Aalto University
School of Electrical Engineering

ELEC-E8125 Reinforcement Learning Actor-critic methods

Ville Kyrki

29.10.2019

Today

- Combining policy gradient with value functions
   - $\rightarrow$ actor-critic methods

Learning goals

- Understand basis of actor-critic approaches.

Motivation

- Even with value function approximation, large state spaces can be problematic.

- Learning parametric policies $\pi(u|x,\theta)$ directly without learning value functions sometimes easier.

- Non-Markov (partially observable) or adversarial situations might benefit from stochastic policies.

Aalto University
School of Electrical
Engineering

Value-based vs policy-based RL

VALUE FUNCTION   POLICY
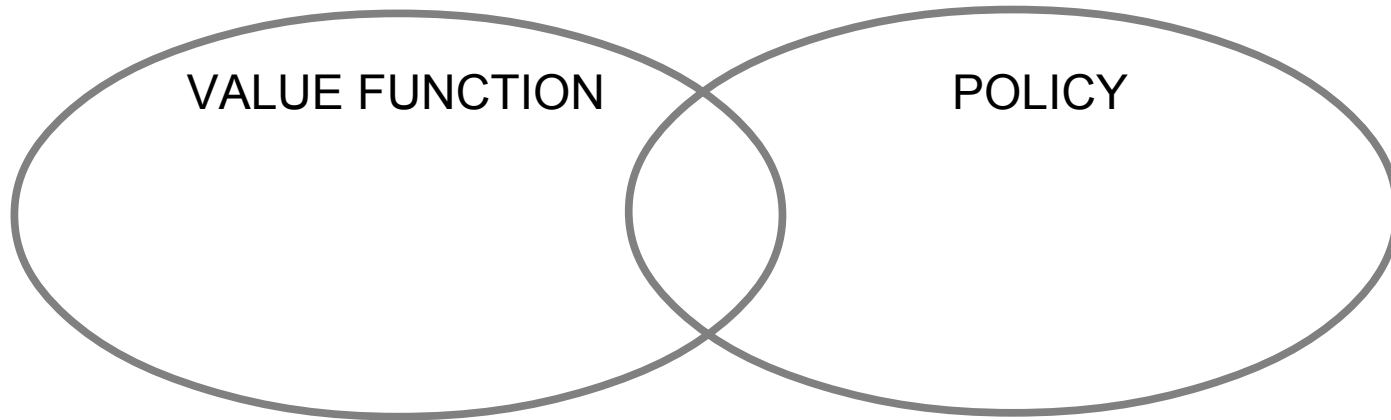
Value-based
· Learnt value function.
· Implicit policy.

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

Can we combine these?

Value-based vs policy-based RL

VALUE FUNCTION   POLICY
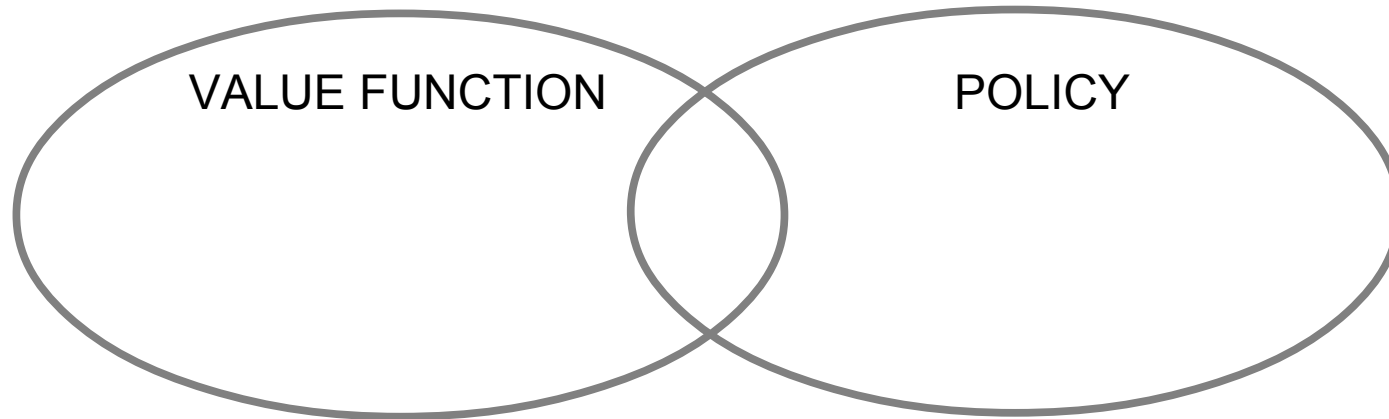
Value-based
· Learnt value function.
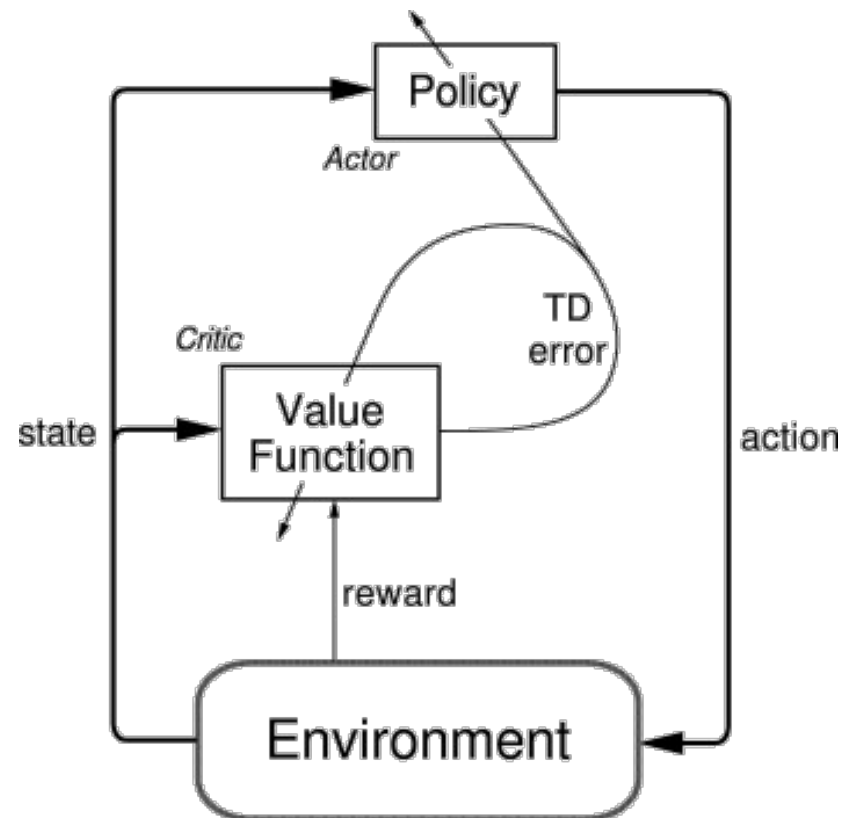· Implicit policy.

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

REINFORCE

1. Run policy, collect $\{\tau_i\}$ $\quad \tau_i = (x_0^i, u_0^i, r_0^i, x_1^i, u_1^i, r_1^i, \ldots)$

2. $\nabla_\theta R(\theta) \approx \dfrac{1}{J} \sum_{i=1}^{J} \left(\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(u_t^i | x_t^i) \left(\sum_{t'=t}^{T} r(x_{t'}^i, u_{t'}^i) \right) \right)$

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

Policy gradient – recap

REINFORCE

1. Run policy, collect $\{\tau_i\}$ $\quad \tau_i = (x_0^i, u_0^i, r_0^i, x_1^i, u_1^i, r_1^i, \dots)$

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

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

What's this?
Does it look familiar?

Policy gradient – recap

REINFORCE

1. Run policy, collect $\{\boldsymbol{\tau}_i\}$ $\quad \boldsymbol{\tau}_i = (\boldsymbol{x}_0^i, \boldsymbol{u}_0^i, r_0^i, \boldsymbol{x}_1^i, \boldsymbol{u}_1^i, r_1^i, \dots)$

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

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

What's this?
Does it look familiar?

$$Q_\pi(x_t, u_t) = \sum_{t=t'}^T E\left[r(x_{t'}^i, u_{t'}^i) | x_t, u_t \right]$$

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

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

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

Remember the baselines?

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

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

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?

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

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

But what does the average mean here?

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

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

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

Remember the baselines?

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

Average is a good baseline: $b_t = \dfrac{1}{J} \sum_{i=1}^{J} Q(x_t^i, u_t^i)$

But what does the average mean here?

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

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

advantage function $A(\boldsymbol{x_t^i}, \boldsymbol{u_t^i})$

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

Determining the advantage

How to find a good estimate for this?

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

Estimate Q, V, or A?

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

$$A(x_t, u_t) = Q(x_t, u_t) - V(x_t)$$

$$Q(x_t, u_t) = r(x_t, u_t) + E_{x_{t+1} \sim \pi(x_{t+1}|x_t, u_t)} \left[V(x_{t+1}) \right]$$

$$A(x_t, u_t) \approx r_{(x_t, u_t)} + V(x_t) - V(x_{t+1})$$

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

- Requires episodic environments to get the value.

- Fitting criterion $L(\phi) = \sum_i \left\| V_\phi(\boldsymbol{x_i}) - y_i \right\|^2$

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

- Identical fitting criterion

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

Any parametric function approximator

Wrap-up: A batch TD actor critic

Batch actor-critic

1. Run policy, collect $\{\boldsymbol{\tau}_i\}$ $\boldsymbol{\tau}_i = (\boldsymbol{x_0^i}, \boldsymbol{u_0^i}, r_0^i, \boldsymbol{x_1^i}, \boldsymbol{u_1^i}, r_1^i, \ldots)$

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

3. Evaluate $A(\boldsymbol{x_t}, \boldsymbol{u_t}) \approx r_{(\boldsymbol{x_t}, \boldsymbol{u_t})} + V_\phi(\boldsymbol{x_t}) - V_\phi(\boldsymbol{x_{t+1}})$

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

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

What about discount?

An on-line TD actor critic (with discount)

learning rate

On-line actor-critic

From lecture 5!

1. Take action $u = \pi(u|x)$

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

3. Evaluate $A(x_t, u_t) \approx r_{(x_t, u_t)} + \gamma V(x_t) - V(x_{t+1})$

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

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

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

Note: TD estimate can be biased.

Challenge: Gradient step sizes

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

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

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

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

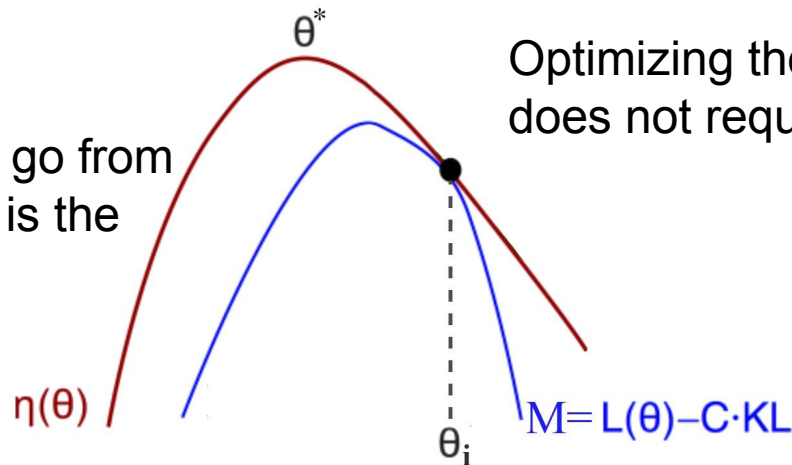$$max_{\theta}\left(R^{IS}_{\theta_{old}}(\theta) - c\, D^{max}_{KL}(\theta_{old}, \theta)\right)$$

where   known constant

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

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!

θ^{*}

$\eta(\theta)$

θ_i

$M = L(\theta) - C \cdot KL$

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:

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

such that

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

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)
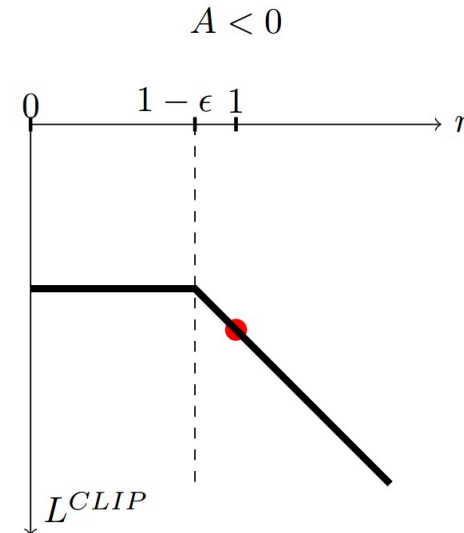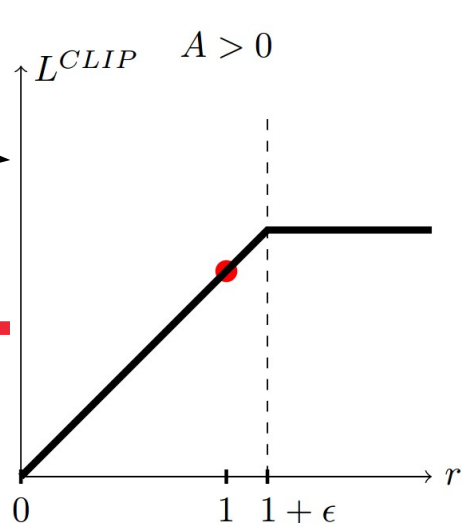
Remember the surrogate advantage?

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

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 $\{\tau_i\}$ $\tau_i = (x_0^i, u_0^i, r_0^i, x_1^i, u_1^i, r_1^i, \dots)$

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

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

   Fit $V_\phi(x_t)$ by minimizing $L(\phi) \equiv \sum_i \left\| V_\phi(x_i) - y_i \right\|^2$ 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: 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: no readings.