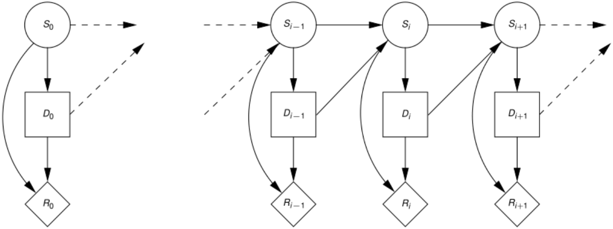
Markovské rozhodovací procesy

- Předpokládáme, že se množina možných stavů S nemění v průběhu času
- **Markovská vlastnost** stav v čase $t+1$ je nezávislý na stavu v čase $t-i$, $i > 0$ při znalosti stavu v čase t, tj.

$$S_{t+1} \perp\!\!\!\perp S_{t-i} | S_t$$

- Toto jsou plně pozorované markovské procesy prvního řádu; markovské procesy vyššího řádu dovolují závislost na více předchozích stavech.

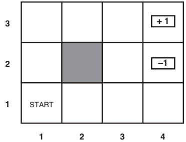


Markovský rozhodovací proces MDP

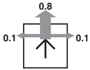
Definition (Markovský rozhodovací proces MDP)

Markovský rozhodovací proces je definuje:

- Množina stavů S v každém časovém bodě stejná
- Počáteční stav s_0
- Množina možných akcí A
- Matice přechodu $T(s, a, s^{|}) \equiv P(s^{|}|s, a)$
- Výplata v každém stavu $R(s)$.
- (discount faktor $\gamma \in < 0, 1 >$).



(a)



(b)

$R(s) = -0.04, \gamma = 1$

Kumulovaná výplata

Protože jde o proces v čase, výplaty musíme sčítat.
Jsou dvě možnosti:

- MDP s konečným horizontem – předem stanoví počet kroků
  - 'klasický' influenční diagram
- nekonečný proces a součet, ale hodnoty v budoucnosti budeme počítat s nižší hodnotou (kdo ví, co bude pak).
  Zvolíme tzv. **diskontní faktor** γ, $0 < \gamma < 1$, a maximalizujeme

$$E(U(s_0, \ldots, s_t, \ldots)) = E(\sum_{t=0}^{\infty} \gamma^t R(s_t))$$

  - Očekávaná hodnota, protože výsledek akcí je nedeterministický.
- γ odpovídá úrokové míře $\frac{1}{\gamma} - 1$, kterou musíme platit.
- především nám zajistí konečnost součtu řady, $U(s_0, \ldots, s_t, \ldots) \le \frac{R_{max}}{(1-\gamma)}$

Strategie (policy)

Cílem je najít takovou **strategii** π^*, která maximalizuje očekávanou výplatu, tj.

$$\pi^* = argmax_\pi E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t)|\pi\right]$$

- Konečný horizont vede k **nestacionární** strategii, tj. různé strategii dle toho, kolik času zbývá do konce. $\pi : History \rightarrow A$
- Nekonečný horizont vede ke **stacionární** strategii, tj. doporučený tah v políčku (stavu) je stejný nezávisle na počtu již provedených tahů.
- Reprezentovat stacionární strategii je snažší $\pi : S \rightarrow A$.
- Máme–li jistotu, že agent musí skončit v cílovém stavu, můžeme počítat s $\gamma = 1$, jinak bychom mohli neustále zvyšovat užitek a neměli bychom zajištěno nalezení optimální strategie. (Šlo by hledat maximální zisk za k kroků.)

Algoritmus řešení MDP Iterací hodnot

input: MDP, stavy S, přechody T, výplata R, diskontní f. γ,ϵ
prom.: U, U', vektory užitku stavů v S, na poč. 0
δ maximální změna užitku v jedné iteraci

repeat

$U \leftarrow U'; \delta \leftarrow 0$

for each state s in S **do**

$U'[s] \leftarrow R[s] + \gamma max_a \sum_{s'} T(s,a,s')U(s')$

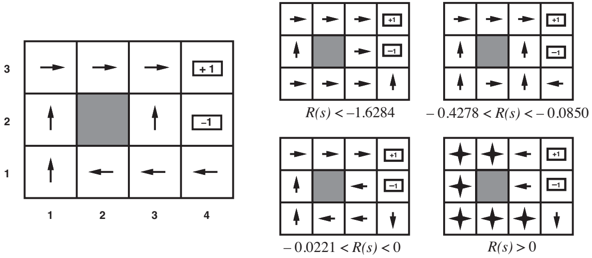
if $|U'[s] - U[s]| > \delta$ **then** $\delta \leftarrow |U'[s] - U[s]|$

until $\delta < \epsilon(1-\gamma)/\gamma$

return U

3	0.812	0.868	0.918	+ 1
2	0.762		0.660	–1
1	0.705	0.655	0.611	0.388
	1	2	3	4

Optimální strategie



(a) (b)

- Výpočet *HODNOTA_STRATEGIE*(π, U, MPD) vyžaduje vyřešení soustavy $|S|$ lineárních rovnic pro $U[s]$ v jednotlivých stavech.

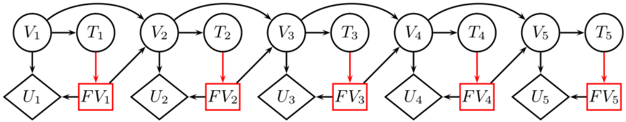
$$U_i[s] = R(s) + \gamma \sum_{s'} T(s, \pi(a), s')U_{i-1}[s']$$

Iterace strategie (policy iteration)

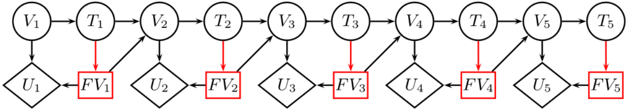
input: MDP, stavy S, přechody T, výplata R, diskontní f. γ,ϵ
prom.: U, vektory užitku stavů v S, na poč. 0
π strategie, na začátku náhodná
repeat
$U \leftarrow HODNOTA\_STRATEGIE(\pi, U, MPD)$
$unchanged? \leftarrow true$
for each state s in S **do**
if $max_a \sum_{s^|} T(s, a, s^|)U[s^|] > \sum_{s^|} T(s, \pi[s], s^|)U[s^|]$
then
$\pi[s] \leftarrow argmax_a \sum_{s^|} T(s, a, s^|)U[s^|]$
$unchanged? \leftarrow false$
until $unchanged?$
return π

- Jediný problém je při velkém počtu stavů, 10^5 rovnic o 10^5 neznámých je dost.
- Možné kombinace iterování hodnot a výpočtu.

Příklad



- Rozhodovací proces na obrázku výše není markovský.
- $\sigma_{FV_5}(T_1, FV_1, T_2, FV_2, T_3, FV_3, T_4, FV_4, T_5)$ hodně velká tabulka



- Rezignujeme na přesnost, aproximujeme.
- Kdybychom pozorovali přímo V_i, šlo by o markovský rozhodovací proces.
- Použitím eliminace na uzly V_i dostaneme (obecnější) markovský proces na $S_i \equiv T_i$.
- $\sigma_{FV_5}(T_1, FV_1, T_2, FV_2, T_3, FV_3, T_4, FV_4, T_5) = \sigma_{FV_5}(T_5^{|})$.
  - už je 'malá' (ne větší než zadání MDP)

Nebo jdeme do POMPD.

Částečně pozorovatelné markovské procesy (POMDP)

- Nejsme schopni pozorovat stav, pouze skrze nepřesné senzory.
- Pointy:
  - Proces je opět makrovský, pokud bereme BELIEF (PRAVDĚPODOBNOSTNÍ ROZLOŽENÍ) na stavech. Těch je ale nekonečně, tedy máme problém.
  - Tzv. Witness algoritmus – hledáme svědka, že naše strategie není optimální.

Zpětnovazební učení
Reinforcement learning

- "Svět"se chová podle MDP
- ale **my neznáme T, R**, matici přechodů a výplat
- opět chceme najít strategii s maximálním očekávaným ziskem.

Přímý odhad očekávané výplaty U

- **Pro fixní strategii**
- provedeme mnoho pokusů
- pro každý startovní stav spočteme průměr přes výplaty ze stavu v pokusech, kdy jsem ze stavu vycházela
- tohle ale ignoruje vztah očekávaných výplat v sousedních stavech a hlavně vyžaduje příliš mnoho pokusů

Odhad matice přechodů T

- Běháme a sbíráme zkušenosti přechodu a výplaty $s, a, s^{|}, r$
- pro každé s, a spočteme frekvence pro $s^{|}$ na základě zkušenosti
- tyto frekvence prohlásíme za odhad $T(s, a, s^{|})$
- zároveň počítáme odhad $R(s, a, s^{|})$

Odhad $U()$ pomocí ADP

Adaptivní dynamické programování

- Vezmeme naučenou matici T
- vezmeme naučené (průměry) okamžité výplaty R
- dosadíme do algoritmů pro řešení MDP.
- Problém je, že pro velké stavové prostory (např. Backgammon) potřebujeme řešit 10^{50} rovnic o 10^{50} neznámých.
- I sama matice T je veliká.

Temporal difference learning (TD learning)

- Odhaduje U bez modelu, **při pevné strategii**, aktualizací po každém kroku.
- **Víme**, odkud jsme přišli – upravíme odhad předchozího stavu.
- Řekněme, že jsem šla ve stavu s dostala r a šla do stavu $s^{|}$
- Pro deterministický proces by stačila úprava:

$$U^{\pi}(s) \leftarrow (r + \gamma U^{\pi}(s^{|}))$$

- aby konvergovalo pro nedeterministický proces, upravíme:

$$U^{\pi}(s) \leftarrow U^{\pi}(s) + \alpha(n)(r + \gamma U^{\pi}(s^{|}) - U^{\pi}(s))$$

- $\alpha(n)$ je parametr rychlosti učení – nejlépe zpočátku blízký 1, máme–li dost zkušeností, blízký 0, vždy mezi 0 a 1.
  Potřebujeme: $\sum \alpha(n) = \infty$ a $\sum \alpha^2(n) < \infty$, např. $\alpha(n) = \frac{1}{n}$ nebo $\alpha(n) = \frac{60}{59+n}$.

Temporal difference learning (TD learning)

- Neučí se tak rychle, jako ADP
- má větší variabilitu dle náhody průchodů
- ale je daleko jednodušší.
- **nepotřebuje model**, resp. model je zahrnut přímo v rovnici aktualizace.

Vztah ADP a TD; prioritized sweeping – ADP aktualizuje pouze stavy, jejichž následníci se hodně změnili.

Aktivní zpětnovazební učení

Už nemá pevnou strategii, strategii volíme. Chceme:

$$\pi^*(s) = argmax_a \sum_{s^|} T(s, a, s^|)U(s^|)$$

Pokud bychom vždy volili akci s maximálním odhadem U, tak neobjevíme nové cesty, které by mohly mít ještě lepší U (experimentálně potvrzeno).

Zkoumat či využívat
(Exploration vs. exploitation)

- Najít optimální rovnováhu mezi využítím a zkoumáním je složité.
- Dost dobrý je princip GLIE (greedy in the limit of infinite exploration):
  - každou akci v každém stavu zkusíme nekonečně krát (tj. vyhneme se v limitě tomu, abychom minuli přechod s nenulovou pravděpodobností)
  - s postupem času více využíváme a méně zkoumáme (v limitě jen využíváme, tj. **greedy**).
  - jedna možnost: s pravděpodobností $\frac{1}{t}$ volit náhodnou akci; konverguje pomalu
  - lepší: přidat váhu málo prozkoumaným akcím, tj. potlačit ty, co známe a jsou špatné
  - tj. rozprostřeme hypotetické velké zisky R^+ na neprozkoumaná území, dokud akci v daném stavu nevyzkoušíme N_e krát,

$$U^+(s) \leftarrow^{(det)} R(s) + \gamma max_a f\left(max_a \sum_{s^|} T(s,a,s^|)U^+(s^|), N(a,s)\right)$$

$f(u,n) = R^+$ pro $n < N_e$, jinak $= u$.

Q–learning (učení se funkce akce–hodnota)

- Definujeme funkci $Q(a,s) = R(s) + \gamma \sum_{s'} T(s,a,s')U(s')$.
- Tedy: $U(s) = max_a Q(a,s)$, a pro optimální Q:
  $Q(a,s) = R(s) + \gamma \sum_{s'} T(s,a,s') max_{a'} Q(a',s')$
- **Pro výběr akce nemusíme znát model T, stačí Q (což je menší matice).**
- Pro učení Q budeme používat TD–učení s rovnicí aktualizace:

$$Q(a,s) \leftarrow Q(a,s) + \alpha(R(s) + \gamma max_{a'} Q(a',s') - Q(a,s))$$

Q–learning ($s^{|}$ nový stav, $r^{|}$ akt. výplata)

Algoritmus Q–learning ($s^{|}$ nový stav, $r^{|}$ akt. výplata)

`First run:`

  $s, a, r \leftarrow null$

  For each s, a

    $Q(s, a) \leftarrow 0$

    $N_{sa} \leftarrow 0$

`Each run:`

if s is not null **then do**

  $N_{sa} ++$

  $Q(a, s) \leftarrow Q(a, s) + \alpha(r + \gamma max_{a^{|}} Q(a^{|}, s^{|}) - Q(a, s))$

if terminal?($s^{|}$) **then** $s, a, r \leftarrow null$

else $s, a, r \leftarrow s^{|}, argmax_{a^{|}} f(Q(a^{|}, s^{|}), N_{sa}[s^{|}, a^{|}]), r^{|}$

return a

Aproximace funkcí

- Tabulky U, Q můžeme počítat tak do rozsahu 10000.
- Backgammon a jiné aplikace potřebují daleko víc 10^{50} až 10^{120}.
- Proto tabulku **aproximujeme funkcí (modelem)**.
- Např. lineárním, tj. pro předem stanovené jevy $f_1, \ldots, f_n$:

$$\hat{U}_\theta(s) = \theta_1 f_1(s) + \ldots + \theta_n f_n(s).$$

- a použijeme metody strojového učení na učení θ.
- Minimalizujeme kvadratickou chybu: $Err = \sum_{j \in pokusy} \frac{(\hat{U}_\theta(s) - u_j(s))^2}{2}$
- v on–line verzi aplikujeme pravidlo:

$$\theta_i \leftarrow \theta_i - \alpha \frac{\partial Err_j(s)}{\partial \theta_i} = \theta_i + \alpha(u_j(s) - \hat{U}_\theta(s)) \frac{\partial \hat{U}_\theta(s)}{\partial \theta_i}.$$

- Aplikací tohoto pravidla na TD–learning dostaneme pro užitek stavů

$$\theta_i \leftarrow \theta_i + \alpha(R(s) + \gamma \hat{U}_\theta(s^|) - \hat{U}_\theta(s)) \frac{\partial \hat{U}_\theta(s)}{\partial \theta_i}.$$

a pro funkci Q:

$$\theta_i \leftarrow \theta_i + \alpha(R(s) + \gamma max_{a^|} \hat{Q}_\theta(a^|, s^|) - \hat{Q}_\theta(a, s)) \frac{\partial \hat{Q}_\theta(a, s)}{\partial \theta_i}.$$

Aplikace

- TD–Gammon
- Inverzní kyvadlo (balancování koštěte)
- vrtulníky
- učení síly kopu v robotím fotbale
- a mnoho dalších.