Specifikacija projekta

Katarina Vujović RA 81/2022 - Osnovi računarske inteligencije

# Definicija problema

Problem koji ovaj projekat rešava je razvoj inteligentnog agenta koji može da donosi odluke u dinamičnom i nepredvidivom okruženju koje je tipa Side-Scroller Shooter-a kao što je Gradius.

Agent ima zadatak da preživljava izbegavanjem prepreka i protivničkih projektila, dok pokušava da maksimizuje svoje poene uništavanjem protivnika u okruženju u kom se agent konstantnom brzinom kreće ka desnoj strani ekrana na kojoj nailazi na protivnike i prepreke. Može će da se kreće u 8 pravaca (gore, dole, levo, desno, i dijagonalno) i ispaljuje projektile. Kraj igre je kada je agent pogođen protivničkim projektilom ili sudaren sa preprekom.

# Motivacija

Motivacija za rešavanje ovog problema je istraživanje ponašanja i napretka agenta koji pod pritiskom događaja u njegovom okruženju konstantno mora da balansira odnos između rizika i nagrade. Simulacijom ovog problema može se doći do zaključaka korisnih za oblasti u kojima je ključno da sistem uračuna višestruke ciljeve (u kontekstu projekta, to su poeni i preživljavanje) kao i neposredne opasnosti koje su pred njim, poput autonomnih vozila i dronova.

# Struktura projekta

Projekat se sastoji iz 2 glavna dela: video igre i RL modela.

Video igra je napravljena u python-u koristeći pygame. Osnovna klasa je klasa *Game* u kojoj se sabira sva logika sa pokretanje igre.

Postoji 5 entiteta: *PlayerSpaceship*, *EnemySpaceship*, *PlayerBullet*, *EnemyBullet*, i *Asteroid*.

Odgovornosti klase *Game* su grupisane u 3 *manager* klase: *DrawManager*, *CollisionManager* i *SpriteManager*.

*DrawManager* je zadužen za svu logiku crtanja potrebnu za pygame.

*CollisionManager* je zadužen za logiku kolizija između entiteta.

*SpriteManager* je zadužen za ažuriranje i generisanje entiteta.

Postoji i klasa *Settings* u kojoj su definisane vrednosti nekih od parametara entiteta.

RL model je napravljen kao *custom gymnasium* okruženje. Klasa koja modeluje ovo okruženje je klasa *Enivronment*. Sadrži definicije funkcija neophodnih da *gymnasium* okruženje može da se trenira, i pomoćne funkcije.

Prostor akcija se sastoji od jednog niza sa 5 binarnih cifara koje obeležavaju da li se agent kreće ka gore, dole, levo, desno i da li puca.

Prostor opservacija se sastoji od koordinata (x, y) sledećih entiteta: agenta, niza agentovih ispaljenih metaka, niza neprijatelja, niza neprijateljevih metaka, niza asteroida. Veličine nizova su ograničene na moguć broj entiteta koji se može naći na ekranu.

Okruženje računa nagradu po koraku na sledeći način:

Model dobija značajnu pozitivnu nagradu za pogođenog neprijatelja, i manju nagradu za gađanje blizu neprijatelja kako bi ga navodila da krene da cilja ka neprijateljima.

Dobija manje negativne nagrade za kretanje blizu asteroida, neprijatelja ili njihovih metaka.

Treniranje agenta vrši funkcija *train().* Koristi se *python* biblioteka *stable-baselines3* i algoritam PPO (Proximal Policy Optimization).

Model koristi prethodno navedeno *Environment* *gymnasium* okruženje. Čuvaju se trenirani modeli i *tensorboard* izveštaji.

Kod za treniranje i kod za posmatranje agenta kako igra bez učenja je podeljen u dve različite funkcije: *train()* i *play()*. Dodatno postoje funkcije *get\_train\_args()* i *get\_play\_args()* koje se koriste za definisanje pomoćnih argumenata pri pokretanju projekta.

# Pokretanje projekta

Projekat se pokreće iz *src* foldera u modu treniranja, ili u modu igranja.

Za pokretanje moda treniranja koristi se komanda:

*python -m PPO.train -m [ime\_modela]*

Opciono se može dodati vreme treniranja u koracima sa *-t* argumentom.

Za pokretanje moda igranja koristi se komanda:

*python -m PPO.play -m [ime\_modela]*

# Rezultat projekta

Rezultat projekta su 2 trenirana modela. Na njima su korišćeni podrazumevane vrednosti za parametre PPO algoritma, i različiti sistemi nagrađivanja koji so donosili različite rezultate. Međutim, zajedničko im je da su krenuli da stagniraju sa napretkom nakon 1 – 1.5h treniranja.

Model koji je bio manje nagrađivani za ubijanje neprijatelja i pucanje u njihovoj blizini su pribegavao gornjoj ivici ekrana, dok je model koji je bio manje kažnjivan i više nagrađivan za pucanje blizu neprijatelja pribegavao gornjoj ivici prozora. Zaključak je da je nasumično generisanje video igre nedovoljno zaista nasumično, te se na gornjoj ivici ekrana češće generišu neprijatelji i asteroidi, dok se na donjoj oni ređe generišu, te pribegavanje gornjoj ivici pristaje agresivnijim modelima, a donjoj pasivnijim.

# Tehnologije

Python: Jezik u kom je razvijen projekat

Stable\_baselines3: Biblioteka za implementaciju RL algoritama

Gymnasium: API standard za RL okruženja

Pygame: Biblioteka za pravljenje video igara

TensorBoard: Alat za posmatranje napretka učenja