



Aalto University
School of Electrical
Engineering

ELEC-E8125 Reinforcement Learning

Actor-critic methods

Joni Pajarinen

10.10.2023

Motivation

<https://www.youtube.com/watch?v=xyJAvghtqIM>

- 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?

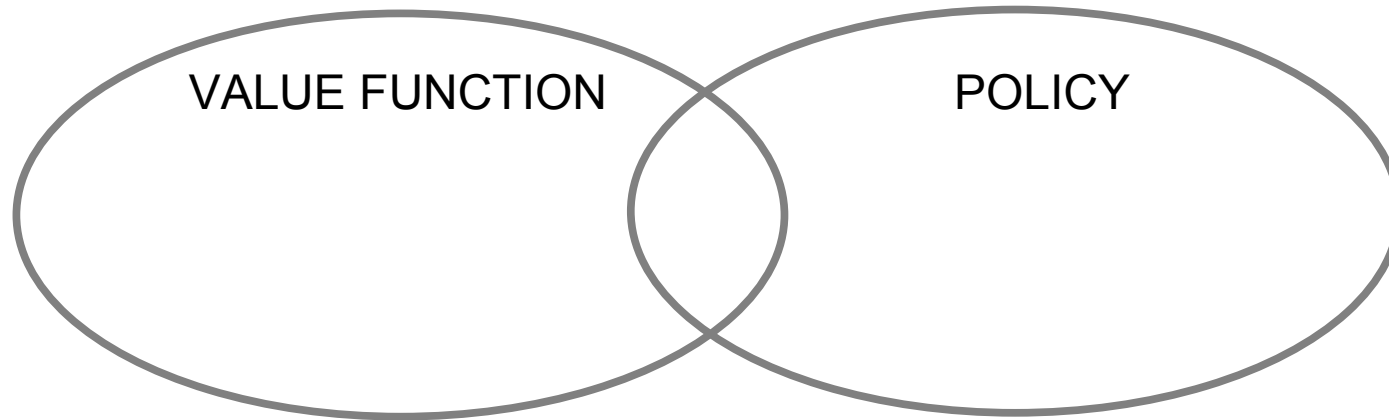
Today

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

Learning goals

- Understand basis of actor-critic approaches

Value-based vs policy-based RL



Value-based

- Learnt value function
- Implicit policy

Actor-critic

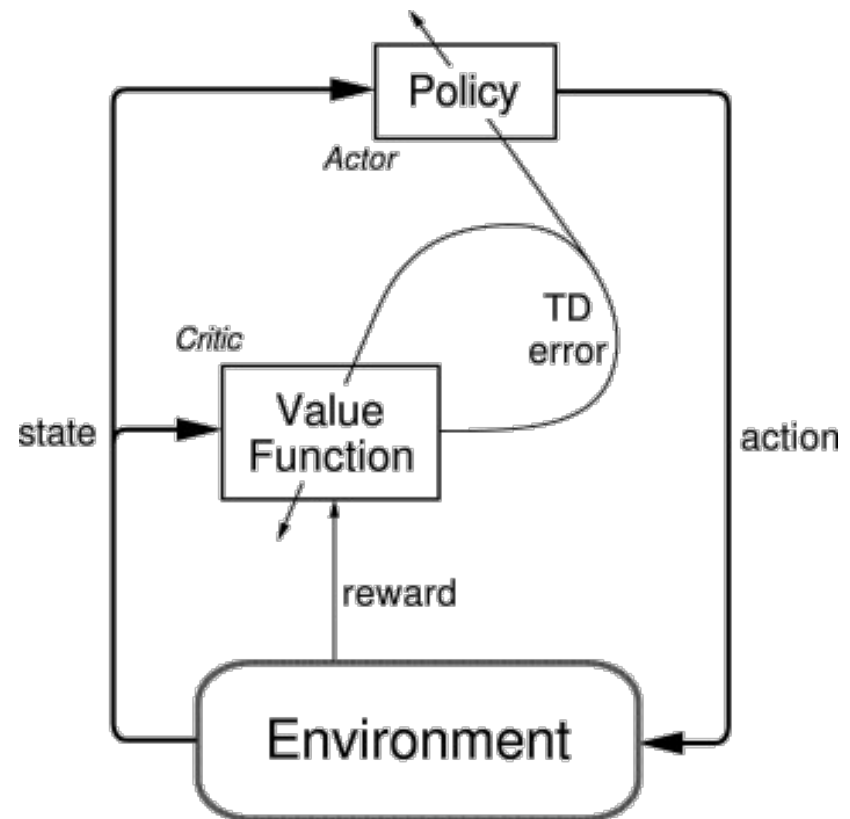
- **Learnt value function**
- **Learnt policy**

Policy-based

- No value function
- Learnt policy

Actor-critic approach – overview

- *Critic* estimates value function
- *Actor* updates policy in direction of critic
- For example, policy gradient where critic estimates value function
 - See previous lectures



Policy gradient – recap

REINFORCE

1. Run policy, collect $\{\tau_i\}$ $\tau_i = (s_0^i, a_0^i, r_0^i, s_1^i, a_1^i, r_1^i, \dots)$
2. $\nabla_{\theta} R(\theta) \approx \frac{1}{J} \sum_{i=1}^J \left(\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t^i | s_t^i) \left(\sum_{t'=t}^T \gamma^{t'} r(s_{t'}^i, a_{t'}^i) \right) \right)$
3. $\theta \leftarrow \theta + \alpha \nabla_{\theta} R(\theta)$

Policy gradient – recap

REINFORCE

1. Run policy, collect $\{\tau_i\}$ $\tau_i = (s_0^i, a_0^i, r_0^i, s_1^i, a_1^i, r_1^i, \dots)$
2. $\nabla_{\theta} R(\theta) \approx \frac{1}{J} \sum_{i=1}^J \left(\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t^i | s_t^i) \underbrace{\left(\sum_{t'=t}^T \gamma^{t'} r(s_{t'}^i, a_{t'}^i) \right)} \right)$
3. $\theta \leftarrow \theta + \alpha \nabla_{\theta} R(\theta)$

What's this?

Does it look familiar?

Policy gradient – recap

REINFORCE

1. Run policy, collect $\{\tau_i\}$ $\tau_i = (s_0^i, a_0^i, r_0^i, s_1^i, a_1^i, r_1^i, \dots)$
2. $\nabla_{\theta} R(\theta) \approx \frac{1}{J} \sum_{i=1}^J \left(\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t^i | s_t^i) \underbrace{\left(\sum_{t'=t}^T \gamma^{t'} r(s_{t'}^i, a_{t'}^i) \right)} \right)$
3. $\theta \leftarrow \theta + \alpha \nabla_{\theta} R(\theta)$

What's this?

Does it look familiar?

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

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

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

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

Remember the baselines?

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

How to find a good baseline for $Q(\mathbf{s}_t^i, \mathbf{a}_t^i)$?

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

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

Remember the baselines?

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

How to find a good baseline for $Q(\mathbf{s}_t^i, \mathbf{a}_t^i)$?

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 $V(\mathbf{s}_t)$ as baseline!

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

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

Remember the baselines?

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

How to find a good baseline for $Q(\mathbf{s}_t^i, \mathbf{a}_t^i)$?

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 $V(\mathbf{s}_t)$ as baseline!

Value of \mathbf{s} compared to the expected value

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

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

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

Determining the advantage

How to find a good estimate for this?

$$\nabla_{\theta} R(\theta) \approx \frac{1}{J} \sum_{i=1}^J \left(\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_t^i | \mathbf{s}_t^i) A(\mathbf{s}_t^i, \mathbf{a}_t^i) \right) \quad \text{Estimate } Q, V, \text{ or } A?$$

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

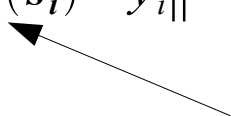
$$A(\mathbf{s}_t, \mathbf{a}_t) = Q(\mathbf{s}_t, \mathbf{a}_t) - V(\mathbf{s}_t)$$

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

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

Thus, knowing V allows approximating A .

Fitting value functions (mostly recap)

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

Fitting value functions (mostly recap)

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

$$L(\phi) = \sum_i \|V_\phi(s_i) - y_i\|^2$$

Any parametric function approximator

Wrap-up: A batch TD actor critic

Batch actor-critic

1. Run policy, collect $\{\tau_i\}$ $\tau_i = (s_0^i, a_0^i, r_0^i, s_1^i, a_1^i, r_1^i, \dots)$
2. Fit $V_\phi(s_t)$
3. Evaluate $A(s_t, a_t) \approx r_{(s_t, a_t)} + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$
4. Evaluate $\nabla_\theta R(\theta) \approx \frac{1}{J} \sum_{i=1}^J \left(\sum_{t=0}^T \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) A(s_t^i, a_t^i) \right)$
5. Update $\theta \leftarrow \theta + \alpha \nabla_\theta R(\theta)$
6. Repeat

ϕ : value function parameters


θ : policy parameters

An on-line TD actor critic

On-line actor-critic

1. Take action $a = \pi(a|s)$
2. Update $V_\phi(s_t)$ using $\phi \leftarrow \phi + \beta (r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)) \nabla_\phi V_\phi(s_t)$
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

learning
rate



From lecture 4!



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

Deep Deterministic Policy Gradient (DDPG): basic idea

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: $\phi_{targ} \leftarrow \xi \phi_{targ} + (1 - \xi) \phi$

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

Deep Deterministic Policy Gradient (DDPG) with batches

Off-policy actor-critic with batch updates and “slow” targets

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

2. Update $Q_{\phi}(s_t, a_t)$ using a sampled batch B from D and

$$\phi \leftarrow \phi + \frac{\beta}{|B|} \nabla_{\phi} \sum_B \left(Q_{\phi}(s_t, a_t) - (r_t + \gamma Q_{\phi_{targ}}(s_{t+1}, \pi_{\theta_{targ}}(s_{t+1}))) \right)^2$$

3. Update $\pi_{\theta}(s_t)$ using $\theta \leftarrow \theta + \frac{\alpha}{|B|} \nabla_{\theta} \sum_B Q_{\phi}(s_t, \pi_{\theta}(s_t))$

4. “Delayed” update of Q and policy parameter targets

using run-time averaging: $\phi_{targ} \leftarrow \xi \phi_{targ} + (1 - \xi) \phi$

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

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 $R_{\theta_{old}}^{IS}(\theta)$ approximates performance difference between previous and updated policies

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

See the importance sampling in effect!

Bounding surrogate advantage

Result: Policy is guaranteed to improve by optimizing

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

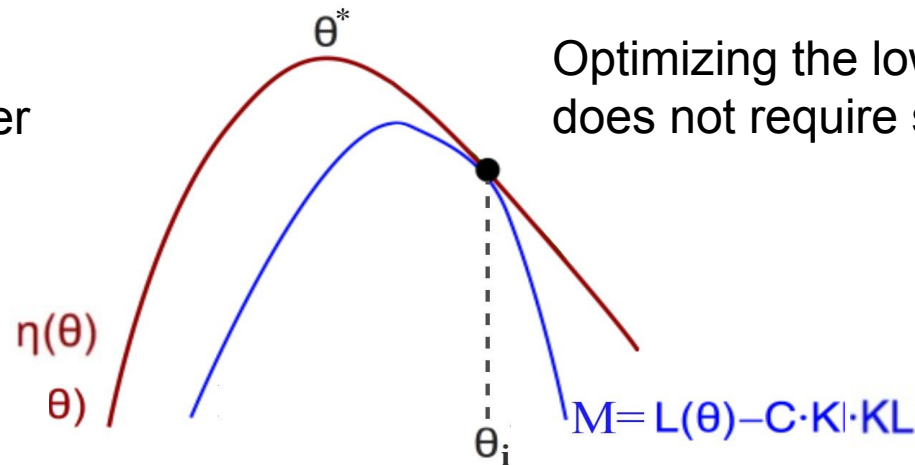
where

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

known constant

is the maximum Kullback-Leibler (KL) divergence between the policies.

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



Trust region policy optimization (Schulman et al. 2015)

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

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

such that

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

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

Trust region policy optimization (Schulman et al. 2015)

Optimize surrogate advantage and constrain KL-divergence:

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

such that

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

In practice, use second order approximation for the KL-divergence constraint (see paper for details)

Nice. This leads to a “natural gradient” [Amari et al., Neural Computation, 1998]: changes to policy are measured in action space, not parameter space unlike standard policy gradient

Not so nice: For policies with many parameters costly to compute

Proximal policy optimization (Schulman et al. 2017)

Remember the surrogate advantage?

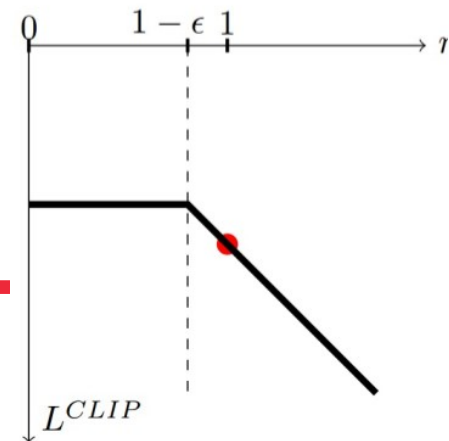
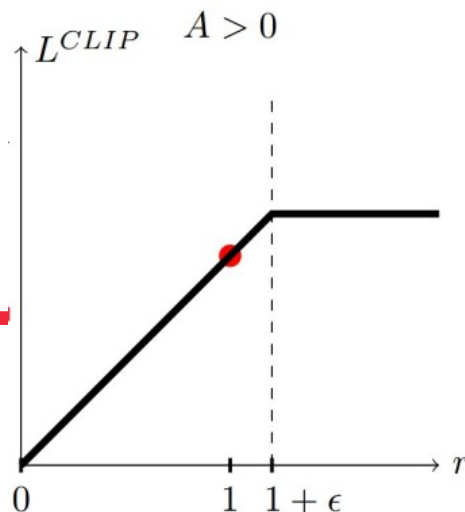
$$R_{\theta_{old}}^{IS}(\theta) = E_{\tau \sim \pi_{\theta_{old}}} \left[\underbrace{\frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{old}}(a_t | s_t)}}_{g_t(\theta)} A^{\pi_{\theta_{old}}}(s_t, a_t) \right]$$

Optimize instead

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

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

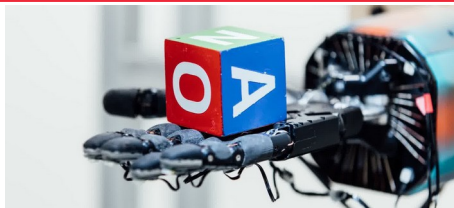
for $i = 1, 2, \dots$ **do**

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

Compute advantage estimates $A(s_t, a_t) \approx r_{(s_t, a_t)} + \gamma V(s_t) - V(s_{t+1})$
using current value function $V_\phi(s_t)$

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

Fit $V_\phi(s_t)$ by minimizing $L(\phi) \equiv \sum_i \|V_\phi(s_i) - y_i\|^2$ using gradient
descent



Other well known 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