Dynamické programovanie
(Strojové učenie II)

M. Mach

Katedra kybernetiky a umelej inteligencie, FEI, TUKE

február 2021 - marec 2024

Princíp rekurzívneho rozkladu

- Je to všeobecná metóda riešenia zložitých problémov pomocou
  - rozdelenia problému na jednoduchšie podproblémy rekurzívnym spôsobom
  - vyriešenia jednotlivých podproblémov
  - skombinovania riešení podproblémov do riešenia problému
- Podproblémy sa
  - neprekrývajú $\rightarrow$ Divide & conquer
    - (príklad: algoritmus quicksort)
  - prekrývajú $\rightarrow$ Dynamické programovanie
    - (príklad: binomické koeficienty)

CIT Centrum inteligentných technológií

Dynamické programovanie

Fibonacci: F(n) = F(n-1) + F(n-2) = F(n-2) + F(n-3) + F(n-2) = ...

```
def fibonacci(n):
    if n <= 1:
        return n
    f = fibonacci(n-1) +
        fibonacci(n-2)
    return f
```

```
def fibonacci(n):
    f = [0, 1]
    for i in range(2,n+1):
        f.append(f[i-1] +
                 f[i-2])
    return f[n]
```

- Riešený problém musí mať dve vlastnosti
  - prekrývajúce sa podproblémy
    - riešenia podproblémov sú viacnásobne používané (reuse)
    - riešenia podproblémov sú memoizované (caching)
  - optimálna štruktúra
    - optimálne riešenie problému pozostáva z optimálnych riešení podproblémov

Dynamické programovanie a MDP

- Rozklad na podproblémy
  - reprezentovaný Bellmanovými rovnicami

$$
\begin{aligned}
v_\pi(s) &= E[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s] \\
q_\pi(s,a) &= E[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a]
\end{aligned}
$$

    pre rekurzívnu dekompozíciu na dve zložky
    - nasledujúci krok
    - ostávajúce kroky
- Podúlohy
  - hodnotové funkcie $v(s)$ a $q(s,a)$
    - používané pre určenie hodnoty všetkých predchodcov (stavov alebo párov stav-akcia)
    - ukladané v tabuľke pre opakované použitie
    - na základe $v_*(s)$ a $q_*(s,a)$ sa určí optimálna politika

Dynamické programovanie pre RL

- Pre počítanie optimálnej politiky na základe perfektného modelu prostredia
  - model v tvare Markovovho rozhodovacieho procesu
  - konečný MDP
  - dynamika procesu daná distribúciou $p(s', r|s, a)$

  označované ako *plánovanie* (vychádza z modelu)
- Použitie pre úlohy
  - diskrétny priestor stavov a akcií - možné riešiť priamo
  - spojitý priestor - priamo kvantovaním alebo zložitejšími prístupmi
- Limitované využitie kvôli
  - predpoklad dostupnosti perfektného modelu
  - veľké výpočtové nároky (iba menšie problémy)

CIT Centrum inteligentných technológií

Vyhodnocovanie politiky π

- Vyhodnocovanie politiky označuje určovanie hodnotových funkcií v_π a q_π pre ľubovoľnú politiku π

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) \left(r + \gamma v_\pi(s')\right)$$

- Je to vlastne sústava lineárnych rovníc
  - počet rovníc daný počtom možných stavov
- Garancia jedinečného riešenia ak platí aspoň jedno
  - $\gamma < 1$ (nutné pre kontinuálnu úlohu)
  - z každého stavu je pri π dosiahnutý terminálny stav (garancia ukončenia epizódy)

Iteračné vyhodnocovanie politiky π (1)

- Iteračné riešenie sústavy rovníc
  - $v_0(s) = 0$ pre terminálne stavy (ak také sú)
  - počiatočná aproximácia pre neterminálne stavy $v_0(s)$ môže byť ľubovoľná
  - je generovaná sekvencia aproximácií hodnotovej funkcie stavu $v_0, v_1, v_2, \ldots$
- Aktualizačné pravidlo (expected update)

$$v_{k+1}(s) = \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) \left(r + \gamma v_k(s')\right)$$

- Ukončenie ak $\max_s \mid v_{k+1}(s) - v_k(s) \mid$ je dostatočne malé

Iteračné vyhodnocovanie politiky π (2)

- Aktualizačné stratégie (aktualizácie očakávania)
  - sweep stratégia
    - dve polia, jedno pre nové hodnoty v_{k+1} a jedno pre staré hodnoty v_k
    - nové hodnoty počítané iba zo starých hodnôt
    - staré hodnoty sa počas výpočtu nových hodnôt nemenia
  - in-place stratégia
    - jedno pole reprezentujúce staré aj nové hodnoty
    - vypočítaná nová hodnota okamžite prepíše starú hodnotu
    - nové hodnoty počítane zo starých aj nových hodnôt
- Konvergencia
  - $\{v_k\}$ konverguje k v_π pre $k \to \infty$
  - in-place
    - konverguje rýchlejšie ako verzia s dvomi poliami
    - rýchlosť konvergencie je ovplyvnená poradím aktualizácie hodnôt stavov

Algoritmus odhadu v_π

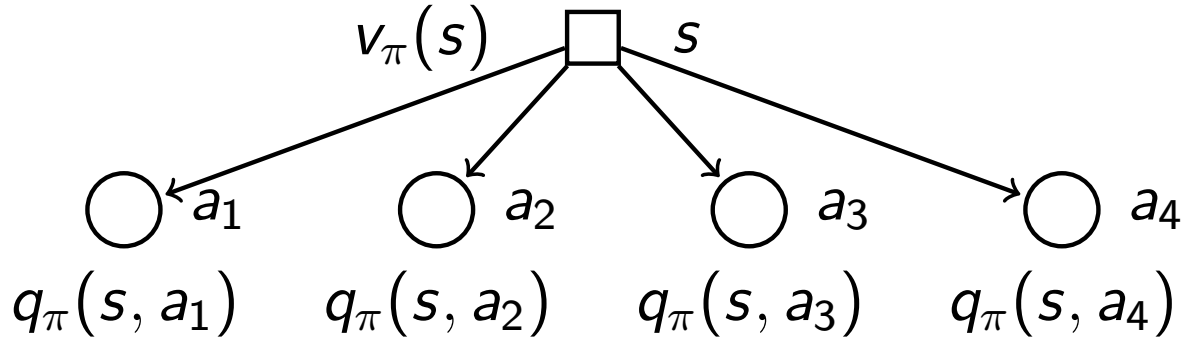
Iterative Policy Evaluation, for estimating $V \approx v_\pi$

Input π, the policy to be evaluated
Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
  $\Delta \leftarrow 0$
  Loop for each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r \,|\, s, a)\big[r + \gamma V(s')\big]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Pozn: in-place verzia

Hľadanie lepšieho výberu akcie



- Vzťah medzi $v_\pi(s)$ a $q_\pi(s, a)$

  $$\min_a q_\pi(s, a) \leq v_\pi(s) \leq \max_a q_\pi(s, a)$$

- Nech pre stav s: $v_\pi(s) < q_\pi(s, a)$
  - $v_\pi(s)$ - ako dobre je od stavu s pokračovať podľa π
  - lepšie je v s nevyberať podľa π ale raz vybrať a a potom pokračovať podľa π
  - ešte lepšie je v s vždy vyberať akciu a

Teorém zlepšovania politiky

- Teorém zlepšovania politiky (Policy Improvement)
  - nech π a π' sú deterministické politiky
  - nech $q_\pi(s, \pi'(s)) \geq v_\pi(s)$ pre všetky $s \in S$
  - potom π' musí byť rovnako dobrá alebo lepšia ako π
- Dokázať TZP znamená dokázať $v_{\pi'}(s) \geq v_\pi(s)$

$$
\begin{aligned}
v_\pi(s) &= q_\pi(s, \pi(s)) \\
&\leq q_\pi(s, \pi'(s)) = E[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = \pi'(s)] \\
&= E_{\pi'}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s] \\
&\leq E_{\pi'}[R_{t+1} + \gamma q_\pi(S_{t+1}, \pi'(S_{t+1})) | S_t = s] \\
&= E_{\pi'}[R_{t+1} + \gamma E[R_{t+2} + \gamma v_\pi(S_{t+2}) | S_{t+1}, A_{t+1} = \pi'(S_{t+1})] | S_t = s] \\
&= E_{\pi'}[R_{t+1} + \gamma E_{\pi'}[R_{t+2} + \gamma v_\pi(S_{t+2}) | S_{t+1}] | S_t = s] \\
&= E_{\pi'}[R_{t+1} + \gamma [R_{t+2} + \gamma v_\pi(S_{t+2})] | S_t = s] \\
&\leq E_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 q_\pi(S_{t+2}, \pi'(S_{t+2})) | S_t = s] \\
&\dots \\
&\leq E_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s] = v_{\pi'}(s)
\end{aligned}
$$

Zlepšovanie politiky (1)

- Rozšírenie TZP na všetky stavy a všetky akcie

$$\begin{aligned}\pi'(s) &= \arg\max_a q_\pi(s,a) \\ &= \arg\max_a E[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = a] \\ &= \arg\max_a \sum_{s'} \sum_r p(s', r|s, a)(r + \gamma v_\pi(s'))\end{aligned}$$

výber najlepšej akcie s uvážením následkov jedného dopredného kroku

- Zlepšovanie politiky ako greedy výber akcií s ohľadom na hodnotovú funkciu pôvodnej politiky

Zlepšovanie politiky (2)

- Ak $a = \pi(s)$ je deterministická politika, tak potom ju môžme skúsiť zlepšiť greedy rozšírením na $\pi'(s)$
  - ak π' priniesla zlepšenie, tak platí (pre aspoň jeden stav)

$$q_\pi(s, \pi'(s)) = \max_a q_\pi(s, a) > q_\pi(s, \pi(s)) = v_\pi(s)$$

  - ak π' nepriniesla zlepšenie, tak platí (pre všetky stavy)

$$q_\pi(s, \pi'(s)) = \max_a q_\pi(s, a) = q_\pi(s, \pi(s)) = v_\pi(s)$$

čo je vlastne Bellmanova rovnica optimality a teda $v_\pi(s) = v_*(s)$ a π je optimálnou politikou

Greedy politika

- Greedy politika spĺňa podmienky teorému
  - je lepšia alebo rovnaká ako tá politika, podľa ktorej boli vytvorené tie hodnotové funkcie, na základe ktorých greedy politika vznikla
  - ak je rovnako dobrá ako pôvodná politika, tak obe sú optimálne
- Zlepšovanie politiky - proces pretvárania politiky na greedy politiku na základe hodnotových funkcií pôvodnej politiky
- Rozšírenie na stochastickú politiku
  - všetky maximalizujúce akcie majú nenulovú pravdepodobnosť výberu
  - ostatné akcie majú nulovú pravdepodobnosť výberu

Iterácia politiky

$$\pi_0 \xrightarrow{E} v_{\pi_0} \xrightarrow{Z} \pi_1 \xrightarrow{E} v_{\pi_1} \xrightarrow{Z} \pi_2 \xrightarrow{E} \ldots \xrightarrow{Z} \pi_* \xrightarrow{E} v_*$$

- Sekvencia monotónne sa zlepšujúcich politík a hodnotových funkcií
  - na základe politiky π a k nej prislúchajúcej hodnotovej funkcie v_π možno vytvoriť lepšiu politiku π',
  - ktorá umožní lepšiu hodnotovú funkciu $v_{\pi'}$
  - ktorá umožní vytvoriť lepšiu politiku π'', ...
- Pre konečný MDP sekvencia
  - má konečný počet iterácií
  - konverguje k optimálnej politike

pretože existuje iba konečný počet deterministických politík ($||A||^{||S||}$)

Algoritmus iterácie politiky

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
   $\Delta \leftarrow 0$
   Loop for each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s)) \left[r + \gamma V(s') \right]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   old-action $\leftarrow \pi(s)$
   $\pi(s) \leftarrow \operatorname{argmax}_a \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma V(s') \right]$
   If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Iterácia hodnôt

- Vnorené cykly sú výpočtovo náročné
- Evaluácia politiky nemusí skonvergovať (v rámci limitu Δ)
  - stačí menej iterácií vyhodnotenia politiky (zmeny správnym smerom aj keď ešte mimo Δ)
  - extrémnym prípadom je iba jedna iterácia
- Kombinácia jednokrokového vyhodnotenia a zlepšovania politiky
  - iba jeden cyklus namiesto vnorených cyklov
  - sekvencia $\{v_k\}$ konverguje k v_*

Kombinácia zlepšovania a ohodnocovania

- Jednokrokové ohodnocovanie politiky ($a = \pi(s)$)

$$v_{k+1}(s) = \sum_{s'} \sum_{r} p(s', r|s, a) \left(r + \gamma v_k(s')\right)$$

- Zlepšovanie politiky (greedy výber akcie)

$$\pi(s) = \arg\max_a \sum_{s'} \sum_{r} p(s', r|s, a) \left(r + \gamma v_k(s')\right)$$

- Kombinácia zlepšovania politiky a skráteného (jednokrokového) ohodnocovania
  - počas iterovania nie je potrebné explicitné vytváranie politiky (až po skonvergovaní v)

$$v_{k+1}(s) = \max_a \sum_{s'} \sum_{r} p(s', r|s, a) \left(r + \gamma v_k(s')\right)$$

  - je to vlastne pravidlo z Bellmanovej funkcie optimality

Algoritmus iterácie hodnôt

Value Iteration, for estimating $\pi \approx \pi_*$

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
| $\Delta \leftarrow 0$
| Loop for each $s \in \mathcal{S}$:
| $v \leftarrow V(s)$
| $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$
| $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
$\pi(s) = \arg\max_a \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$

©*Sutton-Barto: Reinforcement Learning, 2nd ed., 2018*

Algoritmy synchrónneho DP

Algoritmus	Zdrojová teória
Iteratívne vyhodnocovanie politiky	Bellmanova rovnica očakávania
Iterácia politiky	Bellmanova rovnica očakávania + greedy zlepšovanie politiky
Iterácia hodnôt	Bellmanova rovnica optimality

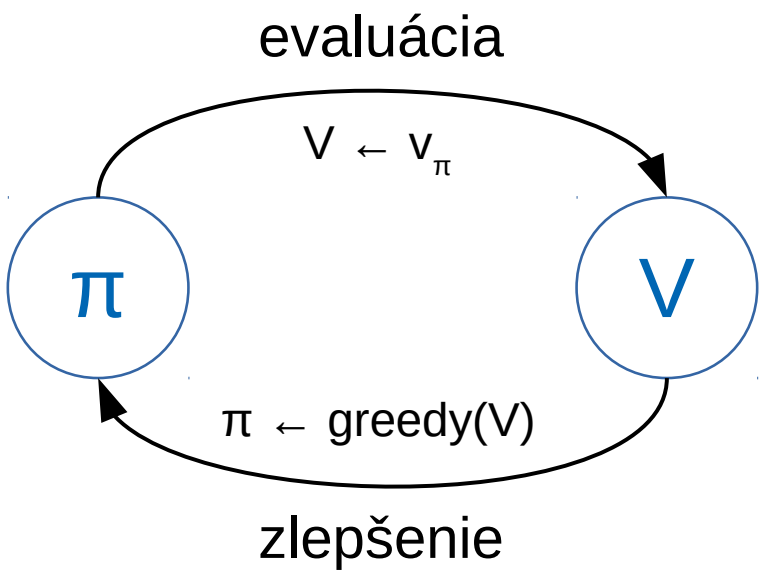
- Založené na práci s hodnotovou funkciou stavu
- Zložitosť je polynomiálna vzhľadom na počet stavov a počet akcií

Asynchrónne DP

- Sekvencia úplných prechodov všetkých stavov je nevhodná ak
  - množina stavov je veľmi veľká
  - nie všetky stavy sú zaujímavé z pohľadu optimálnej politiky
- Asynchrónne DP algoritmy
  - in-place iterácie
  - nie sú systematické prechody celou množinou stavov
    - niektoré stavy môžu byť aktualizované viackrát kým iné budú aktualizované iba raz
    - kvôli konvergencii nemožno stavy vynechať úplne
  - flexibilita ohľadom poradia aktualizácie stavov

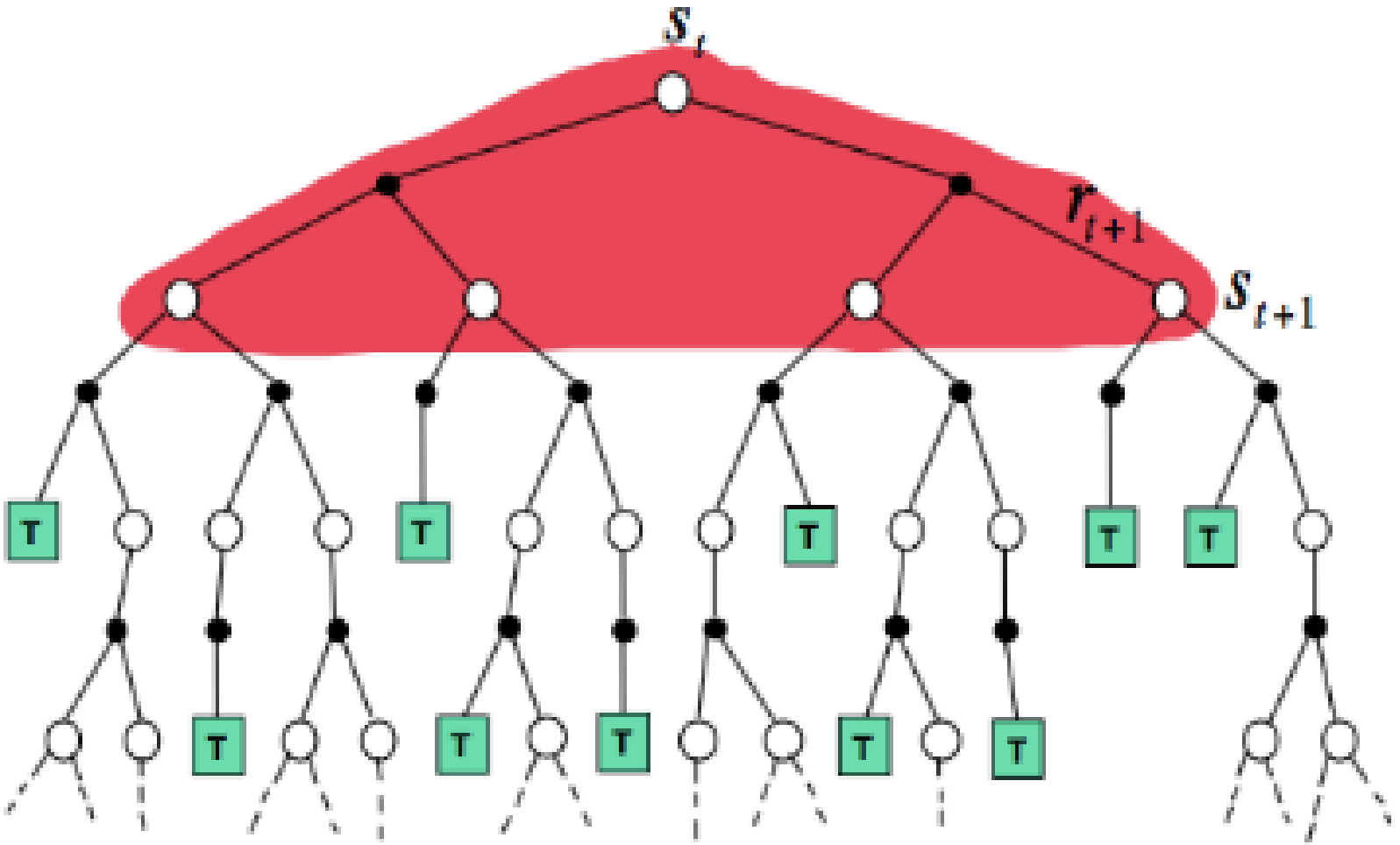
Interakcia evaluácie a zlepšovania

- Interakcia dvoch procesov (Π a V)



- rôzne granularity striedania
  - iterácia politiky
  - iterácia hodnôt
  - asynchrónne DP
- procesy súťažia aj kooperujú
- konvergencia smerom k optimálnym v_* a π_*
  - iba spoločná stabilizácia

Backup diagram



©/lilianweng.github.io