Üzleti Intelligencia

4. Előadás: Monte Carlo és temporális különbségek

Kuknyó Dániel
Budapesti Gazdasági Egyetem

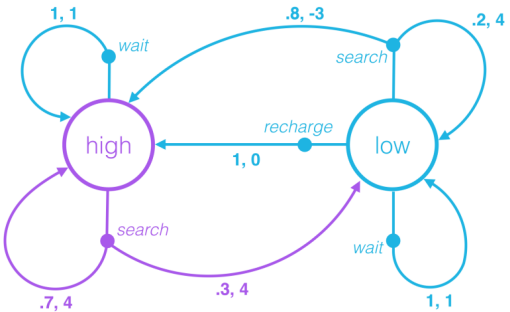
2023/24
1.félév

Ügynök-környezet interakció

Markov döntési folyamat

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

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



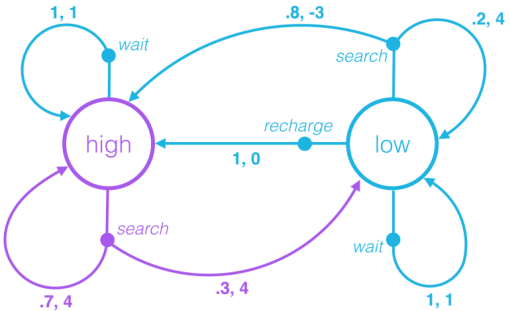
Ügynök-környezet interakció

Az MDP folyamata:

1. Az ügynök s_0 állapotból indul
2. Az ügynök π 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:

$$E_\pi \left(r_1 + \gamma r_2 + \gamma^2 r_3 + ...\right) \rightarrow max$$



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, cselekvésekből 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

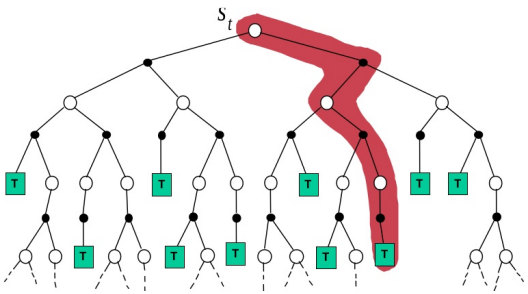
A MC tanítási algoritmus a politika iteráció egy általánosított változata. A megerősítéses tanulásban a MC algoritmusok a mélységi bejárásnak felelnek meg.

MC állapot-érték frissítési szabály

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

- α: 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.

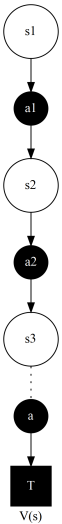
Az **állapot-érték függvény** megadja, mennyi a várható hozam, ha az ügynök adott s állapotban áll, és onnan π politikát követi:

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

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.

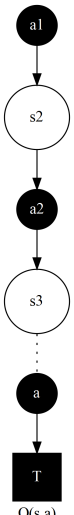
Az **állapot-cselekvés minőség függvény** megadja, mennyi a várható hozam, ha az ügynök adott s állapotban áll, a cselekvést végrehajtja, majd onnan π politikát követi:

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

Ahol

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

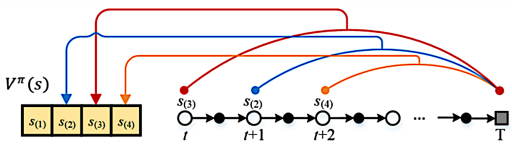
t aktuális időlépéstől a T terminális állapotba vezető időlépésig az ügynök által összegyűjtött diszkontált kumulált hozam.



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 π politika szerint cselekszik.

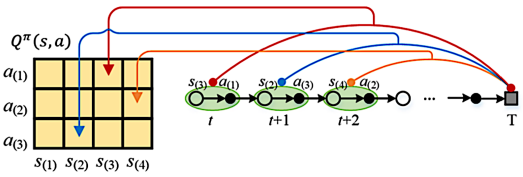
Az értékfüggvény **diszkrét** adatszerkezetben tartja számon a hozamok várható értékét, ezért diszkrét állapotokat és cselekvéseket feltételez.



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 π politika szerint cselekszik.

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

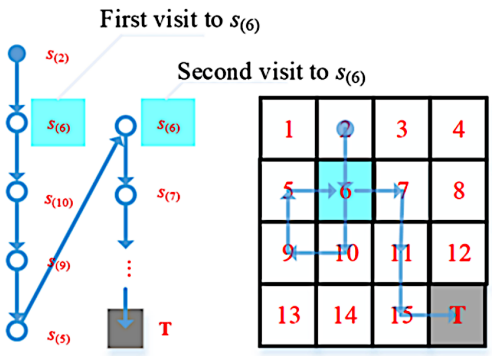


Látogatás alapú Monte Carlo becslés

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 kezelni 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 ∞-hez.



Algoritmus 1: Első látogatás alapú Monte Carlo algoritmus v_π becslésére

Input: A kiértékelendő π politika

$V(s) \leftarrow random()$, minden $s \in S$-re;
$Hozam(s) \leftarrow$ **üres lista**, minden $s \in S$-re;
for $i = 0 \rightarrow max_i$ **do**
    Epizód lejátszása π politikával: $s_0, a_0, r_1, s_1, a_1, r_2, ..., s_{T-1}, a_{T-1}, r_T$;
    $G \leftarrow 0$; /* 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*** $s_0, s_1, ..., s_{t-1}$ **then**
            G hozzáfűzése $Hozam(s_t)$-hez; /* Állapothoz tartozó hozam */
            $V(s_t) \leftarrow Avg\{Hozam(s_t)\}$; /* Hozam várható értéke */
        end
    end
end

Algoritmus 2: Monte Carlo becslés az π_* megkeresésére

$\pi(s) \in A(s)$ 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 π politikával: $s_0, a_0, r_1, s_1, a_1, r_2, ..., s_{T-1}, a_{T-1}, r_T$;
  $G \leftarrow 0$; `/* 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, a_t) ***not in*** $s_0, s_1, ...s_{t-1}$ **then**
      G hozzáfűzése $Hozam(s_t, a_t)$-hez; `/* Állapot-cselekvés pár hozama */`
      $Q(s_t, a_t) \leftarrow Avg\{Hozam(s_t, a_t)\}$; `/* Hozam várható értéke */`
      $\pi(s_t) \leftarrow \underset{a}{argmax} Q(s_t, a)$; `/* Politika frissítése */`
    end
  end
end

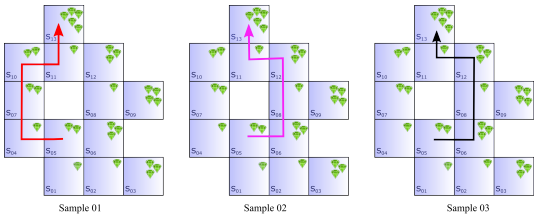
Példa Monte Carlo érték becslésre

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

$$Hozam(01) = \underset{s05}{2} + \underset{s04}{1} + \underset{s07}{2} + \underset{s10}{2} + \underset{s11}{1} + \underset{s13}{5} = 13$$

$$Hozam(02) = \underset{s05}{2} + \underset{s06}{3} + \underset{s08}{1} + \underset{s12}{3} + \underset{s11}{1} + \underset{s13}{5} = 15$$

$$Hozam(03) = \underset{s05}{2} + \underset{s06}{3} + \underset{s08}{1} + \underset{s12}{3} + \underset{s11}{1} + \underset{s13}{5} = 15$$

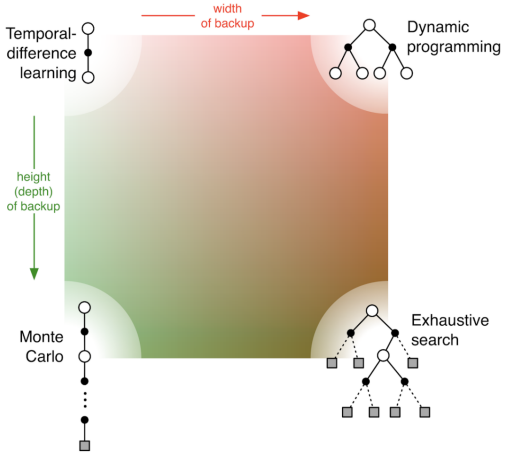


3 Samples starting from State S_{05}

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**.

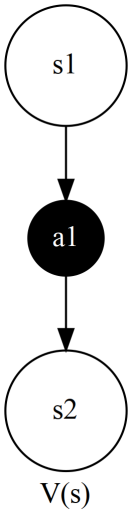


TD becslés

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 **mintavevő 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

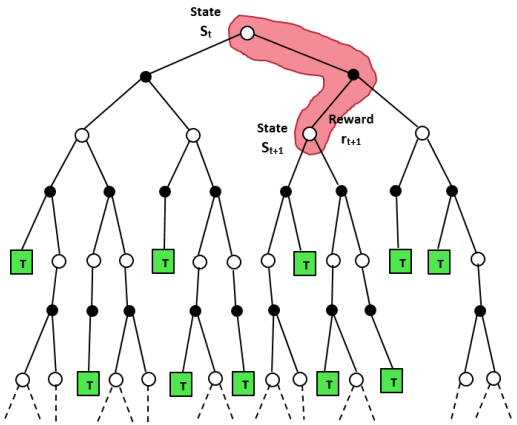
TD állapot-érték frissítési szabály

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

- α: 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:

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



Algoritmus 3: TD algoritmus v_π megbecslésére

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

$V(s) \leftarrow random()$, minden $s \in S$-re; `/* Állapot-értékek inícializálása */`
$V(s_T) \leftarrow 0$; `/* 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` π `szerint */`
    a végrehajtása, r, s megfigyelése;
    $V(s) \leftarrow V(s) + \alpha\left[r + \gamma V(s') - V(s)\right]$;
    $s \leftarrow s'$; `/* Aktuális állapot frissítése */`
  end
end

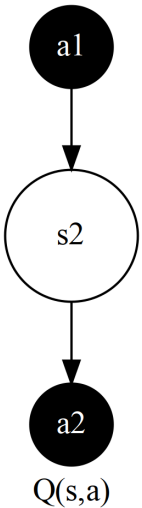
Politikaalapú TD irányítás: SARSA

TD becsléssel lehetséges a v_π mellett a q_π értékfüggvény is. Ebben az esetben nem állapot-állapot átmeneteket hanem állapot-cselekvés állapot-cselekvés átmeneteket tanul az algoritmus. **Formálisan mindkettő Markov folyamat** jutalmazási rendszerrel.

SARSA frissítési szabálya

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

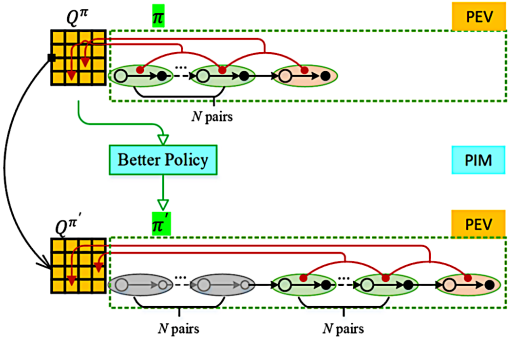
- $Q(s_t, a_t)$: s állapot és a cselekvés minőség függvénye t időlépésben
- $Q(s_{t+1}, a_{t+1})$: a következő állapot és cselekvés minőség függvénye $t+1$ időlépésben



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 gyakoribb a mintavétel nagyobb a számításigény, viszont pontosabb is a becslés.

A **politika javítás** lépésében amikor az ügynök cselekvést választ gyakori az ε-mohó stratégia alkalmazása, ezzel is ösztönözve a felfedezést és nem vélt magas jutalmak megszerzését.



SARSA algoritmusa

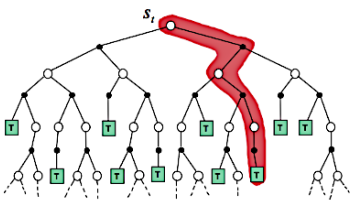
Algoritmus 4: SARSA algoritmus $Q \approx q_*$ megbecslésére

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

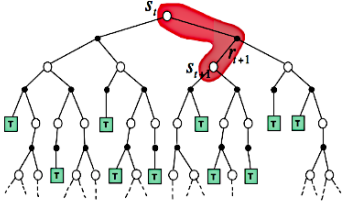
$Q(s, a) \leftarrow random()$, minden $s \in S$-re és $a \in A$-ra;
$Q(s_t, \cdot) \leftarrow 0$; /* Terminális állapot értéke 0 */
for $i = 0 \rightarrow max_i$ **do**
  $s \leftarrow s_0$; /* Kezdőállapot felvétele */
  $a \leftarrow \pi(s)$; /* Cselekvés választása s állapotból π szerint */
  while $s \neq s_T$ **do**
    a végrehajtása, s', r megfigyelése;
    $a' \leftarrow \pi(s')$; /* Cselekvés választása s' állapotból π szerint */
    $Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma Q(s', a') - Q(s, a) \right]$;
    $s \leftarrow s'$, $a \leftarrow a'$; /* Jelenlegi állapot és cselekvés frissítése */
  end
end

Keresés mélysége

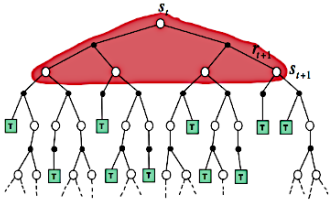
Monte Carlo



Időbeli különbségek



Dinamikus programozás



Frissítési szabályok

Dinamikus programozás

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

Monte Carlo

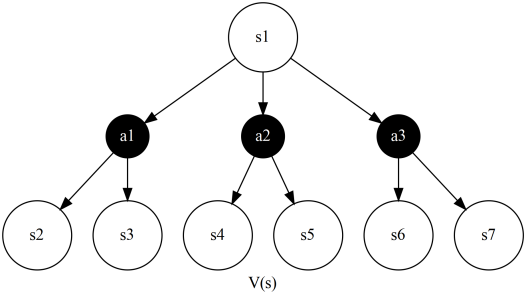
$$V(s_t) \leftarrow V(s_t) + \alpha \left[G_t - V(s_t) \right]$$

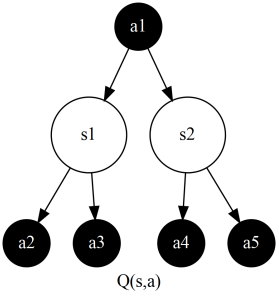
Időbeli különbségek

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

Frissítések mélysége függvény szerint

Dinamikus programozás

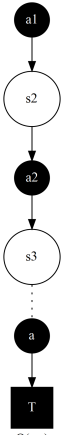




Frissítések mélysége függvény szerint

Monte Carlo

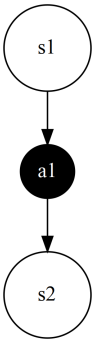
V(s)



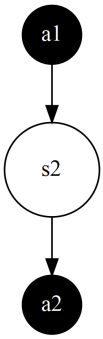
Q(s,a)

Frissítések mélysége függvény szerint

Időbeli különbségek



V(s)



Q(s,a)