

# Üzleti Intelligencia

## 6. Előadás: Mély $Q$ -tanulási architektúrák

Kuknyó Dániel  
Budapesti Gazdasági Egyetem

2023/24  
1.félév

- 1 Bevezetés
- 2 Mély  $Q$ -tanulás (DQN)
- 3 Dupla mély  $Q$ -tanulás (DDQN)
- 4 Párbajozó mély  $Q$ -tanulás

- 1 Bevezetés
- 2 Mély  $Q$ -tanulás (DQN)
- 3 Dupla mély  $Q$ -tanulás (DDQN)
- 4 Párbajozó mély  $Q$ -tanulás

# A $Q$ -tanulás alapjai

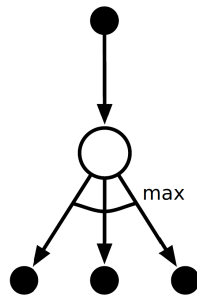
A megerősítéses tanulásban az egyik nagy áttörést egy politikafüggetlen TD algoritmus kifejlesztése hozta el.

Ebben az esetben a becsült **állapot-cselekvés minőség függvény**,  $Q$ , ami megadja, hogy mennyire jövedelmező az ügynöknek  $s$  állapotban  $a$  cselekvést végrehajtani.

## $Q$ -tanulás

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

A  $Q$ -tanulás ezáltal egy teljesen online tanulási algoritmus, ami a követett **politikától függetlenül** garantáltan konvergálni fog a valós  $Q$  értékekhez.



# Dupla Q-tanulás

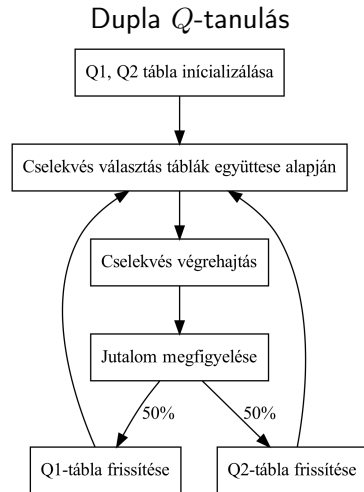
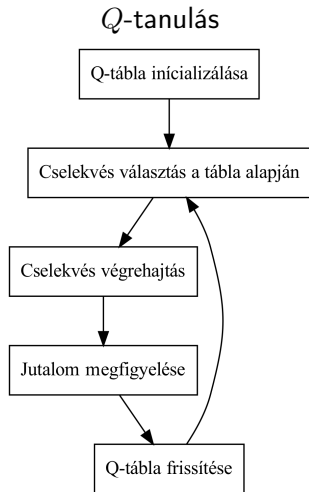
A kettős tanulás ötlete természetesen kiterjed a teljes MDP algoritmusaira. A Q-tanulásban a becsült Q-értékek torzítottak lehetnek, ha alacsony a minta számossága, vagy zaj van a rendszerben. Egy módja a Q-tanulás regularizálásának, ha egy helyet **két Q-táblát tart nyilván az algoritmus**,  $Q_1$ -et és  $Q_2$ -t.

A Q-tanulással analóg dupla Q-tanulás nevű kettős tanulási algoritmus két részre osztja az időlépéseket, **minden lépésnél egy érmét feldobva**. Ha az érme fejre esik, a frissítés a következő:

$$Q_1(s_t, a_t) \leftarrow Q_1(s_t, a_t) + \alpha \left[ r_{t+1} + \gamma Q_2(s_t + 1, \underset{a}{\operatorname{argmax}} Q_1(s_{t+1}, a)) - Q_1(s_t, a_t) \right]$$

Ha az érme pedig írásra esik, akkor ugyanez a frissítés  $Q_1$  és  $Q_2$  felcserélésével történik, így  $Q_2$  frissül. A két közelítő értékfüggvényt teljesen szimmetrikusan kezeli az algoritmus. Például egy  $\varepsilon$ -mohó politika a dupla tanulás esetében az egyes cselekvési értékbecslések **átlagára vagy összegére épülhet**.

# Alapvető $Q$ -tanulási eljárások



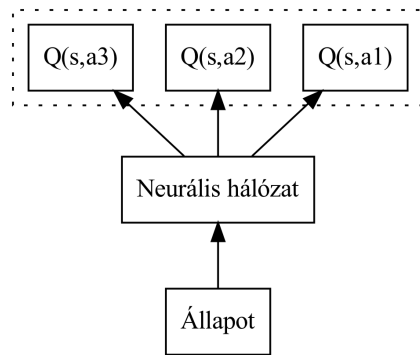
- 1 Bevezetés
- 2 Mély  $Q$ -tanulás (DQN)
- 3 Dupla mély  $Q$ -tanulás (DDQN)
- 4 Párbajozó mély  $Q$ -tanulás

# A $Q$ -hálózat (DQN)

A  $Q$ -hálózat egy természetes kiterjesztése a hagyományos  $Q$ -tanulásnak. A naív  $Q$ -hálózat inputja a **környezetet leíró változók** vektora vagy mátrixa, és az outputja pedig az ügynök számára elérhető **cselekvések  $Q(s, a)$  értéke** minden  $a_1, a_2, \dots, a_n$  cselekvéshez tartozóan.

A cselekvés választáshoz az ügynök kiválasztja a **legnagyobb becsült  $Q$  értéket**, és az ahhoz tartozó cselekvést fogja végrehajtani.

A  $Q$ -hálózat költségfüggvénye az **átlagos négyzetes Bellman hiba**, a paraméter frissítése pedig a költségfüggvény gradiense és a lépésméret szerint történik.



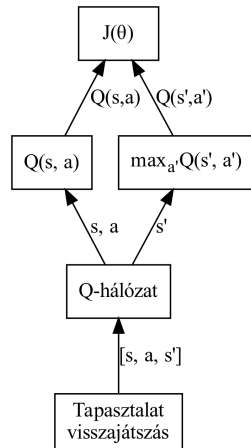


# A DQN költségfüggvénye

A mély  $Q$ -tanulás során a költségfüggvény a négyzetes Bellman hiba várható értéke:

$$J(\theta) = E_{s,a,s' \sim D} \left[ \left( r + \gamma \max_{a'} Q_{\theta}(s', a') - Q_{\theta}(s, a) \right)^2 \right]$$

Ahol  $s, a, s'$  a  $D$  tapasztalat visszajátszási memóriából vett minta. A  $Q$ -hálózat megkeresi az  $s'$  következő állapothoz tartozó legnagyobb értékű cselekvést,  $a'$ -t, és lekérdezi a memóriából vett  $s$  aktuális állapothoz és  $a$  aktuális cselekvéshez tartozó  $Q$ -értéket. Ez a TD hibának egy változata.

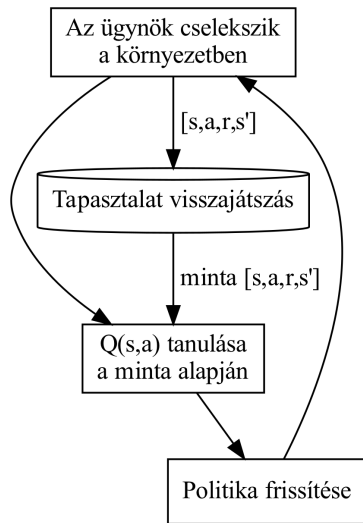


# Tapasztalat visszajátszás

A tapasztalat visszajátszás az egyik változtatás, ami a  $Q$ -hálózat problémáján hivatott segíteni.

A tapasztalat memória ( $D_{replay}$ )  $[s, a, r, s']$  **négyeseket tartalmaz**. Minden alkalommal amikor az ügynök cselekszik, az általa tapasztalt  $s_t, a_t, r_t, s_{t+1}$  elmentődik a tapasztalat memóriába.

Amikor tanulásra kerül a sor, az ügynök **véletlen és rendezetlen mintát** (miniköteget) kap a tapasztalat memóriából, melynek számossága megegyezik a kötegmérettel. Eszerint számolódik ki a költségfüggvény majd frissülnek a  $Q$ -hálózat paraméterei.



---

## Algoritmus 1: Mély $Q$ -tanulás

---

**Input:**  $\tau \sim [0, 1]$  átlagolási ráta, Online hálózat  $Q_\theta$ , célhálózat  $Q_{\bar{\theta}}$ , tapasztalat memória  $D$  inicializálása;

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

**for**  $t = 0 \rightarrow \max_t$  **do**

$s_t$  megfigyelése és  $a_t \sim \pi(s_t, a_t)$  cselekvés választása;

$a_t$  végrehajtása és  $s_{t+1}$ ,  $r_t$  megfigyelése;

$s_t, a_t, r_t, s_{t+1}$  eltárolása a tapasztalat memóriában;

**end**

**for**  $c = 0 \rightarrow \max_c$  **do**

$e_t = s_t, a_t, r_t, s_{t+1} \sim D$ ; /\* Mintavétel a tapasztalat memóriából \*/

        Gradiens ereszkedés végrehajtása a  $\left(r + \gamma \max_{a'} Q_\theta(s', a') - Q_\theta(sa)\right)^2$  hibán:

**end**

**end**

---

Ahol  $t$  a környezetben lejátszott lépések és  $c$  a hálózat tanítási lépések ciklusváltozója.

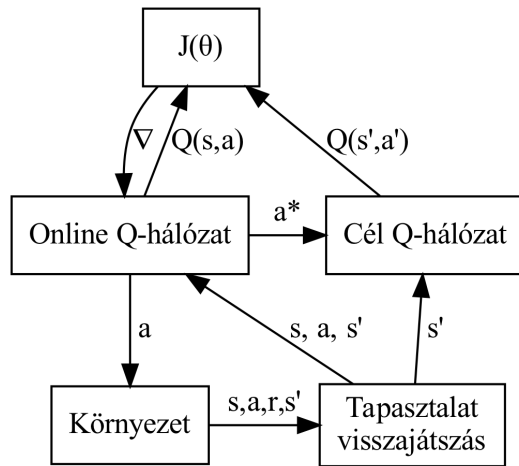
- 1 Bevezetés
- 2 Mély  $Q$ -tanulás (DQN)
- 3 Dupla mély  $Q$ -tanulás (DDQN)
- 4 Párbajozó mély  $Q$ -tanulás

# DDQN

A DDQN algoritmus a dupla  $Q$ -tanulás elveit hajtja végre neurális hálózatokkal: ebben az esetben a két  $Q$ -tábla helyett **két  $Q$ -hálózat** kap helyet az architektúrában:

- Online hálózat  $Q_\theta$ : a cselekvések **kiválasztásáért** felel.
- Célhálózat  $Q_{\bar{\theta}}$ : a választott cselekvés **kiértékeléséért** felel.

Az algoritmus csak az online hálózaton hajt végre paraméter frissítést. A célhálózatot úgy frissíti, hogy az online hálózat paramétereinek értékét átmásolja adott időközönként.



# A DDQN költségfüggvénye

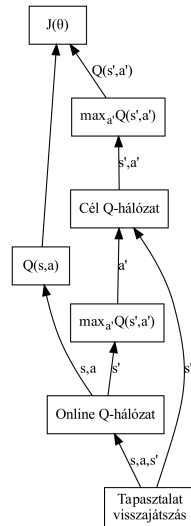
A DDQN architektúrában az optimális cselekvés  $a^*$  a  $Q_\theta$  online hálózat által lesz kiválasztva  $s'$  következő cselekvésre:

$$a^* = \underset{a'}{\operatorname{argmax}} Q_\theta(s', a')$$

A  $Q_{\bar{\theta}}$  célhálózat pedig kiértékeli az online hálózat által választott cselekvést:  $Q_{\bar{\theta}}(s', a^*)$ , majd kiszámolja a TD hibát:

$$J(\theta) = E_{s,a,r,s' \sim D} \left[ (r + \gamma Q_{\bar{\theta}}(s', a^*) - Q_\theta(s, a))^2 \right]$$

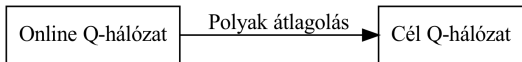
Ahol  $E_{s,a,r,s' \sim D}$  a tapasztalat visszajátzásból származó miniköteg,  $\gamma$  a diszkont ráta.



# A célhálózat frissítése

Adott időközönként az online hálózat  $Q_\theta$  paramétereit át kell másolni a  $Q_{\bar{\theta}}$  célhálózatra, **hiszen a célhálózaton nincs gradiens ereszkedés végrehajtva.**

Ezt meg lehet tenni úgy is, hogy periodikusan explicit másolat készüljön az online hálózatról, viszont egy jobb megoldás, ha Polyak átlagolással (**lágymásolással**) másolódnak át a paraméterek.



Ebben az esetben a  $\tau$  hiperparaméter szabályozza az **átlagolási rátát** vagyis, hogy mekkora súllyal legyenek figyelembe véve az új paraméterek:

## Polyak átlagolás

$$\bar{\theta} \leftarrow \tau\theta + (1 - \tau)\bar{\theta}$$

- $\bar{\theta}$ : Célhálózat paramétere
- $\theta$ : Online hálózat paramétere
- $\tau$ : Átlagolási ráta

---

**Algoritmus 2:** Dupla mély  $Q$ -tanulás (DDQN)

---

**Input:**  $\tau \sim [0, 1]$  átlagolási ráta, Online hálózat  $Q_\theta$ , célhálózat  $Q_{\bar{\theta}}$ , tapasztalat memória  $D$  inicializálása;

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

**for**  $t = 0 \rightarrow \max_t$  **do**

$s_t$  megfigyelése és  $a_t \sim \pi(s_t, a_t)$  cselekvés választása;

$a_t$  végrehajtása és  $s_{t+1}$ ,  $r_t$  megfigyelése;

$s_t, a_t, r_t, s_{t+1}$  eltárolása a tapasztalat memóriában;

**end**

**for**  $c = 0 \rightarrow \max_c$  **do**

$e_t = s_t, a_t, r_t, s_{t+1} \sim D$ ; /\* Mintavétel a tapasztalat memóriából \*/

$Q^*(s_t, a_t) \leftarrow r_t + \gamma Q_\theta(s_{t+1}, \operatorname{argmax}_{a'} Q_{\bar{\theta}}(s_{t+1}, a'))$ ;

        Gradiens ereszkés végrehajtása a  $(Q^*(s_t, a_t) - Q_\theta(s_t, a_t))^2$  hibán;

$\bar{\theta} \leftarrow \tau \theta + (1 - \tau) \bar{\theta}$ ; /\* Paraméterek frissítése \*/

**end**

**end**

---



- 1 Bevezetés
- 2 Mély  $Q$ -tanulás (DQN)
- 3 Dupla mély  $Q$ -tanulás (DDQN)
- 4 Párbajozó mély  $Q$ -tanulás

# Az előny függvény

A megerősítéses tanulás harmadik értékfüggvénye a  $V(s)$  és a  $Q(s, a)$  mellett.

Matematikailag az előny függvény  $s$  állapotra és  $a$  cselekvésre a cselekvés  $Q$  értékének és az állapot értékének a különbsége:

## Előny függvény

Az előny függvény megadja, **ennyivel jobb vagy rosszabb** egy adott  $a$  cselekvést végrehajtani egy  $s$  állapotból a többi elérhető cselekvéshez képest.

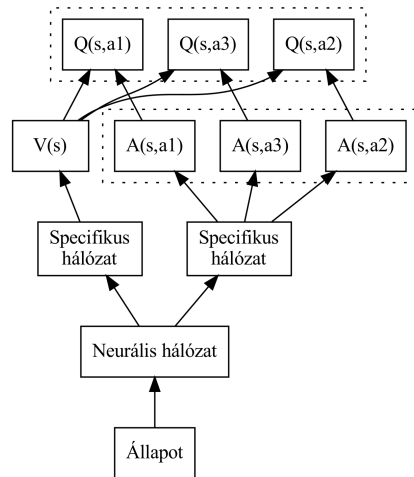
$$A(s, a) = Q(s, a) - V(s)$$

Egy  $A(s, a) = 0$  érték azt jelenti, hogy  $a$  cselekvést végrehajtani olyan jövedelmező, mint az átlagos cselekvés  $s$ -ből. Ha  $A(s, a) > 0$ , a cselekvés jobb, és ha  $A(s, a) < 0$  akkor rosszabb mint az átlagos cselekvés  $s$  állapotból.

# Párbajozó $Q$ -hálózat architektúrája

A párbajozó  $Q$ -tanulás algoritmus **explicit módon kettéválasztja az állapot-érték és előny függvények megbecslését** minden állapot művelet esetén. Ez a szétválasztás teszi lehetővé a hatékonyabb tanulást és jobb általánosítást.

Az előnyfüggvény definíciójából kiindulva a minőségfüggvényt ki lehetne számolni úgy, mint  $Q(s, a) = V(s) + A(s, a)$ , **viszont ez problémás mert nem feltételezhető, hogy a hálózat minden értékre pontos becslést ad.**

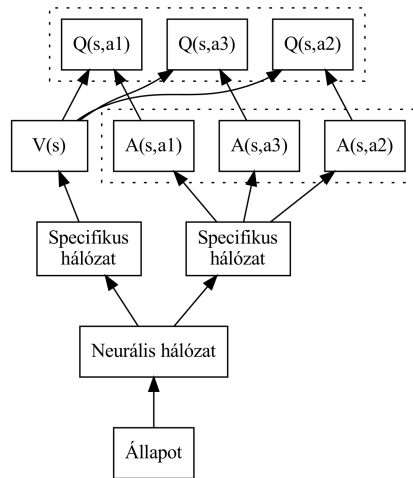


# Párbajozó Q-hálózat architektúrája

A gyakorlatban a  $Q(s, a)$  érték úgy áll elő, hogy a  $V(s)$  állapot-értékhez hozzáadódik a választott cselekvés és a legnagyobb előnyű cselekvés  $A(s, a)$  értékének különbsége:

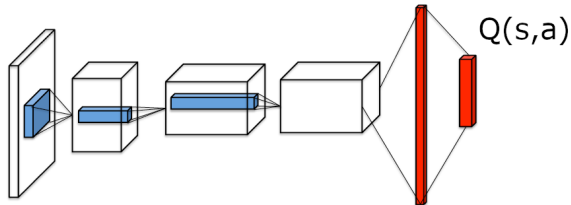
$$Q(s, a) = V(s) + \left( A(s, a) - \frac{1}{|A|} \sum_{a'} A(s, a') \right)$$

Ahol  $|A|$  az elérhető cselekvések száma,  $\frac{1}{|A|} \sum_{a'} A(s, a')$  pedig az elérhető cselekvések előnyeinek átlaga.

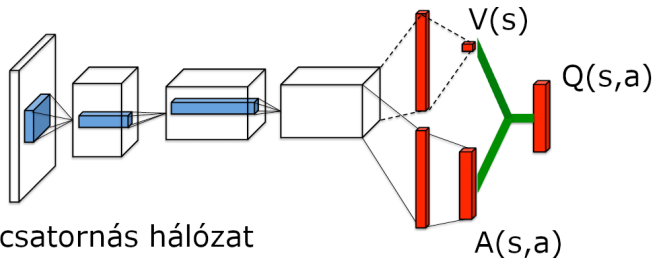


# Az egycsatornás és párbajozó architektúrák összehasonlítása

## Egycsatornás hálózat



## Többscsatornás hálózat



# Dupla párbajozó architektúrák (DDDQN)

A kettős  $Q$ -tanulást meg lehet valósítani párbajozó hálózatokkal is. Mivel mindkettőnek az outputja a  $Q(s, a)$  állapot-cselekvés minőség függvény, az egyetlen változtatás a kettős  $Q$ -tanuláshoz képest a neurális hálózat architektúrájában van. A komponensek elrendezésén nem szükséges változtatni.

A hálózatok között a paraméterek másolását explicit másolással, vagy Polyak átlagolással lehet megoldani. Ezzel a hálózat pontosabban tanul, és ellenállóbb a tanítási mintában jelen lévő zajjal szemben.

