Flappy Bird ágens

2020. November 27.

Feladat

Számos területen nehéz feladatba ütköznénk, ha egy ágens optimális viselkedését kéne előre "kézzel" leírni. Az állapotok száma olyan nagy lehet, hogy nehéz lenne minden egyes esetre jól működő kiértékelő függvényt írni. Erre a problémára egy megoldás a megerősítéses tanulás[1] használata. Ennek lényege, hogy egy tanuló rendszer folyamatosan kap egy visszajelzést arról, hogy jó vagy rossz, amit csinál. Ezt nevezzük jutalomnak (reward) vagy megerősítésnek (reinforcement). Például a snake játékban egy ilyen reward lehetne az, hogy 0 pontot kap minden egyes lépéskor, 1 pontot kap, ha felszed egy kaját és -1 pontot, ha a magába vagy a falnak ütközik. Az ágens célja tehát az lesz, hogy minél több jutalmat összeszedjen. Ezen viselkedés megtanulásához at egyik módszer a Q-tanulás [2]. Az ágens minden állapot-cselekvés párhoz megtanul egy cselekvésértéket. Ha egy adott állapotban a maximális cselekvésértékű cselekvést választja, azzal maximalizálni tudja a jutalma várható értékét.

Ebben a házi feladatban a feladat egy tanuló ágens implementálása, amely képes a Flappy Bird[3] világában minél több pontot összeszedni. Ehhez javasoljuk a táblázatos Q-tanulást használni. A feladatot leegyszerűsítettük úgy, hogy ne legyen túl nagy az állapottér, így beleférjünk az időbe és a memóriába. További egyszerűsítés, hogy az ágens egy kiértékeléskor ugyanazt a pályát kapja. Egy kiértékelés n tanulási iterációt és 1 epoch éles következtetést jelent. Egy iteráció azt lejenti, hogy az ágens megkapja az állapotot és az alapján visszaadja, hogy mit cselekszik; esetünkben 0, ha semmit és 1, ha ugrik. Ha az ágens "meghal", akkor a pálya elölről kezdődik. Egy epoch a pálya elejétől az ágens "haláláig" tart. Minden iteráció után van lehetősége tanulni az ágnesnek, ehhez megkapja az előző állapotot, az új állapotot, a cselekvését és a kapott rewardot. Az epochok és a tanítás végét is függvény jelzi az ágensnek.

A feladat megoldásához kiadunk egy ahhoz nagyon hasonló futtatókörnyezetet, mint amin élesben fog történni a kiértékelés. Kérjük, hogy első körben ezen történjen meg a házi feladat tesztelése. A tanítás $2 \cdot 10^5$ iteráción keresztül történik. Ezután következik a kiértékelés, amin legfeljebb 25 reward pontot lehet elérni, ez jelenti a maximális, 12 házi feladat pontot. A pontozás lineáris 0 reward $\rightarrow$ 0 hf pont és 25 reward $\rightarrow$ 12 hf pont között.

Beadandó

- A HFportálra python és java esetében is egy-egy fájlt kell feltölteni. Ehhez kiadunk egy-egy sablont, amiben a szükséges függvények megtalálhatóak. Ezeket nem érdemes módosítani, különben nem fog lefutni a beadás.
- Java esetén nem lehet használni semmilyen külső csomagot, csakis a java 8 beépített könyvtárait. A Q-táblázatos megoldáshoz előre megírtunk egy egyszerű osztályt a skeletonba, ezt lehet használni, de akár ki is lehet törölni. Az előre kiadott környezetben ezek a segédosztályok szét vannnak szedve külön fájlokba, a beadandó sablonba viszont beletettük, hogy egy fájlban legyen.

[1] http://project.mit.bme.hu/mi_almanach/books/aima/ch21s01
[2] http://project.mit.bme.hu/mi_almanach/books/aima/ch21s03
[3] https://flappybird.io/

- Python 3 esetében lehet használni a numpy-t (1.19.4-es verzió).
- Mindkét esetben a fájlt zip-elve kell feltölteni (fontos, hogy nem a mappát, amiben a fájl van, hanem magát a fájlt kell zip-elni).

Hasznos tudnivalók

A java kiértékelő a FlappyEvaluator.java $main()$ függvényével indul, a python kiértékelő pedig a flappy_evaluator.py file futtatásával. Annak érdekében, hogy a fejlesztési folyamatot érdekesebbé tegyük, készítettünk egyszerű GUI-t mind a java, mind a python megoldáshoz. Java esetében ez is elindul a tanítás végén a $main()$-ben (ezt ki lehet kommentezni). Python esetében a flappy_gui.py-t kell elindítani.

Az interfészről

konstruktor `FlappyAgent()`

- observation space size: integer lista/tömb a lehetséges maximális és minimális értékek különbségével [pozíció y koordinátája, függőleges sebesség, következő hengertől való vízszintes távolság, henger résének magassága] formátumban
- action space: integer lista/tömb a lehetséges cselekvésekkel ([0, 1], ahol 0 a "ne csinálj semmit" és 1 az "ugrás")
- number of iterations: tanulási iterációk száma ($2 \cdot 10^5$)

lépés `step()`
az ágens cselekvését kell visszaadni az aktuális állapot alapján, melyet javaban egy StateDTO objektum, pythonban pedig egy tuple reprezentál. Az értékek $[0; max]$ közé vannak normalizálva.

- java: birdPos (pozíció y koordinátája), birdSpeed (függőleges sebesség), tubeDistance (következő hengertől való vízszintes távolság), henger résének magassága
- python: (pozíció y koordinátája, függőleges sebesség, következő hengertől való vízszintes távolság, tubeHeight (henger résének magassága)

epoch vége `epochEnd()`, `epoch_end()`
minden epoch végén meghívódik

- epoch reward sum: az epochban szerzett rewardok összege

tanulás `learn()`
minden iterációban meghívódik a tanulás során, ez alapján lehet tanítani az ágenst

- old state: előző állapot
- action: az előző állapotban adott cselekvés
- new state: létrejött állapot
- reward: a kapott reward az új állapotban: -1 - "meghalt" az ágens, 1 - átlépett egy hengeren, 0 - egyébként

tanítás vége `trainEnd()`, `train_end()`
jelzi a tanítás végét, ez után kezdődik a kiértékelés

Javaslatok

- A feladat megoldható maximális pontszámmal a Q táblázatos módszerrel.
- Python-ban a futásidő kb. 15 mp, Javaban néhány mp, ha nincs GUI (a mi megoldásunkkal).
- Erősen ajánlott állítgatni az explore/exploit paramétert (akár tanítás közben is), valamint a Q-tanulás frissítési szabályának a paramétereit. A reward-okat is át lehet skálázni az ágensen belül.
- A `trainEnd()/train_end()` függvények is hint-ek.
- A futtatókörnyezetben mindig ugyanaz a pálya generálódik, nem csak kiértékelésenként. Az environment kódokban erre szolgál az "123" random seed. Ez debug célokat szolgál, nyugodtan át lehet írni. Természetesen a HFportál nem így generálja a pályákat, de kiértékelésenként ugyanaz a pálya generálódik.
- Bármi kérdés merül fel, a "Házi feladat" channelbe írjatok, mert akkor lehet, hogy hamarabb kaptok választ. Használjátok a @HF3 taget, mert akkor értesítést is kapunk. Igyekszünk minnél hamarabb válaszolni, de nekünk is év vége van.
- Érdemes a beadás előtti napok előtt elkezdeni a házi megoldását, mert a kiértékelés ideje megnövekszik, ha sokan adjátok be egyszerre; továbbá mi sem azonnal válaszolunk.

Egyéb fontos tudnivalók

- A megoldás forráskódja nem tartalmazhat ékezetes vagy nem ASCII[0:127] karaktert.
- A megoldásnak nem kell, hogy kimenete legyen, ezért ne legyen bennefelejtett debug print, mert errort fog jelezni a kiértékelő.
- A feltöltött megoldás megengedett futásideje CPU-időben bemenetenként 30 másodperc. Időtúllépés esetén a rendszer automatikusan leállítja a kód futását.
- A feltöltött megoldás összesen legfeljebb 400 MB memóriát allokálhat. Ezen érték túllépése esetén a rendszer automatikusan leállítja a kód futását.
- Minden bugot/javítást szívesen várunk, igyekszünk javítani.