Algoritmos para MDP
Monte Carlo Tree Search

Valdinei Freire

(EACH - USP)

Algoritmos para MDP

- VI e PI: processam todos os estados com operador de Bellman

- LAO* e LRTDP: processam apenas estados alcançáveis, mas processam operador de Bellman

- MCTS: processam apenas uma amostra dos estados alcançáveis, não processam o operador de Bellman

MDP Um processo markoviano de decisão (*Markovian Decision Process* – MDP) é definido por uma tupla $\langle \mathcal{S}, \mathcal{A}, T, R \rangle$ onde:

- $s \in \mathcal{S}$ são estados possíveis;
- $a \in \mathcal{A}$ são ações possíveis;
- $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ é a função de transição; e
- $R : \mathcal{S} \times \mathcal{A} \to \Re$ é a função recompensa.

Solução Ótima para MDPs

Solução: política $\pi : \mathcal{S} \rightarrow \mathcal{A}$

Problema de Otimização: $\pi(s) = \arg\max_{\pi \in \Pi} V^{\pi}(s)$

$$V^{\pi}(s) = \mathsf{E}\left[\sum_{t=0}^{\infty} \gamma^{t} r_{t} | s_{0} = s, \pi\right]$$

Programação Dinâmica:

$$V^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} T(s, a, s') V^{\pi}(s')$$

$$V^{\pi^{*}}(s) = V^{*}(s) = \max_{a \in \mathcal{A}} \left\{ R(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} T(s, a, s') V^{*}(s') \right\}$$

Algoritmos PI e VI

Solução Exata a menos de um erro ϵ.

Em cada iteração, todos os estados são atualizados.

Cada execução do operador $\mathcal{T}$ leva $|\mathcal{S}|^2|\mathcal{A}|$.

Cada execução do operador $\mathcal{T}^\pi$ leva $|\mathcal{S}|^2$.

Algoritmos LAO* LRTDP

Se um estado inicial s_0 é considerado nem todos são alcançáveis

- LAO* atualiza apenas estados alcançáveis utilizando a política ótima
  - Pior caso: atualiza todos estados alcançáveis a partir de s_0
  - Melhor caso: atualiza apenas estados alcançáveis utilizando a política ótima
- LRTDP atualiza apenas estados alcançáveis via simulação (estados mais prováveis) e estados improváveis são resolvidos apenas quando necessário.
- LAO* e LRTDP garantem solução exata a menos de um erro ϵ.

Soluções Aproximadas e On-Line

- Ramificação do estado s: $\{s' \in \mathcal{S} | \exists a \in \mathcal{A}\ T(s, a, s') > 0\}$.

- A quantidade de estados alcançáveis é exponencial na ramificação.

- Se a ramificação de um estado é muito grande, a quantidade de estados alcançáveis pode ser intratável.

- A aplicação do Operador de Bellman pode ser intratável.

Conclusão: não é possível encontrar um política parcial fechada nos estados alcançáveis.

Monte Carlo Tree Search

1. Repete
   (a) Percebe o estado s
   (b) Planeja enquanto há tempo e escolhe uma ação a
   (c) Executa ação a escolhida e transita para o estado s'

Um modelo generativo para um MDP é um algoritmo aleatório que, recebe como entrada um par estado-ação (s, a) e retorna uma recompensa r e um estado s', onde o estado s' foi sorteado aleatoriamente de acordo com as probabilidades de transição $T(s, a, \cdot)$ e a recompensa r foi sorteada (usualmente, determinista) da função de recompensa $R(s, a)$.

Métodos de Monte Carlo

Simulação de Variável Aleatória

- consideram amostragem de alguma variável aleatória X
- estima-se propriedades dessa variável aleatória

Estimando $V^{\pi}(s_0)$

- amostra-se N históricos a partir de s_0 até um horizonte H
- calcule o somatório descontado de cada histórico
- calcule a média aritmética

Estima $V^\pi(s_0)$ com N amostras e horizonte H

1. para n entre 1 e N faça
   (a) $V_n \leftarrow 0$
   (b) $s \leftarrow s_0$
   (c) para t entre 0 e $H - 1$ faça
      i. $a_t \leftarrow \pi(s_t)$
      ii. $r_t \leftarrow R(s_t, a_t)$
      iii. $s_{t+1} \sim T(s_t, a_t, \cdot)$
      iv. $V_n \leftarrow V_n + \gamma^t r_t$
2. retorna $\widehat{V}^\pi(s_0) = \dfrac{1}{N}\displaystyle\sum_{n=1}^{N} V_n$

Métodos de Monte Carlo: estatísticas

Note que:

- $V^{\pi}(s_0)$ é uma esperança
- $\widehat{V}^{\pi}(s_0)$ é uma variável aleatória

Perguntas sobre $\widehat{V}^{\pi}(s_0)$:

- Qual é o erro máximo gerado com probabilidade $(1-\delta)$?
- Qual é a probabilidade que o erro não seja menor que ϵ?
- qual horizonte H devo escolher e quantas simulações N devo fazer para garantir um erro máximo ϵ com probabilidade $(1-\delta)$?

Limites e Probabilidades

Theorem 1 (Hoeffding's Inequality)**.** Seja $X_1, \dots, X_N$ variáveis aleatórias independentes e identicamente distribuídas tal que $\mathsf{E}[X_i] = \mu$, $X_i \in [a, b]$ e $\overline{X}_N = \dfrac{1}{N}\displaystyle\sum_{i=1}^{N} X_i$. Então, para qualquer $\epsilon > 0$,

$$\Pr(|\overline{X}_N - \mu| \geqslant \epsilon) \leqslant 2e^{-\frac{2N\epsilon^2}{(b-a)^2}},$$

e com probabilidade pelo menos $1 - \delta$ tem-se:

$$\Pr\left(|\overline{X}_N - \mu| \leqslant \sqrt{\frac{(b-a)^2}{2N} \log \frac{2}{\delta}}\right) \geqslant 1 - \delta.$$

Limites e Probabilidades em MDPs

Considere a função valor com horizonte finito H:

$$V^{\pi,H}(s) = \mathsf{E}\left[\sum_{t=0}^{H-1} \gamma^t r_t | s_0 = s, \pi\right]$$

Considere:

- $R(s,a) \geqslant 0$ para todo $s \in \mathcal{S}$ e $a \in \mathcal{A}$
- $R_{max} = \max_{s \in \mathcal{S}, a \in \mathcal{A}} R(s,a)$
- $V_{max} = \dfrac{R_{max}}{1-\gamma}$

Então: $|V^{\pi,H}(s) - V^{\pi}(s)| \leqslant \gamma^H V_{max}$

Limites e Probabilidades em MDPs

Temos que:

$$
\begin{aligned}
V^{\pi}(s) &= \mathsf{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, \pi\right] \\
&= V^{\pi,H}(s) + \mathsf{E}\left[\sum_{t=H}^{\infty} \gamma^t r_t | s_0 = s, \pi\right] \\
&\leqslant V^{\pi,H}(s) + \mathsf{E}\left[\sum_{t=H}^{\infty} \gamma^t R_{max} | s_0 = s, \pi\right] \\
&= V^{\pi,H}(s) + \frac{\gamma^H}{1-\gamma} R_{max} = V^{\pi,H}(s) + \gamma^H V_{max}
\end{aligned}
$$

Theorem 2. Em um MDP, considere a variável aleatória $V^{\pi}_{T,s_0,N}$ obtida ao avaliar o valor $V^{\pi}(s_0)$ utilizando o método de Monte Carlo com horizonte H e N amostras. Então:

- A probabilidade de o erro ser maior que ϵ tem limite superior:

$$\delta = \frac{2}{\log \frac{2N\epsilon^2}{V_{max}^2}};$$

- Com probabilidade $1 - \delta$ o erro tem limite superior:

$$\epsilon = V_{max}\sqrt{\frac{1}{2N} \log \frac{2}{\delta}};$$

- Para garantir que o erro seja no máximo de ϵ com probabilidade menor que δ, a quantidade máxima de amostras necessária é de:

$$N = \left(\frac{V_{max}}{\epsilon}\right)^2 \frac{1}{2} \log \frac{2}{\delta}.$$

Encontrando a Política Ótima

Iteração de Valor com horizonte finito H

1. defina $V(s, H) = 0$
2. faça para todo passo $n = H - 1$ to 0
   (a) para todo estado $s \in \mathcal{S}$
      i. para toda ação $a \in \mathcal{A}$
$$Q(s, a, n) = R(s, a) + \gamma \sum_{s' \in \mathcal{S}} T(s, a, s') V(s', n + 1)$$
   (b) $V(s, n) = \max_{a \in \mathcal{A}} Q(s, a, n)$
   (c) para todo $s \in \mathcal{S}$
$$\pi(s, n) = \arg\max_{a \in \mathcal{A}} Q(s, a, n)$$
3. retorne π

Algoritmo Sparse Sampling

Algoritmo Sparse Sampling

1. calcule $H = f_H(\gamma, \epsilon, R_{max})$
2. calcule $C = f_C(\gamma, \epsilon, R_{max})$
3. para cada $a \in \mathcal{A}$
   (a) Sorteie C amostras $s_1^{s_0,a}, \ldots, s_C^{s_0,a}$
   (b) Para cada $s_i^{s_0,a}$
      i. $\widehat{V}(s_i^{s_0,a}) = EstimateV(s_i^{s_0,a}, C, H, 1)$
   (c) $\widehat{Q}(a) = R(s_0, a) + \frac{1}{C}\gamma \sum_{i=1}^{C} \widehat{V}(s_i^{s_0,a})$
4. retorna $\arg\max_{a \in \mathcal{A}} \widehat{Q}(a)$

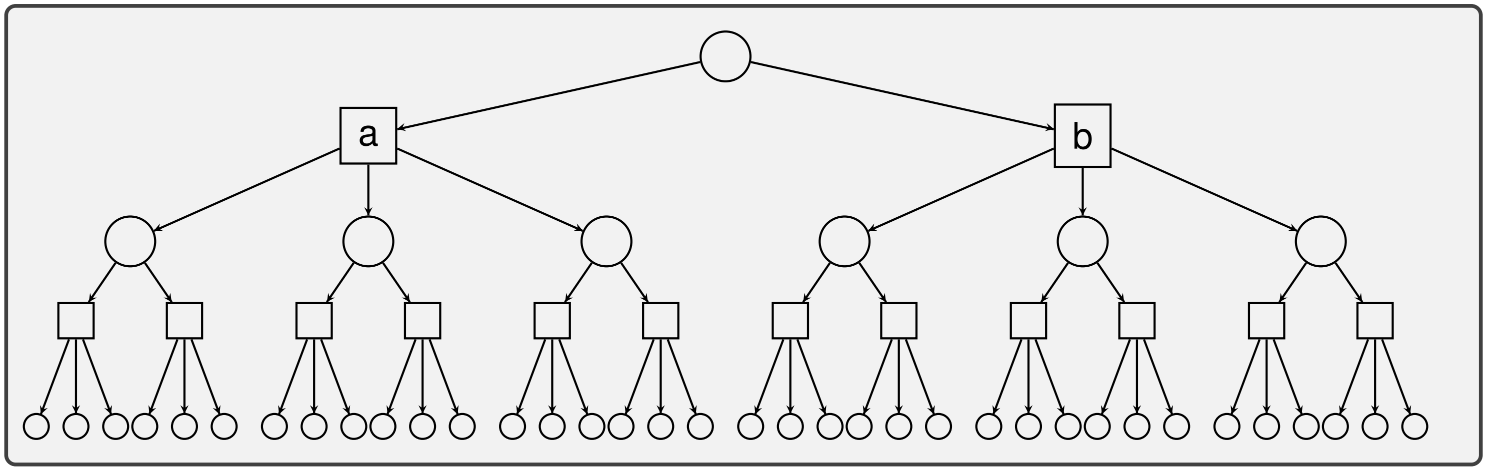
$$\lambda = \frac{\epsilon(1-\gamma)^2}{4}, \qquad H = \left\lceil \log_\gamma \left(\frac{\lambda}{V_{max}} \right) \right\rceil, \qquad C = \frac{V_{max}^2}{\lambda^2} \left(2H \log \frac{kHV_{max}^2}{\lambda^2} + \log \frac{C_{max}}{\lambda} \right)$$

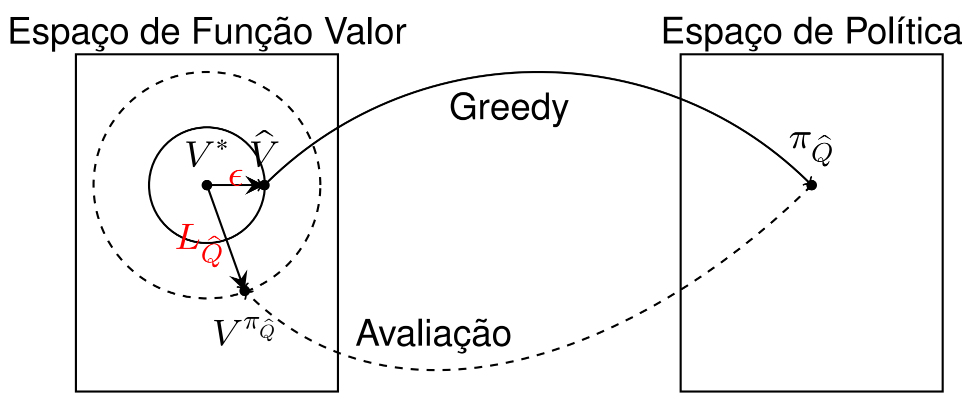
Algoritmo Sparse Sampling

EstimateV(s,C,H,n)

1. se $n = H$ retorna 0
2. para cada $a \in \mathcal{A}$
   (a) Sorteie C amostras $s_1^{s,a}, \ldots, s_C^{s,a}$
   (b) Para cada $s_i^{s,a}$
      i. $\widehat{V}(s_i^{s,a}) = EstimateV(s_i^{s,a}, C, H, n+1)$
   (c) $\widehat{Q}(a) = R(s,a) + \frac{1}{C}\gamma \sum_{i=1}^{C} \widehat{V}(s_i^{s,a})$
3. retorna $\max_{a \in \mathcal{A}} \widehat{Q}(a)$

Algoritmo Sparse Sampling





Algoritmo Sparse Sampling

Theorem 3. Assuma que $\pi_{\widehat{Q}}(s)$ é uma política gulosa resultante de um algoritmo baseado no método de Monte Carlo com base na estimativa $\widehat{Q}$, isto é, $\pi_{\widehat{Q}}(s)$ é uma variável aleatória. Se para todo $s \in \mathcal{S}$ é verdade que:

$$\Pr(|\widehat{Q}(s, \pi^*(s)) - \widehat{Q}(s, \pi_{\widehat{Q}})| \leqslant \epsilon) \leqslant 1 - \delta,$$

então:

$$V^*(s) - V^{\pi_{\widehat{Q}}}(s) \leqslant \frac{2\epsilon - 2\delta(2 - \delta)V_{max}}{1 - \gamma}$$

Algoritmos baseados em Rollout

Sparse Sampling:

- garante solução ϵ-ótima independente de $|\mathcal{S}|$
- estimativa de estados e ações são uniformes
- estados alcançáveis a partir de s_0
- complexidade $O((C \times |\mathcal{A}|)^H)$

Ideia:

- investir mais tempo e memória onde for mais útil
- estados alcançáveis a partir da política ótima

Algoritmos baseados em Rollout

Rollouts:

- inicializa no estado s_0
- escolhe e executa ações até algum critério de parada
- avalia o estado final
- atualiza todos os nós visitados

Exploration vs Exploitation

- Exploration: gastar rollouts em ramos pouco visitados (pode se tornar ramos melhores)
- Exploitation: gastar rollouts em ramos promissores (pode melhorar o ramos das melhores ações)

Algoritmo MC Rollout

1. enquanto não *timeout*
   (a) $search(s_0, 0)$
2. retorna $bestAction(s_0, 0)$

search(s,d)

1. se $Terminal(s)$ então retorne 0
2. se $Leaf(s, d)$ então retorne $Evaluate(s)$
3. $a = selectAction(s, d)$
4. $(s', r) = simulateAction(s, a)$
5. $q = r + \gamma \times search(s', d + 1)$
6. $updateValue(s, a, d, q)$
7. retorne q

Algoritmo UCT

A cada rollout um ou mais nó é adicionado à árvore

As política de rollout dependem dos nós que já estão na árvore

- seleciona ($selectAction$)
- expande ($Leaf$)
- simula ($Evaluate$)
- backup ($updateValue$)

Algoritmo UCT: Backup

UCT - Upper Confidence Tree

- $n(s, a, d)$: número de vezes que a ação a foi executada no estado s no nível d

- $n(s, d)$: número de vezes que o estado s foi visitado no nível d

- $Q(s, a, d)$: recompensa média recebida nas trajetórias executadas depois de executar a ação a no estado s no nível d

$$Q(s, a, d) \leftarrow \frac{n(s, a, d)Q(s, a, d) + \sum_{t=d}^{H-1} \gamma^{t-d} r_t}{n(s, a, d) + 1}$$

Algoritmo UCT: Seleciona

Política de Rollout

$$\pi_{UCT}(s,d) = \operatorname*{arg\,max}_{a \in \mathcal{A}} \left\{ Q(s,a,d) + \beta \sqrt{\frac{\ln n(s,d)}{n(s,a,d)}} \right\}$$

- $Q(s,a,d)$ indica o quão desejável é a ação a no nível d
- $\sqrt{\dfrac{\ln n(s)}{n(s,a,d)}}$ indica quão a foi explorado no nível d
- β indica o compromisso entre exploration e exploitation

Algoritmo UCT: Expande e Simula

Se durante a seleção um estado que não estava na árvore é encontrado:

- insira o estado na árvore e todas as ações disponíveis para aquele estado
- chama simulação a partir daquele estado

Simula:

- Aleatória
- Segundo alguma política heurística
- Retorno o valor de uma função heurística (sem simulação)

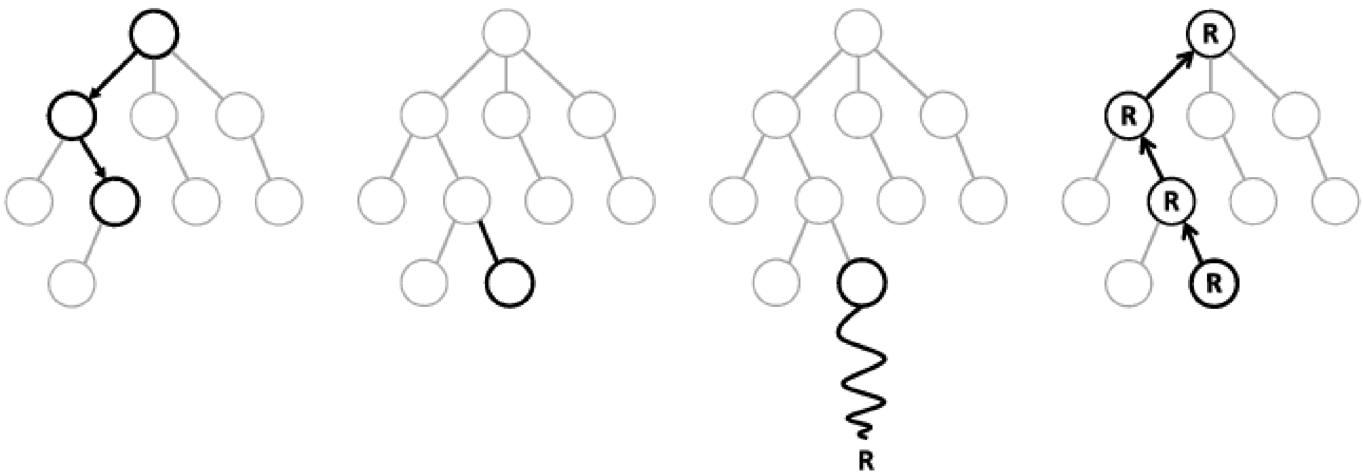
Algoritmo UTC

1. enquanto não *timeout*
   (a) $search(s_0, 0)$
2. retorna $bestAction(s_0, 0)$

search(s,d)

1. se $Terminal(s)$ então retorne 0
2. se $Leaf(s, d)$ então retorne $Evaluate(s)$
3. $a = \arg\max_{a \in \mathcal{A}} \left\{ Q(s, a, d) + \beta \sqrt{\frac{\ln n(s, d)}{n(s, a, d)}} \right\}$
4. $(s', r) = simulateAction(s, a)$
5. $q = r + \gamma \times search(s', d + 1)$
6. $Q(s, a, d) \leftarrow \frac{n(s, a, d)Q(s, a, d) + q}{n(s, a, d) + 1}$
7. $n(s, a, d) \leftarrow n(s, a, d) + 1$
8. $n(s, d) \leftarrow n(s, d) + 1$
9. retorne q

Algoritmo UCT



(a) Selection (b) Expansion (c) Simulation (d) Backpropagation

Algoritmo UCT

Resultado teórico:

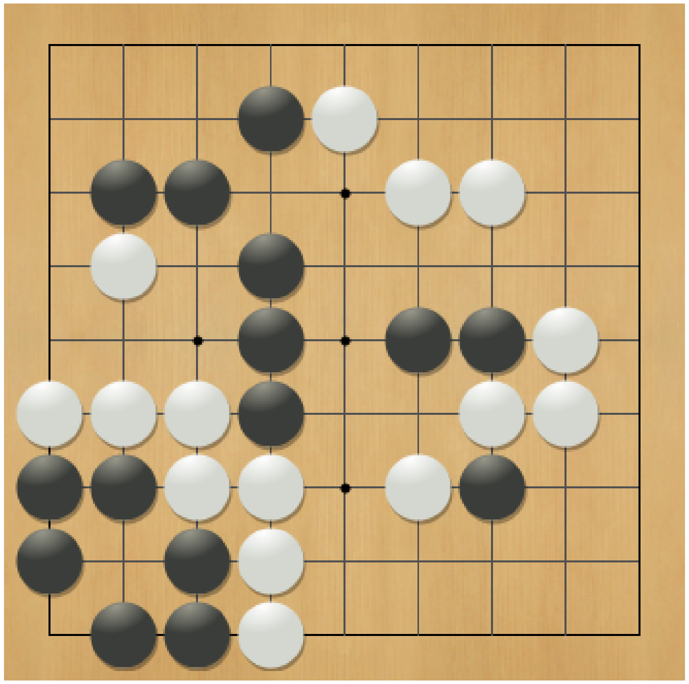
> Consider a finite-horizon MDP with rewards scaled to lie in the $[0, 1]$ interval. Let the horizon of the MDP be D, and the number of actions per state be K. Consider algorithm UCT such that the bias terms of UCB1 are multiplied by D. Then the bias of the estimated expected payoff, $\bar{X}_n$, is $O(log(n)/n)$. Further, the failure probability at the root converges to zero at a polynomial rate as the number of episodes grows to infinity.

Algoritmo UCT

Resultado teórico:

- recompensa limitada
- horizonte finito
- otimalidade (ϵ, δ)-PAC (Probably Approximately Correct)
  - viés decai com $O(log(n)/n)$
  - falha de escolha ótima em s_0 tende a zero

UCT e GO



Jogo Determinista:

- $\min\max$ operador
- muitas ações disponíveis
- heurística obtida de exemplos (self-play ou jogos reais)