Reinforcement Learning

Building a Complete RL System

Samuel Garcin
10 February 2023
Lecture Outline

- What is Gym?
- How to implement your own environment?
- How to implement a RL algorithm?
- How to evaluate your results?
- Demonstration
OpenAI Gym
What is Gym? (Brockman et al., 2016)

- Open source interface for sequential decision processes
- Originally developed by OpenAI Research Lab, currently maintained by the Farama Foundation
- Collection of RL environments
- Standardised interface for RL environments

Can be installed with

```
pip install gym
```
Lots of Interesting Environments! (Brockman et al., 2016)
And many more... (Vinyals et al., 2017; Johnson et al., 2016; Kauten, 2018)
Gym Interface

- `gym.make(<environment_name>)` ➞ gym environment
  Create a gym environment

- `env.reset()` ➞ observation
  Resets environment for a new episode

- `env.step(action)` ➞ observation, reward, done, info
  Take an action and observe new information

- `env.render()`
  Render a visualisation of the current environmental state

- `env.close()`
  Close created environment
Gym Example Snippet

Gym control flow

```python
env = gym.make('CartPole-v0')
obs = env.reset()
done = False
while not done:
    env.render()
    action = agent.choose_action(obs)
    next_obs, reward, done, info = env.step(action)
    obs = next_obs
env.close()
```
Example: Taxi-v3 Environment

- Goal: Pickup passenger and drop it off at destination
- Reward: +20 for successful delivery, −1 at each timestep, −10 for illegal move
- Challenge: navigate gridworld

- Gridworld with 5 × 5 map
- R, G, Y, B - locations
  - B - passenger
  - Y - destination
  - taxi

Observations ∈ [0, 499] including taxi row and col, pass. and dest. index
Example: Taxi-v3 Environment

- Gridworld with 5 × 5 map
- R, G, Y, B - locations
  - B - passenger
  - Y - destination
- | - taxi
- Observations ∈ [0, 499]
  - including taxi row and col,
  - pass. and dest. index
Example: Taxi-v3 Environment

- **Goal:** Pickup passenger and drop it off at destination
- **Reward:**
  + 20 for successful delivery,
  - 1 at each timestep,
  - 10 for illegal move
- **Challenge:** navigate gridworld
  - Gridworld with 5 × 5 map
  - R, G, Y, B - locations
  - B - passenger
  - Y - destination
  - - taxi
- **Observations ∈ [0, 499]** including taxi row and col, pass. and dest. index
- **Actions:**
  South, North, East, West, Pick, Drop
Example: Taxi-v3 Environment

- Goal: Pickup passenger and drop it off at destination
- Reward: +20 for successful delivery, −1 at each timestep, −10 for illegal move
- Challenge: navigate gridworld

- Gridworld with 5 × 5 map
- R, G, Y, B - locations
  B - passenger
  Y - destination
  - taxi
- Observations ∈ [0, 499]
  including taxi row and col, pass. and dest. index
- Actions:
  South, North, East, West, Pick, Drop
Taxi Environment Step I

South

$\text{env}.\text{step}(a)$:

$o = 45$

$a = 0$ (South) $\rightarrow \langle nobs = 154, r = -1, done = \text{False} \rangle$
Taxi Environment Step 1

nobs, r, done, _ = env.step(a):

\[ o = 45 \quad (a=0 \text{ (South)}) \quad \rightarrow \quad \langle \text{nobs} = 154, \quad r = -1, \quad \text{done} = \text{False} \rangle \]
Taxi Environment Step II

West, r, done, _ = env.step(a):

- nobs = 45
- a = 3 (West) → ⟨nobs = 45, r = -1, done = False⟩
nobs, r, done, _ = env.step(a):

\[
o = 45 \xrightarrow{a=3 \text{ (West)}} \langle \text{nobs} = 45, \ r = -1, \ done = \text{False} \rangle
\]
Taxi Environment Step III

Pick/ Drop

\[ o = 45 \]

\[ a = \frac{4}{5} \]

\[ \text{−−−−−−−−−−−−−→⟨nobs = 45, r = -10, done = False}\]
nobs, r, done, _ = env.step(a):

\[ o = 45 \xrightarrow{a=4/5 \text{ (Pick/ Drop)}} \langle \text{nobs}=45, \ r=-10, \ \text{done}=False \rangle \]
Implement your RL Agent
Recap: SARSA

On-Policy TD Control: Sarsa

→ learn \( q_\pi \) and improve \( \pi \) while following \( \pi \)

Updates: \[
Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha [R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]
\]

Exploration: \( \epsilon \)-greedy policy \( \pi \)
Recap: SARSA

On-Policy TD Control: Sarsa

→ learn $q_\pi$ and improve $\pi$ while following $\pi$

Updates: $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha [R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$

Exploration: $\epsilon$-greedy policy $\pi$

| Initialize $Q(s, a), \forall s \in S, a \in A(s)$, arbitrarily, and $Q(terminal-state, \cdot) = 0$
| Repeat (for each episode):
| Initialize $S$
| Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\epsilon$-greedy)
| Repeat (for each step of episode):
| Take action $A$, observe $R, S'$
| Choose $A'$ from $S'$ using policy derived from $Q$ (e.g., $\epsilon$-greedy)
| $Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma Q(S', A') - Q(S, A)]$
| $S \leftarrow S'$; $A \leftarrow A'$;
| until $S$ is terminal |
SARSA Agent Class Structure

• **__init__** Initialise agent and Q-table as dictionary mapping (obs, act) -> q-val

• **act**: $\epsilon$-greedy policy

• **learn**: Update Q-table given new experience

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha [R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$$

• **schedule_hyperparameters**: Update hyperparameters given training progress
And now in Code ...

def act(self, obs):
    act_vals = [self.q_table[(obs, act)] for act in range(self.nActs)]
    max_val = max(act_vals)
    max_acts = [idx for idx, act_val in enumerate(act_vals) if act_val == max_val]

    if random.random() < self.epsilon:
        return random.randint(0, self.nActs - 1)
    else:
        return random.choice(max_acts)
And now in Code ... learn

SARSA Q-Update

```python
def learn(self, obs, action, reward, n_obs, n_action, done):
    target_value = reward + self.gamma * (1 - done) * self.q_table[(n_obs, n_action)]
    self.q_table[(obs, action)] += self.alpha * (target_value - self.q_table[(obs, action)])
    return self.q_table[(obs, action)]
```

\[
Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha [R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]
\]
And now in Code ... schedule_hyperparameters

SARSA $\epsilon$-Scheduling

```python
def schedule_hyperparameters(self, timestep, max_timestep):
    max_deduct, decay = 0.95, 0.07
    self.epsilon = 1.0 - (\min(1.0, \text{timestep}/(\text{decay} \times \text{max_timestep}))) \times \text{max_deduct}
```

![Graph showing epsilon decay for different SARSA decay and max_deduct values over timesteps.](image)
Evaluate your Results
Why do We Evaluate in the First Place?

- It gives our approach credibility
- Empirical evaluation is no proof, but can give strong indication about the strengths and limitations of an approach (when done right!)
Why do We Evaluate in the First Place?

- It gives our approach credibility
- Empirical evaluation is no proof, but can give strong indication about the strengths and limitations of an approach (when done right!)

How to do it right?
What to Evaluate?

**Evaluation Returns**
- Plot mean returns over multiple runs
- Visualise standard deviation or confidence interval
What to Evaluate?

Evaluation Returns

- Plot mean returns over multiple runs
- Visualise standard deviation or confidence interval

Which returns do we plot?
What to Evaluate?

Evaluation Returns
• Plot mean returns over multiple runs
• Visualise standard deviation or confidence interval

Which returns do we plot?
• Execute multiple evaluation runs with $\epsilon = 0$ at fixed intervals
• Evaluation does not involve any learning!
Keep Track of Everything!

Hyperparameters

- Track hyperparameters, here $\epsilon$-decay
- Try various values in a grid- or random-search and find good configuration
SARSA Gridsearch over Learning Rate $\alpha$ for Taxi-v3

Figure 1: $\alpha = 0.9$

Figure 2: $\alpha = 0.3$

Figure 3: $\alpha = 0.7$

Figure 4: $\alpha = 0.2$

Figure 5: $\alpha = 0.5$

Figure 6: $\alpha = 0.1$
Figure 7: Gridsearch overview over learning rate $\alpha$ with half standard deviation as shading
"But it worked last time!"
Common Pitfalls (1)

"But it worked last time!"

• It’s not enough to make it work once!
• Meaningful evaluation achieves consistent performance over multiple randomised runs
• Most RL algorithms have random components (e.g. $\epsilon$-greedy policies)
Is plotting the mean return, even with confidence interval, enough?
Is plotting the mean return, even with confidence interval, enough?
Common Pifalls (2) (Henderson, 2018; Colas et al., 2019)

Is plotting the mean return, even with confidence interval, enough?

Which one is better?
Is plotting the mean return, even with confidence interval, enough?

Which one is better? It's actually the same method!
Is plotting the mean return, even with confidence interval, enough?

Which one is better? It's actually the same method!

Apparently, it's not enough! → Statistical hypothesis testing (Colas et al., 2019) and effective statistical evaluation (Agarwal et al., 2021)
"Why should I use those random seeds? \texttt{random} already delivers random values!"
"Why should I use those random seeds? `random` already delivers random values!"

• Our goal with empirical evaluations is to make meaningful claims about the implemented approach and achieve **reproducible** performance
• Random seeds allow us to fixate random behaviour
• Reproducibility is key for meaningful research
Why should I use those random seeds? `random` already delivers random values!"

- Our goal with empirical evaluations is to make meaningful claims about the implemented approach and achieve **reproducible** performance.
- Random seeds allow us to fixate random behaviour.
- Reproducibility is key for meaningful research.

But NEVER choose/ tune your random seeds!
"Why should I use those random seeds? random already delivers random values!"

- Our goal with empirical evaluations is to make meaningful claims about the implemented approach and achieve **reproducible** performance
- Random seeds allow us to fixate random behaviour
- Reproducibility is key for meaningful research

But NEVER choose/ tune your random seeds!

**Rein in the four horsemen of irreproducibility**

Dorothy Bishop describes how threats to reproducibility, recognized but unaddressed for decades, might finally be brought under control.
Demonstration
All code is available at https://github.com/uoe-agents/Building-a-Complete-RL-System_Demonstration
References


Any questions about this lecture or the demonstration?