Vježba 6 — Podržano učenje (engl. *Reinforcement learning* - RL). Duboko Q-učenje

Priprema

1. Objasnite razliku između nadziranog, nenadziranog i podržanog učenja (engl. *Reinforcement learning* - RL).
2. Objasnite osnovne pojmove u okviru RL: prostor stanja i prostor akcija, funkciju nagrade te razliku između funkcije vrijednosti stanja i funkcije vrijednosti akcije.
3. Na koji se način tipično rješava kompromis između eksploatacije i istraživanja okruženja u RL algoritmima?
4. Objasnite osnovni algoritam Q-učenja. Na koji način se ažuriraju Q-vrijednosti?
5. Objasnite strukturu osnovnog dubokog Q-učenja (engl. *deep Q-learning* - DQN). Čemu služe replay buffer i target mreža?
6. Upoznajte se s okruženjima CartPole i LunarLander u okviru gymanisum biblioteke. Koje akcije i stanja ima pojedino okruženje te kako je definirana nagrada?

Opis zadatka

U okviru ove vježbe potrebno je istrenirati agenta temeljnog na duboko Q-učenju za dva jednostavna okruženja (CartPole i LunarLander) u gymanisum biblioteci. Na raspolaganju je sljedeći programski kod:

DQN_agent.py - implementacija DQN algoritma
DQN_training.py - skripta koja pokreće treniranje DQN agenta na odabranom okruženju
DQN_testing.py - skripta koja pokreće testiranje DQN agenta na odabranom okruženju

Pre-lab pitanja

1) Opišite CartPole i LunarLander okruženja. Zašto je DQN pogodan za ova okruženja?

CartPole, okruženje za podržano učenje u kojem agent upravlja kolicima i balansira štap uspravno što dulje moguće.
Imamo stanja sa 4 varijable: položaj kolica, brzina kolica, kut motke i kutna brzina motke. Akcije su diskretne: pomak lijevo i desno.
LunarLander, složenije okruženje, agent upravlja lunarnim modulom koji treba sigurno sletjeti na označeno polje. Stanje sa osam kontinuiranih varijabli: položaj (x, y), brzina (x, y), kut i kutna brzina, dvije boolean vrijednosti lijevog i desnog kontakta s tlom; a akcije su diskretne: nema potiska, glavni mlaz, lijevi i desni bočni mlaz.

2) Analizirajte programski kod u DQN_agent.py. Opišite DQN strukturu koja se koristi.

U datoteci `DQN_agent.py` implementiran je klasični Deep Q-Network agent. Višeslojna potpuno povezana neuronska mreža koja prima stanje okruženja i vraća Q-vrijednosti za svaku moguću akciju. Periodički ažurira kako bi se stabiliziralo učenje. Ima replay buffer, memoriju fiksne veličine u koju se pohranjuju prijelazi oblika state, action, reward, next_state, done. Agent bira slučajnu akciju s vjerojatnošću epsilon kao faktor istraživanja. Što veći epsilon, veća šansa za istraživanje, dok na epsilon nula je iskorištavanje. Sa bellmanovo ažuriranjem, mreža se trenira minimizacijom MSE gubitka između predviđene Q-vrijednosti i ciljne Q-vrijednosti iz target mreže.

3) Analizirajte DQN_training.py i DQN_testing.py skripte. Koja je temeljna razlika između ove dvije skripte.

DQN_training.py služi za treniranje mreže. Simulira fizički model generirajući podatke iz okružja te ih koristiti za učenje mreže. U početku epsilon je velik, nasumičnost optimizacije je velika te istražuje okolinu. S vremenom se epsilon smanjiva kako bi više iskoristio predikcije i stabilizirao treniranje da sporije konvergira.

DQN_testing.py služi za vrednovanje kvalitete. Provodi simulacije kao i DQN_training.py. Epsilon je na nuli, nema istraživanja. Parametri se ne mijenjaju.

4) Procijenite vrijednosti svih hipeparametara DQN algoritma i očekivano vrijeme treniranja za okruženje CartPole i LunarLander.

batch_size=128,
learning_rate=0.001,
epsilon_start=1.0,
epsilon_min=0.01,
epsilon_decay=0.995,
gamma=0.99,
buff_capacity=10000,
update_target_steps=50

Za CartPole treniranje bi bilo oko 10tak minuta. LunarLander 20tak minuta zbog kompleksnijeg okružja i prostora stanja.

Post-lab pitanja

1) Navedite koje ste konačne vrijednosti hiperparametara koristili prilikom treniranja DQN algoritma za CartPole okruženje i LunarLander okruženje.

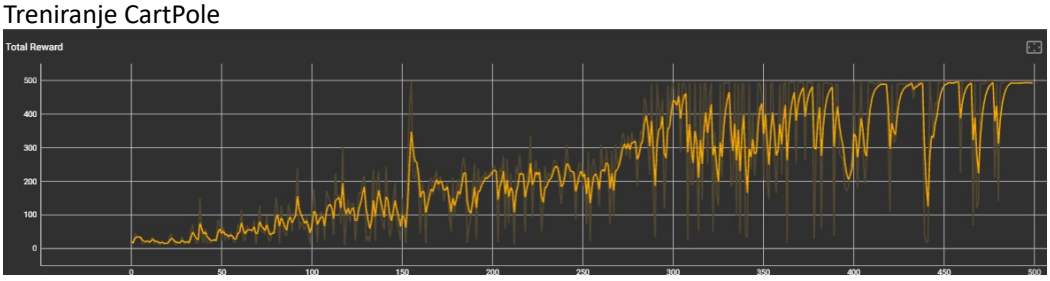
Cart Pole DQN:

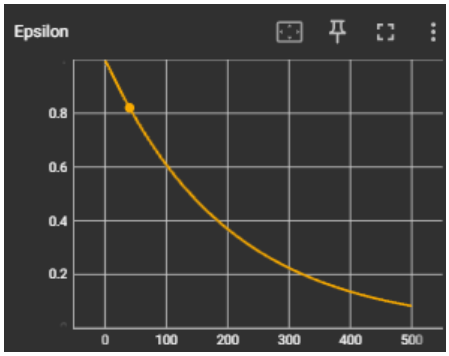
```
batch_size=128,
learning_rate=0.0003,
epsilon_start=1.0,
epsilon_min=0.01,
epsilon_decay=0.995,
gamma=0.99,
buff_capacity=10000,
update_target_steps=50,
```

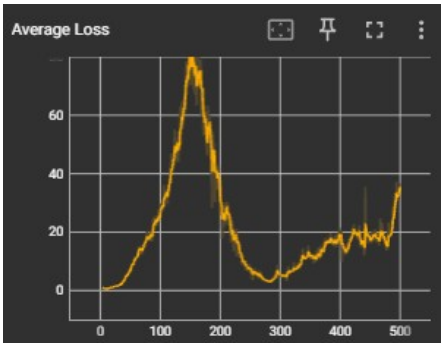
```
Lunar Lander DQN:
    batch_size=128,
    learning_rate=0.00025,
    epsilon_start=0.9999,
    epsilon_min=0.01,
    epsilon_decay=0.995,
    gamma=0.99,
    buff_capacity=10000,
    update_target_steps=100,
```

2) Prikažite tijek treniranja DQN algoritma za CartPole okruženje i LunarLander okruženje (npr. priložite vizualizacije iz Tensorboard). Komentirajte dobivene rezultate.

Treniranje CartPole

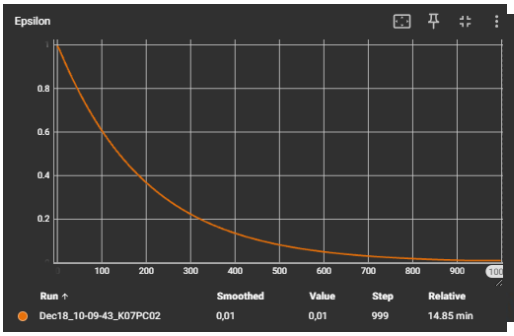


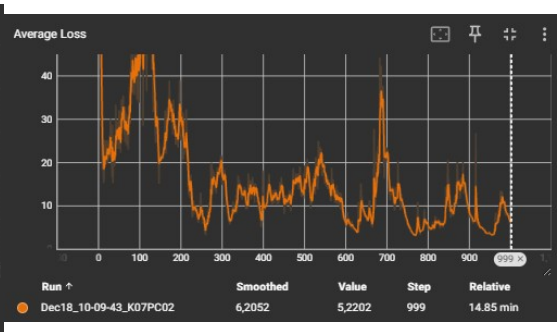


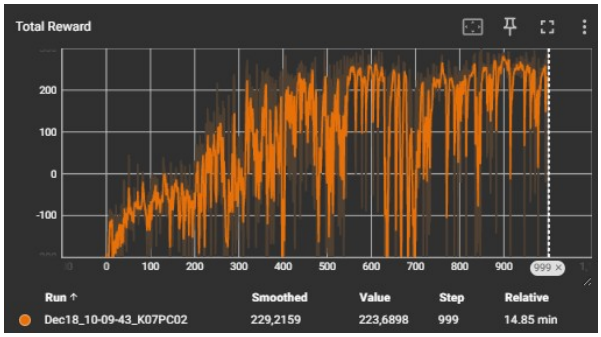


Nakon 400 koraka, treniranje konvergira usporeno prema maksimalnoj nagradi. Pomalo nagrada pada zbog istraživanja slučajnih parametara.

Treniranje LunarLander







Treniranje je vrlo nestabilno. Izgleda da je imao najbolju nagradu oko 890tog koraka.
Oko 600tog koraka nagrada značajno pada.

3) Pokrenite benchmark_DQN.py za konačne modele za oba okruženja na 1000 testnih epizoda te navedite rezultate koje ste postigli.

CartPole

Mean reward: 500.000
Std reward: 0.000
Median reward: 500.000

Agent savršeno rješava zadatak u svim epizodama.

LunarLander:

Mean reward: 226.465
Std reward: 69.587
Median reward: 242.680

Agent postiže uspješno slijetanje u većini epizoda. Uspjeh slijetanja je varijabilan.

4) Osvrnite se na praktičnu primjenu dubokog Q-učenja. Navedite barem jedan konkretan scenarij gdje je ovakav algoritam pogodan i zašto? Koju su izazovi prilikom implementacije ovog algoritma u tom slučaju?

Mogu se primijeniti za autonomnu kontrolu robota ili dronova u svrhu navigacije i manipulacije s objektima u skladištu ili autonomno slijetanje drona.
Diskretan skup akcija (kretanje, potisak motora). Složeni i kontinuirani senzori i kamere. Mogućnost učenja optimalne politike bez eksplicitnog fizičkog modela okruženja.

Izazovi: Velika količina podataka potrebna za treniranje. Rizično istraživanje tijekom učenja. Nestabilnost učenja i osjetljivost na hiperparametre. Razlika između simulacije i stvarnog svijeta.