Aprendizado por Reforço

Carlos H. C. Ribeiro
Professor Adjunto
Divisão de Ciência da Computação
Instituto Tecnológico de Aeronáutica

CTC15

Como programar um "jogador automático" para enfrentar um adversário que às vezes é imperfeito?

Opção 1: MINIMAX (Teoria dos Jogos)

- Idéia: escrever um programa que maximize a expectativa de vitória do jogador, considerando que o adversário jogará de modo a minimizar esta expectativa (expansão da árvore de estados).
- Mas o nosso adversário não joga **sempre** de modo a minimizar a expectativa... MINIMAX não se aplica.

Opção 2: Usar um modelo probabilístico do adversário

- Idéia: escrever um programa que maximize uma expectativa de vitória do jogador, considerando que o adversário jogará de acordo com um modelo.
- Muito bom. Mas primeiro preciso gerar o modelo do nosso adversário...

Opção 3: Usar **Aprendizado por Reforço**

- Primeiro defino uma tabela de números, endereçada pelos possíveis estados do problema.
- Cada número da tabela é uma medida da probabilidade de se vencer o jogo, a partir do estado correspondente. Chamemos esta medida de valor do estado. Inicialmente, carrego a tabela com valores aleatórios, porque a princípio não sei quão bom ou quão ruim um estado é (não desenvolvi nenhum modelo).

Opção 3: Usar **Aprendizado por Reforço**

- Jogo várias vezes contra o adversário. Propago o resultado final de cada jogo aos estados que levaram a isto, modificando seus respectivos valores.
- O resultado final é propagado porque sei o valor esperado associado aos estados finais. Por exemplo, qualquer estado com três X alinhados tem valor 1, e qualquer estado com três O alinhados tem valor 0.

Algumas Idéias Fundamentais

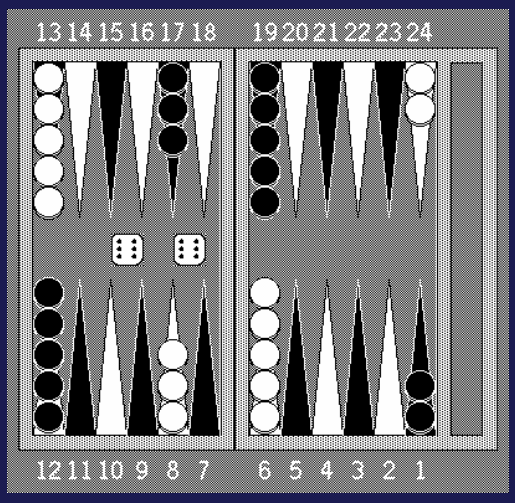
- Aprendizado por experimentação
- "O mundo é o melhor modelo de si mesmo"
- AR é caracterizado por problemas que envolvem o conceito de Autonomia
- AR liga conceitos de IA e Controle Ótimo
- AR é aplicável a problemas reais

Bibliografia Básica: livros

- ***Reinforcement Learning: An Introduction***
  A. Barto / R. Sutton, MIT Press, 1998
  ☺ Didático, princípios bem explicados, bons exercícios
  ☹ Discussão insuficiente de alguns tópicos importantes
- ***Neurodynamic Programming***
  D. Bertsekas / J. Tsitsiklis, Athena, 1996
  ☺ Razoavelmente completo e bem formalizado
  ☹ Um pouco confuso e pesado para uma primeira leitura

TDGammon (Tesauro, 1992)

- Jogo estocástico
- Árvore de estados de alta ramificação: da ordem de 10^{20} estados !!!



TD-Gammon (Tesauro, 1992)

- Aprendizado por Reforço baseado em múltiplos jogos consigo mesmo (*self-play*)
- Tão bom quanto melhores jogadores humanos !!!

Isto é um pouco mais impressionante do que um jogador automático pro Jogo da Velha...
Naturalmente, TD-Gammon chamou muito a atenção para o uso de técnicas de AR em problemas realistas...

1. Caracterizando o problema

- Autonomia: o que é, para que serve
- O modelo básico

2. Uma metodologia para se estudar AR

- Processos Decisórios de Markov
- Programação Dinâmica
- Força Bruta: O Método Monte Carlo
- O Método das Diferenças Temporais
  - O conceito de *bootstrapping*
  - Formulação não-causal
  - Formulação causal

3. AR = Controle Ótimo + Autonomia

- Predição ok... E o Controle?
- Controle Autônomo: Q-learning

1. Caracterizando o problema

Autonomia *(Russel & Norvig, 1997)*

Agente = "Algo" que observa e atua em um ambiente

Autonomia = Capacidade de um agente escolher ações com base **na própria** experiência

Experiência = Interação com o ambiente

Um animal **é** autônomo (mas nenhum agente é *tabula rasa*...)

Uma R.N.A. treinada com supervisão **não é** autônoma

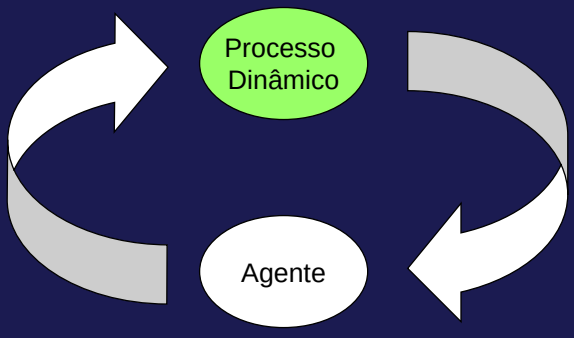
Um programa de inferência típico **não é** autônomo

Por que estudar agentes autônomos?

- Em Engenharia: modelos mais simples e precisos - agente desenvolve modelo necessário e suficiente para a tarefa em questão.
- Em IA: enfoque alternativo com ênfase na atividade situada do agente como fator de aprendizado (IA *nouvelle* - Brooks, Maes, Mataric, etc..).
- Em ciência cognitiva: formalização de conceitos *behavioristas*

2. Uma metodologia para se estudar AR

Agente ⇔ Processo Dinâmico



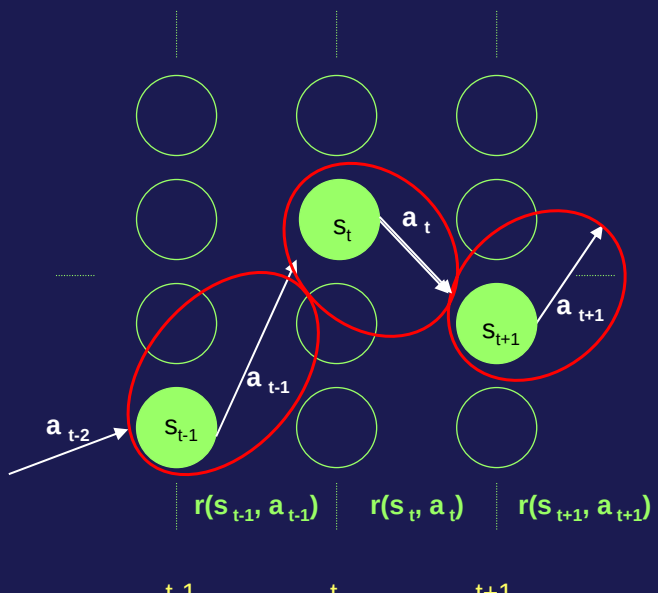
Como estudar agentes autônomos?

A teoria: processos de decisão de Markov (MDP's)

- Estados **s**
- Ações **a**
- Reforços **r(s,a)**
- Transições entre estados
- **Condição de Markov:** Estado atual depende apenas de últimos estado e ação (e possivelmente de algum parâmetro aleatório independente)

O critério: uma função de valor (ou custo) *V*

O objetivo: encontrar política de ações que maximize (minimize) o valor esperado do valor (custo)



Exemplo 1: Controle de Inventário

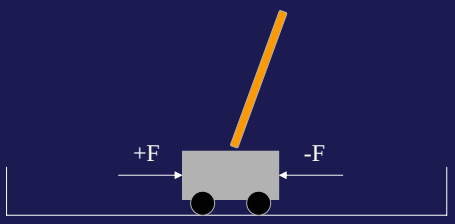
- **Problema: comprar uma quantidade de um certo produto a intervalos regulares (em um total de N intervalos) de modo a satisfazer uma certa demanda**
- Estado: s_k = estoque no começo do período k
- Ação: a_k = compra feita no começo do período k
- Uma perturbação aleatória w_k = demanda no período k, respeitando uma certa distribuição de probabilidade
- Reforço $r_k = r(s_k) + ca_k$, onde $r(s_k)$ é o custo de estocar s_k unidades do produto no período k e c é o custo unitário do produto comprado.

Exemplo 1: Controle de Inventário

- Evolução do estado: $s_{k+1} = s_k + a_k - w_k$
- Função de custo a ser minimizada:

$$V(s_o) = \mathrm{E}\left\{ r(s_N) + \sum_{k=0}^{N-1} \left(r(s_k) + ca_k \right) \right\}$$

Exemplo 2: Pêndulo Invertido



- **Problema: controlar um pêndulo invertido exercendo forças +F ou -F sobre a base do carrinho (controle *bang-bang*). "Controlar" significa não permitir que a barra caia ou que o carrinho choque-se com as paredes.**

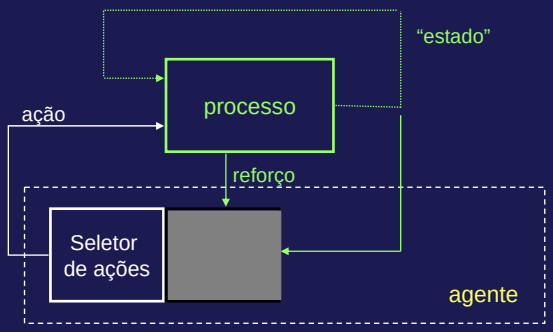
Exemplo 2: Pêndulo Invertido

- Estado: quádrupla $\left(x_t, \dot{x}_t, \theta_t, \dot{\theta}_t\right)$
- Ação: +F ou -F
- Reforço: -1 em caso de falha, senão 0.
- Evolução do estado: $s_{k+1} = f(s_k, a_k)$ (?)
- Possível função de custo a ser minimizada:

$$V(s_o) = \mathrm{E}\left\{\sum_{t=0}^{\infty} \gamma^k r_t\right\}$$

desconto temporal $\gamma < 1$: POR QUÊ?

Modelo Básico Agente ⇔ Processo



SE EU TENHO AS PROBABILIDADES DE TRANSIÇÃO (ou seja, um modelo do processo)

PROBLEMA CLÁSSICO DE CONTROLE ÓTIMO (solução baseada em programação dinâmica)

Controle Ótimo

Idéia fundamental:

- Defino a função de valor $V(s_o) = \sum_{k=0}^{\infty} P(s_k \mid s_{k-1}, a_k) \gamma^k r(s_k)$
- Obtenho uma política de ações que maximize a função de valor

$$V^*(s_o) = \sum_{k=0}^{\infty} P(s_k \mid s_{k-1}, a_k^*) \gamma^k r(s_k) = \max_a \left[\sum_{k=0}^{\infty} P(s_k \mid s_{k-1}, a_k^*) \gamma^k r(s_k) \right]$$

$\mu^* = \{a_0^*, a_1^*, \cdots\}$: política ótima de ações

desconto temporal γ < 1: necessário para problemas sem terminação garantida ...

Controle Ótimo: Operadores Fundamentais

- Operador de Aproximações Sucessivas

$$(T_\mu V)(s) = r(s, \mu(s)) + \gamma \sum_{s' \in S} P(s' \mid s, \mu(s)) V(s')$$

- Operador de Iteração de Valores

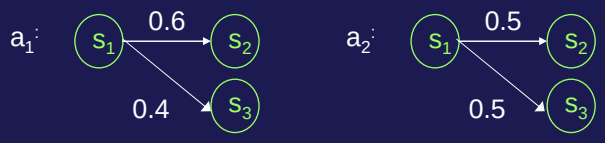
$$(TV)(s) = \max_a \left[r(s,a) + \gamma \sum_{s' \in S} P(s' \mid s, a) V(s') \right]$$

Operadores Fundamentais: Exemplo

Estados s_1, s_2 , desconto $\gamma = 1$ Ações a_1, a_2

Reforços $r(s_1, a_1) = 2$, $r(s_1, a_2) = 1$ Valores $V(s_2) = 1$, $V(s_3) = 2$

políticas μ_1={$\mu_1(s_1)=a_1$, $\mu_1(s_2)$=...}e μ_2={$\mu_2(s_1)=a_2$, $\mu_2(s_2)$=...}



$$\left(T_{\mu_1} V\right)(s_1) = r(s_1, \mu_1(s_1)) + \gamma \sum_{s' \in S} P(s' \mid s, \mu_1(s_1)) V(s') =$$

$$= 2 + 1 \times (0.6 \times 1 + 0.4 \times 2) = 3.4$$

$$\left(T_{\mu_2} V\right)(s_1) = r(s_1, \mu_2(s_1)) + \gamma \sum_{s' \in S} P(s' \mid s, \mu_2(s_1)) V(s') =$$

$$= 1 + 1 \times (0.5 \times 1 + 0.5 \times 2) = 2.5$$

$$\left(TV\right)(s_1) = 3.4$$

Controle Ótimo: Propriedades dos Operadores

- Teorema 1 $V^*(s) = \lim_{N\to\infty} (\mathrm{T}^N V)(s)$
- Corolário 1 $V_\pi(s) = \lim_{N\to\infty} (\mathrm{T}_\pi^N V)(s)$
- Teorema 2 $V^* = \mathrm{T}V^*$ (Equação de Bellman)
  ou seja, V^* é o (único) ponto fixo do operador T
- Corolário 2 $V_\pi = \mathrm{T}_\pi V_\pi$
  ou seja, V_π é o (único) ponto fixo do operador T_π
- Teorema 3 Uma política μ é ótima se e somente se $\mathrm{T}_\pi V^* = \mathrm{T}V^*$

Controle Ótimo: Algoritmos de Programação Dinâmica

- Iteração de Valores
  $$\mu^*(s) = \arg\left[\lim_{N\to\infty} (\mathrm{T}^N V)(s)\right]$$
- Iteração da Política
  - passo 1: avaliar a política atual μ_k
    $$V_{\mu_k} = \mathrm{T}_{\mu_k} V_{\mu_k} \quad \text{ou} \quad V_{\mu_k} \approx \mathrm{T}_{\mu_k}^M V_{\mu_k}$$
  - passo 2: obter uma política melhorada μ_{k+1}
    $$\mu_{k+1}(s) = \arg\left[(\mathrm{T}V_{\mu_k})(s)\right]$$

Autonomia

Modelo do processo não disponível

Probs. de transição desconhecidas

Aprendizado por Reforço = Controle Ótimo + Autonomia

Como aprender autonomamente?

Por enquanto, vamos nos concentrar na estimação do custo *V* para uma política **fixa** de ações (**esqueça** o controle) ...

Solução: tentar minimizar iterativamente (Robbins-Monro) uma **estimativa** da função de custo

ciclo:

- observo novo estado s_t
- defino experiência $\langle s_t, a_t, s_{t+1}, r_t \rangle$
- atualizo estimação de custos (ou espero um pouco...)
- escolho ação $a^{[t+1]}$ em função das novas estimativas

Algoritmo Robbins-Monro

$$r_{t+1} = r_t + \alpha_t \left(g(r_t, \tilde{v}_t) - r_t \right)$$

$\tilde{v}_t$: amostrado de acordo com $p(v \mid r)$

α_t : coef. de aprendizado

r_t converge (com prob. 1) para $\mathrm{E}[g(r_t, \tilde{v})]$ se :

$$\sum_{t=0}^{\infty} \alpha_t = \infty \text{ e } \sum_{t=0}^{\infty} \alpha_t^2 < \infty$$

3 Propostas

- Monte Carlo: uso experiências completas para estimar custos
- TD (0): uso experiência imediata para estimar custos

Proposta 1: Monte Carlo

Idéia: força bruta

Forma geral da iteração:

$$\tilde{V}_{t+N} = \tilde{V}_t + \alpha\left(r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^N r_{t+N} - \tilde{V}_t\right)$$

Um dado estado pode aparecer **mais de um vez** ao longo de uma trajetória

Método "primeira visita": só atualizo o custo para a parte da trajetória relativa à primeira visita.

Método "toda visita": atualizo o custo para toda parte de trajetória relativa à qualquer visita.

☹ **Problema: atualização requer trajetória completa**

Proposta 2: TD(0)

Idéia: Basear aprendizado em **minimização do erro de predição:**

$$\tilde{V}_{t+1}(s_t) = \tilde{V}_t(s_t) + \alpha\left(r(s_t) + \gamma\tilde{V}_t(s_{t+1}) - \tilde{V}_t(s_t)\right)$$

$\tilde{V}_t$: predição no instante t

Um dado estado pode aparecer **mais de um vez** ao longo de uma trajetória

Método "primeira visita": só atualizo o custo para a parte da trajetória relativa à primeira visita.

Método "toda visita": atualizo o custo para toda parte de trajetória relativa à qualquer visita.

TD(0) X Monte Carlo

Random Walk :

Desconto $\gamma = 1$, P(dir)=P(esq)=0.5, α=0.8

$$V(\mathrm{A}) = \frac{1}{6}, V(\mathrm{B}) = \frac{2}{6}, V(\mathrm{C}) = \frac{3}{6}, V(\mathrm{D}) = \frac{4}{6}, V(\mathrm{E}) = \frac{5}{6}$$

Monte Carlo, “primeira visita”

$$D \Rightarrow E \Rightarrow \text{fim}: \tilde{V}_1(D) = 0 + .8(0 + 1 - 0) = .8$$

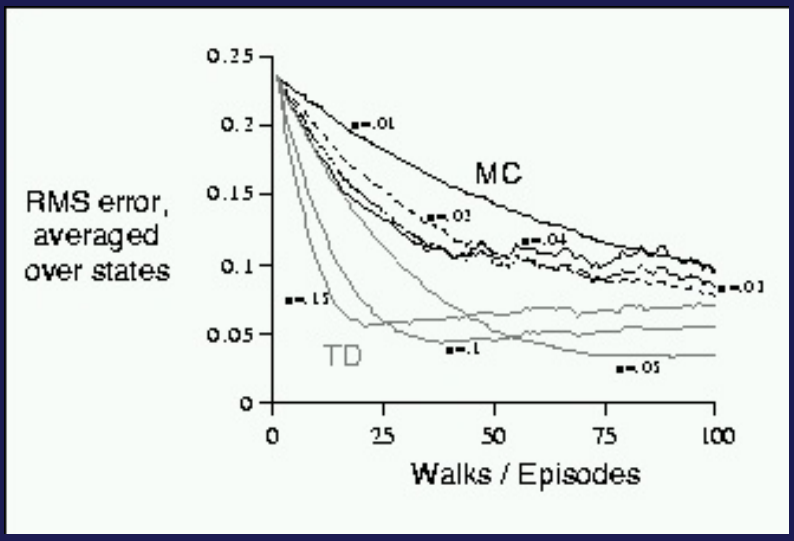
$$D \Rightarrow C \Rightarrow D \Rightarrow E \Rightarrow \text{fim}: \tilde{V}_1(D) = .8 + .8(0 + 0 + 0 + 1 - .8) = .96$$

$$D \Rightarrow C \Rightarrow B \Rightarrow A \Rightarrow \text{fim}$$
$$\tilde{V}_1(D) = .96 + .8(0 + 0 + 0 + 0 - .96) = .192$$

TD(0), “toda visita”

$$D \Rightarrow E \Rightarrow \text{fim}:$$
$$\tilde{V}_1(D) = 0 + .8(0 + 0 - 0) = 0, \quad \tilde{V}_1(E) = 0 + .8(1 + 0 - 0) = .8$$
$$D \Rightarrow C \Rightarrow D \Rightarrow E \Rightarrow \text{fim}:$$
$$\tilde{V}_1(D) = 0 + .8(0 + 0 - 0) = 0, \quad \tilde{V}_1(C) = 0 + .8(0 + 0 - 0) = 0,$$
$$\tilde{V}_1(D) = 0 + .8(0 + .8 - 0) = .64, \quad \tilde{V}_1(E) = .8 + .8(1 + 0 - .8) = .96$$
$$D \Rightarrow C \Rightarrow B \Rightarrow A \Rightarrow \text{fim}:$$
$$\tilde{V}_1(D) = .64 + .8(0 + 0 - .64) = .128, \quad \tilde{V}_1(C) = 0 + .8(0 + 0 - 0) = 0,$$
$$\tilde{V}_1(B) = 0 + .8(0 + 0 - 0) = 0, \quad \tilde{V}_1(A) = 0 + .8(0 + 0 - 0) = 0$$

TD(0) X Monte Carlo



Variantes

- ***Online:*** atualizações feitas assim que $R_t^{(n)}$ é computado.
- ***Offline:*** atualizações "guardadas" e feitas apenas ao final do episódio. Valores V não são modificados durante a execução de uma trajetória.

Aproximando PD para Controle

TD(λ) + Operador de Iteração de Valor + "sem modelo"

Método de Separação de Política "Autônomo"

Tem um operador "min" atrapalhando...

Tentemos então obter **diretamente** o controle usando um método baseado no operador de aproximações sucessivas...

Valores Q

$$Q(s_t,a_t) \stackrel{\Delta}{=} r(s_t,a_t) + \gamma \sum_{s_{t+1}} P(s_{t+1} \mid s_t,a_t) V^*(s_{t+1})$$

$$V^*(s_t) = TV^*(s_t) = \min_a Q(s_t,a_t)$$

$$\therefore Q(s_t,a_t) = r(s_t,a_t) + \gamma \sum_{s_{t+1}} P(s_{t+1} \mid s_t,a_t) \max_a Q(s_t,a_t) = (T^Q Q)(s_t,a_t)$$

onde

$$(T^Q Q)(s,a) \stackrel{\Delta}{=} r(s,a) + \gamma \sum_{s'} P(s' \mid s,a) \min_u Q(s',u)$$

Valores Q

Qual é a diferença **fundamental** entre

$$(T^{Q}Q)(s,a) \stackrel{\Delta}{=} r(s,a) + \gamma \sum_{s'} P(s' \mid s,a) \max_{u} Q(s',u)$$

e

$$(TV)(s) \stackrel{\Delta}{=} \max_{u} \left[r(s,u) + \gamma \sum_{s'} P(s' \mid s,u) V(s') \right]$$

☺ **"min" agora aparece dentro do somátorio... Posso definir um algoritmo do tipo Robbins-Monro para aproximar T^Q !!!**

Q-Learning

Watkins, 1989

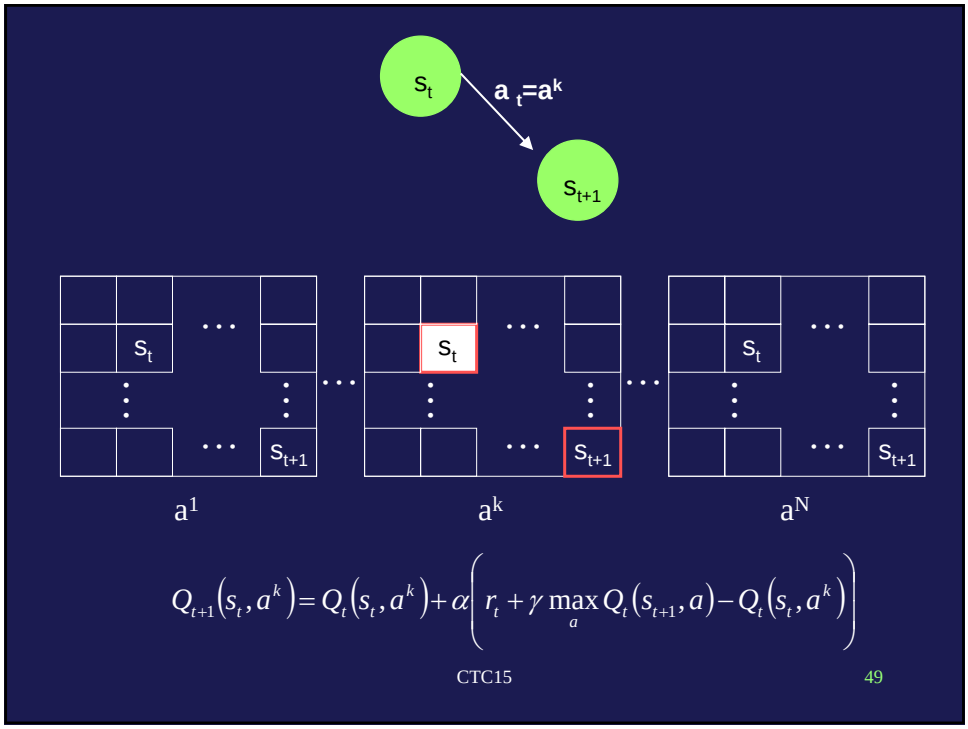
- No tempo *t, o agente*:
  - observa estado s_t e seleciona ação a_t;
- No tempo *t+1, o agente*:
  - observa estado s_{t+1} ;
  - atualiza o **valor de ação** $Q_t(s_t, a_t)$ de acordo com

$$\Delta Q_t(s_t, a_t) = \alpha_t [r_t + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t)]$$

☺ **Aproxima o operador T^Q (iterações Robbins-Monro)**

☺ **Convergência garantida (para representação tabular)**

☺ **Ações de treino podem ser escolhidas livremente**


s_t
$a_t=a^k$
s_{t+1}
s_t
s_{t+1}
s_t
s_{t+1}
s_t
s_{t+1}
a^1
a^k
a^N
$Q_{t+1}(s_t,a^k)=Q_t(s_t,a^k)+\alpha\left(r_t+\gamma\max_a Q_t(s_{t+1},a)-Q_t(s_t,a^k)\right)$