

Üzleti Elemzések Módszertana

11. Előadás: Megerősítéses tanulás

Kuknyó Dániel
Budapesti Gazdasági Egyetem

2023/24
2.félév

1 Bevezetés

2 Politika javítása

3 *Q*-tanulás

1 Bevezetés

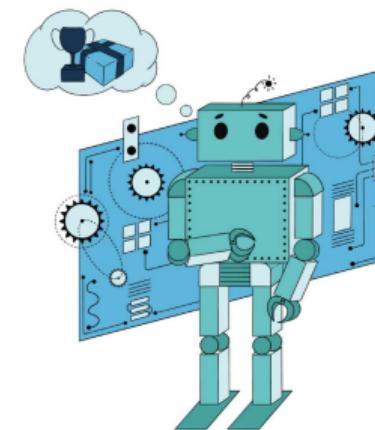
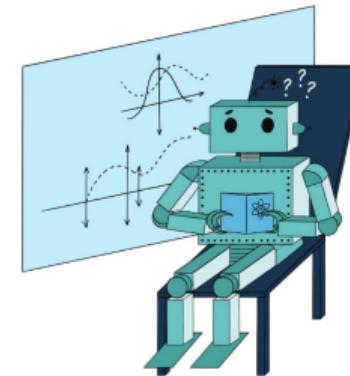
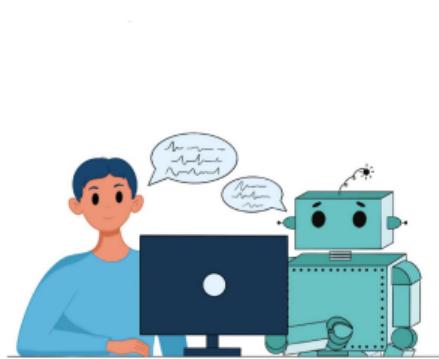
2 Politika javítása

3 Q-tanulás

A gépi tanulás fajtái

A gépi tanulás 3 fő irányzata:

- Felügyelt tanulás
- Felügyelet nélküli tanulás
- Megerősítéses tanulás



Mikor alkalmazható a megerősítéses tanulás?

Az RL olyan problémák esetén használatos, ahol az **algoritmikus vagy hagyományos ML hozzállás nem bizonyul megfelelőnek**, mert nem lehetséges tanító adatot gyűjteni vagy generálni.

Például:

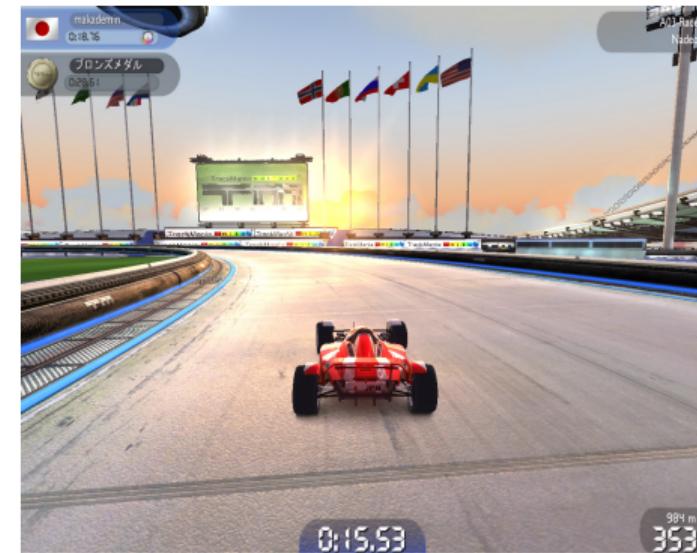
- Robotok
- Autonóm vezetés
- Számítógépes játékok



Felügyelt vagy megerősítéses tanulás?

Adott például egy autóversenyző program. Ha felügyelt tanítás a választott hozzáállás, szükség van egy adatbázisra, amely jellemzi az összes szituációt, és minden szituációhoz tartozóan az elvárt output értéket.

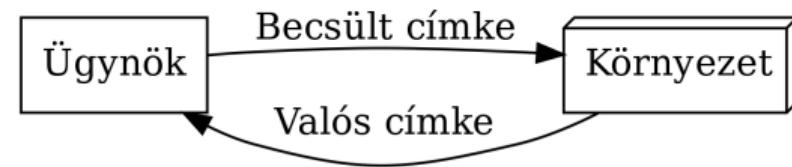
A szituációknak le kell írnia a kocsi helyzetét, a környezet állapotát, a versenytársak helyzetét. Az elvárt outputnak olyan halmazból kell kikerülnie, mint gáz, jobb, bal, fék és ezek kombinációi.



Visszajelzések a megerősítéses tanulásban

A két szemléletmód abban különbözik,
hogy a felügyelő milyen visszajelzéseket ad
a tanulónak.

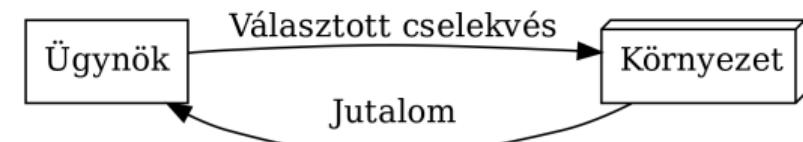
**A felügyelt tanulásban teljes
visszajelzésekéről van szó, mert a válasz
önmagában a megoldás.**



Visszajelzések a megerősítéses tanulásban

A két szemléletmód abban különbözik,
hogy a felügyelő milyen visszajelzéseket ad
a tanulónak.

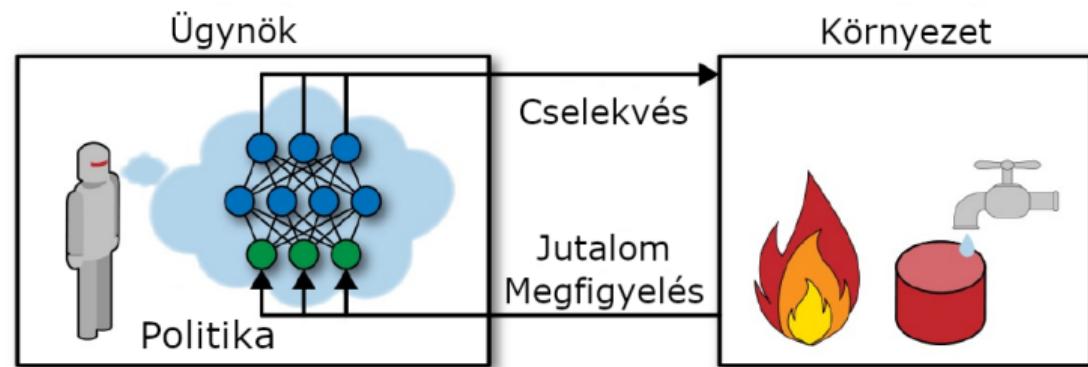
**A megerősítéses tanulásban viszont
csak részlegesek a visszajelzések. A
felügyelő válasza mindenig csak a
megoldás irányába vezet, nem
önmagában a teljes jó megoldás.**



A megerősítéses tanulás komponensei

Ügynök

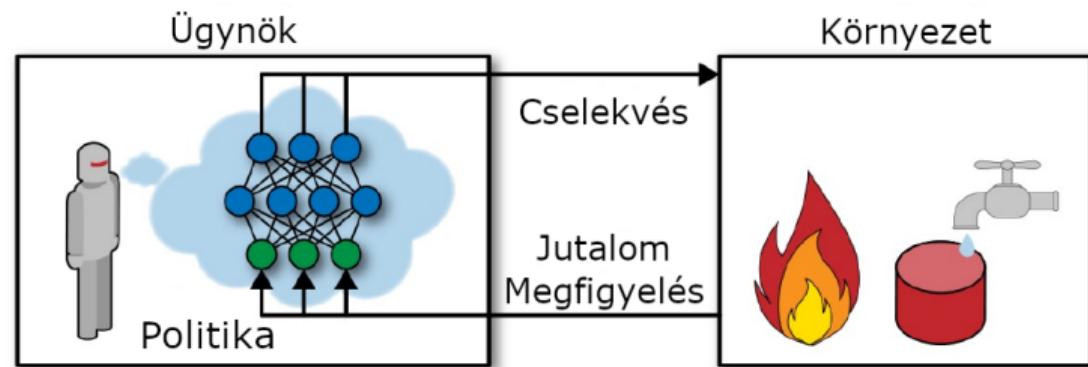
Az autonóm cselekvő, ami a feladat végrehajtására törekszik.



A megerősítéses tanulás komponensei

Környezet

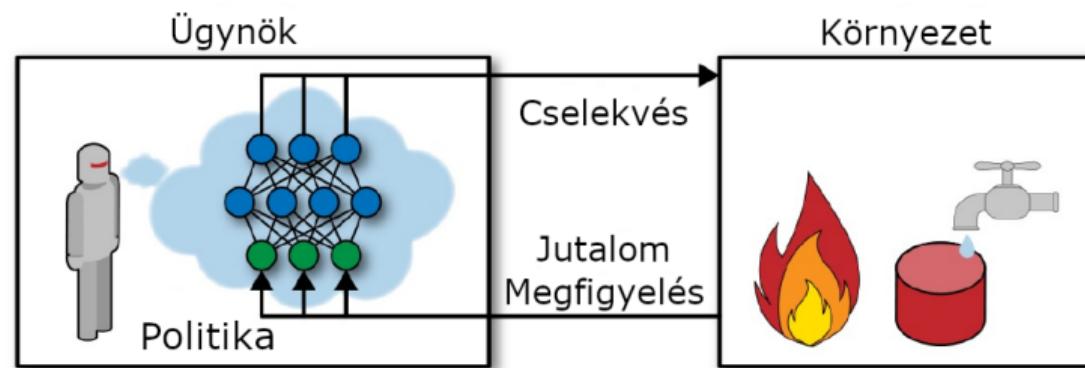
Egy fekete doboz, amely az ügynök cselekvéseinek helyszíne.



A megerősítéses tanulás komponensei

Idő

RL folyamán az időlépések diszkrétek:
 $t \in 1, 2, 3, \dots$



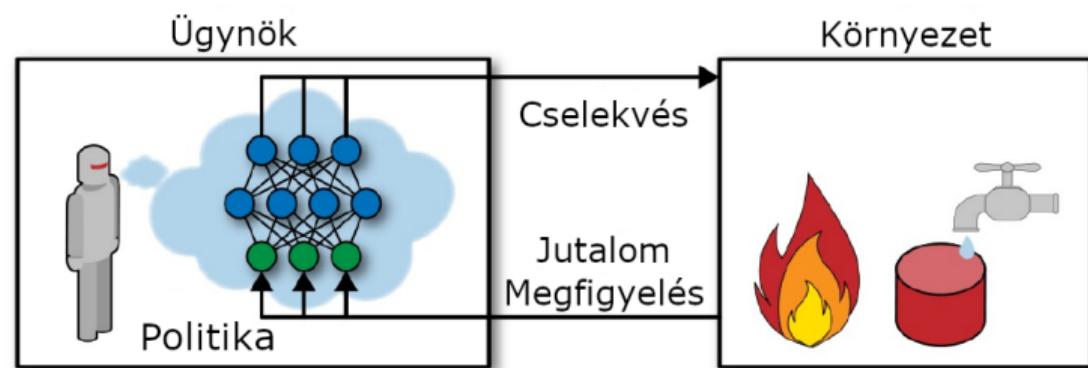
A megerősítéses tanulás komponensei

Állapot

Az ügynök megfigyelése a környezetre vonatkozóan.

A környezetet leíró változók összessége.

Jelölés: $s \in S$, ahol S az összes állapot halmaza.

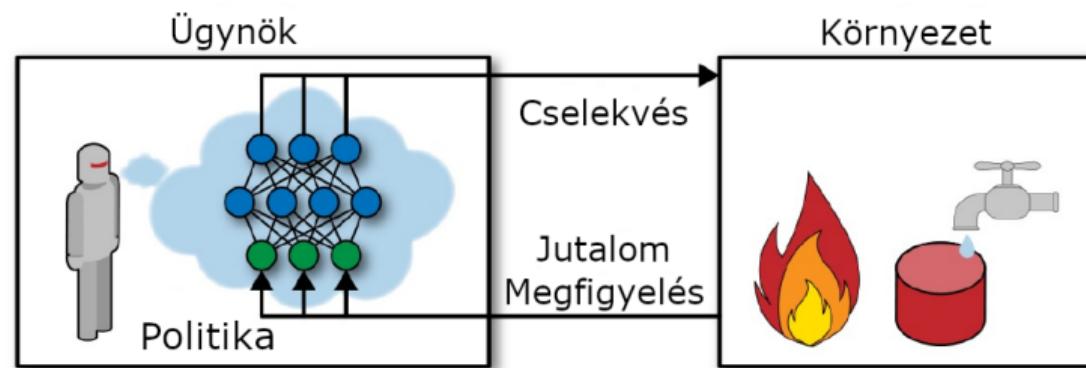


A megerősítéses tanulás komponensei

Jutalom

Az ügynök cselekvésének
jóságát jelző skalár.

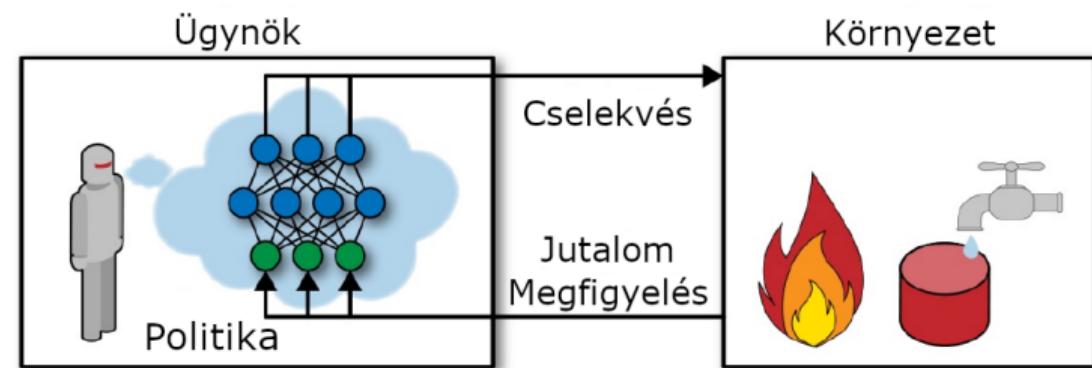
Jelölés: $r \in \mathbb{R}$



A megerősítéses tanulás komponensei

Cselekvés

Az ügynök által végrehajtott művelet, ami a környezetet befolyásolja. Jelölés: $a \in A$, ahol A az összes cselekvés halmaza.



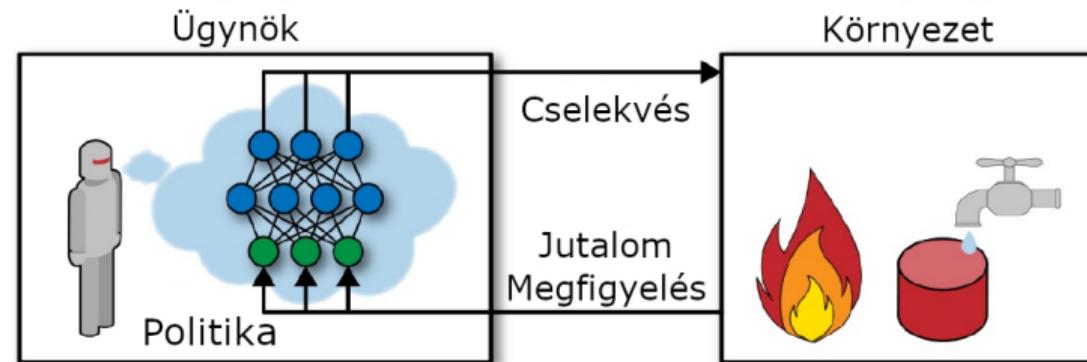
A megerősítéses tanulás komponensei

Politika

Egy állapot \rightarrow cselekvés leképezés. Az ügynök cselekvéseinek szabályait adja meg.

Jelölés:

- Determinisztikus:
 $\pi \in S \rightarrow A$
- Sztochasztikus:
 $\pi \in S \times A \rightarrow [0, 1]$
Röviden: $\pi(s, a)$
Vagy: $\pi(a|s)$



Az ügynök

A megerősítéses tanulásban egy ügynök (**cselekvő**) megfigyeli a környezetet, és ezalapjából cselekvéseket tesz a környezetben. A cselekvéseiért és a környezet változásáért jutalmat kap.

Az ügynök célja, hogy a jutalmakat hosszú távon maximalizálja. Az ügynök lehet:

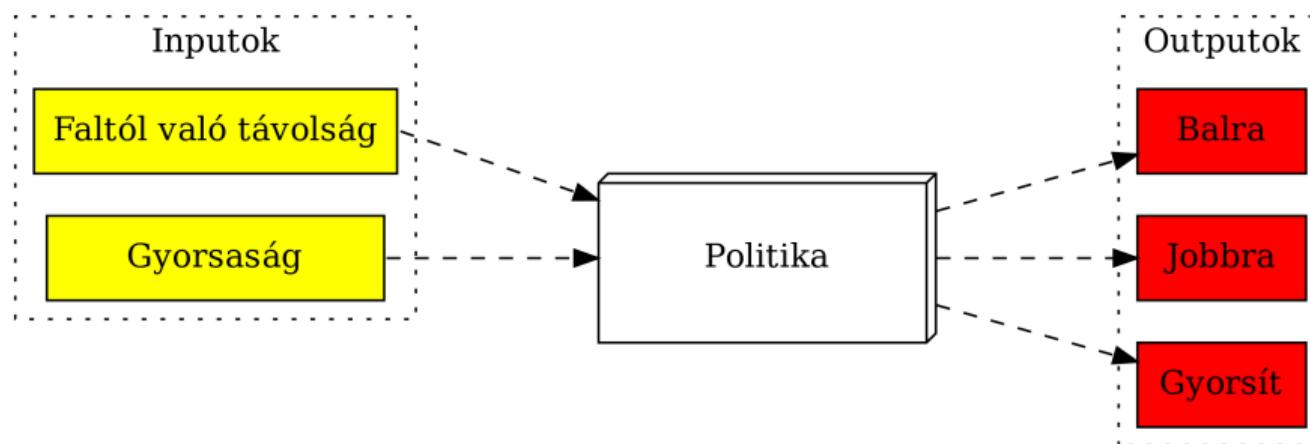
- A program, ami egy robotot irányít
- A program, ami PacMan-t irányítja
- Egy Go-t játszó program
- Lehet egy okos termosztát is
- Kereskedő a tőzsden



Politika

Az az algoritmus, ami az ügynököt irányítja. A politika által határozza meg az ügynök a cselekvéseit.

A politika egy modell, ami a környezetet leíró változókat fogadja bemenetként, és az outputja egy cselekvés a cselekvések választható halmazából.



Interakció a környezettel

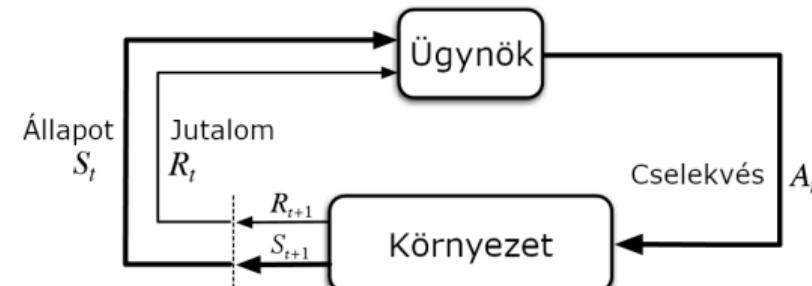
- Az ügynök és a környezet egymásra hatnak. Az ügynök cselekszik, ennek hatására a környezet megváltozik. Az ügynök megfigyeli a környezetet, majd ismét cselekszik:

$$s_1 \rightarrow a_1 \rightarrow s_2 \rightarrow a_2 \rightarrow \dots \rightarrow s_t \rightarrow a_t$$

- A jutalom azonnali, és cselekvés-állapot párosért jár: $R(s, a)$
- A környezet változását az átmeneti valószínűségek adják: $P(s'|s, a)$, ami s' következő állapot valószínűsége s állapotból, a cselekvést követően. Ez a környezet dinamikája.

- Az ügynök célja a lehető legmagasabb jutalom összegyűjtése hosszú távon:

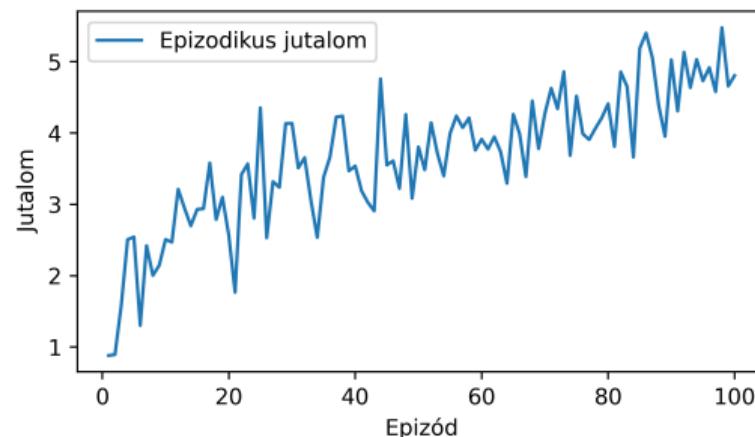
$$E_{\pi}(r_1 + r_2 + r_3 + \dots) \rightarrow \max$$



Epizódok

A megerősítéses tanulás egyetlen tanítási iterációja egy epizód. **Egy epizód addig tart, míg az ügynök el nem ér valamilyen vég/terminális állapotba.** A végállapot lehet:

- A cél teljesítése
- Teljes bukás / halál
- Időkeret lejárása
- Részfeladat teljesítése
- Jutalom összeg összegyűjtése



1 Bevezetés

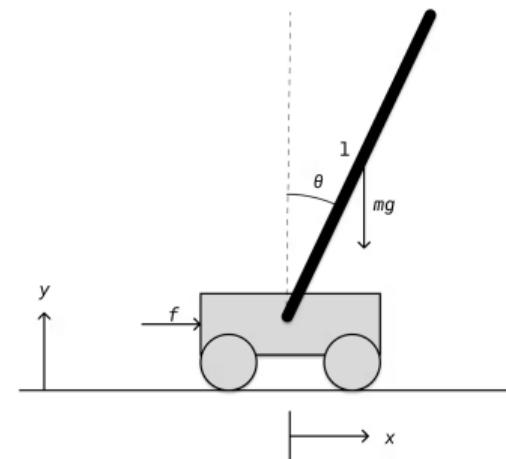
2 Politika javítása

3 Q-tanulás

Egy példa környezet: CartPole

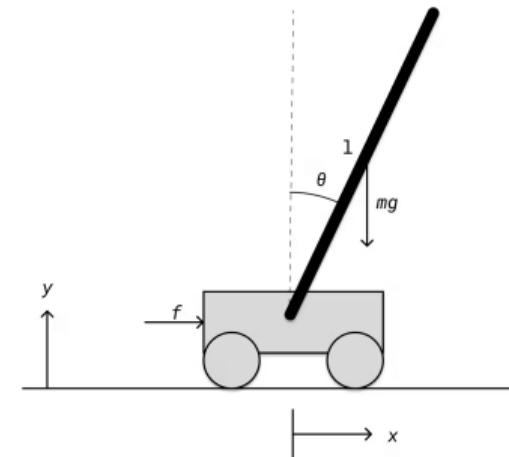
A CartPole egy egyszerű, de kihívást jelentő probléma. Egy oszlop egy szekérre van helyezve, és az ügynök célja, hogy a szekeret mozgatva az oszlopot egyensúlyban tartsa.

A modell tanításához a környezetet az OpenAI Gym megfelelő könyvtárainak segítségével lehet létrehozni. Ennek a környezetnek a feladata biztosítani a környezeti állapotot és a jutalmakat minden lépésben.



Egy példa környezet: CartPole

- **Állapotok:** Az állapot négy valós szám, amelyek a székér pozícióját, sebességét, az oszlop szögét és szögsebességét írják le.
- **Cselekvések:** Két lehetséges cselekvés van: a székér mozgatása balra vagy jobbra.
- **Jutalmak:** minden lépésért, amely során az oszlop nem esik le, a rendszer egy pontot ad. A cél az, hogy minél tovább fenn tartsuk az oszlopot, ezzel maximalizálva a kumulatív jutalmat.
- **Epizód:** Az epizód akkor ér véget, ha az oszlop egy bizonyos szögnél jobban elhajlik, vagy ha a székér kimegy a meghatározott határokon kívülre.



Egy egyszerű kezdeti politika

Az első egy keménykódolt politika: **olyan statikus szabályrendszer, amelyet nem gépi tanulás segítségével ismer meg az ügynök, hanem egy determinisztikus program vezérlí:**

```
for episode in range(n_episodes):
    for step in range(n_steps):
        if obs.theta < 0:
            action = 0
        else:
            action = 1
        obs, reward = env.step(action)
```

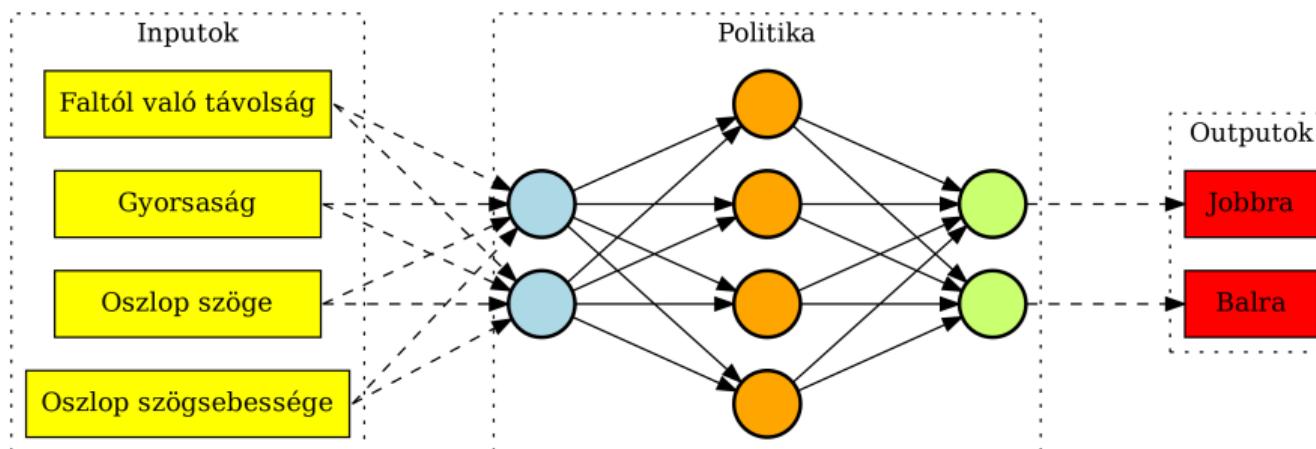
A politika szerint ha a θ oszlop dőlési szöge kevesebb, mint 0, a kocsit balra tolja, egyébként jobbra.

Az `env.step(action)` függvény segítségével hajtja végre az ügynök a választott cselekvését, majd a környezet visszaadja neki a jutalmat és a következő állapotot.

Neurális hálózat politika

A szabályok kézzel való implementálása hosszas és túlságosan specifikus. Ettől egy jobb hozzáállás, ha egy gépi tanulás modell becsüli a cselekvéseket a környezeti változók alapján.

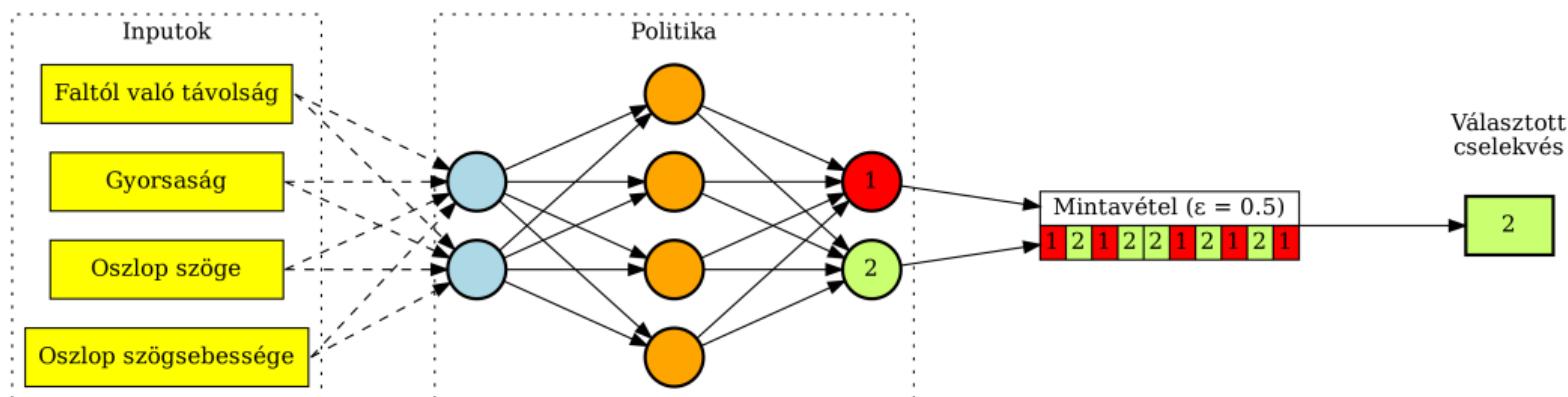
A modell bemenete ebben az esetben a környezeti változók vektora, a kimenete pedig a cselekvés, amit az ügynök végre fog hajtani. A tanítás eljárása pedig a neurális hálózat javításaként értelmezhető.



Politika hálózat működése

A hálózat output neuronjai azt a valószínűségetbecsülik meg, hogy mekkora valószínűsséggel az adott cselekvés lesz a leginkább jövedelmező az ügynök számára.

Ezután történik egy véletlen mintavétel, ahol ε valószínűsséggel a legjobb cselekvés fog szerepelni, $1 - \varepsilon$ valószínűsséggel pedig véletlen cselekvés. Ezzel lesz képes az ügynök felfedezni véletlen cselekvéseket a nem várt, de magas jutalom reményében.

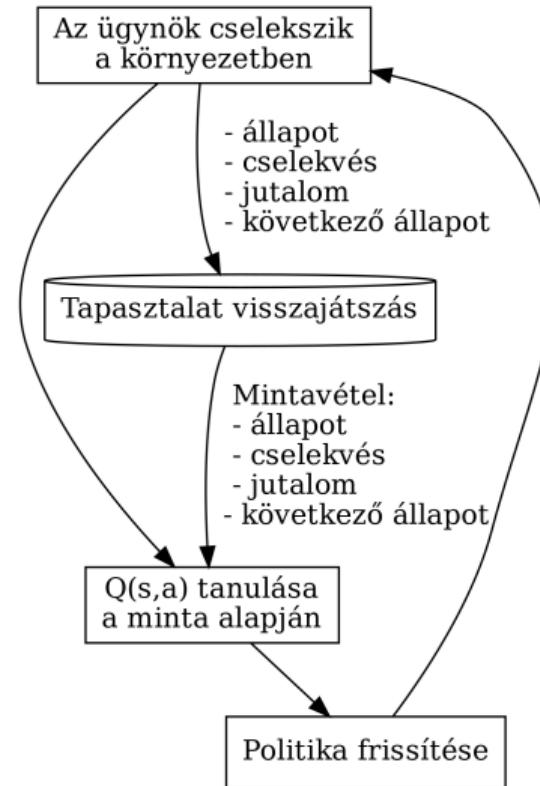


Tapasztalat visszajátszás

A tapasztalat visszajátszás a neurális hálózat alapú ügynökök **belső memóriája**.

A memória $[s, a, r, s']$ négyeseket tartalmaz. minden alkalommal amikor az ügynök cselekszik, az általa tapasztalt s, a, r, s' elmentődik.

Amikor tanulási iterációra kerül a sor az ügynök véletlen és rendezetlen mintát kap a visszajátszásból, melynek számossága megegyezik a kötegmérettel. Eszerint számolja a költségfüggvényt majd frissíti a modell paramétereit.

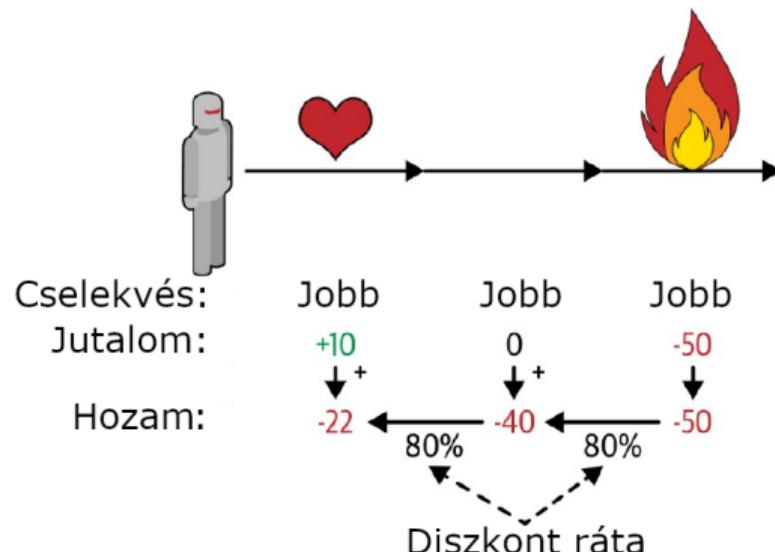


A kredit hozzárendelési probléma

Ha lehetséges lenne tudni minden lépésben, hogy melyik az optimális cselekvés, a neurális hálózat egyszerűen tanítható lenne a keresztrópia minimalizálásával.

Viszont az RL-ben a visszajelzések ritkák és késleltetettek. Az ügynök egyetlen visszajelzése amit kap, a jutalom, és nem mindig az utolsó cselekvés az, ami felelős a jutalomért.

Például: ha az ügynök 100 lépésen keresztül egyensúlyozza a rudat, majd leejt, honnan lehet tudni melyik cselekvés volt a felelős a leejtésért?

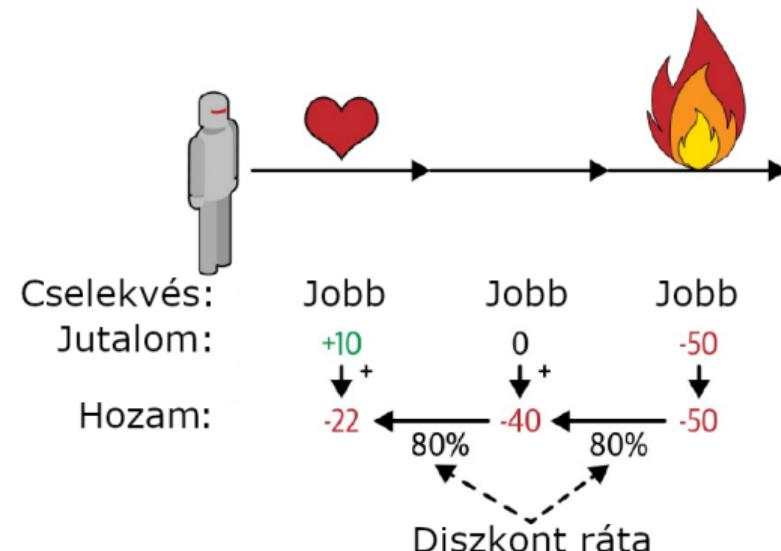


A kredit hozzárendelési probléma

A probléma megoldására az RL bevezet egy γ diszkontálási tényezőt, amely **megadja a jövőbeli jutalmak jelenbeli értékét**. Valamely r jutalom értéke k időlépés után γ^{k-1} .

Példa: ha az ágens háromszor jobbra megy, ezután a három jutalma $[+10, 0, -50]$, és $\gamma = 0.8$ az első cselekvés visszatérése $10 + 0 \cdot \gamma + \gamma^2 - 50 = -22$.

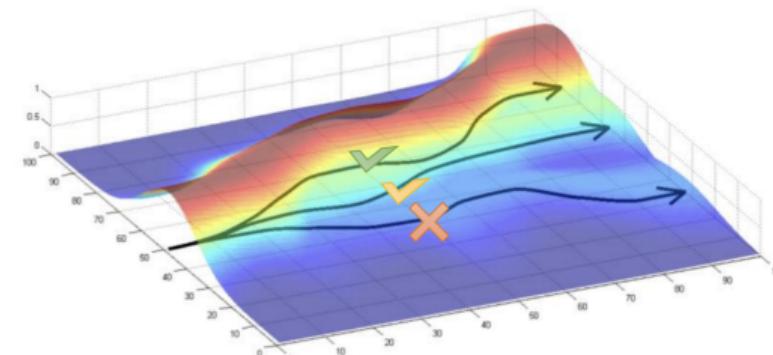
Ha $\gamma = 0$, a modell a jelenbeli jutalmat részesíti előnyben. Ha viszont 1 közel az értéke, a hosszú távú jutalomra törekzik.



Politika gradiens algoritmusok

Ahogy a függvényeknek, úgy egy politikának is meghatározható a gradiense.

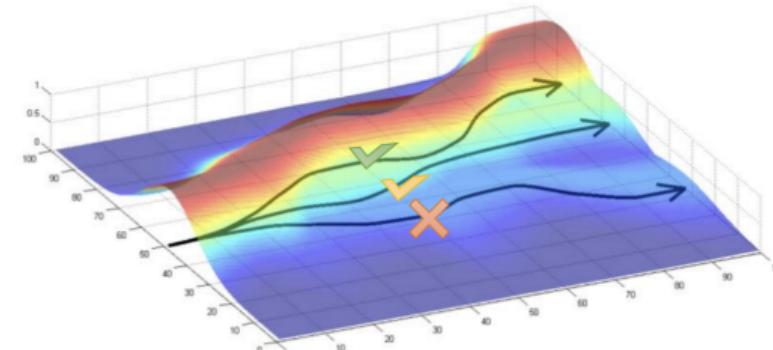
A politika gradiens algoritmusok úgy optimalizálják a modellek paramétereit, hogy mindenkor a magasabb jutalom felé mutató paramétereket növelik.



Politika gradiens algoritmusok

A politika gradiens algoritmus magas szintű lépései:

- ① Az algoritmus epizódokon keresztül futtatja a politikát.
- ② Miután több epizód lezajlott, az algoritmus kiszámítja minden egyik epizód diszkontált kumulatív jutalmát.
- ③ minden epizódhöz kiszámít egy gradiens vektort, majd megszorozza a hozzá tartozó diszkontált jutalommal.
- ④ Az összes, így kapott gradiens vektor átlagolása után az algoritmus egy gradiens ereszkedés lépést hajt végre a politikán.



1 Bevezetés

2 Politika javítása

3 *Q*-tanulás

A Q cselekvés minőség

Állapot-cselekvés minőség függvény

Egy (s, a) állapot-cselekvés páros minőség függvénye valamely π politika szerint a várható hozam, ha az ügynök s állapotból indul, a cselekvést hajtja végre, majd utána π szerint hozza döntéseit:

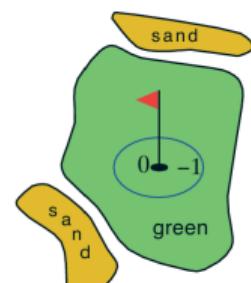
$$Q_\pi(s, a) = E_\pi [G_t | S_t = s, A_t = a] =$$

$$= E_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | S_t = s, A_t = a \right]$$

$$q_*(s, \text{driver})$$

$$\begin{matrix} \text{♀} \\ -3 \end{matrix}$$

$$-2$$

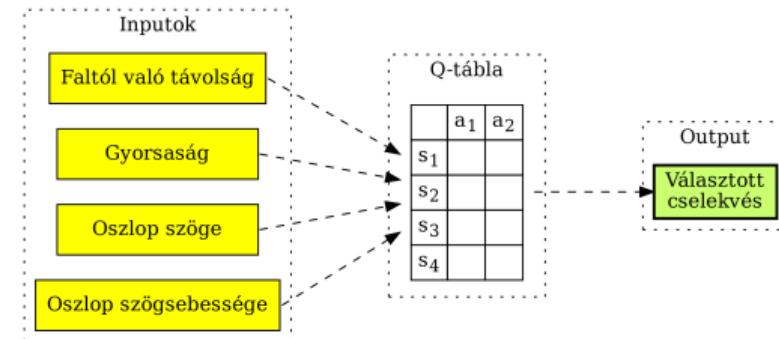


$$-2$$

A *Q*-tanulás

A *Q*-tanulás nem egy politika, hanem egy értékalapú megerősítéses tanulási megközelítés. **Az értékalapú algoritmusok cselekvések és állapotok értékei alapján határozzák meg a politikát.**

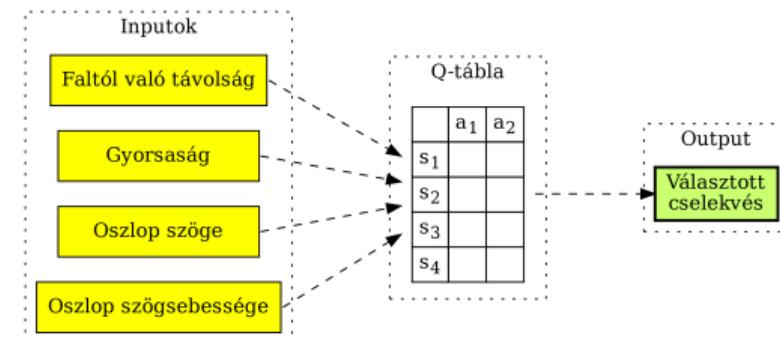
A *Q*-tanulás egy politikafüggetlen tanulási eljárás: az ügynök által végrehajtott politikától függetlenül képes megtalálni az optimális politikához tartozó *Q*-értékeket.



A Q -tanulás

A Q -tanulás általános algoritmusa:

- ① Q -tábla inicializálása véletlen értékekkel
- ② Cselekvés választása a Q -tábla alapján
- ③ Cselekvés végrehajtása a környezetben
- ④ Jutalom és környezet megfigyelése
- ⑤ Q -tábla frissítése
- ⑥ 2-5 lépések ismétlése



Q -tábla frissítése

Q -érték frissítésének szabálya

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right]$$

Ahol:

- s : Aktuális állapot
- a : Aktuális cselekvés
- α : Tanulási sebesség
- r : Jutalom
- γ : Diszkont ráta
- s' : Következő állapot
- a' : Következő cselekvés

Példa Q -táblára:

Állapot	Fel	Le	Bal	Jobb
s_1	0.1	0.6	0.2	0.3
s_2	0.7	0.1	0.4	0.2
s_3	0.2	0.2	0.8	0.4
s_4	0.3	0.1	0.5	0.9
s_5	0.1	0.5	0.4	0.2
s_6	0.3	0.3	0.6	0.2
s_7	0.8	0.3	0.1	0.4
s_8	0.2	0.4	0.3	0.7

Cselekvés kiválasztás és végrehajtás

Mivel az ügynöknek kezdetben nincs ismerete a környezetről, véletlenül kell cselekednie, hogy megismerje mely állapotok és cselekvések milyen jutalommal járnak.

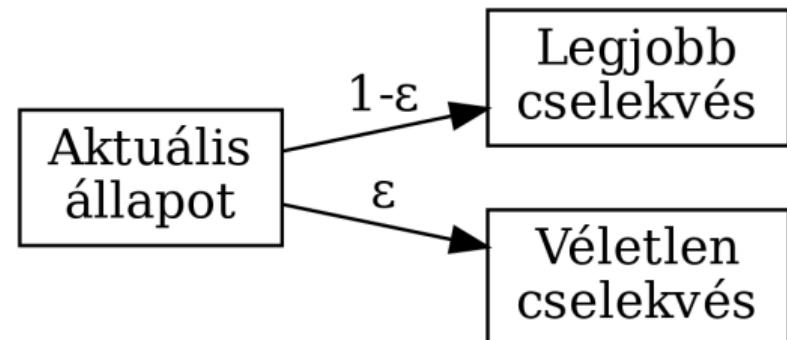
ε -mohó cselekvés választás

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

Ahol A az összes cselekvés halmaza.

Felfedezés: Véletlen cselekvés végrehajtása, ε valószínűsséggel.

Kiszákmányolás: Már ismert, nagy jutalommal járó cselekvés végrehajtása, $1 - \varepsilon$ valószínűsséggel.



Felfedezési stratégia

Amikor a tanulás folyamata elkezdődik, az ügynöknek fel kell fedeznie a környezetét, és véletlen cselekvések által **tapasztalatot szereznie**.

Később, amikor már kellően tapasztalt a **véletlen cselekvéseket felváltják a legnagyobb értékű cselekvések**. Ennek megfelelően az ε együttható csökken.

Az ε csökkenését a párolgási együttható, ε_{decay} határozza meg. minden iterációban a változás:

$$\varepsilon \leftarrow \varepsilon \cdot \varepsilon_{decay}$$

