Üzleti Intelligencia

3. Előadás: Markov döntési folyamatok megoldása

Kuknyó Dániel
Budapesti Gazdasági Egyetem

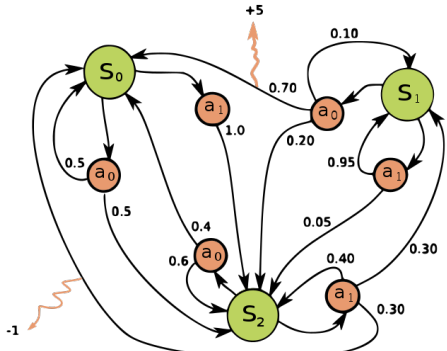
2023/24
1.félév

Az RL modellje

Markov döntési folyamat

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

- 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 halmaza
- s_0: kezdőállapot
- γ: diszkont faktor



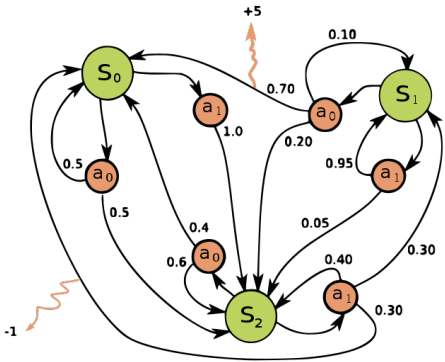
Az RL modellje

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:

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



A mohó ügynök

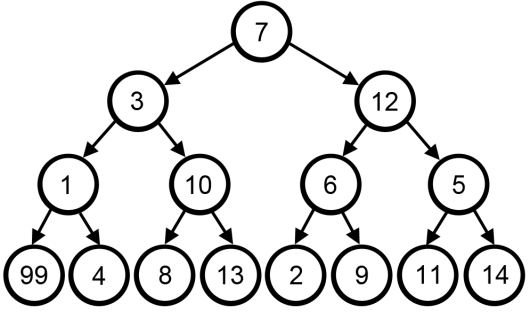
A legegyszerűbb cselekvés kiválasztási szabály, ha az ügynök mindig azt a cselekvést választja, ami számára a lehető legnagyobb várható hozammal rendelkezik.

Mohó cselekvés választás

Mohó politika mindig azt a cselekvést fogja választani, amelyik - egy lépéses távlatban - a lehető legnagyobb várható jutalommal fog járni az ügynök számára v_π szerint.

$$A_t = \underset{a}{argmax}\, Q_t(a)$$

- Mi lenne a mohó politika ebben az estben?
- Mindig ez a legjobb megoldás?
- A legjobb megoldás mindig mohó?



A mohó ügynök

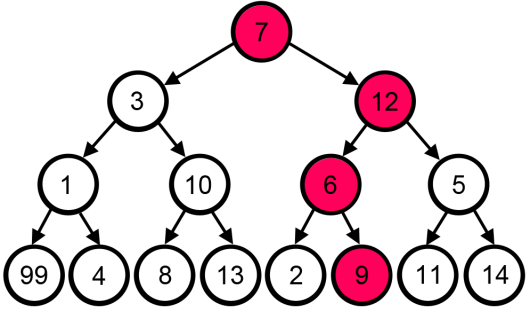
A legegyszerűbb cselekvés kiválasztási szabály, ha az ügynök mindig azt a cselekvést választja, ami számára a lehető legnagyobb várható hozammal rendelkezik.

Mohó cselekvés választás

Mohó politika mindig azt a cselekvést fogja választani, amelyik - egy lépéses távlatban - a lehető legnagyobb várható jutalommal fog járni az ügynök számára v_π szerint.

$$A_t = \underset{a}{argmax}\, Q_t(a)$$

- Mi lenne a mohó politika ebben az estben?
- Mindig ez a legjobb megoldás?
- A legjobb megoldás mindig mohó?



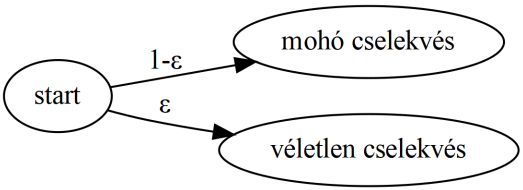
Az ε-mohó stratégia

Egy másik lehetőség, ha adott valószínűséggel az ügynök véletlen cselekvést hajt végre remélve, hogy ezzel elér egy olyan állapotba amelyhez nagy jutalom tartozik. A véletlen cselekvés a **felfedezés**, és végrehajtásának valószínűsége ϵ.

ε-mohó cselekvés választás

$$A_t \leftarrow \begin{cases} \underset{a}{argmax} Q(a) & P{=}1{-}\varepsilon \\ a{\sim}A & P{=}\varepsilon \end{cases}$$

Az ügynök tehát ε valószínűséggel véletlen cselekvést választ az ismeretlen, de nagyobb jutalom reményében. Ez a **felfedezés** művelete.
ε valószínűséggel pedig a már ismert és a legnagyobb várható jutalommal járó cselekvést hajtja végre. Ez a **kizsákmányolás** művelete.



Példák

A következő valós példák alkalmasak a felfedezés/kizsákmányolás dilemma bemutatására:

- Étterem választás:
  - **Kizsákmányolás**: elmész a kedvenc éttermedbe.
  - **Felfedezés**: elmész egy új étterembe, hátha találsz egy jobbat mint a kedvenced.
- Online hirdetés:
  - **Kizsákmányolás**: a legjobb reklám megmutatása a felhasználónak.
  - **Felfedezés**: egy új reklám megmutatása a felhasználónak, hátha tetszik neki.
- Olajfúrás:
  - **Kizsákmányolás**: Egy meglévő helyen fúrás az olajért.
  - **Felfedezés**: Egy új helyen fúrás.
- Klinikai kezelés:
  - **Kizsákmányolás**: A bevált kezelés alkalmazása.
  - **Felfedezés**: Új kezelés kipróbálása.

A rabló probléma

A k-karú rabló problémája egy elméleti megerősítéses tanulás probléma. A játékos egy rablógépen játszik, amelynek k karja van.
Minden karhúzás után egy állandó eloszlásból választott jutalmat kap az ügynök. Az ügynök célja, hogy olyan politikát válasszon, ami az elvárt hozamot maximalizálja 1000 cselekvés vagy *időlépés* után.

A rabló probléma

Az ügynöknek számon kell tartania, mennyi a jutalom várható értéke, ha adott egy a cselekvés. Ez a $Q(s, a)$ állapot-cselekvés minőség függvény. A rabló problémában csak egy állapot van, ezért elég csak a cselekvésekhez tartozóan számon tartani:

$$q_*(a) = E\left[r_t | A_t = a\right]$$

A jutalom várható értéke:

$$Q_n = \frac{r_1 + r_2 + ... + r_{n-1}}{n - 1}$$

Algoritmus a rabló megoldására

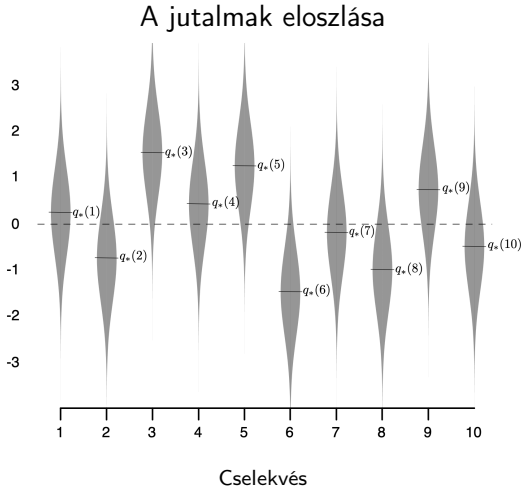
Algorithm 1: Egyszerű rabló algoritmus

$Q(a) \leftarrow 0 \; for \; a = 1 \rightarrow k;$ /* Cselekvés minőségének függvénye */
$N(a) \leftarrow 0 \; for \; a = 1 \rightarrow k;$ /* Kar meghúzásainak a száma */
for $t = 1 \rightarrow max_t$ **do**
  $p = random(0, 1);$ /* Véletlen szám 0 és 1 között */
  if $p > \varepsilon$ **then**
    $a \leftarrow \underset{a}{argmax} Q(a);$ /* Legnagyobb ismert jutalom cselekvése */
  else
    $a \leftarrow a \sim A;$ /* Véletlen cselekvés */
  end
  $r \leftarrow env(a);$ /* Cselekvés végrehajtása a környezetben */
  $N(a) \leftarrow N(a) + 1;$ /* Cselekvés számlálójának növelése */
  $Q(a) \leftarrow Q(a) + \frac{1}{N(a)} [r - Q(a)];$ /* Q-érték frissítése */
end

Egy példa rabló

Hogy meg lehessen mérni a mohó és ε-mohó stratégiák teljesítményét, szükség van egy teszt rablóra. A példában szereplő egy 10-karú rabló. Minden karhoz tartozóan a jutalmak eloszlása Gauss-i eloszlást követ 1 varianciával, viszont nem 0 átlaggal.

Valamelyik karok nagyobb valószínűséggel járnak magas jutalommal mint a többi. Az ügynök feladata megtalálni melyik kartól remélhet nagyobb jutalmat. Ehhez szükség van arra, hogy végig próbálja őket.

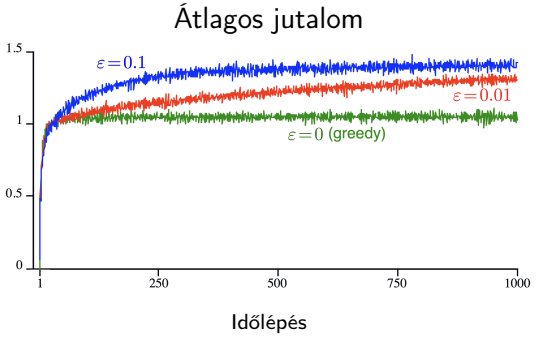


A futás teljesítménye

Az algoritmus 1000 időlépésen keresztül futott $\varepsilon = 0, \varepsilon = 0.01, \varepsilon = 0.001$ hiperparaméterekkel. Minél nagyobb a ε érték, annál nagyobb a felfedezés valószínűsége.

Mindegyik módszer megbecsülte az állapot-cselekvés minőség függvényt a rabló minden karára a mozgóátlagolás technikájával. A diagramon a várható jutalom mértékét mutatja az időlépések függvényében.

A mohó stratégia kezdetben gyorsabban javult mint a többi, de kisebb értékre konvergált a futásidő végére.

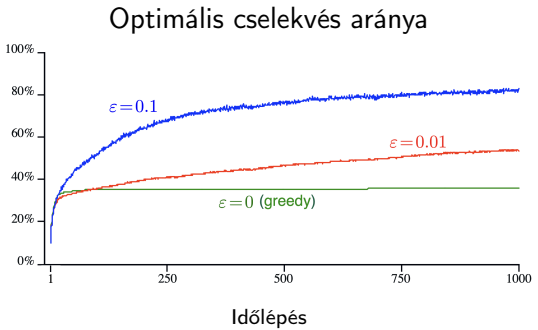


A futás teljesítménye

Az ábra azt mutatja, hogy a mohó módszer csak a feladatok mintegy 30%-ában találta meg az optimális műveletet. Az ε-mohó módszerek végül jobban teljesítettek, mert folytatták a felfedezést és javították az esélyüket az optimális művelet felismerésére.

Az $\varepsilon = 0.1$ módszer többet fedezett fel, és általában korábban megtalálta az optimális műveletet, de soha nem választotta ki azt több mint 91%-ban.

Az $\varepsilon = 0.01$ módszer lassabban javult, de végül mindkét teljesítménymérőn jobban teljesített.



Dinamikus programozás alapjai

A dinamikus programozás egy gyűjtőfogalom olyan algoritmusokra amelyekkel kiszámolható az optimális politika *ha adott egy tökéletes környezeti modell egy Markov döntési folyamatként*.

A klasszikus dinamikus programozási algoritmusok ritkák a megerősítéses tanulásban mert egy tökéletes környezeti modellt feltételeznek és mert rendkívül erőforrásigényesek.

Dinamikus programozás

A DP algoritmusok a komplex problémákat alproblémákra bontják, majd a végső megoldást az alproblémák megoldásaiból állítják elő. Ehhez két tulajdonságnak kell érvényesnek lennie:

- **Optimális alstruktúra**: Az almegoldásoknak felhasználhatóknak kell lenniük a probléma megoldására.
- **Átfedésben lévő alproblémák**: Bizonyos alproblémák megoldásait többször is fel lehet használni hasonló feladatok elvégzéséhez.