

# Üzleti Intelligencia

## 4. Előadás: Monte Carlo és temporális különbségek

Kuknyó Dániel  
Budapesti Gazdasági Egyetem

2023/24  
1.félév

1 Ismétlés

2 Monte Carlo

3 Időbeli különbségek

4 SARSA

5 *n*-lépéses különbségek

6 Összefoglalás

## 1 Ismétlés

## 2 Monte Carlo

## 3 Időbeli különbségek

## 4 SARSA

## 5 *n*-lépéses különbségek

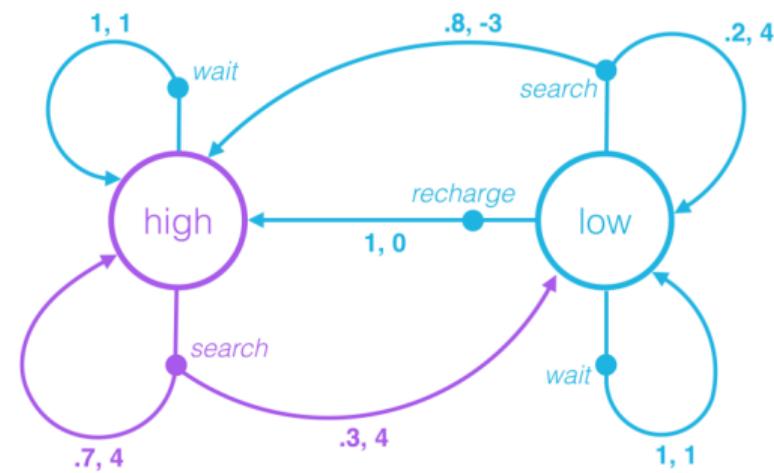
## 6 Összefoglalás

# Ügynök-környezet interakció

## Markov döntési folyamat

$$MDP(S, A, P, R, s_0, \gamma)$$

- $S$ : állapotok halmaza
- $A$ : cselekvések halmaza
- $P : S \times A \times S \rightarrow [0, 1]$ : állapotátmeneti valószínűségek
- $R : S \times A \rightarrow \mathbb{R}$ : azonnali jutalmak
- $s_0$ : kezdőállapot
- $\gamma$ : diszkont faktor



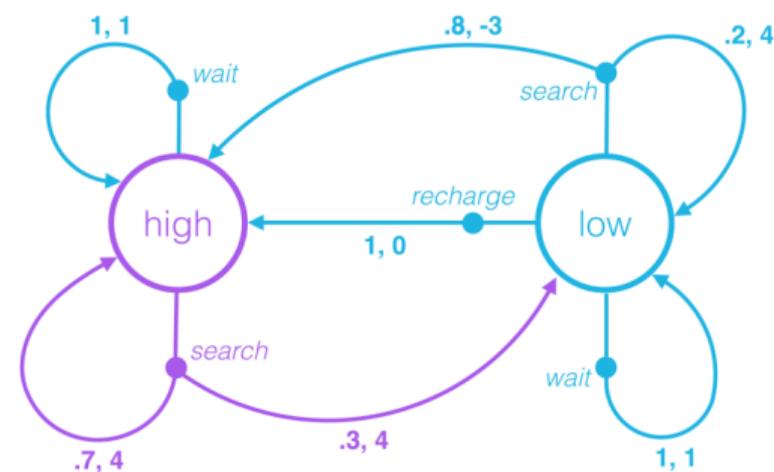
# Ügynök-környezet interakció

Az MDP folyamata:

- ① Az ügynök  $s_0$  állapotból indul
- ② Az ügynök  $\pi$  politika szerint cselekszik:  
 $a_t \sim \pi(s_t)$
- ③ A környezet reagál a cselekvésre, és visszaadja az ügynöknek  $r_{t+1}$  jutalmat és  $s_{t+1}$  következő állapotot
- ④ Ez ismétlődik amíg a kilépési kritérium be nem teljesül

Cél: Az optimális politika megtalálása. A politika optimális, ha a hozamának várható értéke maximális:

$$E_\pi (r_1 + \gamma r_2 + \gamma^2 r_3 + \dots) \rightarrow \max$$



# Monte Carlo becslés

A dinamikus programozási algoritmusok futtatásához szükség volt a környezet dinamikájának modelljére. Ez a tulajdonságuk gyakran használhatatlanná teszik őket a gyakorlatban, mert a környezeti dinamika vagy nem ismert vagy nem kiszámítható sok esetben.

A Monte Carlo (**MC**) módszerek ezzel szemben nem igényelnek előzetes tudást a feladat elvégzéséhez: csak **tapasztalat** szükséges ahhoz, hogy megtanuljanak elvégezni egy feladatot. Ezt úgy érik el, hogy mintát vesznek állapotokból, cselekvésekkel és jutalmakból, majd az eredményeket átlgolják.



# Monte Carlo becslés

## Monte Carlo szimuláció

Számítási algoritmusok olyan osztálya, ami a véletlen vagy bizonytalan komponens hatását analizálja vagy szimulálja sztochasztikus folyamatokban.



# Monte Carlo a megerősítéses tanulásban

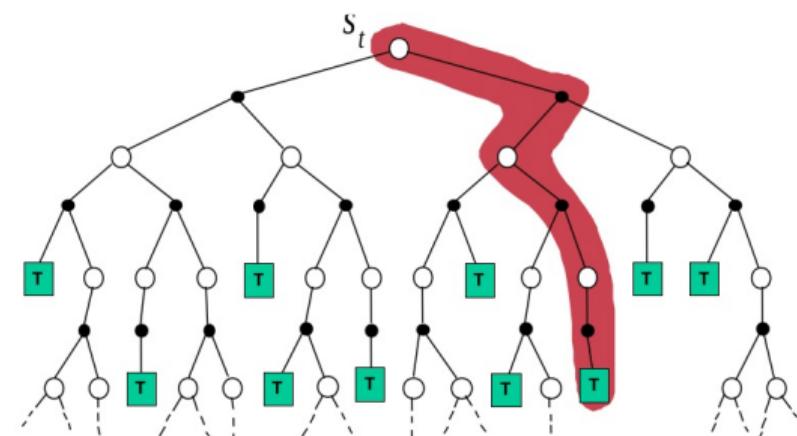
A MC tanítási algoritmus a politika iteráció egy általánosított változata. A megerősítéses tanulásban a MC algoritmusok a mélyiségi bejárásnak felelnek meg.

## MC állapot-érték frissítési szabály

$$V(s_t) \leftarrow V(s_t) + \alpha [G_t - V(s_t)]$$

- $\alpha$ : Tanulási sebesség
- $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$ : Diszkontált kumulált hozam

A központban egy egyszerű ötlet áll: epizódonkénti nyers tapasztalatok alapján tanul anélkül, hogy szüksége lenne a környezet dinamikájának modelljére.



# Értékfüggvények a Monte Carlo tanításban

Mivel a MC epizódonként vesz mintát a hozamokból, az értékfüggvények csak sok epizód hozamának átlagolásaként számítódhatnak ki.

Az állapot-érték függvény megadja, mennyi a várható hozam, ha az ügynök adott  $s$  állapotban áll, és onnan  $\pi$  politikát követi:

$$V_\pi(s) = \text{Avg} \{G_{t:T} | s_t = s\}$$

Ahol

$$G_{t:T} = \sum_{k=0}^{T-t} \gamma^k r_{t+k}$$

$t$  aktuális időlépéstől a  $T$  terminális állapotba vezető időlépésig az ügynök által összegyűjtött diszkontált kumulált hozam.



# Értékfüggvények a Monte Carlo tanításban

Mivel a MC epizódonként vesz mintát a hozamokból, az értékfüggvények csak sok epizód hozamának átlagolásaként számítódhatnak ki.

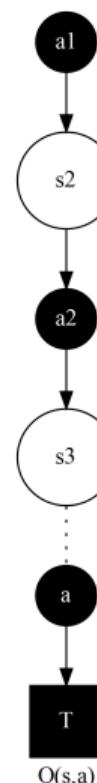
Az **állapot-cselekvés minőség függvény** megadja, mennyi a várható hozam, ha az ügynök adott  $s$  állapotban áll,  $a$  cselekvést végrehajtja, majd onnan  $\pi$  politikát követi:

$$Q_\pi(s, a) = \text{Avg} \{G_{t:T} | s_t = s, a_t = a\}$$

Ahol

$$G_{t:T} = \sum_{k=0}^{T-t} \gamma^k r_{t+k}$$

$t$  aktuális időlépéstől a  $T$  terminális állapotba vezető időlépésig az ügynök által összegyűjtött diszkontált kumulált hozam.

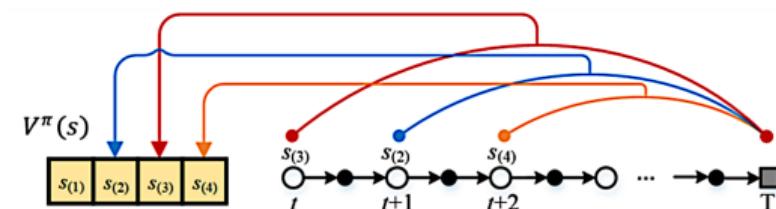


$Q(s,a)$

# MC politika kiértékelés

A  $V_\pi(s)$  állapot-érték függvény egy állapotonkénti vektor, amely minden állapot esetén számon tartja az adott  $s$  állapotból indulva összegyűjtött jutalmak várható értékét ha az ügynök egy adott  $\pi$  politika szerint cselekszik.

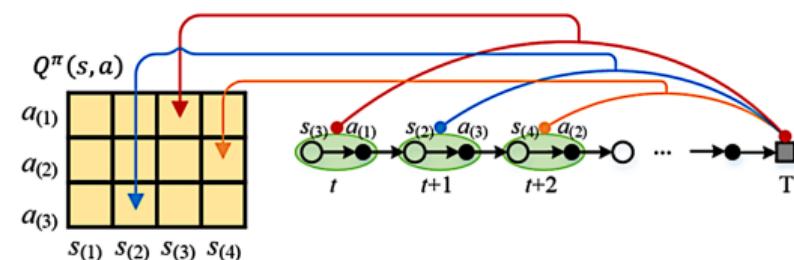
Az értékfüggvény **diszkrét** adatszerkezetben tartja számon a hozamok várható értékét, ezért diszkrét állapotokat és cselekvéseket feltételez.



# MC politika kiértékelés

A  $Q_\pi(s, a)$  állapot-cselekvés minőség függvény egy mátrix, amely az állapot-cselekvés párokhoz tartozóan tárolja el, hogy egy adott  $s$  állapotból  $a$  cselekvést végrehajtva mennyi a jutalom várható értéke ha az ügynök egy adott  $\pi$  politika szerint cselekszik.

Az értékfüggvény **diszkrét** adatszerkezetben tartja számon a hozamok várható értékét, ezért diszkrét állapotokat és cselekvéseket feltételez.

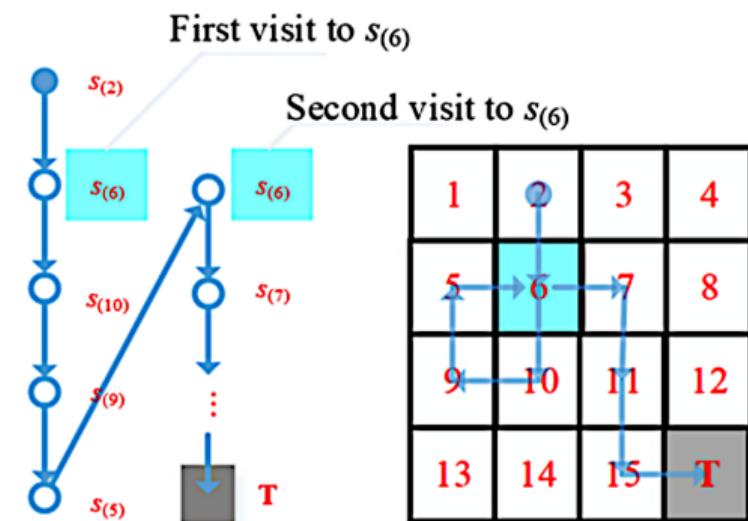


# Látogatás alapú Monte Carlo becslés

Az ügynök-környezet interakció alatt az ügynök egy adott  $s$  állapotot többször is meglátogathat egy epizód alatt. Az ábrán ez a 6-os állapot, amit kétszer látogatott meg az ügynök.

Ezt kétféleképpen tudja kezelní az MC algoritmusa: vagy az első látogatást, vagy minden látogatást fog figyelembe venni az értékfüggvények számításakor. Az első látogatás alapú MC esetén csak az első látogatás hozamát fogja figyelembe venni átlagoláskor, míg az összes látogatás alapú esetén mindegyiket.

Mindkét esetben a valós értékhez való konvergencia garantált, ahogy a látogatások száma közelít a  $\infty$ -hez.



---

**Algoritmus 1:** Első látogatás alapú Monte Carlo algoritmus  $v_\pi$  becslésére

---

**Input:** A kiértékelendő  $\pi$  politika

$V(s) \leftarrow \text{random}()$ , minden  $s \in S$ -re;

$Hozam(s) \leftarrow \text{üres lista}$ , minden  $s \in S$ -re;

**for**  $i = 0 \rightarrow \max_i$  **do**

    Epizód lejátszása  $\pi$  politikával:  $s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_{T-1}, a_{T-1}, r_T$ ;

$G \leftarrow 0$ ; /\* Hozam inícializálása 0 értékkel \*/

**for**  $t = T - 1 \rightarrow 0$  **do**

$G \leftarrow \gamma G + R_{t+1}$ ; /\* Hozam diszkontálása és kumulálása \*/

**if**  $s_t$  not in  $s_0, s_1, \dots, s_{t-1}$  **then**

$G$  hozzáfűzése  $Hozam(s_t)$ -hez; /\* Állapothoz tartozó hozam \*/

$V(s_t) \leftarrow \text{Avg}\{Hozam(s_t)\}$ ; /\* Hozam várható értéke \*/

**end**

**end**

**end**

---

---

## Algoritmus 2: Monte Carlo becslés $\pi_*$ keresésére

---

$\pi(s) \in A(s)$  véletlenszerűen, minden  $s \in S$ -re;

$Q(s, a) \in \mathbb{R}$  véletlenszerűen, minden  $s \in S$ -re és  $a \in A$ -ra;

$Hozam(s, a) \leftarrow$  üres lista minden  $s \in S$ -re és  $a \in A$ -ra;

**for**  $i = 0 \rightarrow max_i$  **do**

$s_0 \in S$  és  $a_0 \in A$  véletlenszerű kiválasztása;

Epizód lejátszása  $s_0, a_0$ -ból  $\pi$  politikával:  $s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_{T-1}, a_{T-1}, r_T$ ;

$G \leftarrow 0$ ; /\* Hozam inicializálása 0 értékkel \*/

**for**  $t = T - 1 \rightarrow 0$  **do**

$G \leftarrow \gamma G + R_{t+1}$ ; /\* Hozam diszkontálása és kumulálása \*/

**if**  $(s_t, a_t)$  **not in**  $s_0, s_1, \dots, s_{t-1}$  **then**

$G$  hozzáfűzése  $Hozam(s_t, a_t)$ -hez; /\* Állapot-cselekvés párok hozama \*/

$Q(s_t, a_t) \leftarrow Avg\{Hozam(s_t, a_t)\}$ ; /\* Hozam várható értéke \*/

$\pi(s_t) \leftarrow argmax_a Q(s_t, a)$ ; /\* Politika frissítése \*/

**end**

**end**

**end**

---

# Példa: Monte Carlo értékbecslés egy látogatás esetén

Az ügynök 3 tanítási iteráción keresztül játszott. minden esetben a végállapot  $s_T = s_{13}$ . Ez 3 tanítási mintának felel meg:

$$Hozam(01) = \frac{2}{s_{05}} + \frac{1}{s_{04}} + \frac{2}{s_{07}} + \frac{2}{s_{10}} + \frac{1}{s_{11}} + \frac{5}{s_{13}} = 13$$

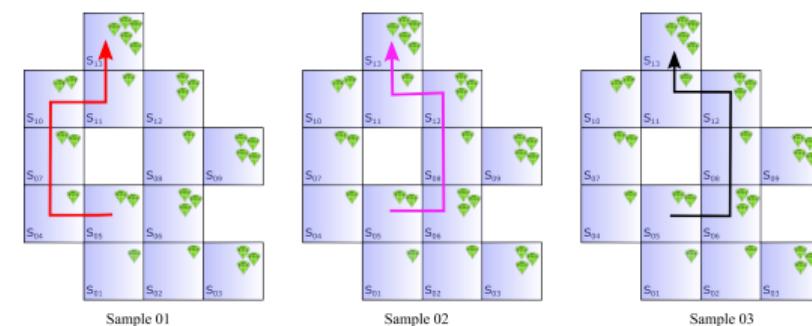
$$Hozam(02) = \frac{2}{s_{05}} + \frac{3}{s_{06}} + \frac{1}{s_{08}} + \frac{3}{s_{12}} + \frac{1}{s_{11}} + \frac{5}{s_{13}} = 15$$

$$Hozam(03) = \frac{2}{s_{05}} + \frac{3}{s_{06}} + \frac{1}{s_{08}} + \frac{3}{s_{12}} + \frac{1}{s_{11}} + \frac{5}{s_{13}} = 15$$

Tehát a megfigyelt átlagos hozam  $s_{05}$  állapotból indulva:

$$V_\pi(s_{05}) = \frac{13 + 15 + 15}{3} = 14.33$$

Megjegyzés:  $\gamma = 1$



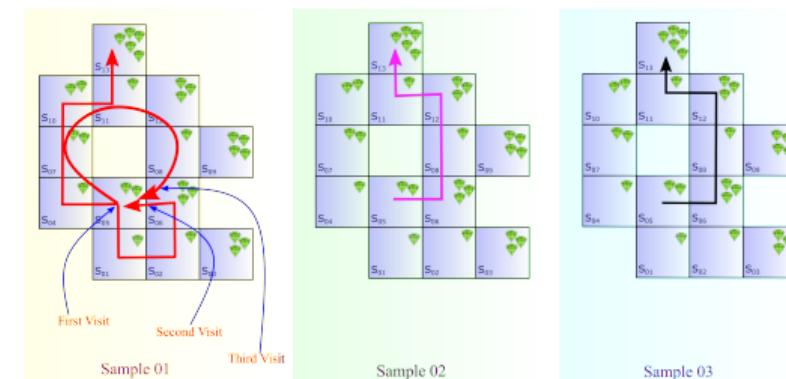
# Példa: Monte Carlo értékbecslés több látogatás esetén

Ha az ügynök többször látogatta meg ugyanazt az állapotot a tanítási iterációk során, a tanítási iteráció során kezelní kell a több látogatásból adódó jutalom többletet. Az első látogatás alapú MC tanulás során az ügynök első látogatásától a jutalom kumulálódva számítódik.

Ebben az esetben  $s_{05}$  állapot értéke:

$$V_\pi(s_{05}) = \frac{33 + 15 + 15}{3} = 21$$

Megjegyzés:  $\gamma = 1$



1 Ismétlés

2 Monte Carlo

3 Időbeli különbségek

4 SARSA

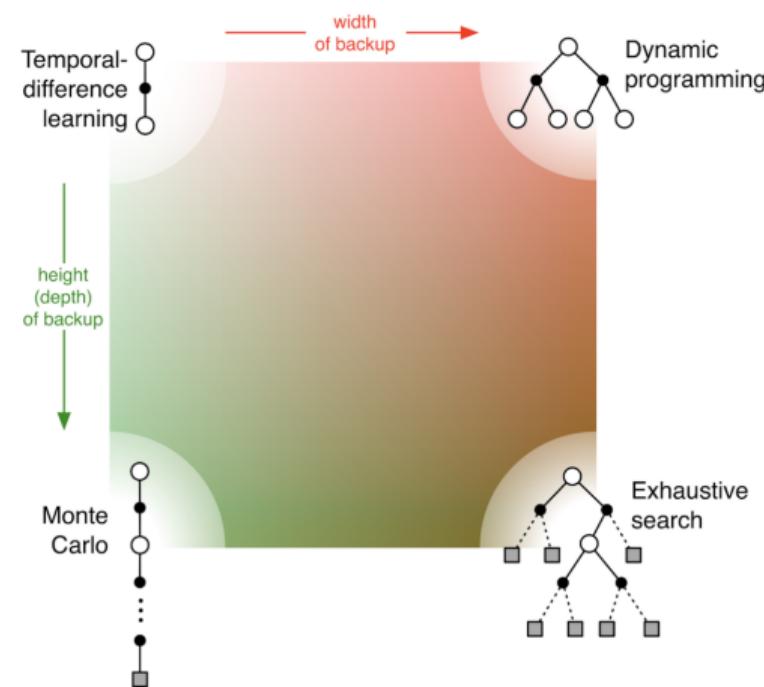
5 *n*-lépéses különbségek

6 Összefoglalás

# Időbeli különbségek

Az egyik központi ötlete a megerősítéses tanulásnak az időbeli különbségek (**TD**) tanító algoritmusa. A TD kombinálja a MC és DP algoritmusok ötleteit.

Csakúgy mint a MC, nyers tapasztalatokból tud tanulni, ezért **nincs szüksége a környezet dinamikájának modelljére**. És hasonlóan a DP algoritmusokhoz, az ügynöknek **nem szükséges teljes epizódokat lejátszania**.



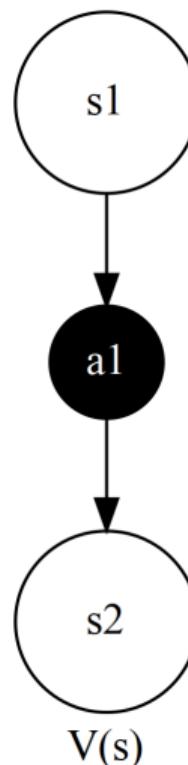
## TD becslés

Az ábrán az időbeli különbségek frissítési diagramja látható.

Az  $s_1$  állapothoz tartozó állapot-érték csak a következő állapot és a következő állapothoz tartozó jutalom alapján frissül.

A TD tanítás csak az egyes lépések által kapott jutalomból vesz mintát, és ez alapján számítja ki adott állapotonként a hozamok várható értékét. Az MC és TD becslés a **mintavezvő frissítések** algoritmusainak körébe tartoznak, mert ahhoz, hogy ki lehessen számolni, előre meg kell ismerni a következő állapothoz (vagy állapotokhoz) tartozó jutalmakat.

A DP algoritmusok esetén nem mintavétel történik, hanem egy teljes sokaság alapján egy várható érték számolódik ki.



# TD a megerősítéses tanulásban

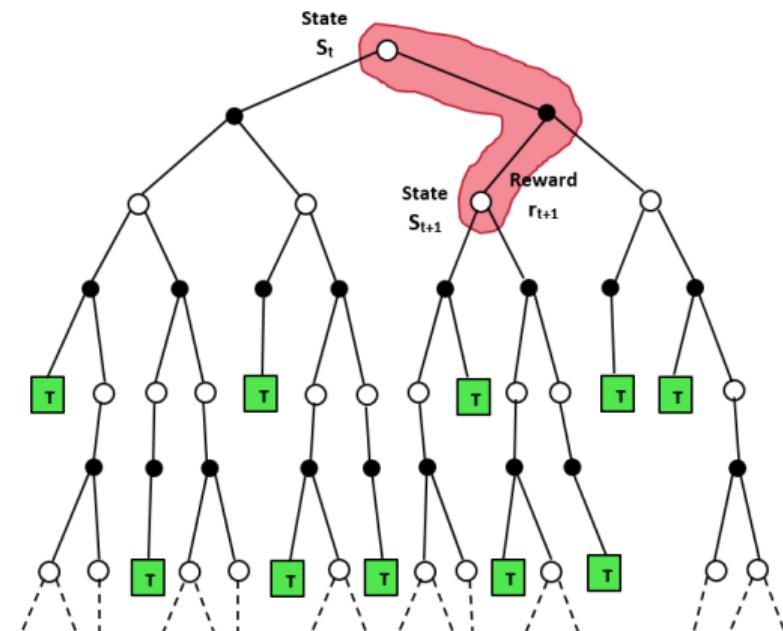
## TD állapot-érték frissítési szabály

$$V(s_t) \leftarrow V(s_t) + \alpha [r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$$

- $\alpha$ : Tanulási sebesség
- $r_{t+1}$ : Következő jutalom

A zárójelben lévő kifejezés a **TD hiba**, ami a  $s_t$  jelen állapot értéke és a jobb becslés,  $r_{t+1} + \gamma V(s_{t+1})$  közötti távolságot adja meg. A megerősítéses tanulásban számos helyen jelen van:

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$$



---

**Algoritmus 3:** TD algoritmus  $v_\pi$  megbecslésére

---

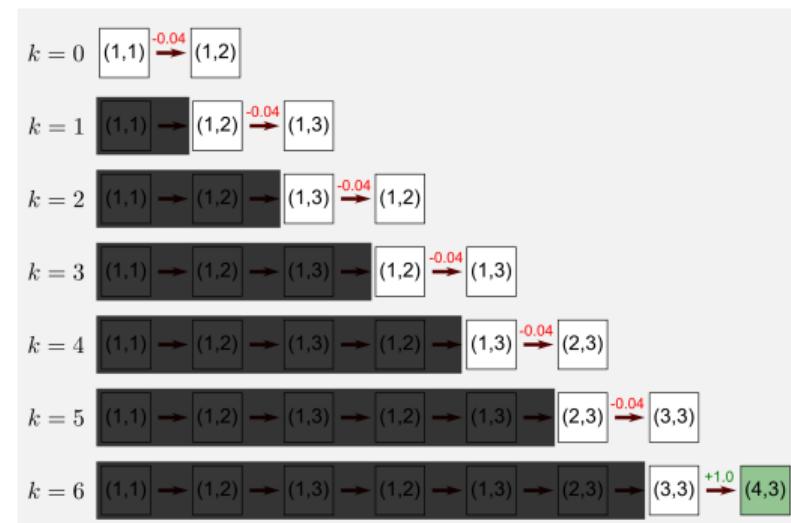
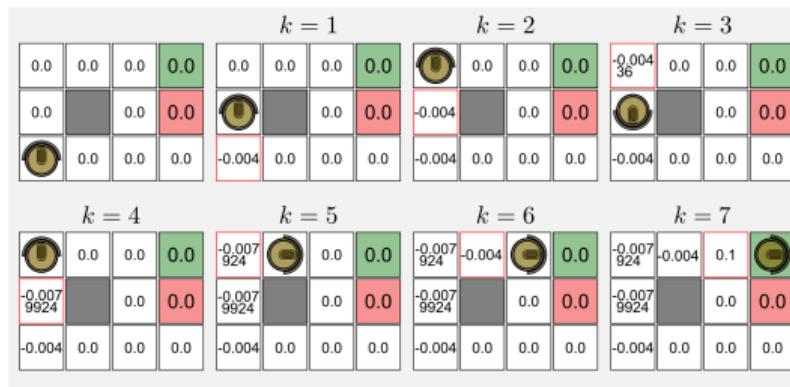
**Input:**  $\pi$  politika ami kiértékelésre kerül;  $\alpha \in [0, 1]$  tanulási sebesség

```
 $V(s) \leftarrow \text{random}()$ , minden  $s \in S$ -re; /* Állapot-értékek inicializálása */  
 $V(s_T) \leftarrow 0$ ; /* Terminális állapot értékének 0-ra állítása */  
for  $i = 0 \rightarrow \max_i$  do  
   $s \leftarrow s_0$ ; /* Kezdőállapot felvétele */  
  while  $s \neq s_T$  do  
     $a \leftarrow \pi(s)$ ; /* Cselekvés választás  $s$  állapotból  $\pi$  szerint */  
    /* Végrehajtása,  $r, s'$  megfigyelése;  
     $V(s) \leftarrow V(s) + \alpha [r + \gamma V(s') - V(s)]$ ;  
     $s \leftarrow s'$ ; /* Aktuális állapot frissítése */  
  end  
end
```

---

# Példa TD becslésre

A robot célja eljutni a bal alsó kockáról a zöld kockára jutalomért. Ha a piros kockára lép negatív jutalmat kap. Az ábrán egy epizód eredménye látható.



1 Ismétlés

2 Monte Carlo

3 Időbeli különbségek

4 SARSA

5 *n*-lépéses különbségek

6 Összefoglalás

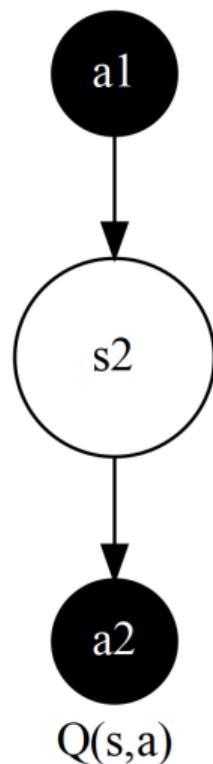
# Politikaalapú TD irányítás: SARSA

TD becsléssel lehetséges a  $v_\pi$  mellett a  $q_\pi$  értékfüggvény is. Ebben az esetben nem állapot-állapot átmeneteket hanem állapot-cselekvés állapot-cselekvés átmeneteket tanul az algoritmus. **Formálisan minden kettő Markov folyamat jutalmazási rendszerrel.**

## SARSA frissítési szabálya

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

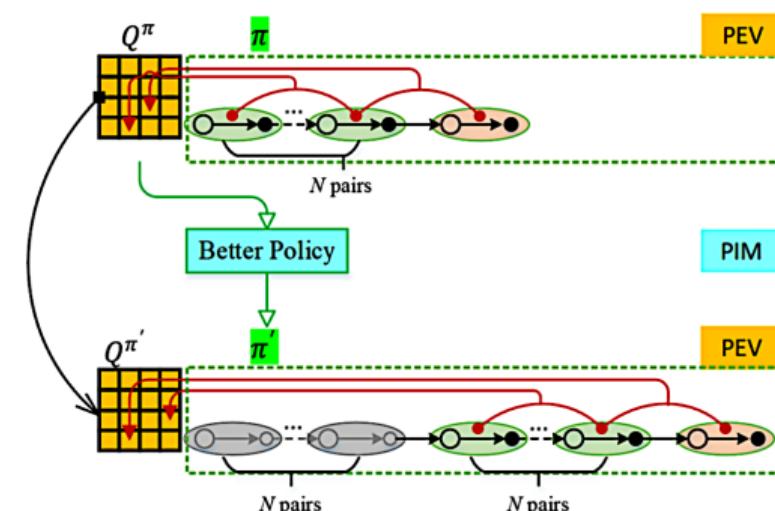
- $Q(s_t, a_t)$ :  $s$  állapot és  $a$  cselekvés minőség függvénye  $t$  időlépésben
- $Q(s_{t+1}, a_{t+1})$ : a következő állapot és cselekvés minőség függvénye  $t + 1$  időlépésben



# A SARSA folyamat modellje

A SARSA frissítési szabálya szerint a tanítási epizód minden lépése alkalmával **politika kiértékelés** fog végrehajtódni, amikor egy  $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$  sorozattal találkozik a modell. Ha gyakrabban mintavétel nagyobb a számításigény, viszont pontosabb is a becslés.

A **politika javítás** lépései között amikor az ügynök cselekvést választ gyakori az  $\epsilon$ -móhó stratégia alkalmazása, ezzel is ösztönözve a felfedezést és nem vélt magas jutalmak megszerzését.



# SARSA algoritmusa

---

**Algoritmus 4:** SARSA algoritmus  $Q \approx q_*$  megbecsülésére

---

**Input:**  $\alpha$ : tanulási sebesség;  $\varepsilon > 0$ : algoritmus hibahatára

$Q(s, a) \leftarrow \text{random}()$ , minden  $s \in S$ -re és  $a \in A$ -ra;

$Q(s_t, \cdot) \leftarrow 0$ ; /\* Terminális állapotot értéke 0 \*/

**for**  $i = 0 \rightarrow \max_i$  **do**

$s \leftarrow s_0$ ; /\* Kezdőállapot felvétele \*/

$a \leftarrow \pi(s)$ ; /\* Cselekvés választása  $s$  állapotból  $\pi$  szerint \*/

**while**  $s \neq s_T$  **do**

a végrehajtása,  $s'$ ,  $r$  megfigyelése;

$a' \leftarrow \pi(s')$ ; /\* Cselekvés választása  $s'$  állapotból  $\pi$  szerint \*/

$Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma Q(s', a') - Q(s, a)]$ ;

$s \leftarrow s'$ ,  $a \leftarrow a'$ ; /\* Jelenlegi állapot és cselekvés frissítése \*/

**end**

**end**

---

## 1 Ismétlés

## 2 Monte Carlo

## 3 Időbeli különbségek

## 4 SARSA

## 5 *n*-lépéses különbségek

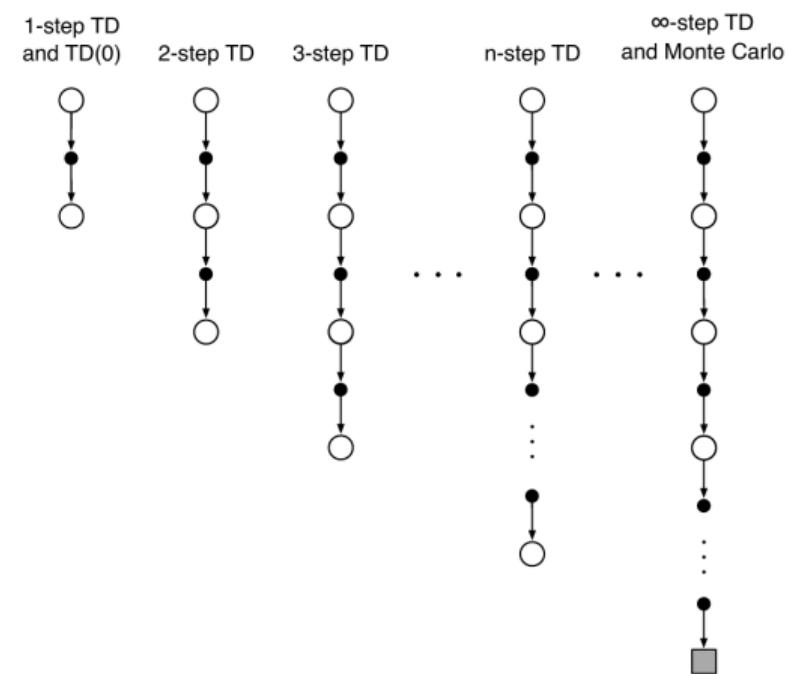
## 6 Összefoglalás

# MC és TD egyesítése

A MC eljárások az aktuális állapottól egészen **az epizód végéig** megfigyelt állapotok alapján számolják ki az új  $v_\pi(s)$  értéket.

A TD módszer pedig **csak a következő állapotig** kapott jutalom alapján frissíti az értékfüggvényt.

A köztes módszer, hogy az algoritmus  $n$  köztes állapotból származó jutalmat vegye figyelembe. Például ha  $n = 2$  akkor következő 2 állapot és a hozzájuk tartozó jutalom ami számít.

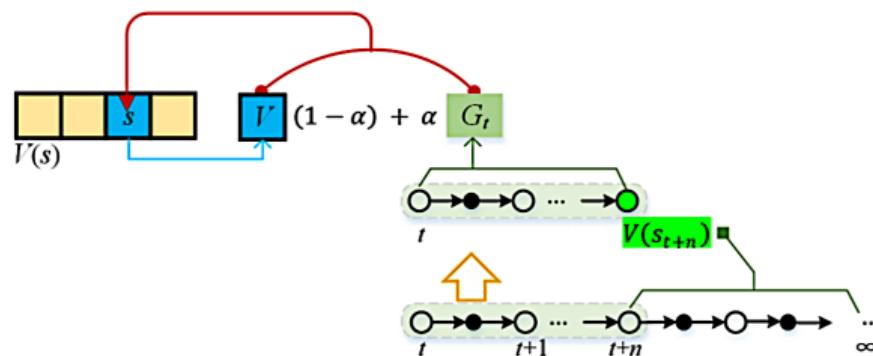


TD( $n$ ) értékfüggvényei

$n$  lépéses diszkontált hozam

$$G_{t:t+n} = r_{t+1} + \gamma r_{t+2} + \dots + \gamma^{n-1} r_{t+n} + \gamma^n V_{t+n-1}(s_{t+n})$$

A hozam végén az utolsó,  $t + n - 1$ -edik állapot állandó értéke szerepel, ezzel kompenzáálva az összes,  $n$ -edik állapot után szereplő állapot várható hozamáért.

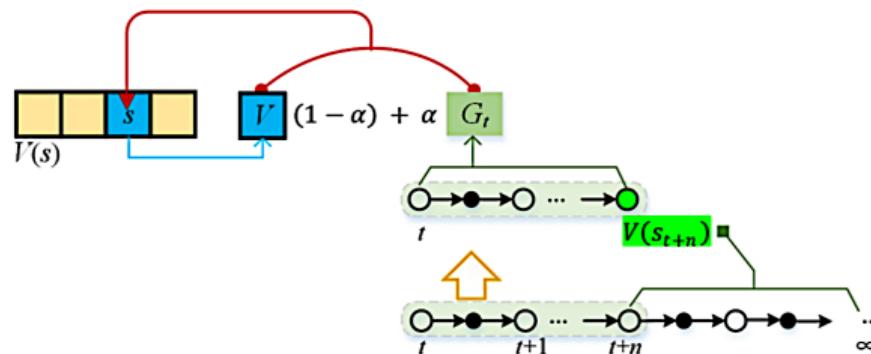


TD( $n$ ) értékfüggvényei

$n$  lépéses állapot-érték frissítési szabály

$$V_{t:t+n} = V_{t+n-1}(s_t) + \alpha [G_{t:t+n} - V_{t+n-1}(s_t)]$$

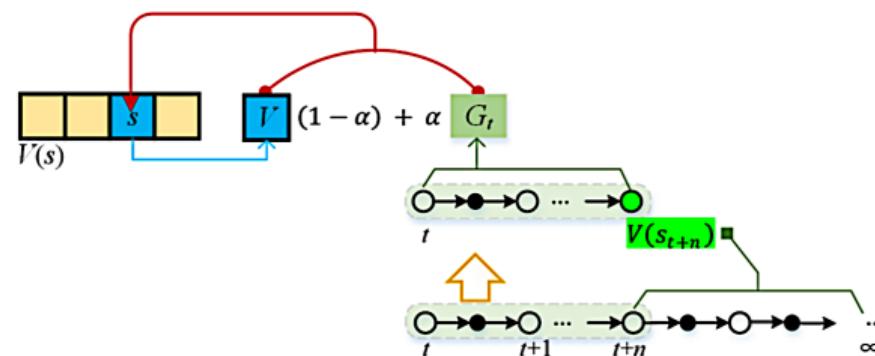
Megegyezik az eddig vizsgált állapot-érték függvényivel kivéve, hogy az  $n$  lépéses diszkontált hozamot használja a teljes hozam helyett.



TD( $n$ ) értékfüggvényei $n$  lépéses SARSA

$$Q_{t+n}(s_t, a_t) = Q_{t+n-1}(s_t, a_t) + \alpha [G_{t:t+n} - Q_{t+n-1}(s_t, a_t)]$$

Megegyezik az eddig vizsgált állapot-cselekvés minőség függvényel kivéve, hogy az  $n$  lépéses hozamot használja a teljes hozam helyett.



## 1 Ismétlés

## 2 Monte Carlo

## 3 Időbeli különbségek

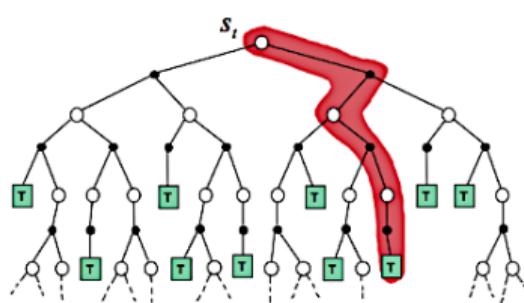
## 4 SARSA

## 5 *n*-lépéses különbségek

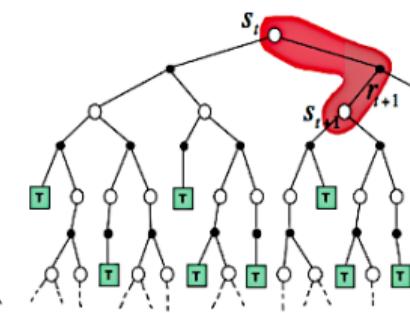
## 6 Összefoglalás

## Keresés mélysége

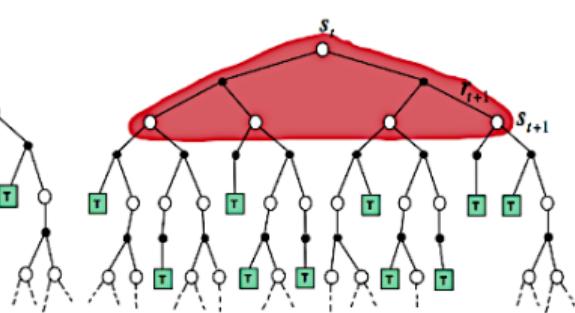
Monte Carlo



Időbeli különbségek



Dinamikus programozás



# Frissítési szabályok

## Dinamikus programozás

$$V(s_t) \leftarrow E_{\pi} [r_{t+1} + \gamma V(s_{t+1})]$$

## Monte Carlo

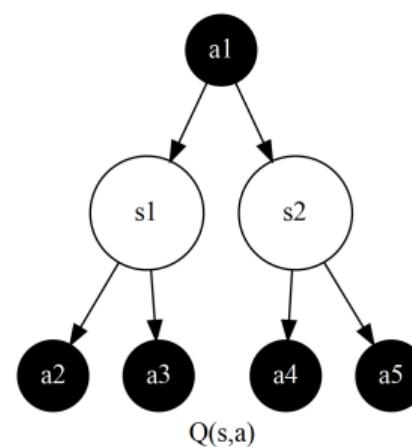
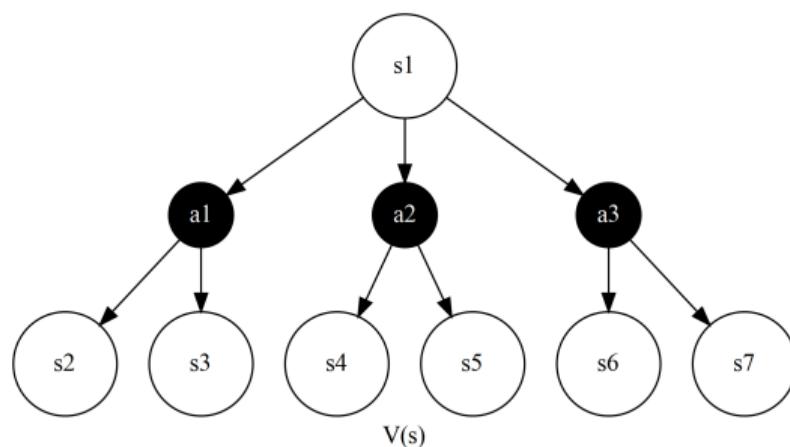
$$V(s_t) \leftarrow V(s_t) + \alpha [G_t - V(s_t)]$$

## Időbeli különbségek

$$V(s_t) \leftarrow V(s_t) + \alpha [r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$$

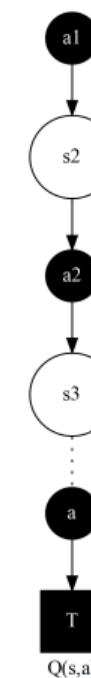
## Frissítések mélysége függvény szerint

## Dinamikus programozás



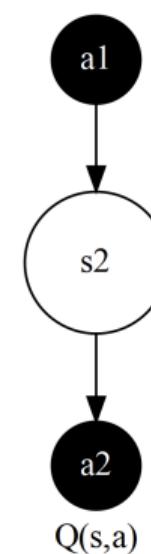
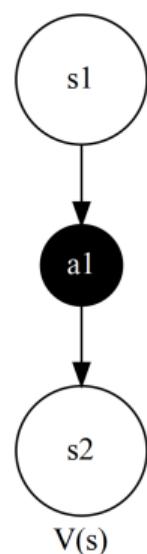
## Frissítések mélysége függvény szerint

## Monte Carlo



## Frissítések mélysége függvény szerint

## Időbeli különbségek



## Frissítések mélysége függvény szerint

 $n$  lépéses különbségek