Reinforcement Learning Lab

Lesson 10: Actor-Critic (A2C)

Luca Marzari and Alberto Castellini

University of Verona
email: luca.marzari@univr.it

Academic Year 2023-24

Environment Setup

The first step for the setup of the laboratory environment is to update the repository and load the miniconda environment.

- Update the repository of the lab:

```
cd RL-Lab
git stash
git pull
git stash pop
```

- Activate the *miniconda* environment:

```
conda activate rl-lab
```

Today Assignment

In today's lesson, we will implement the A2C algorithm to solve the CartPole-v1 problem. In particular, the file to complete is:

```
RL-Lab/lessons/lesson_10_code.py
```

Inside the file, two Python functions are partially implemented. The objective of this lesson is to complete them.

- **def training_loop()**
- **def A2C()**

Your task is to fill in the code where there are #TODO, or "pass" words. Expected results can be found in:

```
RL-Lab/results/lesson_10_results.png
```

A2C to implement

Multi trajectory Actor-Critic (A2C): training loop

```
Input: a differentiable policy parameterization $\pi(a|s,\theta)$, with its optimizer $\rightarrow$ *Actor* network + optimizer
a differentiable state-value function parameterization $\hat{v}(s,w)$, with its optimizer $\rightarrow$ *Critic* network + optimizer
Update frequency $\eta$
  Initialize memory buffer $\mathcal{B}$
  Initialize environment
  for each episode do
      $state \leftarrow env.reset()$
      while $state$ is not terminal do
          $action \sim \pi(\cdot|state,\theta)$
          Take $action$ in the environment and observe $next\_state, reward, done$
          $\mathcal{B} \leftarrow state, action, reward, next\_state, done$
          if $done$ then
              terminate the episode
          $state \leftarrow next\_state$
      if episode % $\eta$ == 0 then
          update Actor
          update Critic
          $\mathcal{B} \leftarrow \emptyset$
```

Multi trajectory Actor-Critic (A2C): update phase

Input: a differentiable policy parameterization $\pi(a|s, \theta)$, with its optimizer $\rightarrow$ *Actor* network + optimizer
a differentiable state-value function parameterization $\hat{v}(s, w)$, with its optimizer $\rightarrow$ *Critic* network + optimizer
Discount factor γ
Memory buffer $\mathcal{B}$

```
First we update the Critic network $\hat{v}(s, w)$ multiple times to improve efficiency using a shuffle of $\mathcal{B}$ to avoid overfitting the data
for $n$ iterations do
    Shuffle $\mathcal{B}$
    Get $states, actions, rewards, next\_states, dones$ from $\mathcal{B}$
    targets $\leftarrow rewards + (1 - dones) \cdot \gamma \cdot \hat{v}(next\_states, w)$
    predictions $\leftarrow \hat{v}(states, w)$
    loss $\leftarrow MSE(preditions, targets)$
    $w \leftarrow$ Backpropagation loss
Then we update the Actor network $\pi$
probabilities $\leftarrow \pi(states)$
$\mathcal{A} \leftarrow rewards + (1 - dones) \cdot \gamma \cdot \hat{v}(next\_states) - \hat{v}(states)$
objective $\leftarrow -MEAN(\log(probabilities) \cdot \mathcal{A})$
$\theta \leftarrow$ Backpropagation objective
```

A2C (theory)

In A2C, the update rule consists of replacing $G(t) - \hat{v}(s_t)$ (as seen in REINFORCE with baseline in the previous lecture) with the advantage:

$$J_\pi = \sum_{t=0}^{T} (log \pi_\theta(a_t|s_t) \cdot \mathcal{A}(s_t, a_t)) \tag{1}$$

In general, obtaining $\mathcal{A}(s_t, a_t)$ can be hard and requires two additional neural networks (i.e., $Q(s_t, a_t)$ and $V(s_t)$). However, the advantage function $\mathcal{A}$ can be simplified as follows:

$$\mathcal{A}(s_t, a_t) = r_t + \gamma \cdot \hat{v}(s_{t+1}) - \hat{v}(s_t) \tag{2}$$

Exploiting this formulation, we can use only one additional network to estimate $\hat{v}(s_t)$; this neural network is called critic.

Shaping

These operations require different reshaping: remember to always print the shape of the tensors and arrays before and after all the algebraic operations.

A2C (code snippets)

Following the update function for the critic:

```
# Compute the MSE between the current prediction and the real advantage (Eq. 2)
target = rewards + (1 - dones.astype(int)) * gamma * critic_net(next_states)
prediction = critic_net(states)
mse = mean((prediction - target)**2)
#hint (in keras): mse = tf.math.square(prediction - target.numpy())
#mse = tf.math.reduce_mean(mse)
```

(1 - dones) ensures that the discount factor is applied correctly based on whether an episode has terminated or not.

The critic can be updated multiple times to improve efficiency but requires to shuffle the buffer to avoid overfitting the data:

```
for _ in range(10):
    np.random.shuffle(memory_buffer)
    states, rewards, next_states, actions, dones = ...
```

We recall that in CartPole-v1 an episode is *done* when either the variable *terminated* or *truncated* returned by *env.step(action)* is true.

Following the update function for the actor:

```
# The update rule for the actor directly implements Eq. 1, using the
# simplified version for the advantage (Eq. 2)
predictions = actor_net(states)
# hint (keras): probs =
#[entry[actions[idx]] for idx, entry in enumerate(predictions)]
# hint (torch): probs = predictions.gather(-1, actions[:,None]).squeeze()
log_probs = log(probs)
adv_a = rewards + gamma * critic_net(next_states).numpy()).reshape(-1)
adv_b = critic_net(states).numpy().reshape(-1)
objective = -mean(log_probs * (adv_a - adv_b))
```

Differences with One-step Actor-Critic (Sutton and Barto)

- single vs multi trajectory updates
- multiple critic updates with memory buffer shuffle for enhanced efficiency and generalization
- fixed discounted factor γ (we do not exploit $\mathcal{I}$, i.e., a γ decay)

Expected Results

Why A2C?

There are many advantages to using an actor-critic architecture.

- You can train the critic off-policy (improve data efficiency).
- You can update over multiple trajectories.

REINFORCE Baseline vs A2C

As an additional exercise, you can compare A2C with *REINFORCE baseline* implemented in the previous lesson.

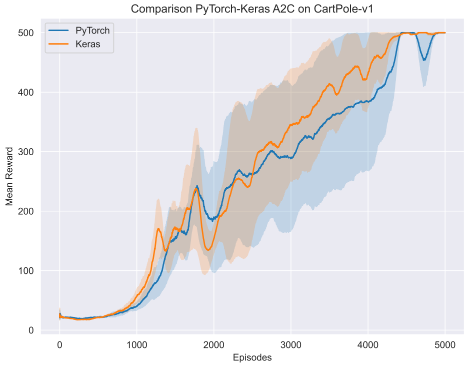

Figure: Mean and Std error using 3 random seeds for PyTorch A2C implementation. Obtaining this result requires time (stop the training after a few episodes if you notice an incremental mean reward).