Aalto University
School of Electrical Engineering

ELEC-E8125 Reinforcement Learning
Actor-critic methods

Ville Kyrki

13.10.2020

Today

- Combining policy gradient with value functions
  → actor-critic methods

Learning goals

- Understand basis of actor-critic approaches.

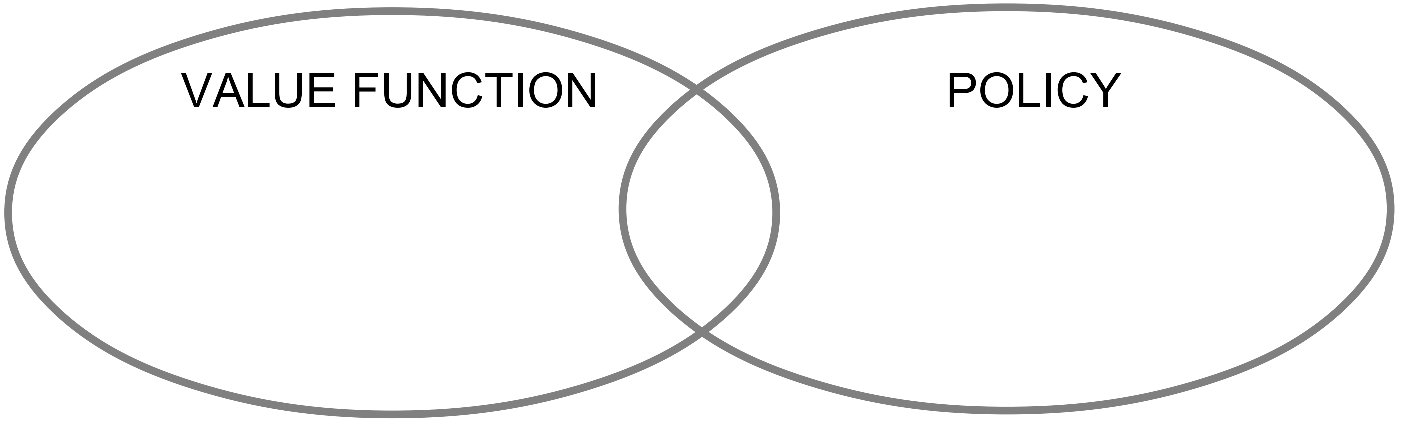
Motivation

- Policy gradient (PG) methods may be often ineffective in terms of requiring lots (and lots and lots) of data because of high variance of gradient estimates.
  - Similar to MC approaches for value function estimation.

- Temporal difference (TD) approaches have smaller variance compared to MC but they cannot handle stochastic policies or continuous action spaces like PG.

- Can we combine PG with something like TD?

Value-based vs policy-based RL



Value-based
- Learnt value function.
- Implicit policy.

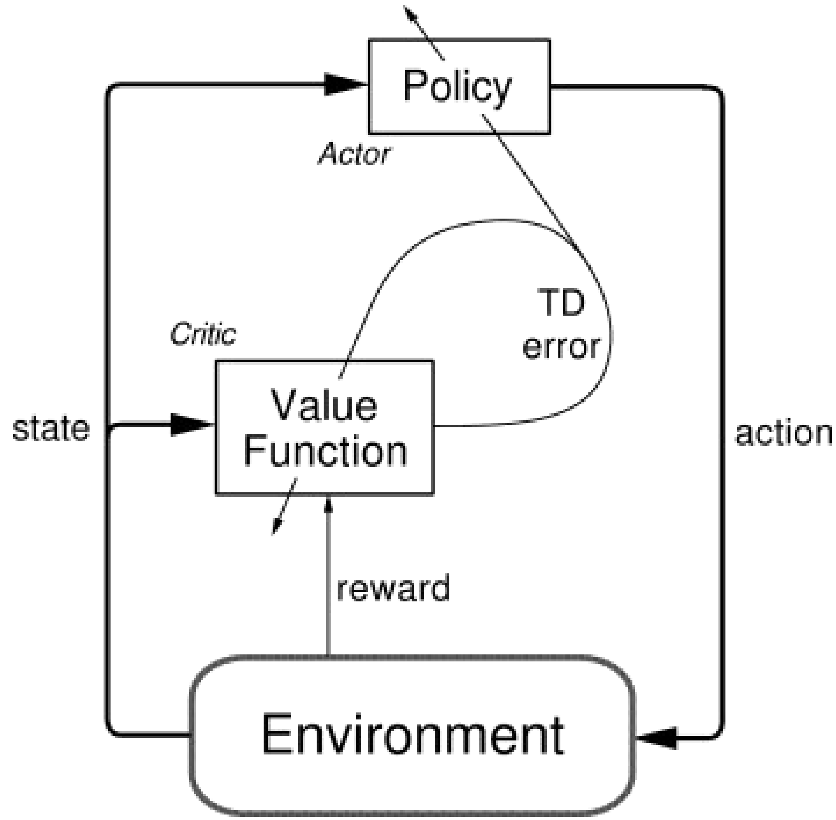
Actor-critic
- **Learnt value function.**
- **Learnt policy.**

Policy-based
- No value function.
- Learnt policy.

Actor-critic approach – overview

- *Critic* estimates value function.
- *Actor* updates policy in direction of critic.

- For policy gradient, critic estimates value function.
  - See previous lectures.



Policy gradient – recap

Note: Discount omitted for getting shorter notation

REINFORCE

1. Run policy, collect $\{\boldsymbol{\tau}_i\}$ $\quad \boldsymbol{\tau}_i = (\boldsymbol{s}_0^i, \boldsymbol{a}_0^i, r_0^i, \boldsymbol{s}_1^i, \boldsymbol{a}_1^i, r_1^i, \ldots)$
2. $\nabla_\theta R(\theta) \approx \frac{1}{J} \sum_{i=1}^{J} \left(\sum_{t=0}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}} (\boldsymbol{a}_t^i | \boldsymbol{s}_t^i) \left(\sum_{t'=t}^{T} r(\boldsymbol{s}_{t'}^i, \boldsymbol{a}_{t'}^i) \right) \right)$
3. $\theta \leftarrow \theta + \alpha \nabla_{\boldsymbol{\theta}} R(\theta)$

Policy gradient – recap

REINFORCE

1. Run policy, collect $\{\boldsymbol{\tau}_i\}$ $\quad \boldsymbol{\tau}_i = (\boldsymbol{s}_0^i, \boldsymbol{a}_0^i, r_0^i, \boldsymbol{s}_1^i, \boldsymbol{a}_1^i, r_1^i, \ldots)$
2. $\nabla_\theta R(\theta) \approx \frac{1}{J} \sum_{i=1}^{J} \left(\sum_{t=0}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}} (\boldsymbol{a}_t^i | \boldsymbol{s}_t^i) \underbrace{\left(\sum_{t'=t}^{T} r(\boldsymbol{s}_{t'}^i, \boldsymbol{a}_{t'}^i) \right)}_{} \right)$
3. $\theta \leftarrow \theta + \alpha \nabla_{\boldsymbol{\theta}} R(\theta)$

What's this?
Does it look familiar?

Policy gradient – recap

REINFORCE

1. Run policy, collect $\{\boldsymbol{\tau}_i\}$ $\boldsymbol{\tau}_i = (\boldsymbol{s}_0^i, \boldsymbol{a}_0^i, r_0^i, \boldsymbol{s}_1^i, \boldsymbol{a}_1^i, r_1^i, \ldots)$
2. $\nabla_\theta R(\theta) \approx \frac{1}{J} \sum_{i=1}^{J} \left(\sum_{t=0}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}} (\boldsymbol{a}_t^i | \boldsymbol{s}_t^i) \underbrace{\left(\sum_{t'=t}^{T} r(\boldsymbol{s}_{t'}^i, \boldsymbol{a}_{t'}^i) \right)} \right)$
3. $\theta \leftarrow \theta + \alpha \nabla_{\boldsymbol{\theta}} R(\theta)$

What's this?
Does it look familiar?

$$Q_\pi(s_t, a_t) = \sum_{t=t'}^{T} E\left[r(s_{t'}^i, a_{t'}^i) | s_t, a_t \right]$$

Q is true expected reward, unlike the estimate in step 2.
This would reduce variance of the gradient estimate.

$$\nabla_\theta R(\theta) \approx \frac{1}{J} \sum_{i=1}^{J} \left(\sum_{t=0}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}} (\boldsymbol{a}_t^i | \boldsymbol{s}_t^i) Q(\boldsymbol{s}_t^i, \boldsymbol{a}_t^i) \right)$$

$$\nabla_{\boldsymbol{\theta}} R(\boldsymbol{\theta}) = E_{\boldsymbol{\theta}}\left[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\tau})\left(R(\boldsymbol{\tau}) - \boxed{b}\right)\right]$$

Remember the baselines?

$$\nabla_{\theta} R(\theta) \approx \frac{1}{J} \sum_{i=1}^{J} \left(\sum_{t=0}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t^i | \boldsymbol{s}_t^i) \left(Q(\boldsymbol{s}_t^i, \boldsymbol{a}_t^i) - b \right) \right)$$

Average is a good baseline: $b_t = \frac{1}{J} \sum_{i=1}^{J} Q(s_t^i, a_t^i)$

But what does the average mean here?

$$\nabla_{\boldsymbol{\theta}} R(\boldsymbol{\theta}) = E_{\boldsymbol{\theta}}\left[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\tau})(R(\boldsymbol{\tau}) - \boxed{b})\right]$$

Remember the baselines?

$$\nabla_\theta R(\theta) \approx \frac{1}{J}\sum_{i=1}^{J}\left(\sum_{t=0}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t^i | \boldsymbol{s}_t^i)\left(Q(\boldsymbol{s}_t^i, \boldsymbol{a}_t^i) - b\right)\right)$$

Average is a good baseline: $b_t = \frac{1}{J}\sum_{i=1}^{J} Q(s_t^i, a_t^i)$

But what does the average mean here?

b approximates the state value function *V(x)!*

$$\nabla_\theta R(\theta) \approx \frac{1}{J}\sum_{i=1}^{J}\left(\sum_{t=0}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t^i | \boldsymbol{s}_t^i)\left(Q(\boldsymbol{s}_t^i, \boldsymbol{a}_t^i) - V(\boldsymbol{s}_t^i)\right)\right)$$

$$\nabla_{\boldsymbol{\theta}} R(\boldsymbol{\theta}) = E_{\boldsymbol{\theta}}\left[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\tau})(R(\boldsymbol{\tau}) - \boxed{b})\right]$$

Remember the baselines?

$$\nabla_\theta R(\theta) \approx \frac{1}{J} \sum_{i=1}^{J} \left(\sum_{t=0}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t^i | \boldsymbol{s}_t^i) \left(Q(\boldsymbol{s}_t^i, \boldsymbol{a}_t^i) - b \right) \right)$$

Average is a good baseline: $b_t = \frac{1}{J} \sum_{i=1}^{J} Q(s_t^i, a_t^i)$

But what does the average mean here?

b approximates the state value function *V(x)!*

$$\nabla_\theta R(\theta) \approx \frac{1}{J} \sum_{i=1}^{J} \left(\sum_{t=0}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t^i | \boldsymbol{s}_t^i) \underbrace{\left(Q(\boldsymbol{s}_t^i, \boldsymbol{a}_t^i) - V(\boldsymbol{s}_t^i) \right)}_{} \right)$$

advantage function $A(\boldsymbol{s}_t^i, \boldsymbol{a}_t^i)$

$$\nabla_\theta R(\theta) \approx \frac{1}{J} \sum_{i=1}^{J} \left(\sum_{t=0}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t^i | \boldsymbol{s}_t^i) A(\boldsymbol{s}_t^i, \boldsymbol{a}_t^i) \right)$$

Determining the advantage

How to find a good estimate for this?

$$\nabla_\theta R(\theta) \approx \frac{1}{J} \sum_{i=1}^{J} \left(\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta (\boldsymbol{a}_t^i | \boldsymbol{s}_t^i) A(\boldsymbol{s}_t^i, \boldsymbol{a}_t^i) \right)$$

Estimate *Q*, *V*, or *A*?

V has the fewest parameters, so let's estimate it (from data).
But how to then get *A*?

$$A(\boldsymbol{s}_t, \boldsymbol{a}_t) = Q(\boldsymbol{s}_t, \boldsymbol{a}_t) - V(\boldsymbol{s}_t)$$

$$Q(\boldsymbol{s}_t, \boldsymbol{a}_t) = r(s_t, a_t) + \gamma E_{s_{t+1} \sim \pi(s_{t+1} | s_t, a_t)} [V(\boldsymbol{s}_{t+1})]$$

$$A(\boldsymbol{s}_t, \boldsymbol{a}_t) \approx r_{(\boldsymbol{s}_t, \boldsymbol{a}_t)} + \gamma V(\boldsymbol{s}_{t+1}) - V(\boldsymbol{s}_t)$$

Thus, knowing *V* allows approximating *A.*

How to fit *V?*

Does this look familiar?

Fitting value functions (mostly recap)

- Episodic batch fitting: (1) gather data, (2) fit (least squares) over gathered data.
- Data = state-value pairs $\left\{\left(\boldsymbol{s}_t^i, \underbrace{\sum_{t'=t}^{T} r_{t'}^i}_{y_t^i}\right)\right\}$
- Requires episodic environments to get the value.
- Fitting criterion $L(\phi)=\sum_i \left\|V_\phi(\boldsymbol{s}_i)-y_i\right\|^2$

Any parametric function approximator

But what about non-episodic?
What do we do then?

Fitting value functions (mostly recap)

- Non-episodic batch fitting: (1) gather data, (2) fit (least squares) over gathered data.
- Data = state-value pairs $\left\{\left(\boldsymbol{s}_t^i, \underbrace{r_t^i + V\left(\boldsymbol{s}_{t+1}^i\right)}_{y_t^i}\right)\right\}$
- Identical fitting criterion

$$L(\phi) = \sum_i \left\| V_\phi(\boldsymbol{s}_i) - y_i \right\|^2$$

Any parametric function approximator

Wrap-up: A batch TD actor critic

Batch actor-critic

1. Run policy, collect $\{\boldsymbol{\tau}_i\}$ $\quad \boldsymbol{\tau}_i = (\boldsymbol{s}_0^i, \boldsymbol{a}_0^i, r_0^i, \boldsymbol{s}_1^i, \boldsymbol{a}_1^i, r_1^i, \ldots)$
2. Fit $V_\phi(\boldsymbol{s}_t)$
3. Evaluate $A(\boldsymbol{s}_t, \boldsymbol{a}_t) \approx r_{(\boldsymbol{s}_t, \boldsymbol{a}_t)} + \gamma V(\boldsymbol{s}_{t+1}) - V(\boldsymbol{s}_t)$
4. Evaluate $\nabla_\theta R(\theta) \approx \frac{1}{J} \sum_{i=1}^{J} \left(\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta (\boldsymbol{a}_t^i | \boldsymbol{s}_t^i) A(\boldsymbol{s}_t^i, \boldsymbol{a}_t^i) \right)$
5. Update $\theta \leftarrow \theta + \alpha \nabla_\theta R(\theta)$

What about discount?

An on-line TD actor critic (with discount)

On-line actor-critic

1. Take action $\boldsymbol{a}=\pi(\boldsymbol{a}|\boldsymbol{s})$
2. Update $V_\phi(\boldsymbol{s}_t)$ using $\overbrace{\phi \leftarrow \phi+\underset{\text{learning rate}}{\beta}\left(r_t+\gamma V_\phi(\boldsymbol{s}_{t+1})-V_\phi(\boldsymbol{s}_t)\right)\nabla_\phi V_\phi(\boldsymbol{s}_t)}^{\text{From lecture 5!}}$
3. Evaluate $A(\boldsymbol{s}_t, \boldsymbol{a}_t)\approx r_{(s_t, a_t)}+\gamma V(\boldsymbol{s}_{t+1})-V(\boldsymbol{s}_t)$
4. Evaluate $\nabla_\theta R(\theta)\approx\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t^i|\boldsymbol{s}_t^i)A(\boldsymbol{s}_t^i, \boldsymbol{a}_t^i)$
5. Update $\theta\leftarrow\theta+\alpha\nabla_{\boldsymbol{\theta}}R(\theta)$

In practice, even this works best in batches (decreases variance in gradient estimates).

Note: TD estimate can be biased.

Challenge: Gradient step sizes

$$\theta \leftarrow \theta + \alpha \nabla_{\theta} R(\theta)$$

<u>Gradient step size</u> affects convergence (speed) greatly but is difficult to set.

Incorrect step size may lead to divergence or slow convergence.

How to guarantee policy improvement?

Reformulating policy gradient through surrogate advantage

- How to predict performance of updated policy (since we do not have data about it yet)?
- Surrogate advantage $R_{\theta_{old}}^{IS}(\theta)$ approximates performance difference between previous and updated policies

$$R_{\theta_{old}}^{IS}(\theta) = E_{\tau \sim \pi_{\theta_{old}}}\left[\frac{\pi_{\theta}(\boldsymbol{a}_t|\boldsymbol{s}_t)}{\pi_{\theta_{old}}(\boldsymbol{a}_t|\boldsymbol{s}_t)} A^{\pi_{\theta_{old}}}(\boldsymbol{s}_t, \boldsymbol{a}_t)\right]$$

See the importance sampling in effect!

Can we find a lower bound for this?
Yes, using KL-divergence.

Bounding surrogate advantage

Result: Policy is guaranteed to improve by optimizing

$$max_{\theta}\left(R^{IS}_{\theta_{old}}(\theta) - c\, D^{max}_{KL}(\theta_{old}, \theta)\right)$$

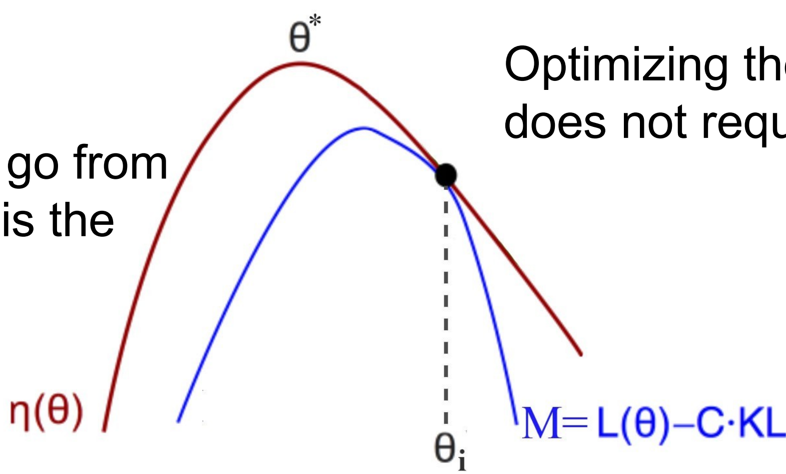
known constant

where

$$D^{max}_{KL}(\theta_{old}, \theta)$$

is the maximum Kullback-Leibler divergence between the policies.

Intuition: The further you go from current policy, the larger is the penalty.



Optimizing the lower bound function does not require step size!

In practice leads to slow convergence, not easy to optimize.

Trust region policy optimization (Schulman et al. 2015)

Instead of lower bound, optimize surrogate advantage and constrain KL-divergence:

$$max_\theta R^{IS}_{\theta_{old}}(\theta)$$

such that

$$\bar{D}_{KL}(\theta_{old}, \theta) \equiv E_{\tau \sim \pi_{\theta_{old}}}\left[D_{KL}\left(\pi_\theta(.|\boldsymbol{s}), \pi_{\theta_{old}}(.|\boldsymbol{s})\right)\right] \leq \delta$$

Intuition: Limit the policy parameter change such that the actions do not change too much in the relevant part of state space.

In practice, this is still (too) costly to compute and the constraint is approximated (details in the paper).

Next: a simple and practical way to implement the same idea (and it even works well usually).

Proximal policy optimization (Schulman et al. 2017)

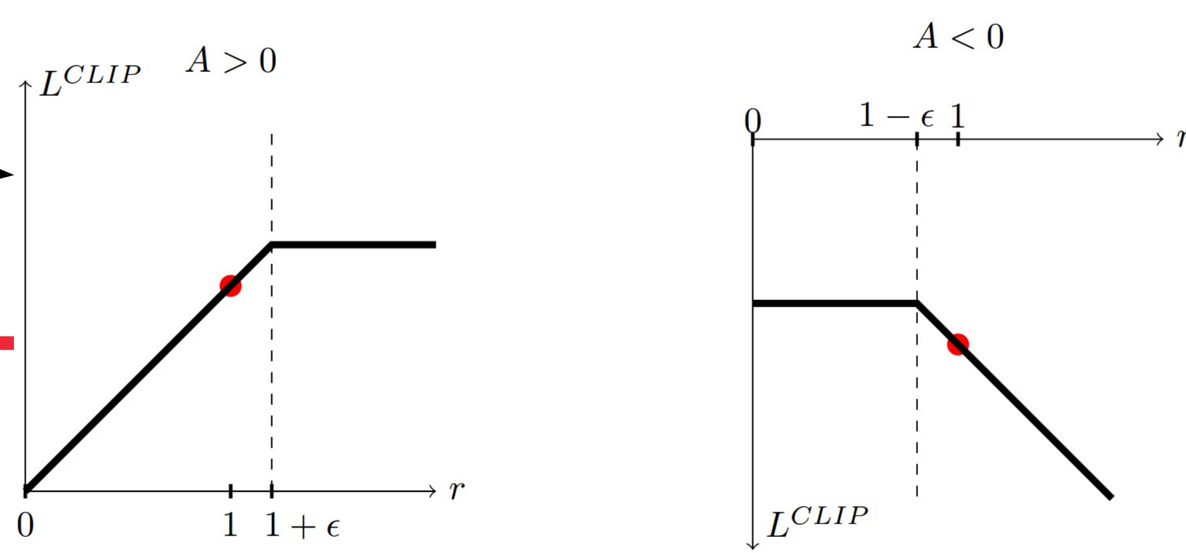
Remember the surrogate advantage?

$$R^{IS}_{\theta_{old}}(\theta) = E_{\tau\sim\pi_{\theta_{old}}}\left[\underbrace{\frac{\pi_\theta(\boldsymbol{a}_t|\boldsymbol{s}_t)}{\pi_{\theta_{old}}(\boldsymbol{a}_t|\boldsymbol{s}_t)}}_{g_t(\theta)} A^{\pi_{\theta_{old}}}(\boldsymbol{s}_t, \boldsymbol{a}_t)\right]$$

Optimize instead

$$L^{CLIP}(\theta) = E_{\tau\sim\pi_{\theta_{old}}}\left[min\left(g_t(\theta)A, clip\left(g_t(\theta), 1-\epsilon, 1+\epsilon\right)A\right)\right]$$

Looks horrible, look at the figure instead.
In practice: Limit how much benefit there is for changes.



Proximal policy optimization (Schulman et al. 2017)

Other variants possible

Algorithm: PPO

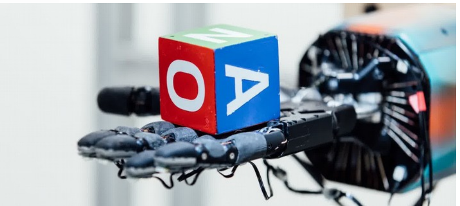
for i = 1, 2, … do

Run policy, collect trajectories $\{\boldsymbol{\tau}_i\}$ $\quad \boldsymbol{\tau}_i = (\boldsymbol{s}_0^i, \boldsymbol{a}_0^i, r_0^i, \boldsymbol{s}_1^i, \boldsymbol{a}_1^i, r_1^i, \ldots)$

Compute advantage estimates $A(\boldsymbol{s}_t, \boldsymbol{a}_t) \approx r_{(\boldsymbol{s}_t, \boldsymbol{a}_t)} + \gamma V(\boldsymbol{s}_{t+1}) - V(\boldsymbol{s}_t)$ using current value function $V_\phi(\boldsymbol{s}_t)$

Update policy by maximizing $L^{CLIP}(\theta)$ for K epochs of stochastic gradient ascent

Fit $V_\phi(\boldsymbol{s}_t)$ by minimizing $L(\phi) \equiv \sum_i \left\| V_\phi(\boldsymbol{s}_i) - y_i \right\|^2$ using gradient descent



PPO is a standard baseline at the moment.

Summary

- Actor-critic approaches allow addressing continuing problems and continuous action spaces.
- They may also learn faster than policy gradient because variance of policy gradient estimate is reduced.

- TRPO/PPO aim to control extent of policy update steps to avoid oscillation/divergence due to large updates.

Next: Optimal control – Toward model-based RL

- Even with a critic, policy-based approaches often require huge amounts of data.
- Can we somehow benefit even more from earlier experiences?

- Reading: Introduction to Linear Quadratic Regulation (R. Platt).
- Note: No quiz for next week.