Üzleti Intelligencia

6. Előadás: Mély Q-tanulási architektúrák

Kuknyó Dániel
Budapesti Gazdasági Egyetem

2023/24
1.félév

A Q-tanulás alapjai

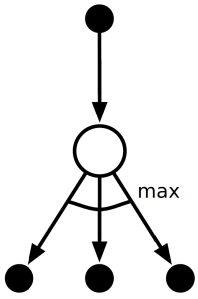
A megerősítéses tanulásban az egyik nagy áttörést egy politikafüggetlen TD algoritmus kifejlesztése hozta el.

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

Q-érték frissítése

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

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



Dupla Q-tanulás

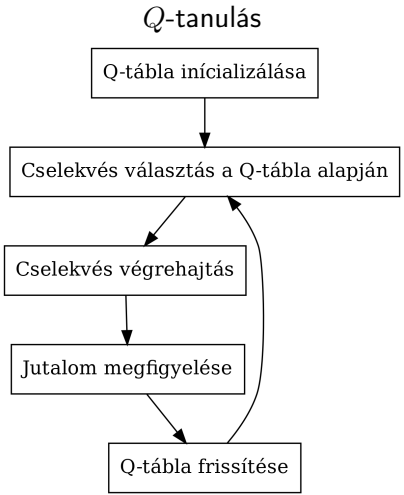
A kettős tanulás ötlete kiterjed a teljes MDP algoritmusaira. A Q-tanulásban a becsült Q-értékek torzítottak lehetnek, ha alacsony a minta számossága, vagy zaj van a rendszerben. Egy módja a Q-tanulás regularizálásának, ha egy helyet **két Q-táblát tart nyilván az algoritmus**, Q_1-et és Q_2-t.

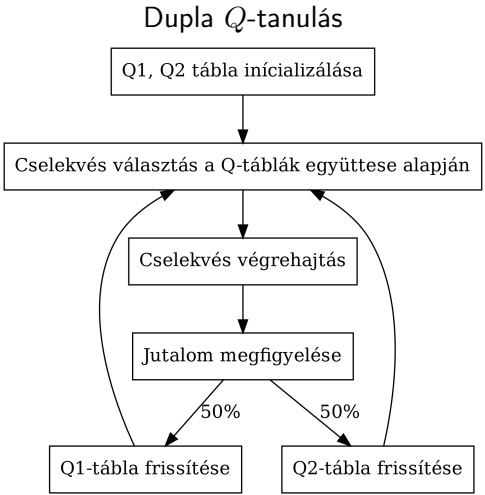
A Q-tanulással analóg dupla Q-tanulás nevű kettős tanulási algoritmus két részre osztja az időlépéseket, minden lépésben 50% valószínűséggel a frissítés a következő:

$$Q_1(s,a) \leftarrow Q_1(s,a) + \alpha \left[r + \gamma Q_2 \left(s', \underset{a'}{argmax} Q_1(s',a') \right) - Q_1(s,a) \right]$$

A többi esetben Q_1 és Q_2 felcserélésével történik, így Q_2 frissül. A két táblát teljesen szimmetrikusan kezeli az algoritmus. Például egy ε-mohó politika a dupla tanulás esetében az egyes cselekvési értékbecslések **átlagára vagy összegére épülhet**.

Alapvető Q-tanulási eljárások



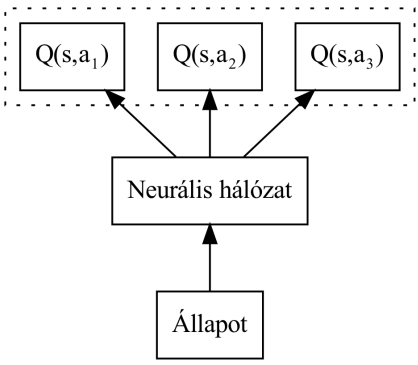


A Q-hálózat (DQN)

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 $a_1, a_2, .., a_n$ 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.

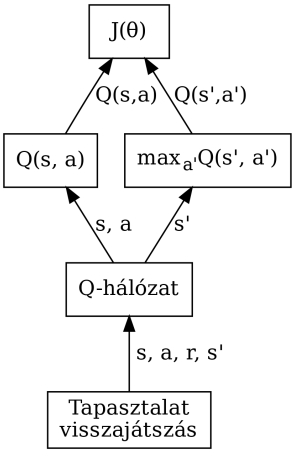


A DQN költségfüggvénye

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

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

Ahol s, a, r, 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.

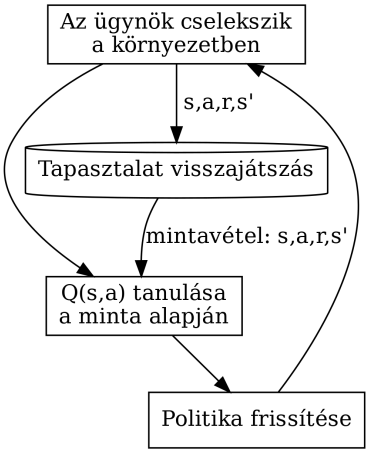


Tapasztalat visszajátszás

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, a, r, s' 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

Q-hálózat Q_θ inicializálása, tapasztalat memória D inicializálása;
for $i = 0 \rightarrow max_i$ **do**
  for $t = 0 \rightarrow max_t$ **do**
    s megfigyelése és $a \sim \pi(s, a)$ cselekvés választása;
    a végrehajtása és s', r megfigyelése;
    s, a, r, s' eltárolása a tapasztalat memóriában;
  end
  for $c = 0 \rightarrow max_c$ **do**
    $e = s, a, r, s' \sim D$; `/* Mintavétel a tapasztalat memóriából */`
    Gradiens ereszkedés végrehajtása $\left(r + \gamma \max_{a'} Q_\theta(s', a') - Q_\theta(s, a)\right)^2$ hibán;
  end
end

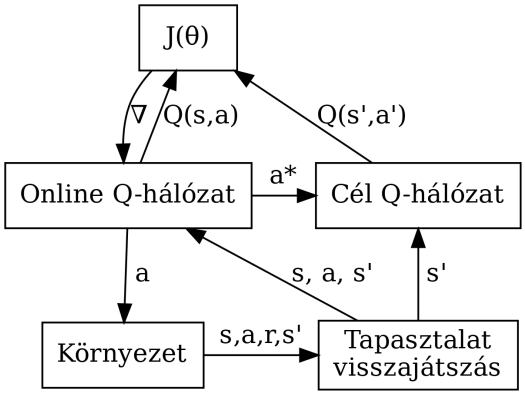
Ahol t a környezetben lejátszott lépések és c a hálózat tanítási lépések ciklusváltozója.

DDQN

A DDQN algoritmusa 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_θ: 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

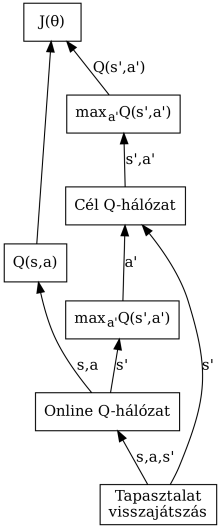
A DDQN architektúrában az optimális cselekvés a^* a Q_θ online hálózat által lesz kiválasztva s' következő állapotra:

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

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

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

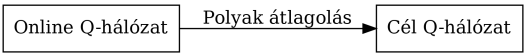
Ahol $\{s, a, r, s' \sim D\}$ a tapasztalat visszajátszásból származó miniköteg, γ a diszkont ráta.



A célhálózat frissítése

Adott időközönként az online hálózat Q_θ 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 (**lágy másolással**) másolódnak át a paraméterek.



Ebben az esetben a τ 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 paraméterei
- θ: Online hálózat paraméterei
- τ: Átlagolási ráta

Algoritmus 2: Dupla mély Q-tanulás (DDQN)

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

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

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

    s megfigyelése és $a \sim \pi(s, a)$ cselekvés választása;

    a végrehajtása és s', r megfigyelése;

    s, a, r, s' eltárolása a tapasztalat memóriában;

  end

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

    $e = s, a, r, s' \sim D$; `/* Mintavétel a tapasztalat memóriából */`

    $Q^*(s, a) \leftarrow r + \gamma Q_\theta(s', argmax_{a'} Q_{\bar{\theta}}(s', a'))$;

    Gradiens ereszkedés végrehajtása a $(Q^*(s, a) - Q_\theta(s, a))^2$ hibán;

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

  end

end

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 $Q(s, a)$ minőségének és s állapot $V(s)$ értékének a különbsége:

Előny függvény

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

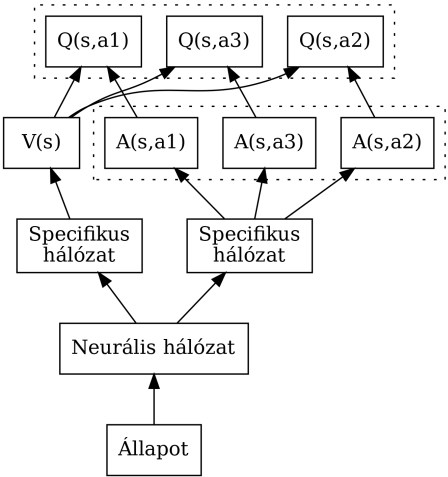
$$A(s, a) = Q(s, a) - V(s)$$

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 algoritmusa **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**.

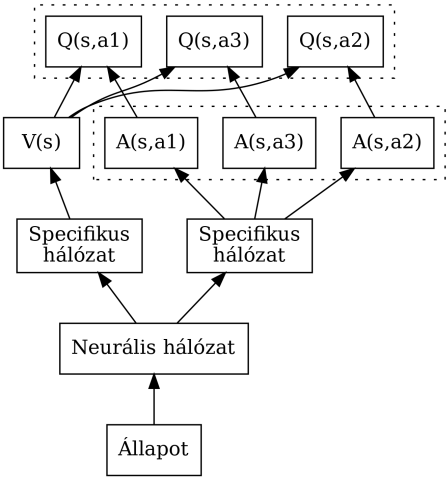


Párbajozó Q-hálózat architektúrája

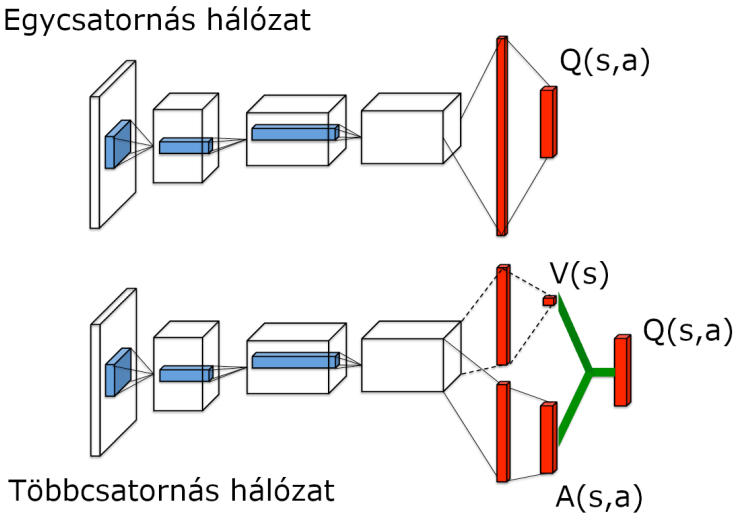
A gyakorlatban a $Q(s,a)$ érték úgy áll elő, hogy a $V(s)$ állapot-értékhez hozzáadódik a választott cselekvés előnyének és az összes cselekvés átlagos előnyének különbsége:

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

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



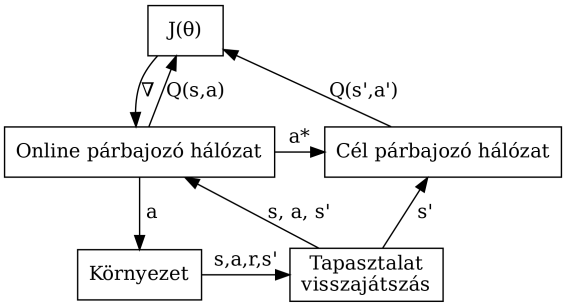
Az egycsatornás és párbajozó architektúrák összehasonlítása



Dupla párbajozó architektúrák (DDQN)

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 módosítani.

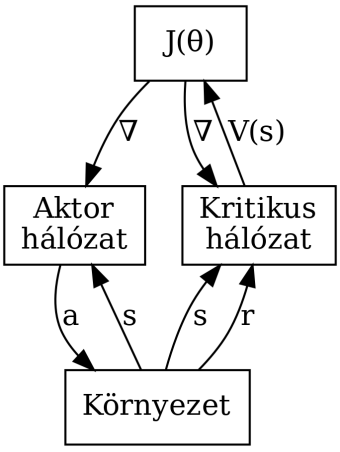
A hálózatok között a paraméterek másolását explicit másolással, vagy Polyak átlagolással lehet megoldani. Ezzel a hálózat pontosabban tanul, és ellenállóbb a tanítási mintában jelen lévő zajjal szemben.



Aktor-kritikus architektúra

Az aktor-kritikus architektúra egy kettős tanulási eljárás, amely kombinálja a politikaalapú (**aktor**) és az értékalapú (**kritikus**) eljárásokat a tanítási folyamat javítása és a gyorsabb konvergencia érdekében.

- **Aktor**: Az állapothoz tartozó cselekvés kiválasztásáért felel. Egy politikát tanul meg, ami az állapotokhoz cselekvéseket rendel. Célja, hogy maximalizálja a várható kumulált hozamot.
- **Kritikus**: Feladata megbecsülni a $V(s)$ állapot-érték függvényt. A kritikus segíti az aktort azzal, hogy visszajelzést ad neki az általa megtett cselekvések minőségéről.



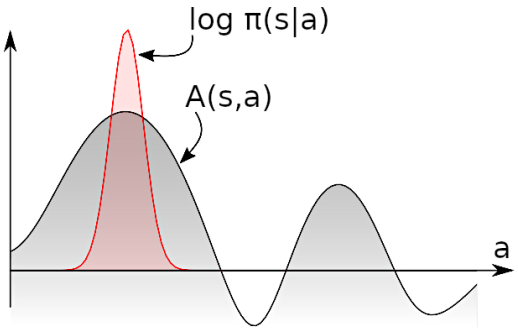
Az aktor-kritikus költségfüggvény

Az előnyalapú aktor-kritikus eljárásokban az aktor az $A(s,a)$ előnyfüggvényt becsüli meg a saját neurális fejében. A kritikus pedig egy logaritmikus valószínűségi eloszlást.

Az AAC költségfüggvénye

$$J(\theta) = \sum_{t=0}^{T-1} log\pi_\theta(a|s)A(s,a)$$

Ahol $\pi_\theta(a|s)$ az a valószínűség, hogy a π_θ szerint mekkora valószínűséggel hajtja végre az ügynök a cselekvést s állapotból, és $A(s,a)$ az előnyfüggvény.



A modellezés folyamata

1. **Inicializáció**: Az aktor és kritikus hálózatok kezdősúlyainak megadása: θ, ϕ, tanulási sebesség beállítása: α
2. **Tapasztalat gyűjtése**: az aktor interakcióba lép a környezettel
3. **Előny kiszámítása**: $A(s,a) = (V(s) - (r + V(s')))^2$ megadja, hogy egy adott a cselekvés mennyire jövedelmező a többihez képest
4. **Aktor frissítése**: gradiens ereszkedés a $log\pi_\theta(a|s)A(s,a)$ előny értéken
5. **Kritikus frissítése**: gradiens ereszkedés a $(V(s) - (r + V(s')))^2$ hibán
6. **Iteráció** folytatása a kilépésig

Algoritmus 3: Előnyalapú aktor-kritikus tanítás

Aktor $\pi_\theta(s)$ és kritikus $V_\phi(s)$ inicializálása;
for $i = 0 \rightarrow max_i$ **do**
    s inicializálása a kezdőállapottal;
    while $s \neq s_T$ **do**
        $a \leftarrow \pi_\theta(s)$; `/* Az aktor cselekvést választ */`
        a végrehajtása a környezetben, r és s' megfigyelése;
        $A(s,a) \leftarrow r + \gamma V_\phi(s') - V_\phi(s)$; `/* Előny kiszámítása */`
        $\theta \leftarrow \theta + \alpha \nabla log \pi_\theta(a|s) A(s,a)$; `/* Aktor frissítése */`
        $\phi \leftarrow \phi + \alpha \nabla A(s,a)^2$; `/* Kritikus frissítése */`
    end
end

Ahol θ az aktor, és ϕ a kritikus paraméterlistája.