Algoritmos para MDP
Prioritized Sweeping, LRTDP, LAO*

Valdinei Freire

(EACH - USP)

Exemplos

- Ações cardinais (N,S,L,O). Na linha “Det.” as ações são deterministas. Na linha “Prob.” as ações resultam com 0.5 de chance, caso contrário o agente fica parado. Custo de 1 por ação.

Prob.	s_0				G
Det.					

- Ações cardinais (N,S,L,O). As ações são deterministas, a menos do rio. No rio, as ações resultam com 0.5 de chance, caso contrário o agente volta para o estado inicial s_0.

s_0	rio	G

Iteração de Valor ($\gamma = 1$)

1.00	1.00	1.00	1.00	0
1.00	1.00	1.00	1.00	1.00

2.00	2.00	2.00	1.50	0
2.00	2.00	2.00	2.00	1.00

3.00	3.00	2.75	1.75	0
3.00	3.00	3.00	2.00	1.00

4.00	3.88	3.25	1.88	0
4.00	4.00	3.00	2.00	1.00

4.94	4.56	3.56	1.94	0
5.00	4.00	3.00	2.00	1.00

5.75	5.06	3.75	1.97	0
5.00	4.00	3.00	2.00	1.00

6.38	5.41	3.86	1.98	0
5.00	4.00	3.00	2.00	1.00

6.69	5.63	3.92	1.99	0
5.00	4.00	3.00	2.00	1.00

6.84	5.78	3.96	2.00	0
5.00	4.00	3.00	2.00	1.00

6.92	5.87	3.98	2.00	0
5.00	4.00	3.00	2.00	1.00

Iteração de Valor ($\gamma = 0.9$, custo nulo, $V_G = 1$)

0.0000	0.0000	0.0000	0.4500	1.0000	0.3174	0.4715	0.6231	0.8114	1.0000
0.0000	0.0000	0.0000	0.0000	0.9000	0.5905	0.6561	0.7290	0.8100	0.9000
0.0000	0.0000	0.2025	0.6525	1.0000	0.4085	0.5074	0.6455	0.8151	1.0000
0.0000	0.0000	0.0000	0.8100	0.9000	0.5905	0.6561	0.7290	0.8100	0.9000
0.0000	0.0911	0.3848	0.7436	1.0000	0.4496	0.5236	0.6573	0.8168	1.0000
0.0000	0.0000	0.7290	0.8100	0.9000	0.5905	0.6561	0.7290	0.8100	0.9000
0.0410	0.2141	0.5078	0.7846	1.0000	0.4680	0.5314	0.6633	0.8176	1.0000
0.0000	0.6561	0.7290	0.8100	0.9000	0.5905	0.6561	0.7290	0.8100	0.9000
0.1148	0.3916	0.5816	0.8031	1.0000	0.4763	0.5376	0.6664	0.8179	1.0000
0.5905	0.6561	0.7290	0.8100	0.9000	0.5905	0.6561	0.7290	0.8100	0.9000

Prioritized Sweeping

Iteração de Valor

- Cada iteração atualiza todos estados
- Atualizações podem trazer pouca mudança

Prioritized Sweeping

- Cada iteração atualiza apenas um estado
- Estado escolhido por ordem de prioridade
- Ideia: Resíduo $\|V_k(s) - V_{k+1}(s)\|$
- Convergência: evitar *starvation* (morte de fome)

Notação

Operador de Bellman aplicado a uma função valor $V(s)$

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

Política gulosa (*greedy*) a partir de uma função valor $V(s)$

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

Função Resíduo a partir de uma função valor $V(s)$

$$Res^V(s) = |(\mathcal{T}V)(s) - V(S)|$$

Prioritized Sweeping

1. inicializa $V_0(s)$ arbitrariamente
2. inicializa $H_0(s)$ arbitrariamente
3. faça para toda iteração $k \geqslant 0$
   (a) escolha estado com maior prioridade:
   $$s_k \leftarrow \arg\max_{s \in \mathcal{S}} H_k(s)$$
   (b) aplica operador de Bellman em s_k
   $$V_{k+1}(s_k) = (\mathcal{T} V_k)(s)$$
   e mantém o valor para os outros estados $s \neq s_k \in \mathcal{S}$: $V_{k+1}(s) = V_k(s)$
   (c) para todo $s \neq s_k \in \mathcal{S}$ gere a nova função de prioridade $H_{k+1}(s)$
   enquanto não atinge critério de parada
4. retorne a política π^V

Funções de Prioridade

Prioritized Sweeping

- Inicializa $H_0(s)$ aleatoriamente com números não negativos

- Para todo $s \in \mathcal{S}$:

$$H_{k+1}(s) = \begin{cases} \max\{H_k(s), \Delta_k \cdot \max_{a \in \mathcal{A}} T(s_k|s,a)\}, & \text{se } s \neq s_k \\ \Delta_k \cdot \max_{a \in \mathcal{A}} T(s_k|s,a), & \text{se } s = s_k \end{cases}$$

onde

$$\Delta_k = |V_{k+1}(s_k) - V_k(s_k)|$$

Generalized Prioritized Sweeping

- Para todo $s \in \mathcal{S}$:

$$H_k(s) = |(\mathcal{T}V_k)(s) - V_k(s)|$$

Prioritized Sweeping e Generalized Prioritized Sweeping convergem.

Funções de Prioridade

Considere um MDP com estados absorvedores $\mathcal{G}$.

- Backward Value Iteration
  - Atualiza estados de acordo com sua proximidade de $\mathcal{G}$
  - $\{s \in \mathcal{S} : \exists a \in \mathcal{A}\ T(s, a, \mathcal{G}) > 0\}$

- Topological Value Iteration
  - Particiona os estados em componentes fortemente conectados
  - Executa VI nos estados de um componente até convergência
  - Seleciona os componentes a partir da meta (BVI)

Shortest Stochastic Path

MDP com:

- estados absorvedores $\mathcal{G}$
- estados iniciais s_0
- existe política própria
- políticas não própria tem custo infinito

Solução: política parcial π_{s_0} definida apenas para estados alcançáveis a partir de s_0 quando π_{s_0} é executada.

Find-and-Revise

Grafo de Conectividade de um MDP: um grafo dirigido no qual as arestas são hiperarestas (uma fonte, mas vários destinos). O hipergrafo $G_{\mathcal{S}}$ representa as transições de um MDP.

Alcançabilidade: um estado s_n é alcançável de s_1 em um hipergrafo G, se existe um caminho entre s_1 e s_n em G.

Grafo de Conectividade com estado inicial s_0: é o hipergrafo G_{s_0} que contém o vértice s_0 e todos os estados s' alcançáveis de s_0, mais suas respectivas hiperarestas.

Grafo Guloso de Conectividade de um função valor V com estado inicial s_0: é o hipergrafo $G_{s_0}^V$ que contém o vértice s_0 e todos os estados s' alcançáveis de s_0 ao executar qualquer política gulosa segundo V, mais suas respectivas hiperarestas apontadas por tais políticas.

Find-and-Revise

1. inicializa $V(s) = h(s) \leqslant V^*(s)$
2. enquanto existe $s \in G_{s_0}^V$ tal que $Res^V(s) > \epsilon$
   (a) FIND um estado $s \in G_{s_0}^V$ com $Res^V(s) > \epsilon$
   (b) REVISE $V(s) = (\mathcal{T}V)(s)$
3. retorne a política parcial π^V

$h(s)$ é uma função heurística admissível.

LAO*

- Mantém subconjuntos $\widehat{G}^V_{s_0} \subseteq \widehat{G}_{s_0}$ e para os subconjunto de G_{s_0} e $G^V_{s_0}$

- A cada iteração aumenta a fronteira de $\widehat{G}_{s_0}$, atualiza V e reconstrói $\widehat{G}^V_{s_0}$

- A atualização de V é feita de forma completa

- Pode-se utilizar qualquer algoritmo para atualizar V (VI, PI, PS, etc.)

LAO*

1. inicializa $V(s) = h(s) \leqslant V^*(s)$
2. $F \leftarrow \{s_0\}$
3. $I \leftarrow \emptyset$
4. $\widehat{G}_{s_0} \leftarrow \{s_0\}$
5. $\widehat{G}^V_{s_0} \leftarrow \{s_0\}$
6. enquanto existe $s \in F \cap G^V_{s_0}$ e $s \notin \mathcal{G}$ faça
   (a) $s \leftarrow$ algum estado não meta em $F \cap G^V_{s_0}$
   (b) $F \leftarrow F \backslash \{s\}$
   (c) $F \leftarrow F \cup \{x \notin I : \exists a \in \mathcal{A}\, T(s, a, x) > 0\}$
   (d) $I \leftarrow I \cup \{s\}$
   (e) $\widehat{G}_{s_0} \leftarrow \{I \cup F\}$
   (f) $Z \leftarrow \{s$ e todos estados que, executando política gulosa, podem alcançar $s\}$
   (g) Atualize V para cada estado em Z, considerando estados em $s' \in F$ como estados terminais com valor $h(s')$
   (h) Reconstrua $\widehat{G}^V_{s_0}$ sobre estados de $\widehat{G}^V_{s_0}$
7. retorne a política parcial $\pi^V_{s_0}$ que começa em s_0 e determinada pela função V

LAO*

Variações:

- ILAO*: expande todos os nós da fronteira, atualiza o valor de cada estado em $\widehat{G}_{s_0}$ apenas uma vez de traz para frente

- RLAO*: expande os nós a partir do meta

- BLAO*: mantém dois grafos, um a partir da meta e outros a partir do estado inicial

LRTDP

Labeled Real-Time Dynamic Programming

- considera estado inicial
- simula o MDP
- atualiza estados visitados
- etiqueta estados cujo valores convergiu
- converge se todos estados alcançáveis de s_0 alcança a meta

LRTDP

1. $V(s) = h(s) \leqslant V^*(s)$
2. enquanto s_0 não foi etiquetado *Solved*
   (a) $s \leftarrow s_0$
   (b) $visited \leftarrow EMPTYSTACK$
   (c) enquanto s não foi etiquetado *Solved*
      i. $visited.PUSH(s)$
      ii. se $s \in \mathcal{G}$ então **break**
      iii. $a \leftarrow \pi^V(s)$
      iv. $V(s) \leftarrow (\mathcal{T}V)(s)$
      v. $s \sim T(s, a, \cdot)$
   (d) enquanto $visited \neq EMPTYSTACK$
      i. $s \leftarrow visited.POP(s)$
      ii. se não $CHECKSOLVED(s, \epsilon)$ então **break**
3. retorne a política π^V

$CHECKSOLVED(s, \epsilon)$

1. $rv = TRUE$
2. $open \leftarrow EMPTYSTACK$
3. $closed \leftarrow EMPTYSTACK$
4. se s não foi etiquetado *Solved* então $open.PUSH(s)$
5. enquanto $open \neq EMPTYSTACK$
   (a) $s \leftarrow open.POP(s)$
   (b) $closed.PUSH(s)$
   (c) se $Res^V(s) > \epsilon$ então
      i. $rv = FALSE$
      ii. **continue**
   (d) $a \leftarrow \pi^V(s)$
   (e) para todo $s' \in \{x \in \mathcal{S} : T(s, a, x) > 0\}$ faça
      i. se s' não foi etiquetado *Solved* e s' não está em $open \cup closed$ então
         A. $open.PUSH(s')$
6. se $rv = true$ então
   (a) para todo $s' \in closed$ etiquete s' como *Solved*
   caso contrário
   (a) para todo $s' \in closed$ faça $V(s') \leftarrow (\mathcal{T}V)(s')$

Heurísticas

Conhecimento do Domínio

- Distância Manhattan

- Distância Euclideana

Sem conhecimento do Domínio

- $h(s) = 0$

- MDP relaxado:
  - MDP determinista
  - MDP Fatorado

Horizonte Infinito

Como utilizar LRTDP e LAO* com fator de desconto?

Lembre-se que uma possível semântica para γ é a chance de continuar vivo.

Construa um novo MDP M' com a nova função de transição T', onde g é o único estado meta:

$$T'(s,a,s') = \begin{cases} \gamma T(s,a,s') & \text{se } s' \neq g \\ (1-\gamma) + \gamma T(s,a,s') & \text{se } s' = g \end{cases}$$

Referências

Lihong Li and Michael L. Littman. Prioritized sweeping converges to the optimal value function. Technical Report DCS-TR-631, Rutgers University, 2008.

Andrew W. Moore and Christopher G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. Machine Learning, 13:103–130, 1993.

Blai Bonet and Hector Geffner. Labeled RTDP: Improving the convergence of real-time dynamic programming. In Proceedings of the First International Conference on Automated Planning and Scheduling, pages 12–21, 2003.

Eric A. Hansen and Shlomo Zilberstein. LAO*: A heuristic search algorithm that finds solutions with loops. Artificial Intelligence, 129:35–62, 2001.

Planning with Markov Decision Processes: An AI Perspective. Mausan and Andrey Kolobov.