**Tic Tac Toe haladó programozás beadandó  
Készitette: Hajós Imre**

**Kód futtatásának előkövetelményei:**

Python telepítése, környezeti változó beállítása

A colab nem támogatja a pygame-t ezért visual code vagy hasonló kódszerkesztó telepítése, és a python extension telepítése.

A pygame python modul telepítése: pip install pygame

linearregression használatához: pip install scikit-learn

A project mappában legyenek ott a kódfájl mellet a szükséges png fájlok.

**Project munka:**

Feladatnak egy egyszerű tictactoe játék elkészítését választottam pygame segítségével, amelyben bot ellen lehet játszani. A menüben lehet választani a kezdési lehetőséget, illetve a játékpálya alatt található egy státusz ablak rész, amelytől visszajelzéseket kapunk a játékállapottal kapcsolatban.

Program indításával megjelenik a játék menü felülete, itt billentyűzetről kell választani 1-es vagy 2-es gombot megnyomva, hogy akarunk kezdeni. A 4-5-6-os gombal lehet állítani a játék nehézséget, illetve lehet választani betanított model elleni játékot. Ha választottunk kezdő pozíciót, akkor meghívódik a játék felület, amely a játékpályát és a státusz ablakrészt tartalmazza. A felhasználó játékos lépéseit a játékfelületre való kattintással olvassa be. Játékos megjegyzések: célszerű megvárni a botlépését, mert ha közben kattingatunk összevissza, nem várt módon olvassa be a lépést. Minden felület módosító műveletnél frissül alul az állapot szöveg és a játék végén kiírja az eredményt, valamint visszatér a játékmenübe, ha végeztünk ki x-eljük az ablakot, vagy leállítjuk a futtatást.

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás A képen képernyőkép, szöveg, Betűtípus, szám látható

Automatikusan generált leírás A képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírás

**Pygameröl röviden:**

A Pygame egy platformfüggetlen Python-modul, amit videójátékok fejlesztésére hoztak létre. Magában foglalja a számítógépes grafikákat, a hang- és programkönyvtárakat, amiket a Python programozási nyelvre fejlesztettek ki.

A képen szöveg, szoftver, Multimédiás szoftver, képernyőkép látható

Automatikusan generált leírás

Ez egy nagyon alap pygame alkalmazás, ami tartalmazza a három fő elemet:  
- a játék ablakot  
-a játék loopot  
-az esemény kezelőt

Importáljuk pygame modult, a pygame.init() inicializálja.  
A screen tartalmazza a játék ablakot, ami paraméterként átveszi a szélességet és a magasságot.  
A ciklusban fut az eseménykezelő, ami felveszi a billletyűnyomásokat vagy egérkattintásokat, és ha megvan határozva valami művelet hozzájuk, akkor végrehajtja.

**Kódmagyarázat:**

A képen szöveg, képernyőkép, szoftver látható

Automatikusan generált leírás

draw és game\_started változók állapot követő flagek. TTT egy 3x3as lista (mátrix), ami a játéktáblát reprezentálja, folyamatban lévő játék állásával folyamatosan frissül. winner a gyöztest tartalmazza, ha van. Néhány segédváltozó a kevesebb gépelésért. Inicializáljuk a pygame modulokat, létrehozzuk a programablakot, megcímezzük, és betölltjük a képeket, és méretezzük.

**függvények:**

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

szövegkiíráshoz segéd függvény, átveszi a szükséges paramétereket, font beállítja a betű stílust és a méretet, text\_surface ráhúzza a szövegre és beállítja a színt, a screen.blit „négyzetbe” foglalja és kirajzolja egy off-screen bufferbe. A képernyőn megjelenéshez updatelni kell az ablakot.

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

kitölti az ablakot fehérrel és kiírja a választási lehetőségeket, draw\_text függvény hívások után meg kell hívni a pg.display.update() fv-t, amely frissíti az ablakot a változásokkal. A váltózoktól függően jelenik meg a játéknehézség vagy belehet kapcsolni a betanított modelt.

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

Létrehozza a játékfelületet a fill() és a draw.line() azonnal módosítja a felületet, utóbbi paraméterezésével készítjük el a táblaszerkezetet, és az alatta elhelyezkedő 100pixel magas status ablakot, amit azonnal kitölt.

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

ternary operátorral meghatározzuk az üzenetet, közben ellenőrizzük a döntetlen állást, és a draw\_textel kiiratjuk a meghatározott pozícióban.

A képen szöveg, képernyőkép, szoftver látható

Automatikusan generált leírás

Ezzel a fvel minden megtett lépés után ellenőrizzük a játék állapotát. A TTT mátrixban tároljuk a játék állását. Ha vízszintesen vagy függőlegesen vagy átlósan minden cellában ugyanazt a karaktert találjuk(ami nem üres) akkor beállítjuk a winner változót az adott karakterrel. a másik kimenetel, hogy egyik cella se üres és nem elégíti ki az előző feltételeket és beállítja draw flaget.  
Valamelyik teljesülésével a programlogika leállítja a mérközést.

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

segéd fv lépés kirajzoláshoz. global definiálással, a fven belül módosítani tudjuk a globális változókat. A sor és oszlop paraméterekből koordinátákat képez, frissít a „játéktáblát”, a soron következő játékost, és kirajzolja az aktuális játékos karakterét az adott pozícióba

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

Kattintás pozícióját beolvassa, feldolgozza,ellenőrzi hogy ürese, kirajzoltatja, ellenőrzi van-e győztes, ellenőrzi a flageket és meghívja a bot fv-t.

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

Ez a funkció biztosítja, hogy a játék újrakezdéskor minden beállítás visszaálljon az alaphelyzetbe.

**Bot műkődése:**

**Segédfügvények:**

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

best\_score felvétele, amibe tároljuk a legtöbb pontot, minden lehetséges lépéshez lefuttatjuk a minimax fv-t, a TTT módosítása nem okoz problémát mert úgyis visszaállítjuk a változtatásokat, először false flaget állítunk be, az ellenfél játékos lépését akarjuk minimalizálni, így szímuláljuk végig az összes lépés sorozatot, végül a legjobb lépést visszakapja botMove() . Listába összegyűjtöm az eddigi legjobb lépéseket, frissítem, hogy ténylegesen a legjobb lépések közül válasszon. Így nem csak egy lépés sorozatot fog végig csinálni, hanem változatosabb lesz.

A képen szöveg, képernyőkép, képernyő, szoftver látható

Automatikusan generált leírás

átveszi a játéktáblát, a mélységet, és hogy a bot lépését vagy játékos lépését követjük, illetve az eddig legjobb eredményt tároljuk, ennek főleg korlát beállításánál van szüksége.  
Először le ellenőrzi a játék állapotot, hogy van-e győztes vagy döntetlen, majd elvégzi miden lehetséges lépésre a szimulációt, attól függően hogy épp a játékos vagy a bot lépését nézi minimalizál vagy maximalizál. Elkerüli azokat a lépéseket, ami előnyösebb a játékosnak. Rekurzívan meghívódik, amig nem kap egy végeredményt. A nagyon alacsony mélységikorlát megegyezik egy egyszerű randomizációval, mert egyszerűen nem tudja feltérképezni a lépés lehetőségeket., hogy értelmes következtetést vonjon le, főleg ezért adom vissza az eddigi eredményt, hogy ne None-t adjon, vissza és a lépés listából random tudjon választani. Mondjuk közepes nehézséghez egy 5 körüli korláttal, már a legtöbb lépést végig vezeti, és ezekböl a lépéshalmazból már az elejétöl tud választani. 9-nél már végig tudja szimulálni az összes lépést, ezért nem lehet legyőzni. és egy lépést hibázol vesztesz.

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

ellenőrzi, hogy van e játék végi állapot, kulcsfontosságú a lépéssorozat kiértékelése szempontjábol.

A képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírás

megkapja az egyik legjobb lépést, és kiíratja.

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

Akartam gépi tanuláshoz valami nagyon egyszerű megoldással próbálkozni. A bemeneti értékek tábla állapotok, amibe az első érték azt jelenti, hogy hányszor van 2 karaktere egy sor/oszlop/átlóban, a második érték ugyanez a játékosnak, a harmadik érték a bot kontrolálja a középső cellát, az utolsó pedig mennyi üres sarok cella van. A kimeneti érték pedig melyik az előnyösebb állapot, mert a az összes lépésböl a legjobbat pontszámut választjuk ki, igaz l a lineáris regresszió célja egy folytonos kimenetel előrejelzése, itt most azt próbáljuk elérni, hogy egy "pontszámot" jósoljon meg minden lehetséges lépéshez. Minél több adattal lehet pontosítani a modelt, vagy előnyösebb mértékekkel.  
Létrehozzuk a modelt, és ráillesztjük a mintát.

Az extract\_features kiszámolja a szükséges adatokat a jelenlegi tábla állapotra, ezt minden egyes lehetséges lépésre elvégzi és a legjobbat küldi tovább a botMove()-nak.

**Q learning-es megoldás**

1.lépés bot tanítása

A képen szöveg, képernyőkép, szoftver látható

Automatikusan generált leírás

Automatikus játék szimulálás a q-table feltöltéséhez. Inicializáljuk a táblát, véletlenszerűen eldöntjük a kezdő játékost és addig futtatunk egy játékot míg játék végi állapotot nem kapunk, az episode fv bemenet megadja, hány játékot szimuláljon le. Az akutális játékba az action függvényvissza ad egy lehetséges lépést, beáállítjuk a szükséges változókat az update q tables meghívásához, ami frissíti a q-tablet. Az exploration ratet episodeonként csökkentjük, ami azt határozza, meg h új lépéseket fedezzen fel vagy az eddigieket optimalizálja. A reward függvény pontozza a játékállapotokat: nyerés 1, vesztés -1, döntetlen 0, kis lépés büntetés -0,5, hogy rövidebb lépés sorozatot válasszon.

A képen szöveg, képernyőkép, szoftver látható

Automatikusan generált leírás

Az aktuális játék állapotban lehetséges lépéseket fedezi fel. Bele hardcodeoltam, hogy a nyertes lépéseket azonnal megtegye, vagy blokkolja az ellenfél nyerését. Epsilon-greedy policy bevezetése, exploit, azaz az eddigi lépésekre támaszkodik, vagy exploration, azaz új lépést próbál ki. A tanítási folyamatban az exploration rate 1-ről indul és folyamatosan szorozzuk a decayel pl 0,999, szóval eleinte új lépéseket próbál ki, majd fokozatosan nagyobb esélyel kezd választani a q tableből.(training fv-ben folyamatosan csökkentjük az exploration ratet).Az aktuális állapotban összeslehetséges lépésre összeszedjük az értékeket(ha még nem létezik 0), majd a legjobb értékű lépésekből választunk.(ha még a táblában nem létezik, akkor az összes értéke 0, és egy random lépést dob vissza)

A képen képernyőkép, szöveg, Betűtípus látható

Automatikusan generált leírás

Alap képlet(Bellman egyenlet):  
A képen Betűtípus, kézírás, kalligráfia, tipográfia látható

Automatikusan generált leírás

A fv átveszi az állapotot, a lépést, a kiszámolt jutalmat, a lépés utáni állapotot, és az ebből lehetséges lépéseket.  
Megkeressük az új állapot legoptimálisabb lépés lehetőségének az értékét, az állapot-lépés pár értéket(ha nem létezik akk 0), a képlet alapján kiszámoljuk, és frissítjük a táblát.  
A learning rate meghatározza, mennyire módosítjuk a jelenlegi értéket, a discount faktor súlyozza jövőbeli értékeket, a reward az azonnali jutalom  
A cél, hogy a q-tábla fokozatosan közelítsen az optimális értékekhez, így az ágens egyre jobb döntéseket tudjon hozni a tanulás során. Ha sokáig tart a tanítás lehet csökkenteni az epsodeokat vagy fájlba lementeni

Összefoglalva:

Beállítjuk a Q-learning paramétereket:  
LEARNING\_RATE = 0.1 # Alpha tanulási ráta  
DISCOUNT\_FACTOR = 0.9 # Gamma kedvezményes faktor  
EXPLORATION\_RATE = 1.0 # Epsilon felfedezési rta  
EXPLORATION\_DECAY = 0.999 # csökkenés  
MIN\_EXPLORATION\_RATE = 0.01 # minimum

Minél többször futtatjuk a tréning során a botot, annál pontosabb és optimalizáltabb Q-táblát kapunk. Kezdetben az ágens felfedezi a lehetséges állapot-lépés párokat, hogy minél több információt gyűjtsön a környezetről. Ahogy az exploráció mértéke fokozatosan csökken, az ágens az addig megszerzett adatok alapján egyre inkább az optimális lépéseket részesíti előnyben.

A valódi játék során az ágenst alacsony explorációs értékkel indítjuk, hogy a Q-táblából választott lépéseire támaszkodjon, így a korábbi tanulás eredményét hasznosítva a lehető legjobb döntéseket hozza.A képen szöveg, képernyőkép látható

Automatikusan generált leírásA program „törzse” megnyitjuk a menüt és elindítjuk a loopot, amiben az eseménykezelő fogadja az inputokat. Ha megnyomjuk az ablakon az x-et kilép. A menüben az egyest billentyűzetről megnyomva beállítja a flageket és megnyitja a játék majd a legalsó elif ág fog futni amíg folyamatban van a játék. Ha a bot kezd annyi a különbség, hogy ahhoz megfelelően állítja be a flageket és először meghívja a botMove()-t mert egyébként mindig a UserClick után hívódik meg.  
A játék végén reseteli a flageket és felugrik a menü. Kiegészíttem a nehézség a beállításával, és a másik fajta bot bekapcsolásával, újra meghívom e menüt, hogy fríssüljön az ablak az aktuális beállítással.

források:  
<https://www.youtube.com/watch?v=y9VG3Pztok8&t=301s&ab_channel=CodingWithRuss>  
<https://www.youtube.com/watch?v=IL_PMGVxEUY>  
<https://en.wikipedia.org/wiki/Tic-tac-toe>  
<https://www.neverstopbuilding.com/blog/minimax>  
https://www.geeksforgeeks.org/q-learning-in-python/