Flappy Laser Chicken ágens

2021. november 22.

Környezet

A Flappy Bird[1] játék világában a csirkék generációkon át szinte gondtalanul élhették az életüket, néhány szembe jövő zöld oszloptól eltekintve semmilyen veszély nem leselkedett rájuk. Ez azonban gyökeresen megváltozott, amikor egy sötét napon egy idegen világból megérkeztek a sasok. E nagytestű madarak folyamatosan terrorizálták a náluk fizikailag jóval visszamaradottabb csirkéket, akik ennek következtében azóta a kihalás szélére kerültek. A kétségbeesett csirkék utolsó reménye a lézerfegyverek technológiája, amelyet sok generáción keresztül, a legnagyobb titoktartás mellett tökéletesre fejlesztettek. E házi feladat során ránk hárul a nemes feladat, hogy ezen eszközök felhasználásával segítsünk a csirkéknek bebizonyítani, hogy a sasok szervezetének fizikai felépítéséből származó előnyök ellensúlyozhatók technológiai fejlesztések segítségével.

Feladat

Ebben a házi feladatban a cél egy tanuló ágens implementálása, amely képes a Flappy Bird világának sasokkal és lézerfegyverekkel kiegészített változatában minél több pontot összeszedni. Ehhez **javasoljuk a táblázatos Q-tanulás használatát**. A feladatot leegyszerűsítettük úgy, hogy ne legyen túl nagy az állapottér, így beleférjünk az időbe és a memóriába. Egy kiértékelés $1.500.000$ tanulási iterációt és 100 epochnyi éles következtetést jelent. Egy iteráció azt jelenti, hogy az ágens megkapja a legfrissebb játékállapotot, és az alapján visszaadja, hogy mit cselekszik. Esetünkben **0: ha semmit, 1: ha ugrik, és 2: ha lő**. 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 $15 \cdot 10^5$ iteráción keresztül történik. Ezután következik a kiértékelés, amely 100 darab játékból (epoch-ból) áll, amelyek mindegyikén legfeljebb 25 reward pontot lehet elérni. Az ezen 100 darab játékban elért reward-ok átlaga képezi a pontozás alapját. A pontozás lineáris átlagos 4 reward $\rightarrow$ 2 hf pont és átlagos 9 reward $\rightarrow$ 12 hf pont között, 9 fölötti átlagos reward-ra természetesen a maximális, 12 pont jár. Amennyiben az átlagos reward 4-nél kevesebb, úgy a beadás nem ér pontot.

A játék folyamata és a jutalomrendszer

A játék során az ágens egy madarat irányít, amely folyamatosan jobbra halad. Alapból a *gravitációs erő* lefelé húzza, ezt ellensúlyozandó lehetősége van bármely időpillanatban *ugrani*, amellyel rövid idő alatt növelni tud a repülési magasságán. Amennyiben nekiütközik a pálya szélének (tehát "aljának" vagy "tetejének"), illetve valamely szembejövő oszlopnak, akkor meghal, és -1 büntetésben részesül, illetve minden oszlop után, amelyen sikeresen (tehát anélkül hogy meghalt volna) áthaladt, 1 jutalmat kap. Az ágens nagyjából $50-50\%$ eséllyel találkozhat szembe oszloppal, vagy sassal. Amennyiben sassal kerül szembe, úgy egy vízszintes irányú lövéssel megpróbálhatja likvidálni a ragadozó madarat, amely találat esetén meghal, és ekkor az ágens 1

[1] `https://flappybird.io/`

jutalomban (reward) részesül. Hogyha azonban a sas eléri az ágens által irányított madarat, akkor az ágens meghal, és -1 büntetésben részesül. Végül pedig, mivel a csirkék által kifejlesztett lézerfegyverek meglehetősen sok energiát fogyasztanak, így a legközelebbi sast el nem találó (tehát kvázi "fölösleges") lövések után az ágens -0.25 büntetést kap.

Beadandó

- A **Moodle**-re 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 11.0.7 beépített könyvtárait. A Q-táblázatos megoldáshoz előre megírtunk egy egyszerű osztályt a skeletonba, ezt lehet használni, de akár ki is lehet törölni.
- Python (3.7.3-as verzió) esetében lehet használni a numpy-t (1.16.2-es verzió).
- Mindkét esetben pontosan egy fájlt kell feltölteni: *FlappyAgent.java* vagy *flappy_agent.py* névvel.

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 objektum típusa, következő objektumtól való vízszintes távolság, következő objektum magassága] formátumban
- action space: integer lista/tömb a lehetséges cselekvésekkel ([0, 1, 2], ahol 0 a "ne csinálj semmit", 1 az "ugrás" és 2 a "lövés")
- number of iterations: tanulási iterációk száma ($15 \cdot 10^5$)

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), objectType (következő objektum típusa, tehát 0: oszlop és 1: sas), objectDistance (következő objektumtól való vízszintes távolság), objectHeight (következő objektum magassága)
- python: (pozíció y koordinátája, függőleges sebesség, következő objektum típusa (0: oszlop és 1: sas), következő objektumtól való vízszintes távolság, következő objektum 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 vagy kilőtt egy sast, -0.25 - lőtt egyet és célt tévesztett, 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. 35 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.
- Bármi kérdés merül fel, a "Kérdések és válaszok" 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 tudunk azonnal válaszolni.

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 jelezhet a kiértékelő.
- A feltöltött megoldás megengedett futásideje CPU-időben 40 másodperc (python) ill. 10 másodperc (java). Időtúllépés esetén a rendszer automatikusan leállítja a kód futását.
- A feltöltött megoldás összesen legfeljebb 500 MB (python) ill. 1000 MB (java) memóriát allokálhat. Ezen érték túllépése esetén a rendszer automatikusan leállítja a kód futását.
- A 2. beadástól kezdve 10 percenként lehet újra próbálkozni.
- Az újonnan beadott megoldás eredménye felülírja az előzőt.
- Minden bugot/javaslatot szívesen fogadunk, igyekszünk javítani.