Tipps for coding exercise on PPO

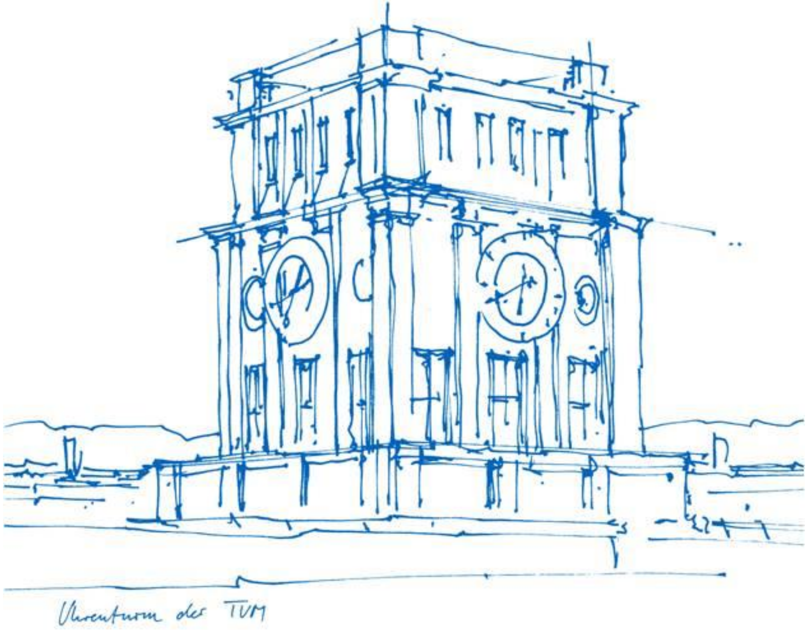
Prof. Dr. Maximilian Schiffer

Professorship of Operations and Supply Chain Management

TUM School of Management

Technical University of Munich

WiSe21/22



3rd coding exercise: PPO

- Find optimal pole balancing policy using proximal policy optimization (PPO) → same problem as for the coding exercise presented this week
- Use the same gym environment as the one that you used for the exercise from last week, so you can compare your results
- Try to think about the different ingredients that you will need:
  - A neural network for the actor and for the critic
  - A memory/buffer where you store states, actions, rewards from one trajectory through an episode & that for N actors
  - A function that calculates advantage estimates
  - A function that updates weights of both critic and actor
- You can use the code framework introduced this week or implement according to your own taste

Pseudo code for PPO-Clip

Initialize network weights w, θ_{old}

Repeat:

For N actors:

$\tau \leftarrow \{s_1, a_1, ..., s_T, a_T, r_T, s_{T+1}\} \sim \pi(s, a; \theta_{\text{old}})$

Compute advantage estimates $A_1, ..., A_T$ based on $V(s; w)$

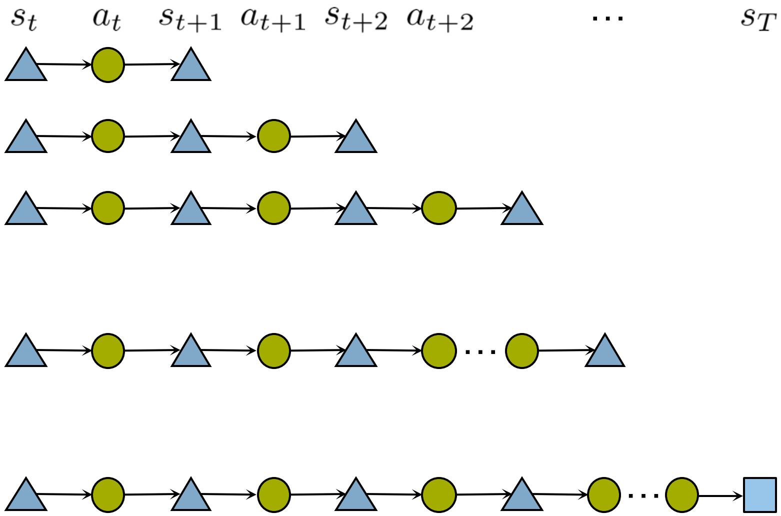
$$\theta_{\text{new}} \leftarrow \arg\max_{\theta_{\text{new}}} \frac{1}{NT} \sum_{\tau} \sum_{t} L_t^{\text{CLIP}}$$

Update w

$\theta_{\text{old}} \leftarrow \theta_{\text{new}}$

n-step temporal-difference learning

Idea: we assume a tabular setting for now and **let TD target look n steps into the future**



$n = 1$ TD (1-step) $G_t^{(1)} = r_t + \gamma V(s_{t+1})$

$n = 2$ 2-step $G_t^{(2)} = r_t + \gamma r_{t+1} + \gamma^2 V(s_{t+2})$

$n = 3$ 3-step

⋮

n-step

⋮

$n = \infty$ Monte Carlo $G_t^{(\infty)} = r_t + \gamma r_{t+1} + \ldots + \gamma^{T-1-t} r_{T-1}$

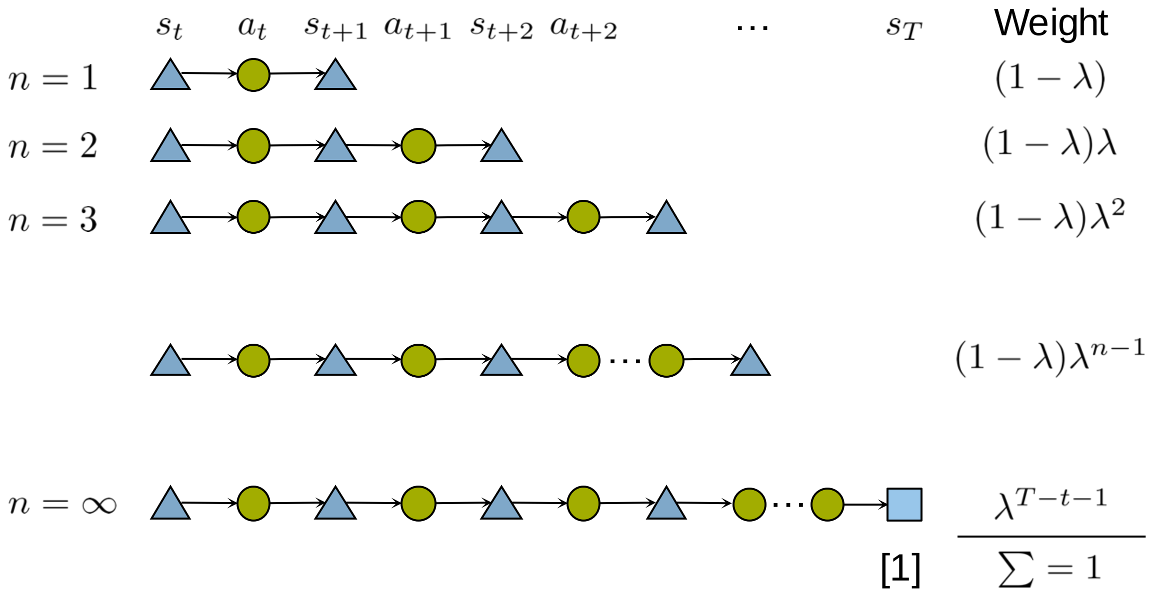
[1]

New update rule

- We define the n-step return $G_t^{(n)} = \sum_{k=t}^{t+n-1} \gamma^{k-t} \cdot r_k + \gamma^n V(s_{t+n})$
- We obtain a new update rule for the tabular value function estimate: $V(s_t) \leftarrow V(s_t) + \alpha \left(G_t^{(n)} - V(s_t) \right)$

Forward-view TD(λ): averaging n-step returns

Idea: we combine all n-step returns $G_t^{(n)}$ into one weighted average, the λ-return G_t^{λ}



New update rule

- We define the λ-return

$$G_t^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}$$

- For episodes with terminal state T

$$G_t^{\lambda} = (1-\lambda)\sum_{n=1}^{T-t-1} \lambda^{n-1} G_t^{(n)} + \lambda^{T-t-1} G_t$$

- We obtain a new update rule called forward-view TD(λ):

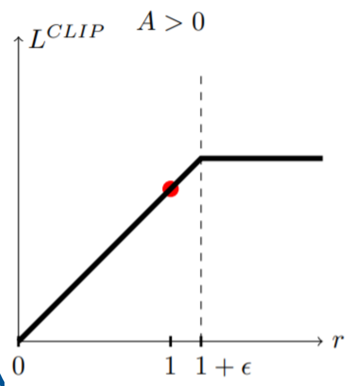
$$V(s_t) \leftarrow V(s_t) + \alpha\left(G_t^{\lambda} - V(s_t)\right)$$

PPO-Clip: the effect of clipping

Case 1: positive advantage

- The objective term reduces to

$$L^{\text{CLIP}} = \min\left(\frac{\pi_{\theta_{\text{new}}}(s,a)}{\pi_{\theta_{\text{old}}}(s,a)},\ 1+\epsilon\right) \cdot A_{\pi_{\theta_{\text{old}}}}(s,a)$$



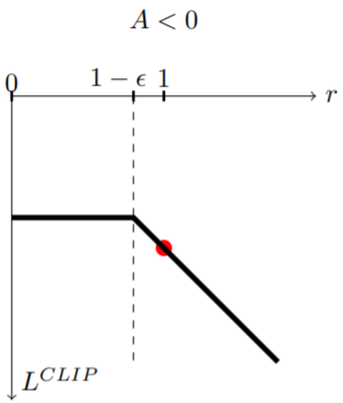
[10]

- Since the advantage is positive, maximizing the objective means to maximize $\pi_{\theta_{\text{new}}}$
- Due to the min, increasing the ratio of old and new policy r beyond $1+\epsilon$ does not change the objective

Case 2: negative advantage

- The objective term reduces to

$$L^{\text{CLIP}} = \max\left(\frac{\pi_{\theta_{\text{new}}}(s,a)}{\pi_{\theta_{\text{old}}}(s,a)},\ 1-\epsilon\right) \cdot A_{\pi_{\theta_{\text{old}}}}(s,a)$$



[10]

- Since the advantage is negative, maximizing the objective means to minimize $\pi_{\theta_{\text{new}}}$
- Due to the max, decreasing the ratio of old and new policy r below $1-\epsilon$ does not change the objective

Moving the new policy too far from the old one does not have a positive effect on the objective