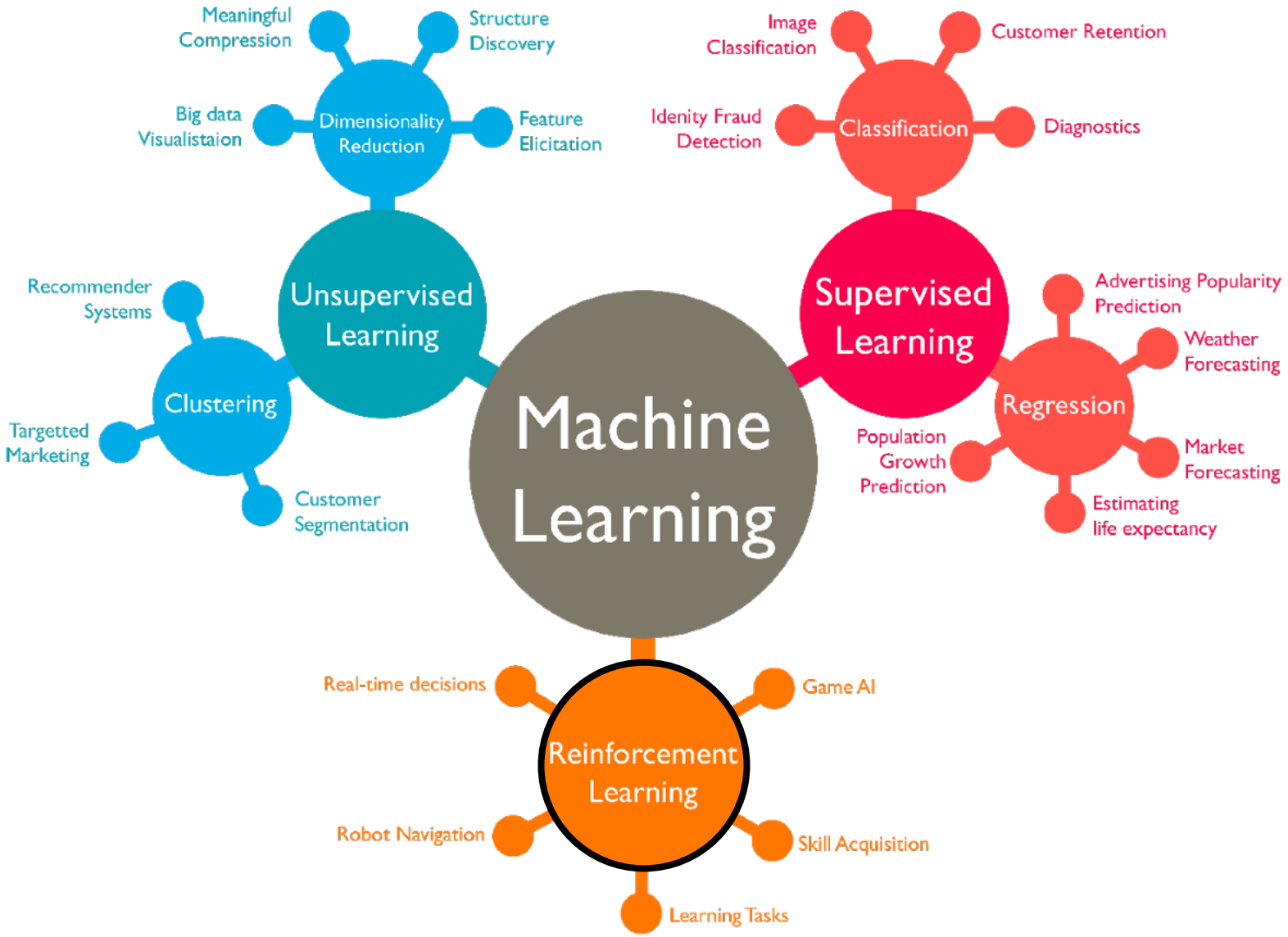
Aprendizagem automática

- Arthur Samuel, 1956

"Machine learning", jogo de damas

- Tom M. Mitchell, 1997

"A computer program is said to **learn** from experience *E* with respect to some class of tasks *T* and performance measure *P* if its performance at tasks in *T*, as measured by *P*, improves with experience *E*."


Machine Learning
Unsupervised Learning
Dimensionality Reduction
Meaningful Compression
Structure Discovery
Big data Visualistaion
Feature Elicitation
Clustering
Recommender Systems
Targetted Marketing
Customer Segmentation
Supervised Learning
Classification
Image Classification
Customer Retention
Idenity Fraud Detection
Diagnostics
Regression
Advertising Popularity Prediction
Weather Forecasting
Market Forecasting
Estimating life expectancy
Population Growth Prediction
Reinforcement Learning
Real-time decisions
Game AI
Robot Navigation
Skill Acquisition
Learning Tasks

Aprendizagem…

- Aprendizegem supervisionada: treino com input/output
  - Exemplos de fruta categorizados em banana, maçã, laranja, etc.
- Aprendizagem não supervisionada: treino com input apenas
  - Exemplos de fruta que vão sendo separados pelas suas características
- Aprendizagem por reforço
  - Não requer treino prévio
  - Agente monitoriza respostas às ações (recompensa)

Aprendizagem por reforço

A tarefa de **aprendizagem por reforço** consiste em usar recompensas que são observadas para **aprender** (ou avaliar) uma política (*policy*) ótima (ou perto de ótima) para o ambiente

política = escolha de ação

Aprendizagem por reforço

- Multi-agent hide and seek

https://www.youtube.com/watch?v=kopoLzvh5jY

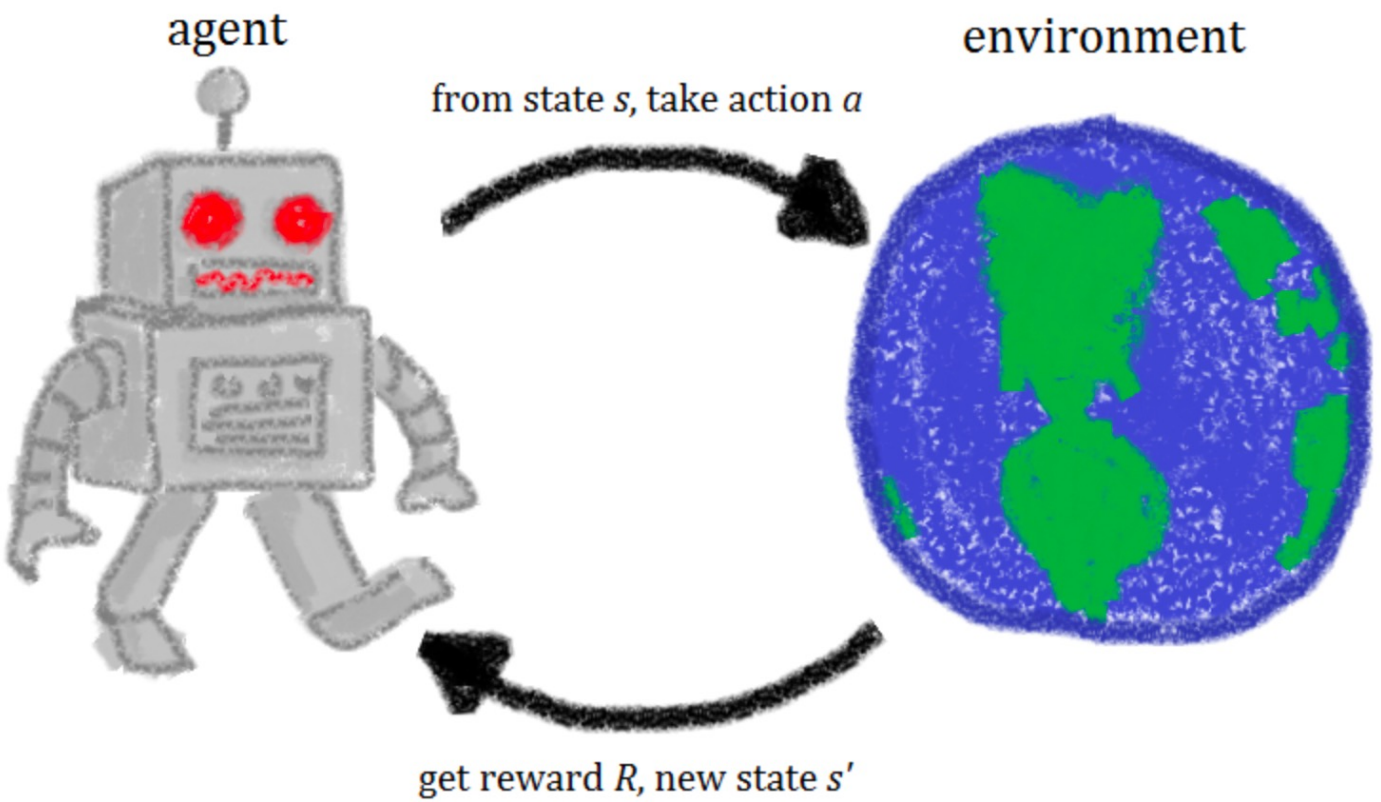
Aprendizagem por reforço

A aprendizagem por reforço pode ser considerada uma **abordagem que envolve toda a IA**:

um agente é colocado num ambiente e tem de aprender a comportar-se para aí ter sucesso

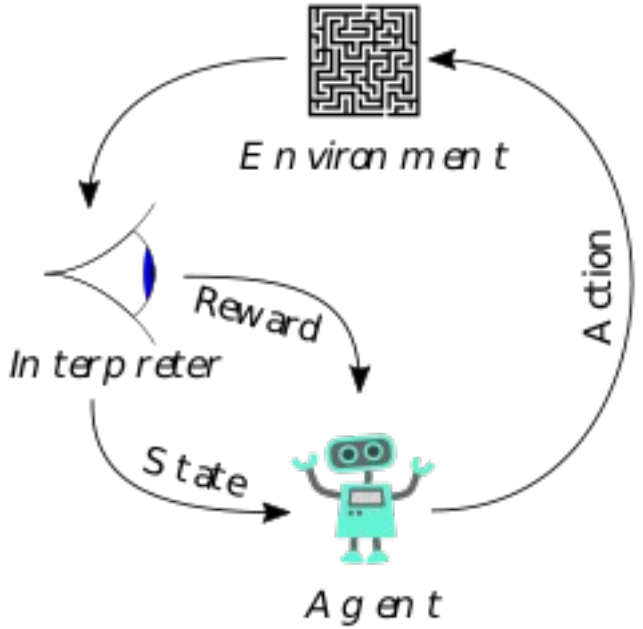
Saber mais… https://youtu.be/nyjbcRQ-uQ8

Aprendizagem por reforço



Aprendizagem por reforço

- Ideia base:
  - Receber feedback na forma de **recompensa** (**reforço** em Psicologia)
  - Agente tem de (aprender a) agir para *maximizar reforços* esperados
  - Aprendizagem baseada nos exemplos de output **observados**



Exemplos: recompensas em jogos

- Xadrez
  - Só recebe feedback no fim do jogo (*sparse rewards*)
  - Ou a cada momento teria de classificar cada uma das jogadas…
- Ping pong
  - Recebe feedback por cada ponto

- Mais simples dar recompensa do que modelar comportamento!
- Aprendizagem por reforço estudada em psicologia animal desde o século XIX!

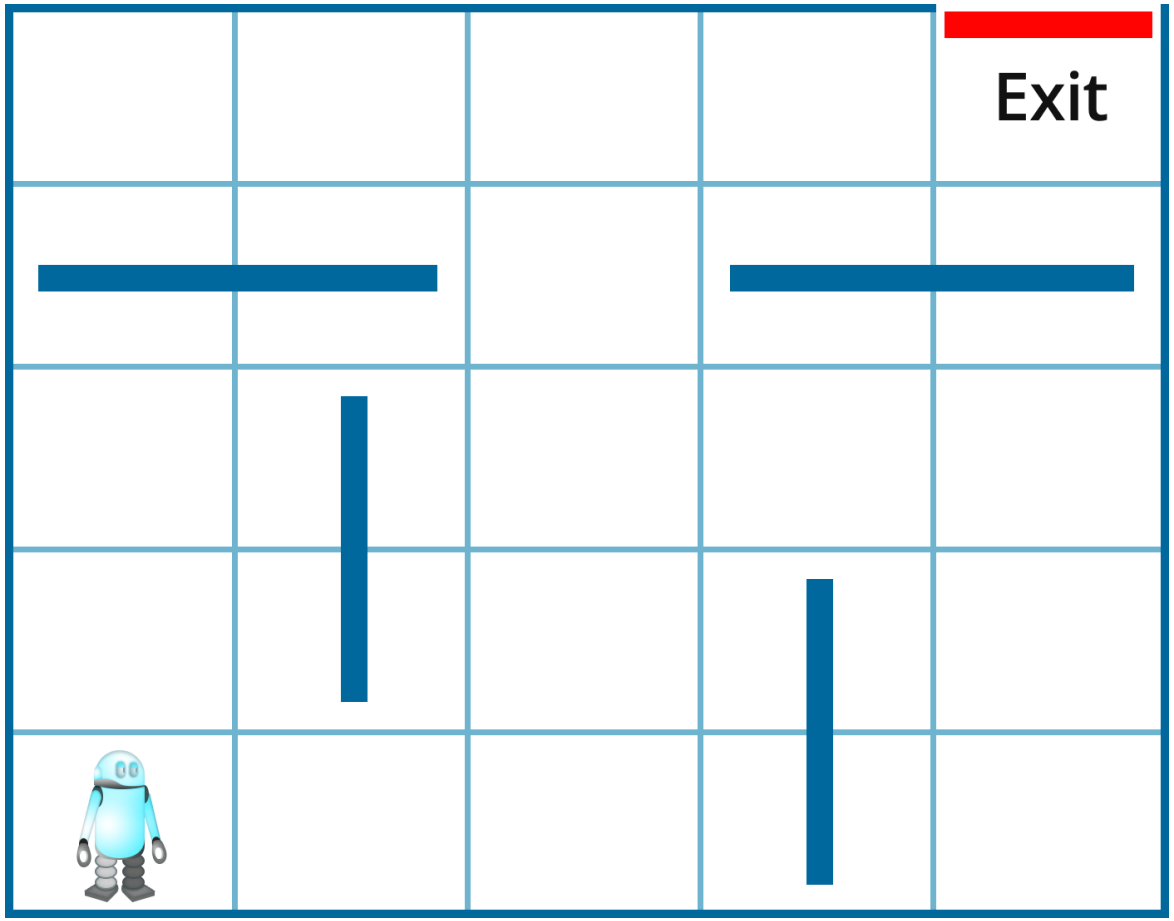
Modelação do problema

A cada momento, o agente tem de selecionar uma ação

Como resultado da ação, recebe uma recompensa e atualiza estado

- Espaço de estados: $X = x_1, x_2, ..., x_n$
- Espaço de ações: $A = a_1, a_2, ..., a_m$
- Função de recompensa: $X \times A \rightarrow R$ (ou $X \times A \times X \rightarrow R$ no caso de não determinismo)

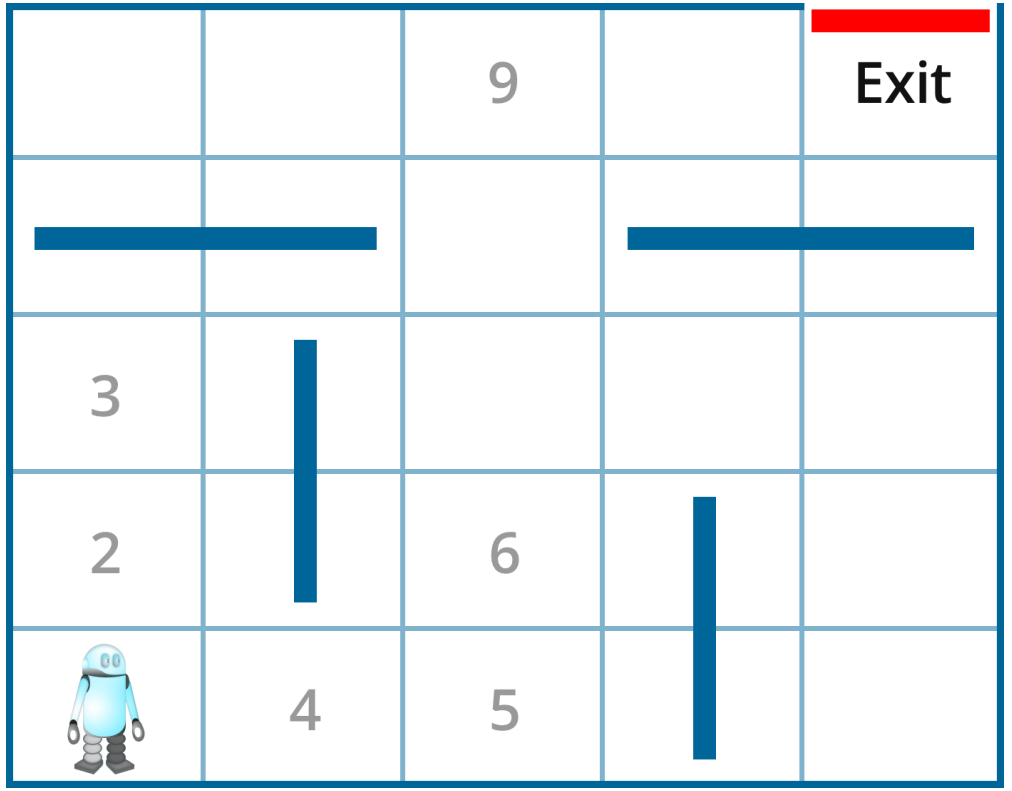
Exemplo: ambiente



Estados? Ações? Recompensas?

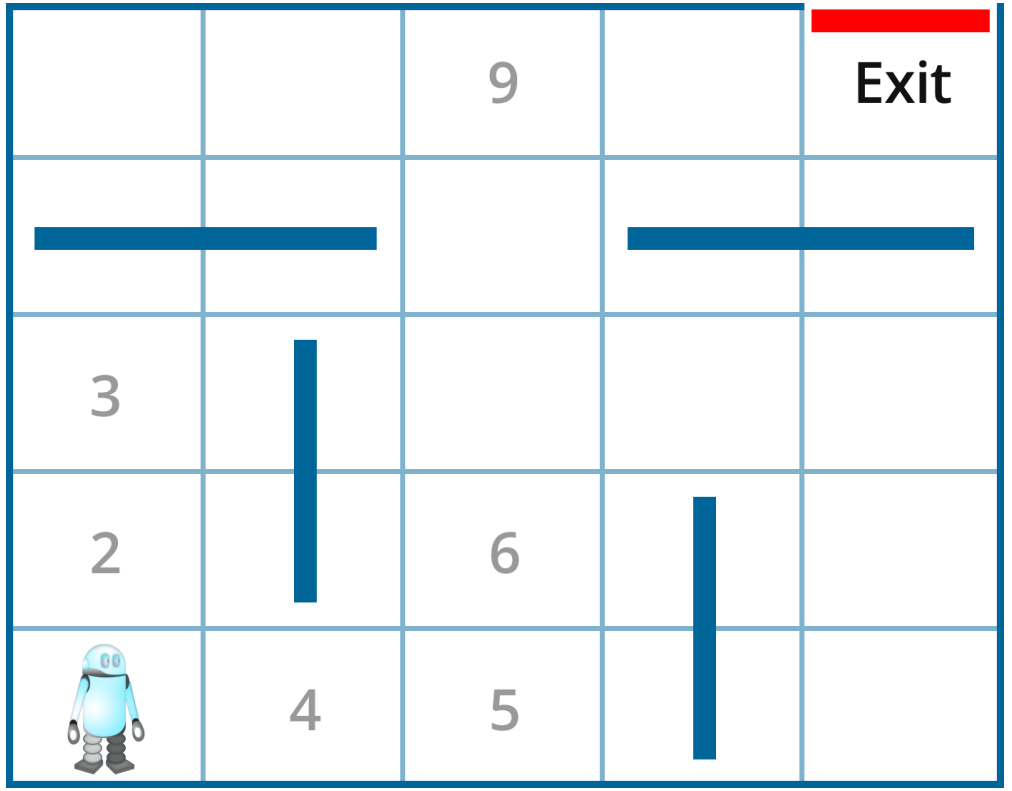
Exemplo: estados

- Numerar cada posição
- Posição inicial: 1



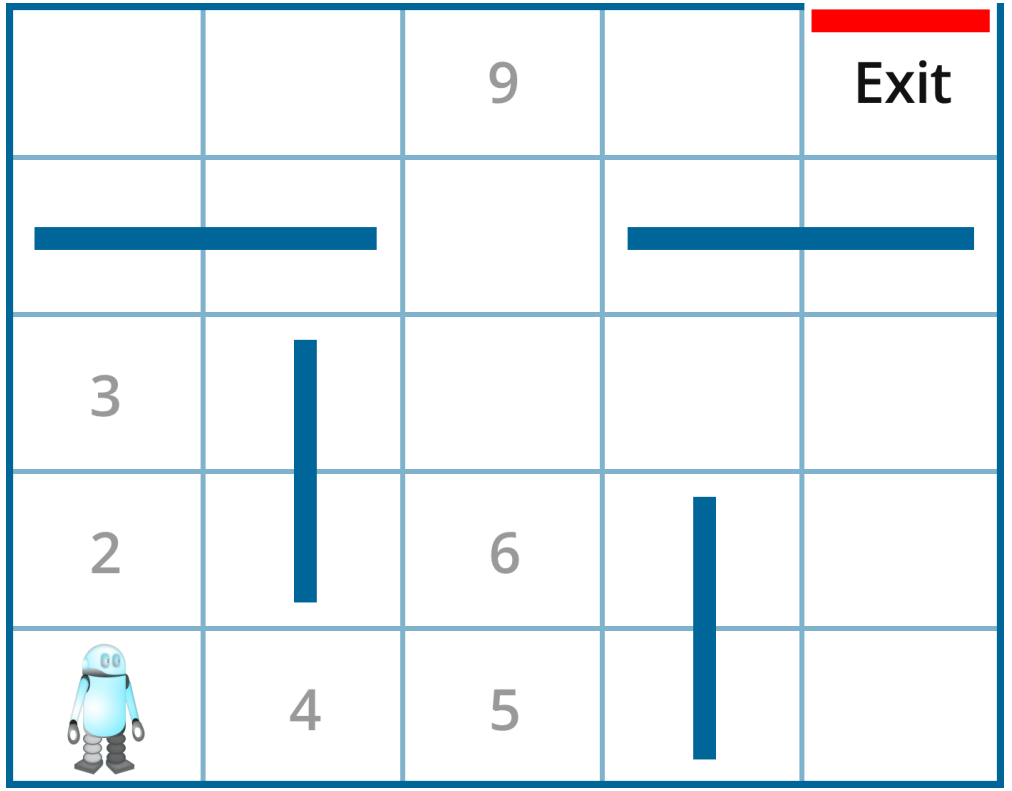
Exemplo: ações

- Cima, baixo, esquerda, direita
- Limitações *apenas* junto à fronteira
  - Por exemplo, a partir da posição 4 são possíveis ações cima, esquerda e direita



Exemplo: recompensas

- -1 quando bate num obstáculo
- -0.01 quando alcança posição vazia
- 100 quando chega à posição Exit



Aprendizagem por reforço: categorias

- Model-based
  - Agente usa modelo de transição do ambiente para ajudar a interpretar recompensas e tomar decisões em relação a ações
  - Modelo pode ser inicialmente desconhecido

- Model-free (exº Q learning)
  - Agente não conhece nem aprende modelo do ambiente
  - Modelo aprende como se comportar
    - Aprendizagem ação-utilidade (Q-function)
    - Procura de uma política que mapeia estados para ações

Aprendizagem por reforço: passiva e ativa

- Aprendizagem por reforço passiva
  - Política é fixa
  - Tarefa consiste em aprender utilidade dos estados (ou pares estado-ação)
- Aprendizagem ativa (exº Q-learning)
  - Agente precisa de explorar o ambiente
  - Aprende a política

Aprendizagem por reforço passiva

- Agente já tem uma política π (fixa) que determina as suas ações
- Agente pretende determinar função de utilidade $U^{\pi}(s)$
- Agente não conhece…
  - Modelo de transição P(s'|s,a), i.e. probabilidade de alcançar estado s' a partir de estado s depois de realizar ação a
  - Função de recompensa R(s,a,s')

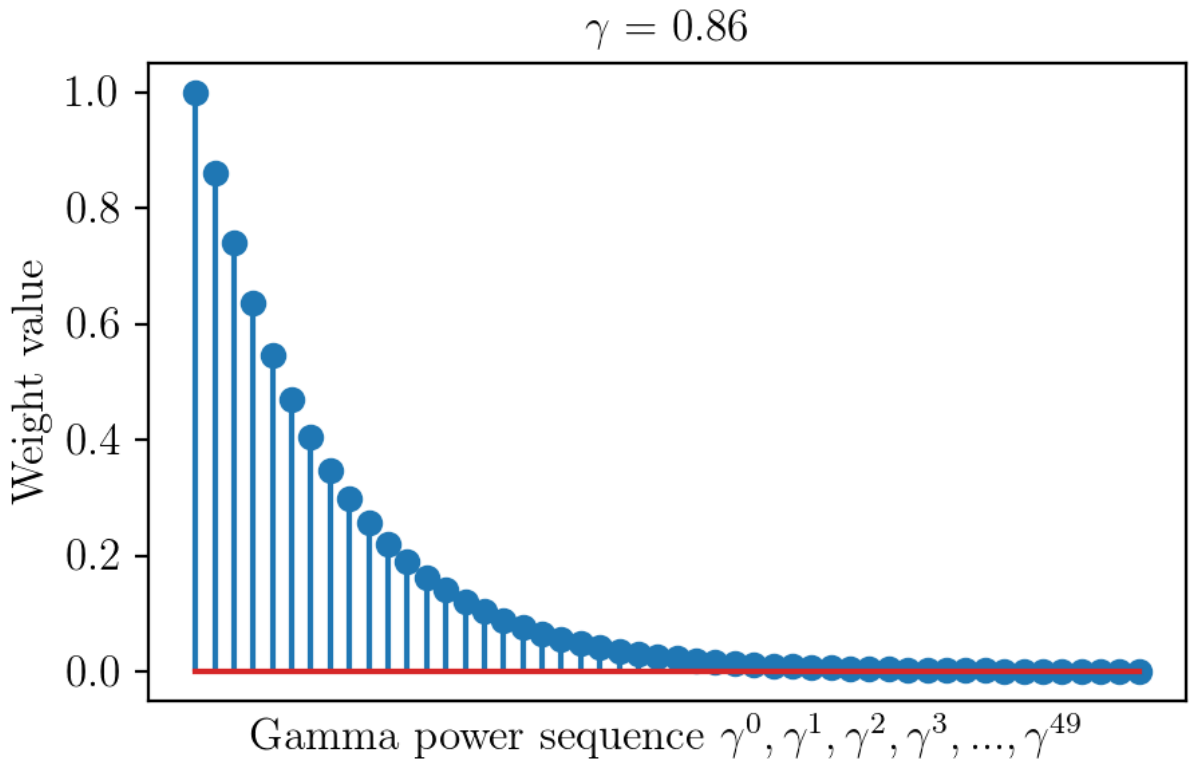
Aprendizagem Passiva

- A política do agente é fixa
- O objetivo do agente é avaliar a qualidade de uma política ótima
- O agente precisa de aprender a utilidade esperada $U^{\pi}(s)$ para cada estado s

$$U^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(S_t, \pi(S_t), S_{t+1})\right]$$

- $R(S_t, \pi(S_t), S_{t+1})$ *é a recompensa para um estado* S_t *dada a ação* $\pi(S_t)$ que alcança S_{t+1}, S_t *é um estado qualquer alcançado no instante t ao executar a política* π a partir do estado S_0=s, γ^t *é um fator de desconto gamma (entre 0 e 1)*

Fator de desconto: exemplo γ^t =0.86



γ^t

Valor de diminui ao longo do tempo

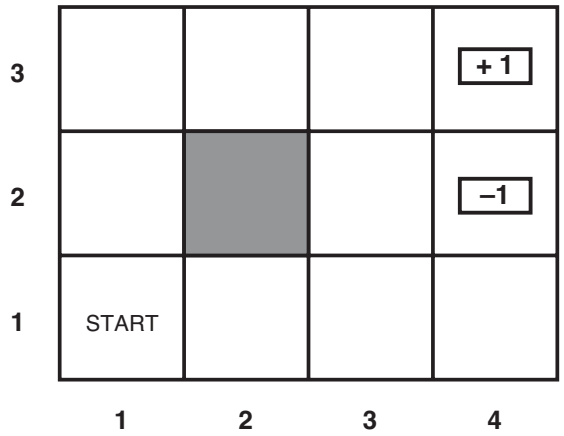
i.e. recompensa inicial é mais relevante

Como estimar a utilidade?

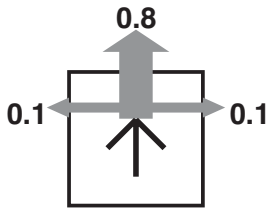
- **Utilidade** de um agente definida pela **função** de recompensa
- O agente executa uma **trajetória**, i.e. sequência de transições estado-ação até atingir um estado terminal
- A cada trajetória é associado um valor que é usado para estimar o valor de utilidade
- A desvantagem deste método é o facto de assumir que a utilidade de cada estado é independente, o que implica demorar a convergir

$$U^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^{t} R(S_t, \pi(S_t), S_{t+1})\right]$$

Exemplo: modelo de transição



(a)



(b)

Figure 17.1 (a) A simple 4×3 environment that presents the agent with a sequential decision problem. (b) Illustration of the transition model of the environment: the "intended" outcome occurs with probability 0.8, but with probability 0.2 the agent moves at right angles to the intended direction. A collision with a wall results in no movement. The two terminal states have reward +1 and –1, respectively, and all other states have a reward of –0.04.

Exemplo: política óptima

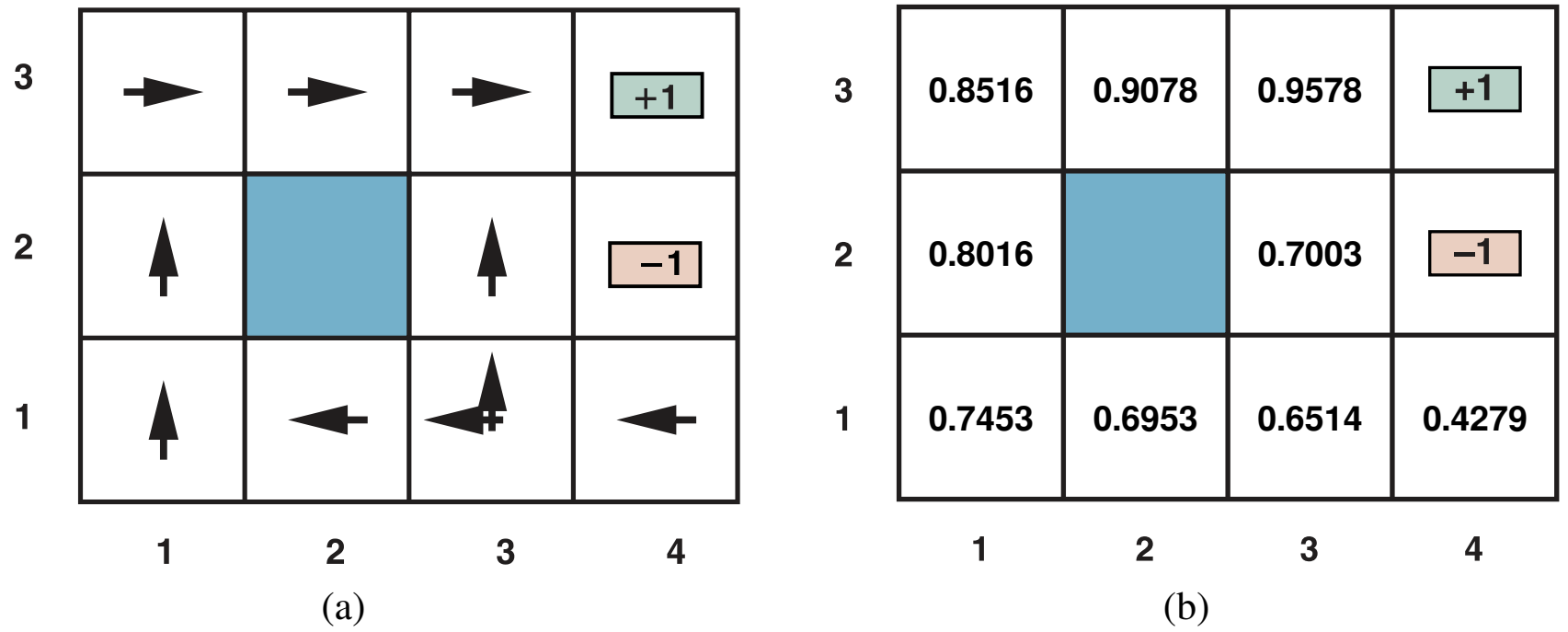
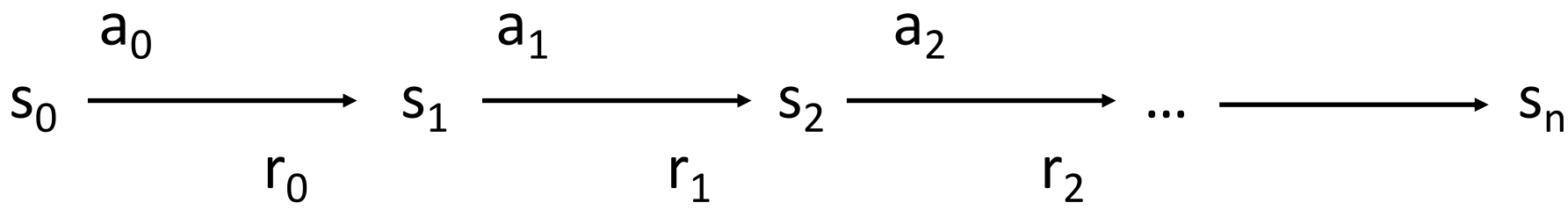


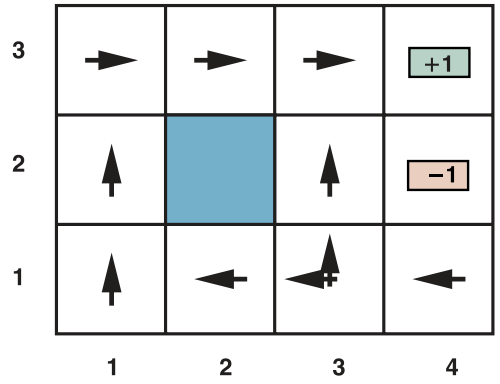
Figure 23.1 (a) The optimal policies for the stochastic environment with $R(s,a,s') = -0.04$ for transitions between nonterminal states. There are two policies because in state (3,1) both *Left* and *Up* are optimal. We saw this before in Figure 16.2. (b) The utilities of the states in the 4×3 world, given policy π.

Trajetoria: estado, ação, recompensa

Sequência do estado inicial s_0 ao estado objetivo s_n



Exemplos de três trajetórias



$$(1,1) \xrightarrow[Up]{\textbf{-.04}} (1,2) \xrightarrow[Up]{\textbf{-.04}} (1,3) \xrightarrow[Right]{\textbf{-.04}} (1,2) \xrightarrow[Up]{\textbf{-.04}} (1,3) \xrightarrow[Right]{\textbf{-.04}} (2,3) \xrightarrow[Right]{\textbf{-.04}} (3,3) \xrightarrow[Right]{\textbf{+1}} (4,3)$$

$$(1,1) \xrightarrow[Up]{\textbf{-.04}} (1,2) \xrightarrow[Up]{\textbf{-.04}} (1,3) \xrightarrow[Right]{\textbf{-.04}} (2,3) \xrightarrow[Right]{\textbf{-.04}} (3,3) \xrightarrow[Right]{\textbf{-.04}} (3,2) \xrightarrow[Up]{\textbf{-.04}} (3,3) \xrightarrow[Right]{\textbf{+1}} (4,3)$$

$$(1,1) \xrightarrow[Up]{\textbf{-.04}} (1,2) \xrightarrow[Up]{\textbf{-.04}} (1,3) \xrightarrow[Right]{\textbf{-.04}} (2,3) \xrightarrow[Right]{\textbf{-.04}} (3,3) \xrightarrow[Right]{\textbf{-.04}} (3,2) \xrightarrow[Up]{\textbf{-1}} (4,2)$$

Recompensa de uma trajetória

$$U^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(S_t, \pi(S_t), S_{t+1})\right]$$

- Assumir $\gamma = 1$
- Trajetória $(1,1) \overset{\textbf{-.04}}{\underset{Up}{\rightarrow}} (1,2) \overset{\textbf{-.04}}{\underset{Up}{\rightarrow}} (1,3) \overset{\textbf{-.04}}{\underset{Right}{\rightarrow}} (1,2) \overset{\textbf{-.04}}{\underset{Up}{\rightarrow}} (1,3) \overset{\textbf{-.04}}{\underset{Right}{\rightarrow}} (2,3) \overset{\textbf{-.04}}{\underset{Right}{\rightarrow}} (3,3) \overset{\textbf{+1}}{\underset{Right}{\rightarrow}} (4,3)$

- Recompensa de -.04*6+1=0.76 para estado (1,1)
- Recompensas de 0.80 e 0.88 para estado (1,2)
- Recompensas de 0.84 e 0.92 para estado (1,3)
- …
- Com número de trajetórias infinito, média converge para valores na Figura 23.1(b)

Aprendizagem por reforço ativa

- Agente passivo usa política para determinar ação
  - E determina função de utilidade
- Agente ativo decide que ações tomar
  - Aprende uma política

Função de recompensa V

- Função V(x) calcula a recompensa obtida após uma **trajetória** com T passos que começa no estado x em t=0

$$V(x) = \sum_{t=0}^{T-1} (\gamma^t r_t)$$

- Valor de gamma $0 \leqslant \gamma < 1$ é o fator de desconto
  - Permite abordagem *greedy*: determina a importância de recompensas futuras, valorizando recompensas com menor valor de t
- r_t é a recompensa recebida no instante t

Q-learning

- ***Q*-learning** é um algoritmo de aprendizagem por reforço para aprender a utilidade/qualidade das ações e consequentemente que ações deve tomar
  - "Q" refere-se a qualidade
- Não requer a existência de um modelo (é chamado "model-free") do ambiente
  - Pode lidar com problemas com transições estocásticas e recompensas, sem necessitar de adaptações
- Encontra uma política que maximiza o valor esperado da recompensa total para uma sequência de passos a partir do estado atual
  - Converge para política ótima com probabilidade 1

Função de recompensa Q

- Função Q : X × A → R

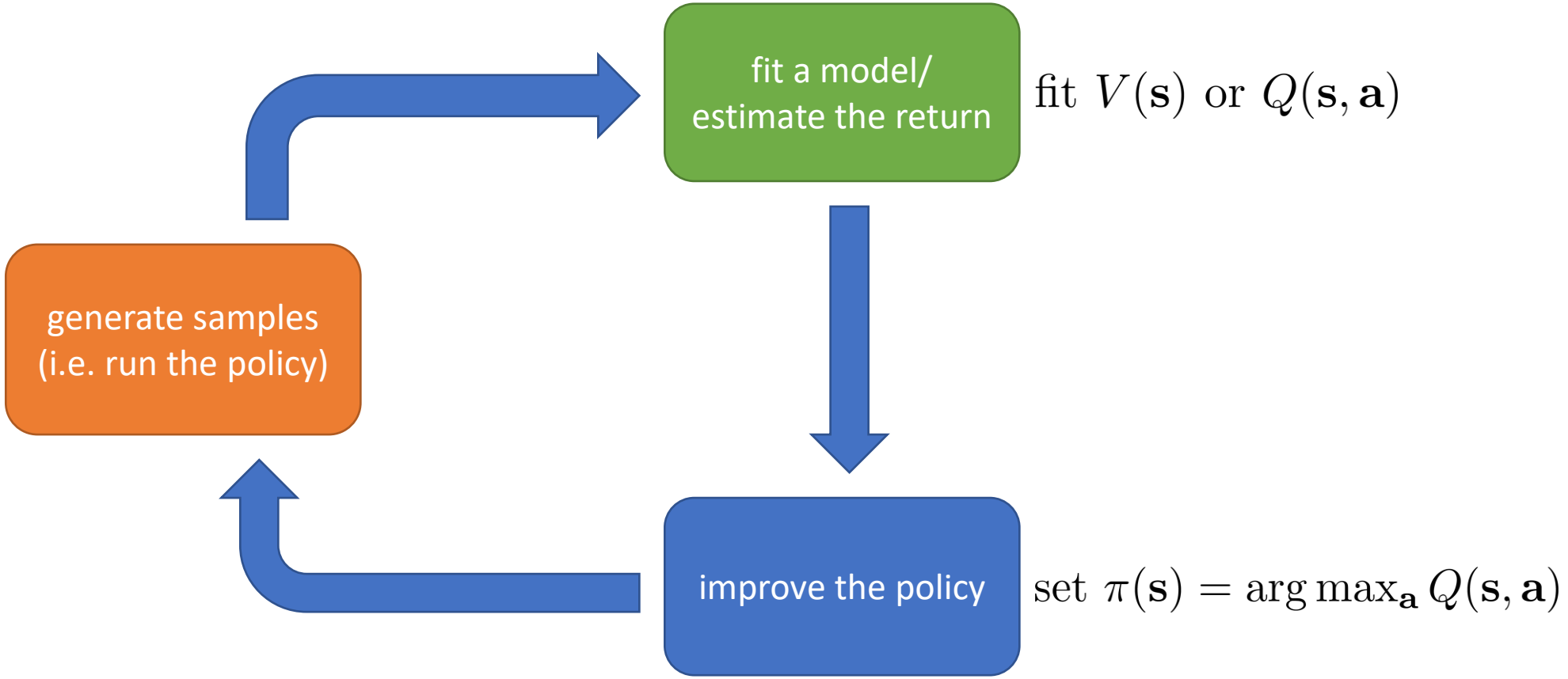
  calcula o mesmo que V mas para um par estado / ação

- Uma **política** π(a|x) : X × A → [0, 1] define a probabilidade de se executar cada ação a para cada estado x
- π* é a política que maximiza a recompensa total
- π* é obtida a partir de Q*

$$\pi^*(x) = argmax_{a \in A} Q^*(x, a)$$

Aprender função Q*

- Para encontrar a política ótima π^* para um dado ambiente, apenas precisamos de aprender a função Q*
- Utilizando Q-learning podemos aproximar a função $Q*$ a partir de **trajetórias de exemplo** no ambiente
- Para um agente num dado estado seguir a política ótima apenas tem que selecionar a ação que maximiza o valor de Q* para esse estado


fit a model/
estimate the return
fit $V(\mathbf{s})$ or $Q(\mathbf{s}, \mathbf{a})$
generate samples
(i.e. run the policy)
improve the policy
set $\pi(\mathbf{s}) = \arg\max_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a})$

Aprender função Q*

- Caraterização de um passo de uma trajetória (s_t, a_t, s_{t+1}, r_t):

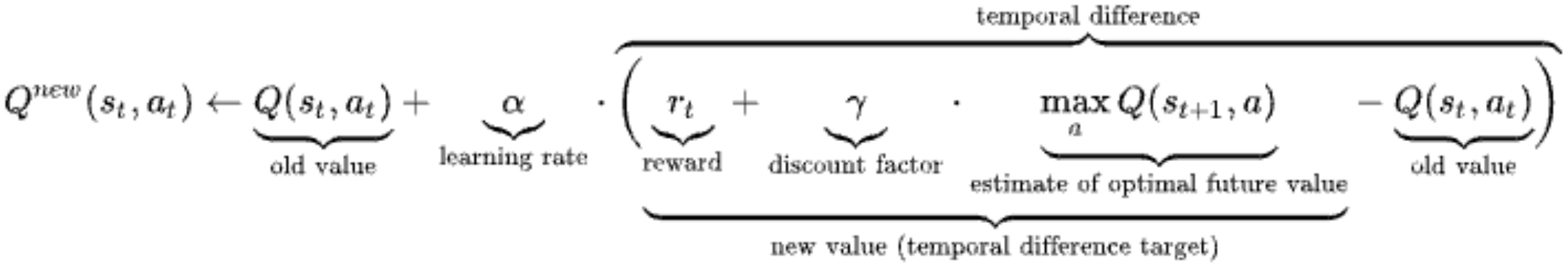
  s_t: estado antes realizar ação a
  a_t: ação utilizada
  s_{t+1}: estado após realizar ação a_t
  r_t: recompensa obtida por realizar ação a_t no estado s_t

- Regra de update ($0<\alpha\leqslant 1$ é a *learning rate)*

$$Q^{new}(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\Big(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}}\Big)}^{\text{temporal difference}}$$

$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{\text{new value (temporal difference target)}}$$

Regra de update



- $Q^{new}(s_t,a_t)$ resulta da **soma** de 3 parcelas
  - $(1-\alpha)\ Q(s_t,a_t)$: valor inicial pesado pela learning rate α; α próximo de 1 altera Q mais depressa
  - $\alpha\ r_t$: recompensa obtida se ação a_t é executada em s_t pesada por α
  - $\alpha\ \gamma\ \max_a Q(s_{t+1},a)$: máxima recompensa que pode ser obtida a partir do estado s_{t+1} pesada por α

Aprendizagem Q-learning

- Inicializar matriz Q a zeros
- Começar a explorar ações: Para cada estado, selecionar uma ação possível (a_t) para esse estado (s_t)
- Transitar para o próximo estado (s_{t+1}) como resultado da ação (a_t)
- Para todas as possíveis ações *a* a partir do estado s_{t+1} selecionar a ação que tem o maior valor de Q
- Atualizar os valores da matriz Q usando a regra de update
- Próximo estado s_{t+1} passa a estado atual
- Se o estado objetivo é alcançado, terminar e repetir o processo

Matriz Q-Learning: exemplo

Initialized

Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	327	0	0	0	0	0	0
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	499	0	0	0	0	0	0

Training

Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	328	-2.30108105	-1.97092096	-2.30357004	-2.20591839	-10.3607344	-8.5583017
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	499	9.96984239	4.02706992	12.96022777	29	3.32877873	3.38230603

Reinforcement learning solves a particular kind of problem where decision making is sequential, and the goal is long-term, such as game playing, robotics, resource management, or logistics.

Andriy Burkov, The Hundred Page Machine Learning Book