ELEC-E8125 Reinforcement Learning Actor-critic methods

Joni Pajarinen
11.10.2022

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

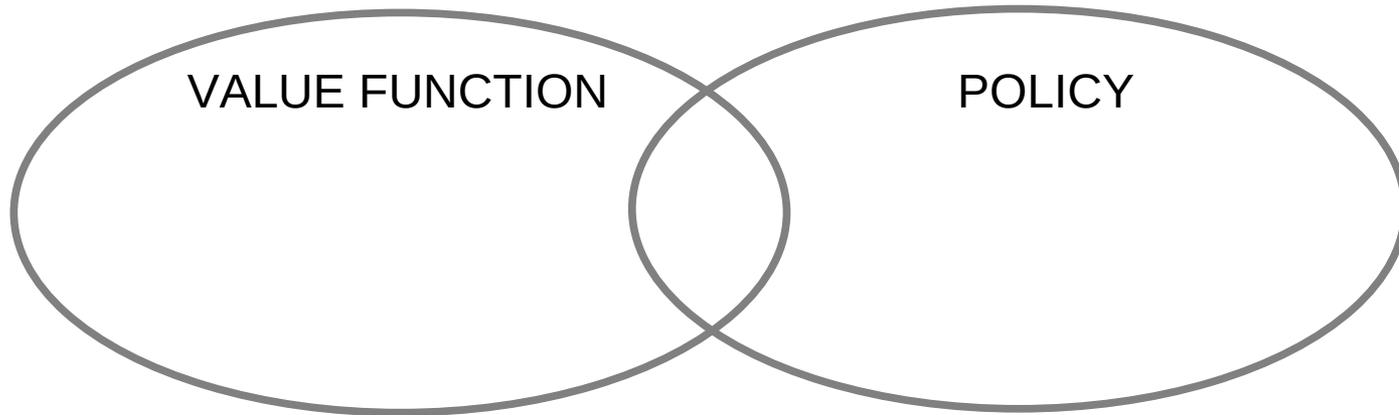- Can we combine PG with something like TD?

A'' Aalto University
School of Electrical
Engineering

Today

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

Learning goals

- Understand basis of actor-critic approaches

Value-based vs policy-based RL

VALUE FUNCTION   POLICY

Value-based   **Actor-critic**   Policy-based
· Learnt value function   · **Learnt value function**   · No value function
· Implicit policy   · **Learnt policy**   · Learnt policy

Aalto University
School of Electrical
Engineering

Actor-critic approach – overview

- *Critic* estimates value function

- *Actor* updates policy in direction of critic

- For example, policy gradient where critic estimates value function
  - See previous lectures
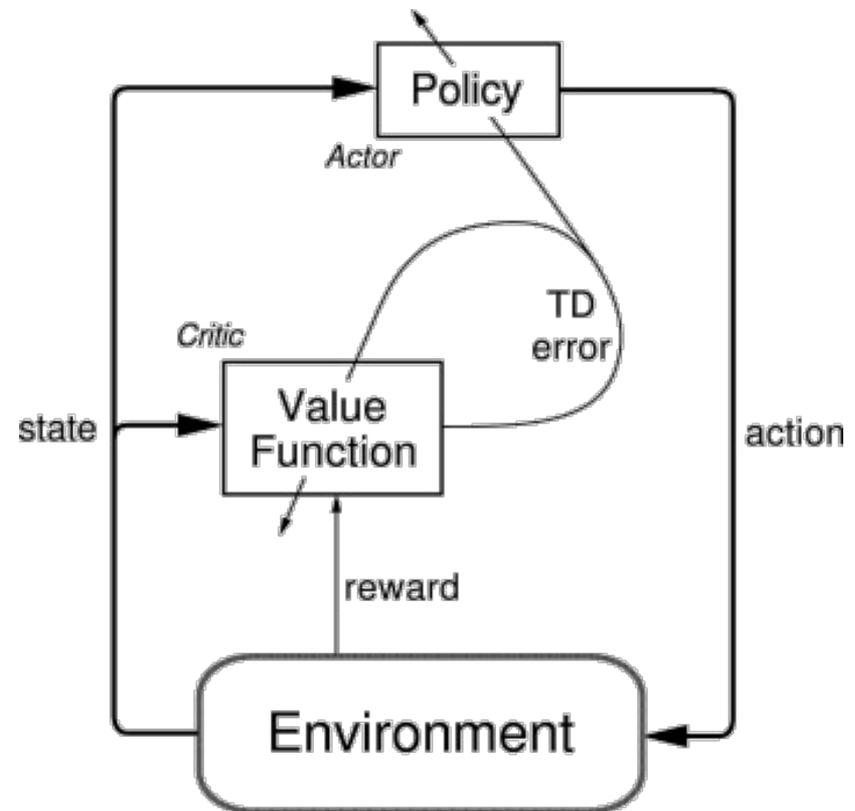
Policy gradient – recap

Note: Discount omitted to get shorter notation

REINFORCE

1. Run policy, collect $\{\tau_i\}$ $\tau_i = (s_0^i, a_0^i, r_0^i, s_1^i, a_1^i, r_1^i, \ldots)$

2. $\nabla_\theta R(\theta) \approx \frac{1}{J} \sum_{i=1}^{J} \left(\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \left(\sum_{t'=t}^{T} r(s_{t'}^i, a_{t'}^i) \right) \right)$

3. $\theta \leftarrow \theta + \alpha \nabla_\theta R(\theta)$

Policy gradient – recap

REINFORCE

1. Run policy, collect $\{\tau_i\}$ $\quad \tau_i = (s_0^i, a_0^i, r_0^i, s_1^i, a_1^i, r_1^i, \dots)$

2. $\nabla_\theta R(\theta) \approx \dfrac{1}{J} \sum_{i=1}^J \left(\sum_{t=0}^T \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \underbrace{\left(\sum_{t'=t}^T r(s_{t'}^i, a_{t'}^i) \right)}_{} \right)$

3. $\theta \leftarrow \theta + \alpha \nabla_\theta R(\theta)$

What's this?
Does it look familiar?

_

Policy gradient – recap

REINFORCE

1. Run policy, collect $\{\tau_i\}$ $\tau_i = (s_0^i, a_0^i, r_0^i, s_1^i, a_1^i, r_1^i, \ldots)$

2. $\nabla_\theta R(\theta) \approx \dfrac{1}{J} \sum_{i=1}^{J} \left(\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \underbrace{\left(\sum_{t'=t}^{T} r(s_{t'}^i, a_{t'}^i) \right)} \right)$

3. $\theta \leftarrow \theta + \alpha \nabla_\theta R(\theta)$

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.

$$\nabla_\theta R(\theta) \approx \dfrac{1}{J} \sum_{i=1}^{J} \left(\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) Q(s_t^i, a_t^i) \right)$$

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

Remember the baselines?

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

How to find a good baseline for $Q(s_t^i, a_t^i)$?

Baseline: a function that does not affect the expected gradient, that is, one that does not depend on the <u>action</u> or new policy parameters.

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

Remember the baselines?

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

How to find a good baseline for $Q(s_t^i, a_t^i)$?

Baseline: a function that does not affect the expected gradient, that is, one that does not depend on the action or new policy parameters.

We can use value function $V(s_t)$ as baseline!

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

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

Remember the baselines?

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

How to find a good baseline for $Q(s_t^i, a_t^i)$?

Baseline: a function that does not affect the expected gradient, that is, one that does not depend on the action or new policy parameters.

We can use value function $V(s_t)$ as baseline!   Value of *a* compared to the expected value

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

advantage function $A(s_t^i, a_t^i)$

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

Determining the advantage

How to find a good estimate for this?

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

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

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

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

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

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

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 $\left\{ \left(s_t^i, \underbrace{\sum_{t'=t}^{T} r_{t'}^i}_{y_t^i} \right) \right\}$

- Requires episodic environments to get the value

- Fitting criterion $L(\phi) = \sum_i \left\| V_\phi(s_i) - y_i \right\|^2$

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 $\left\{\left(\boldsymbol{s_t^i}, \underbrace{r_t^i + V(\boldsymbol{s_{t+1}^i})}_{y_t^i}\right)\right\}$

- 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 $\{\boldsymbol{\tau_i}\}$ $\qquad \boldsymbol{\tau_i} = (\boldsymbol{s_0^i}, \boldsymbol{a_0^i}, r_0^i, \boldsymbol{s_1^i}, \boldsymbol{a_1^i}, r_1^i, \ldots)$

2. Fit $V_\phi(\boldsymbol{s_t})$

3. Evaluate $A(\boldsymbol{s_t}, \boldsymbol{a_t}) \approx r_{(\boldsymbol{s_t}, \boldsymbol{a_t})} + \gamma V_\phi(\boldsymbol{s_{t+1}}) - V_\phi(\boldsymbol{s_t})$

4. Evaluate $\nabla_\theta R(\theta) \approx \dfrac{1}{J} \sum_{i=1}^{J} \left(\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(\boldsymbol{a_t^i}|\boldsymbol{s_t^i}) A(\boldsymbol{s_t^i}, \boldsymbol{a_t^i}) \right)$

5. Update $\theta \leftarrow \theta + \alpha \nabla_\theta R(\theta)$

6. Repeat

What about discount?

An on-line TD actor critic (with discount)

On-line actor-critic

learning rate

1. Take action $a = \pi(a|s)$

From lecture 4!

2. Update $V_\phi(s_t)$ using $\phi \leftarrow \phi + \beta \left(r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t) \right) \nabla_\phi V_\phi(s_t)$

3. Evaluate $A(s_t, a_t) \approx r_{(s_t, a_t)} + \gamma V(s_t) - V(s_{t+1})$

4. Evaluate $\nabla_\theta R(\theta) \approx \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) A(s_t^i, a_t^i)$

5. Update $\theta \leftarrow \theta + \alpha \nabla_\theta R(\theta)$

6. Repeat

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

Note: TD estimate can be biased.

Deep Deterministic Policy Gradient (DDPG)

Off-policy actor-critic with batch updates and "slow" targets

1. Add data to replay buffer using $a = \pi_\theta(s) + \epsilon$, $where\ \epsilon \sim N(0, \sigma^2)$

2. Update $Q_\phi(s_t, a_t)$ using

$$\phi \leftarrow \phi + \beta \left(r_t + \gamma Q_{\phi_{targ}}(s_{t+1}, \pi_{\theta_{targ}}(s_{t+1})) - Q_\phi(s_t, a_t) \right) \nabla_\phi Q_\phi(s_t, a_t)$$

3. Update $\pi_\theta(s_t)$ using $\theta \leftarrow \theta + \alpha \nabla_\theta Q_\phi(s_t, \pi_\theta(s_t))$

4. "Delayed" update of Q and policy parameter targets using run-time averaging: $\phi_{targ} \leftarrow \xi \phi_{targ} + (1 - \xi)\phi$

$$\theta_{targ} \leftarrow \xi \theta_{targ} + (1 - \xi)\theta$$

**Aalto University
School of Electrical
Engineering**

[Lillicrap et al. 2016, ICLR]

Challenge: Gradient step sizes

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

Gradient step size 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 $R^{IS}_{\theta_{old}}(\theta)$ approximates performance difference between previous and updated policies

$$R^{IS}_{\theta_{old}}(\theta) = E_{\tau \sim \pi_{\theta_{old}}} \left[\frac{\pi_\theta(\boldsymbol{a_t}|\boldsymbol{s_t})}{\pi_{\theta_{old}}(\boldsymbol{a_t}|\boldsymbol{s_t})} A^{\pi_{\theta_{old}}}(\boldsymbol{s_t}, \boldsymbol{a_t}) \right]$$

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

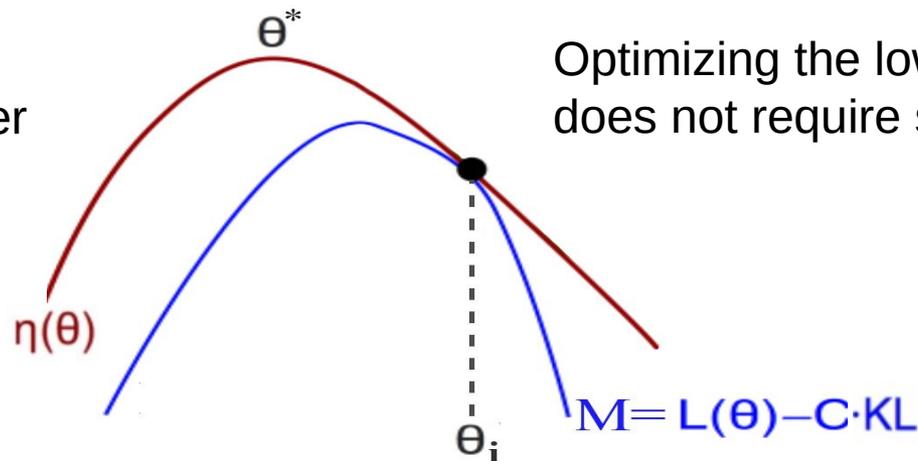$$max_{\theta}\left(R_{\theta_{old}}^{IS}(\theta) - c\,D_{KL}^{max}(\theta_{old},\theta)\right)$$

where   known constant

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

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

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

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

For policies with many parameters, this is still (too) costly to compute and the constraint is approximated (details in the paper).

Next: another way to implement the same idea.

Proximal policy optimization (Schulman et al. 2017)
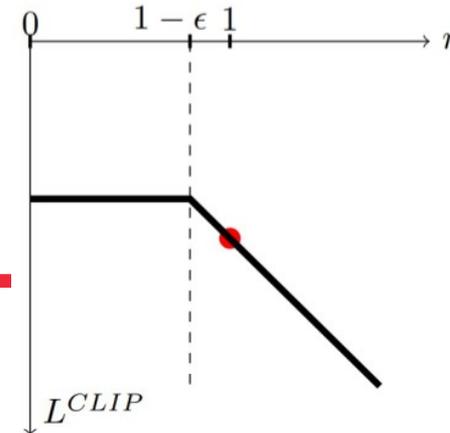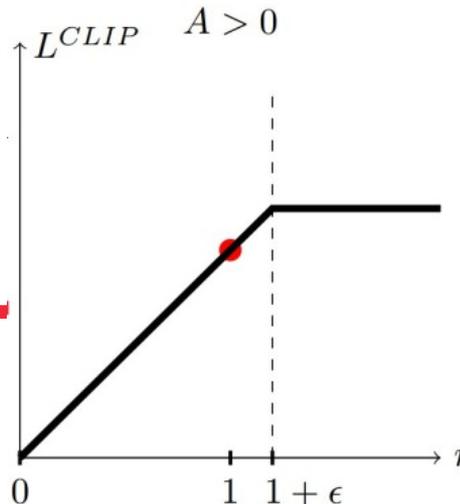
Remember the surrogate advantage?

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

Optimize instead

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

Looks horrible, look at the figure instead.
In practice: limit influence of policy change.

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 $A(\boldsymbol{s_t}, \boldsymbol{a_t}) \approx r_{(\boldsymbol{s_t}, \boldsymbol{a_t})} + \gamma V(\boldsymbol{s_t}) - V(\boldsymbol{s_{t+1}})$
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.

Recent successful algorithms

- Twin Delayed DDPG (TD3), 2018
  - Similar to DDPG but improves value function approximation
  - Off-policy

- Soft Actor Critic (SAC), 2018
  - Q-function learning encouraging policy randomness (exploration)
  - Stochastic policy for continuous actions
  - Off-policy

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: Model-based RL

- Even with critic, policy-based approaches often require huge amounts of data

- Can we somehow benefit even more from earlier experiences?

- Reading: Sutton & Barto 8 - 8.2