

# Лабораториска вежба 2 - Учење со ПОТТИКНУВАЊЕ

## Gym - Python библиотека




Gym е Python библиотека која поддржува развој и споредба на алгоритми за учење со поттикнување. Документацијата на оваа библиотека е достапна на следната [страница](#). Инсталација на библиотеката:

```
pip install gymnasium
```





Околина може да се креира на следниот начин:

```
env = gym.make(env_name)
```

каде `env_name` е името на околината. Библиотеката gym обезбедува голем број на [околин](#) кои може да се користат. Пример за околин:

-  FrozenLake-v0
-  Taxi-v3
-  MountainCar-v0

Некои атрибути на околината кои може да се користат:

-  Action\_space - валидни акции
-  Observation\_space - валидни состојби
-  Reward\_range - ранг на наградата која агентот ја добива
-  Env - дополнителни информации за околината

Околината се ресетира со повик на функцијата `reset()`:

```
env.reset()
```

Оваа функција ја враќа почетната состојба.

Со повик на функцијата `render()` може да се визуелизира тековната состојба:

```
env.render()
```

За движење низ околината се користи функцијата `step`:

```
state, reward, done, info = env.step(action)
```

каде `action` е акцијата која се презема. Оваа функција како резултат враќа 4 вредности:

- State - објект кој ја претставува следната состојба на околината
- Reward - награда која се добива по преземање на акцијата action
- Done - информација дали епизодата е завршена (ако е завршена, околината треба да се ресетира)
- Info - дополнителни информации специфични за конкретната околина

## Задачи

### Задача 1 (15 поени)

За околината „FrozenLake-v1“ одредете ја Q функцијата со **q learning** со следните вредности за `discount_factor`: 0.5 и 0.9, и `learning_rate`: 0.1 и 0.01. Тестирајте со различни вредности за број на епизоди. Може да ја користите имплементацијата достапна во скриптата **q\_learning.py**. При учење на Q табелата во секој чекор на случаен начин избирајте акција која треба да се преземе.

Визуелизирајте го движењето на агентот низ оваа околина. Тестирајте ја добиената функција во 50 и 100 итерации. Колкав е просечниот број на чекори потребни за стигнување до целта? Колкава е просечната награда? Дали резултатите се менуваат ако се користи  $\epsilon$ -greedy политика?

### Задача 1 (15 поени)

За околината „Taxi-v3“ одредете ја Q функцијата со **q learning** со следните вредности за `discount_factor`: 0.5 и 0.9, и `learning_rate`: 0.1 и 0.01. Тестирајте со различни вредности за број на епизоди. Може да ја користите имплементацијата достапна во скриптата **q\_learning.py**. При учење на Q табелата во секој чекор избирајте ја најдобрата акција која треба да се преземе.

Визуелизирајте го движењето на агентот низ оваа околина. Тестирајте ја добиената функција во 50 и 100 итерации. Колкав е просечниот број на чекори потребни за стигнување до целта? Колкава е просечната награда? Дали резултатите се менуваат ако наместо најдобрата акција се избира акција на случаен начин при учење на Q табелата? Дали резултатите се менуваат ако се користи  $\epsilon$ -greedy политика?

### Задача 3 (20 поени)

За околината „MountainCar-v0“ одредете ја Q функцијата со **q learning** со користење  $\epsilon$ -greedy политика без  $\epsilon$ -decay. Тестирајте со различни вредности за број на епизоди. **Напомена:** потребно е да направите дискретизација на состојбите. Може да ја користите имплементацијата достапна во скриптата **q\_learning.py**.

Визуелизирајте го движењето на агентот низ оваа околина. Тестирајте ја добиената функција во 50 и 100 итерации. Колкав е просечниот број на чекори потребни за стигнување до целта? Колкава е просечната награда? Дали резултатите се менуваат ако се користи  $\epsilon$ -decay?