Üzleti Intelligencia

6. Előadás: Q-tanulás

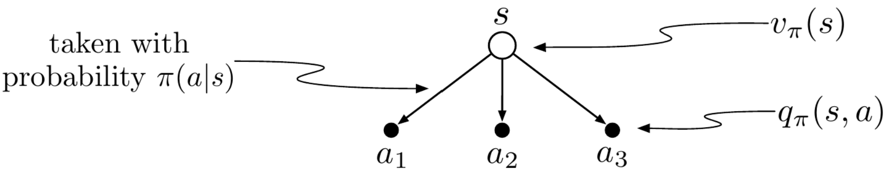
Kuknyó Dániel
Budapesti Gazdasági Egyetem

2023/24
1.félév

Bevezetés a Q-tanulásba

A megerősítéses tanulásban a $Q_\pi(s,a)$ minőségfüggvény megadja, hogy mennyire jövedelmező az ügynöknek, ha s állapotban állva, a cselekvést végrehajtva majd onnan π politikát követve mekkora a várható hozam.

Ha adott minden állapotra és cselekvésre az optimális $Q(s,a)$ érték, onnan levezethető a π_* optimális politika, amelynek várható hozama maximális.



$$q_\pi(s,a) = E_\pi\left[G_t | s_t = s, a_t = a\right] = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\right]$$

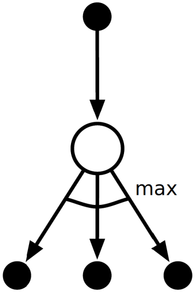
Q-tanulás: politikafüggetlen TD irányítás

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

Ebben az esetben a becsült állapot-cselekvés minőség függvény, Q, direkten becsüli meg q_* optimális állapot-cselekvés minőség függvényt a követett politikától teljesen függetlenül.

Q-tanulás

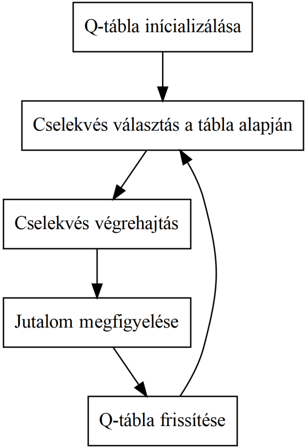
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$



Az egyetlen különbség a SARSA algoritmusától, hogy a referencia a következő állapotból elérhető legjobb minőségű cselekvés értéke: $\max_a Q(s_{t+1}, a)$.

Q-tanulás eljárása

A Q-tanulás folyamata



Példa Q-táblára

	a_0	a_1	a_2	a_3
s_0	0.76	0.41	0.92	-0.14
s_1	-0.65	0.31	-0.07	0.55
s_2	0.23	-0.99	0.67	-0.43
s_3	-0.81	0.79	-0.58	0.17
s_4	0.62	-0.28	0.96	-0.72
s_5	-0.36	0.08	-0.51	0.64

A Q-tábla tárolja el a $Q(s, a)$ értékeket minden $s \in S$ és $a \in A$ párosra.

Algoritmus 1: Q-tanulás algoritmusa $\pi \approx \pi_*$ megbecslésére

Input: α tanulási sebesség; $\varepsilon > 0$ hibahatár
$Q(s,a) \leftarrow random()$ for $s \in S$, $a \in A$; `/*` Q `értékek inícializálása */`
$Q(s_T, \cdot) \leftarrow 0$; `/* Terminális állapot 0-ra állítása */`
for $i = 0 \to max_i$ **do**
  s inícializálása;
  while $s \neq s_T$ **do**
    $a \leftarrow \pi(s)$; `/* Cselekvés választása` π `szerint, pl.` ε`-mohó */`
    a végrehajtása, r és s' megfigyelése;
    $Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_a Q(s',a) - Q(s,a) \right]$;
    $s \leftarrow s'$; `/*` s `frissítése */`
  end
end

A politikát a Q-tábla határozza meg.

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.

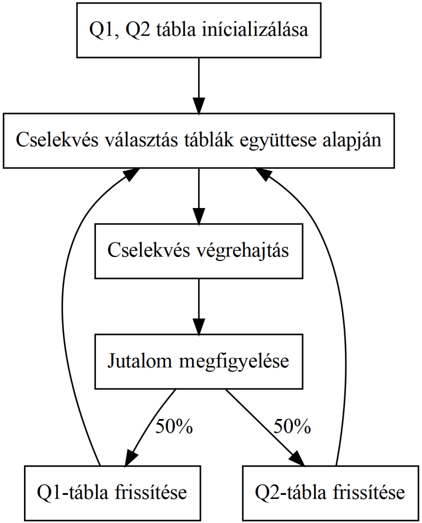
A Q-tanulással analóg dupla Q-tanulás nevű kettős tanulási algoritmus két részre osztja az időlépéseket, **minden lépésnél egy érmét feldobva**. Ha az érme fejre esik, a frissítés a következő:

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

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 ε-mohó politika a dupla tanulás esetében az egyes cselekvési értékbecslések **átlagára vagy összegére épülhet**.

Dupla Q-tanulás eljárása

Mivel két Q táblát kell nyilván tartania, az algoritmus memóriaigénye megkétszereződik. A becsült értékek viszont jóval torzítatlanabbak lesznek, mint az egyszeres Q tanulás esetében. A lépésenkénti számításigény nem növekszik az extra Q-tábla bevonásával.



Algoritmus 2: Dupla Q-tanulás algoritmusa

Input: α tanulási sebesség; $\varepsilon > 0$ hibahatár
$Q_1(s,a) \leftarrow random()$, $Q_2(s,a) \leftarrow random()$ for $s \in S$, $a \in A$;
$Q_1(s_T,\cdot) \leftarrow 0$, $Q_2(s_T,\cdot) \leftarrow 0$; `/* Terminális állapotok 0-ra állítása */`
for $i = 0 \rightarrow max_i$ **do**
  s inícializálása;
  while $s \neq s_T$ **do**
    $a \leftarrow \pi(s)$; `/*` Cselekvés választása π szerint, pl. ε-mohó `*/`
    a végrehajtása, r és s' megfigyelése;
    if $p > 0.5$ **then**

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

    else

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

    end
    $s \leftarrow s'$; `/*` s frissítése `*/`
  end
end

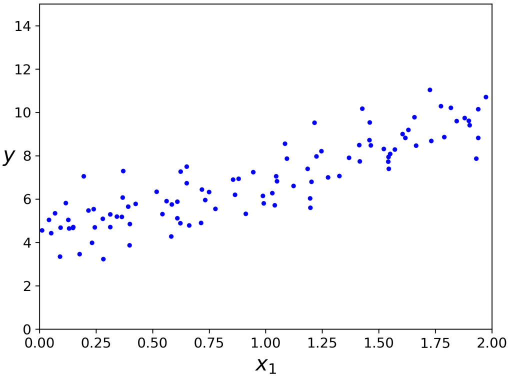
Függvény illesztés alapjai

A függvény illesztés eljárása szerint valamely x **független változóból** vett minta alapján szeretnénk előre jelezni egy y **függő változó** értékét azért, hogy feltárja az adatpontok közötti mintázatokat.

Két eljárása ismert:

- **Regresszió**: tárgya egy folytonos változó
- **Osztályozás**: tárgya egy diszkrét változó

A függvény illesztés eredménye a **modell**.



Függvény illesztés alapjai

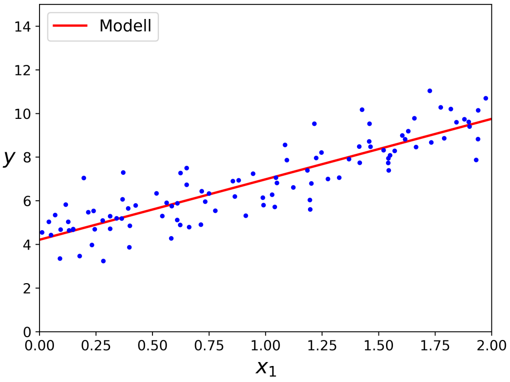
Az illesztett modell alakját és viselkedését a **paraméterei** határozzák meg, amelyek együtthatókként viselkednek a modell egyenletében. A lineáris modell egyenlete:

$$y_i = \theta_0 + \theta_1 x_i + \varepsilon_i$$

Ahol:

- θ_0: az y tengely metszéspontja, vagy eltolás
- θ_1: az egyenes meredeksége
- ε: a véletlen hiba, amit a modell nem tud előre jelezni

$\theta = [\theta_0, \theta_1, ..., \theta_n]$ a paraméterek vektora.

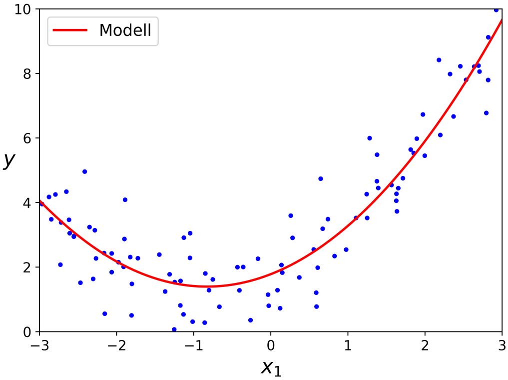


Függvény illesztés több paraméterrel

Függvényt lehetséges nemlineáris adatokra is illeszteni. Ebben a példában a minta adatpontok kvadratikusak, **nem írhatók le egy lineáris egyenlettel**. A modellnek ebben az esetben 3 paramétere van: θ_0, θ_1 és θ_2. Az illesztett modell egyenlete:

$$y = 1.78 + 0.93x + 0.56x^2$$

Tehát ebben az esetben $\theta_0 = 1.78$, $\theta_1 = 0.93$ és $\theta_2 = 0.56$.

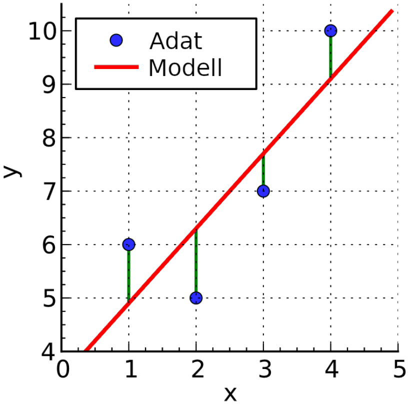


A költségfüggvény

A modell illesztő algoritmusok mindegyike úgy találja meg az optimális függvényt, hogy valamilyen költségfüggvényt minimalizál. A leggyakoribb ilyen költségfüggvény az **átlagos négyzetes hiba**:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x)_i)$$

- y_i: Megfigyelt adatpont
- $f(x)_i$: Modell által adott becslés



A gradiens

A függvény illesztés célja, hogy megtalálja azt a modellt, ami a legjobban illeszkedik az adatpontokra, tehát **minimalizálja a költségfüggvényt** (MSE).

Gradiens

Olyan vektor, amely megmutatja hogyan változik a függvény, és megadja a legnagyobb változás irányát minden dimenzióban.

$$df = \nabla f * dx$$

A gradiens segítségével meg lehet határozni, merre és mennyivel érdemes változtatni a paramétereket a célfüggvény értékének csökkentése érdekében.