Lecture 09:
Enhancing Policy gradient

Radoslav Neychev

Sber RL course
Spring 2022

References

These slides are deeply based on Practical RL course week09 slides. Special thanks to YSDA team for making them publicly available.

Original slides link: week09_policy_II

Let τ be a trajectory $(s_0, a_0, s_1, a_1, ...)$

Recall:

$$J(\pi) = E_{\tau \sim \pi}\left[\sum_{t=0}^{T} \gamma^t r(s_t)\right]$$ - **goodness** of the π, π depends on θ

Policy gradient theorem:

$$\nabla_\theta J(\pi) \approx E_{(s,a)\sim\pi}\left[Q^\pi(s,a)\nabla_\theta log\, \pi(a|s)\right]$$

Let τ be a trajectory $(s_0, a_0, s_1, a_1, ...)$

Recall:

$$J(\pi) = E_{\tau\sim\pi}\left[\sum_{t=0}^{T} \gamma^t r(s_t)\right]$$ - **goodness** of the π, π depends on θ

Policy gradient theorem:

$$\nabla_\theta J(\pi) \approx E_{(s,a)\sim\pi}\left[Q^\pi(s,a)\nabla_\theta log\,\pi(a|s)\right]$$

Problem: Value of learning rate doesn't guarantee degree of policy change.

Let's use smarter optimization method.

Optimization

Suppose we want to optimize $\mathrm{F}(\theta)$ *by* θ in local neighbourhood

We approximate F by $\hat{F}(\theta) \approx F(\theta_0) + \nabla F(\theta)^T(\theta - \theta_0)$

Minimizing $\hat{F}(\theta)$ is the same as minimizing $\nabla F(\theta)^T(d), d = \theta - \theta_0$

We want to find d that

1) minimize $\nabla F(\theta)^T(d)$

2) $\boxed{d^T d} < \epsilon$

Distance

Using Lagrange multipliers, we find that $\mathrm{d}_{opt} \propto -\nabla F(\theta)$

Optimization

Now, let's measure distance using non-identical matrix K:

We want to find d that

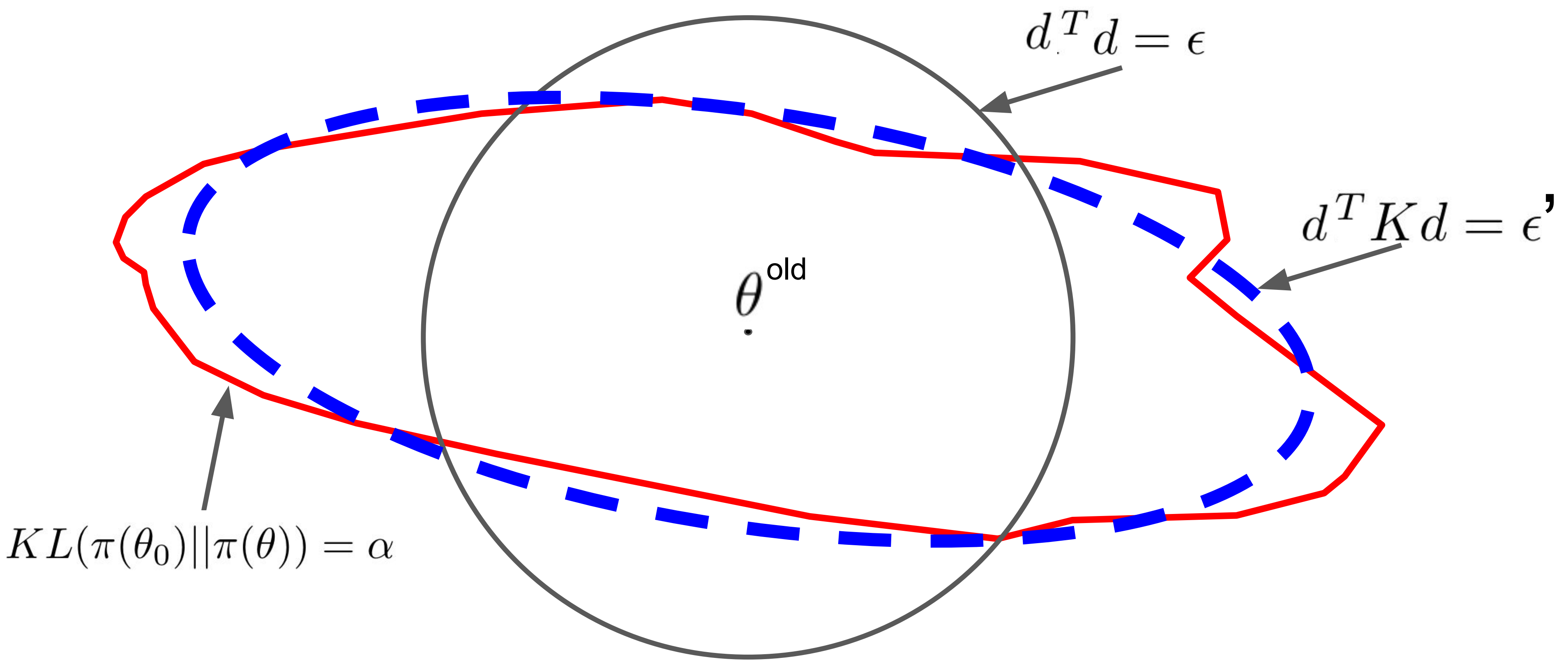
1) minimize $\nabla F(\theta)^T(d)$

2) $\boxed{d^T K d} < \epsilon$

Distance

Using Lagrange multipliers, we find that $\mathrm{d}_{opt} \propto -K^{-1}\nabla F(\theta)$

Space of parameters



Natural Policy Gradient

Suppose:

$$KL(\pi(\theta_0)||\pi(\theta)) \approx 0.5*(\theta-\theta_0)^T K(\theta-\theta_0) = 0.5*d^T K d, K = \nabla_\theta^2 KL(\pi(\theta_0)||\pi(\theta))$$

Solve constrained equation: find vector d that

1) Minimize $\nabla J^T d$

2) $d^T K d < \epsilon$

Solution: $\mathrm{d}_{opt} \propto K^{-1} \nabla J(\theta)$

New update rule: $\theta_{t+1} = \theta_t - \alpha K^{-1} \nabla J(\theta)$

Natural Policy Gradient

Suppose:

$$KL(\pi(\theta_0)||\pi(\theta)) \approx 0.5*(\theta-\theta_0)^T K(\theta-\theta_0) = 0.5*d^T K d, K = \nabla_\theta^2 KL(\pi(\theta_0)||\pi(\theta))$$

Solve constrained equation: find vector d that

1) Minimize $\nabla J^T d$

2) $d^T K d < \epsilon$

Solution: $\mathrm{d}_{opt} \propto K^{-1} \nabla J(\theta)$

New update rule: $\theta_{t+1} = \theta_t - \alpha K^{-1} \nabla J(\theta)$

Problems: Value of learning rate still doesn't guarantee degree of policy change.
It may be too hard to compute inverse of K.

If we want to find $K^{-1}\nabla J(\theta)$ we may solve $Kx = \nabla J(\theta)$

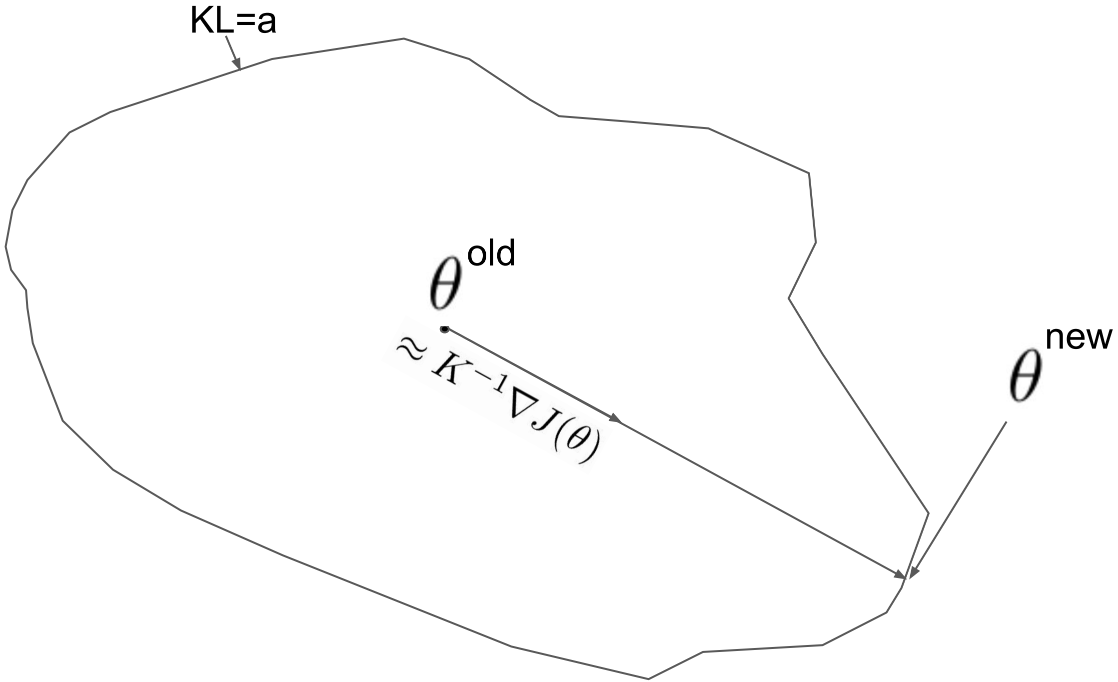
Matrix K is positive-definite so we can use **conjugate gradients**

Number of iterations k allows us to trade-off between precision and time.

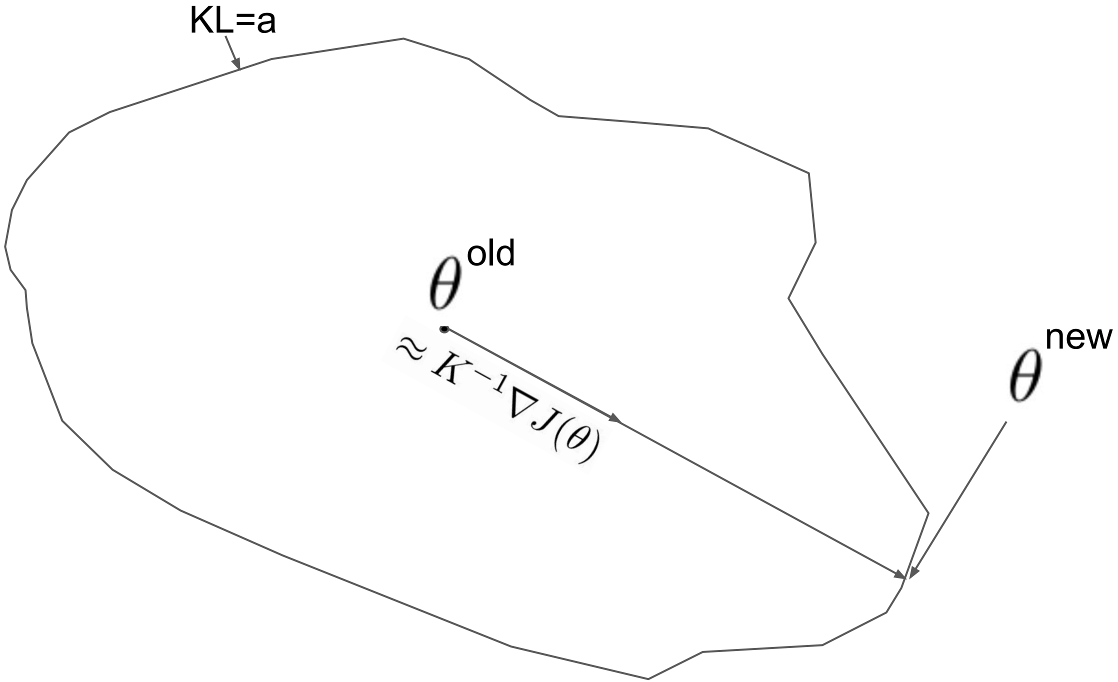
As a result: $\Delta\theta \approx \alpha K^{-1}\nabla J(\theta)$

Last problem: Value of learning rate still doesn't guarantee degree of policy change.

Linear search!

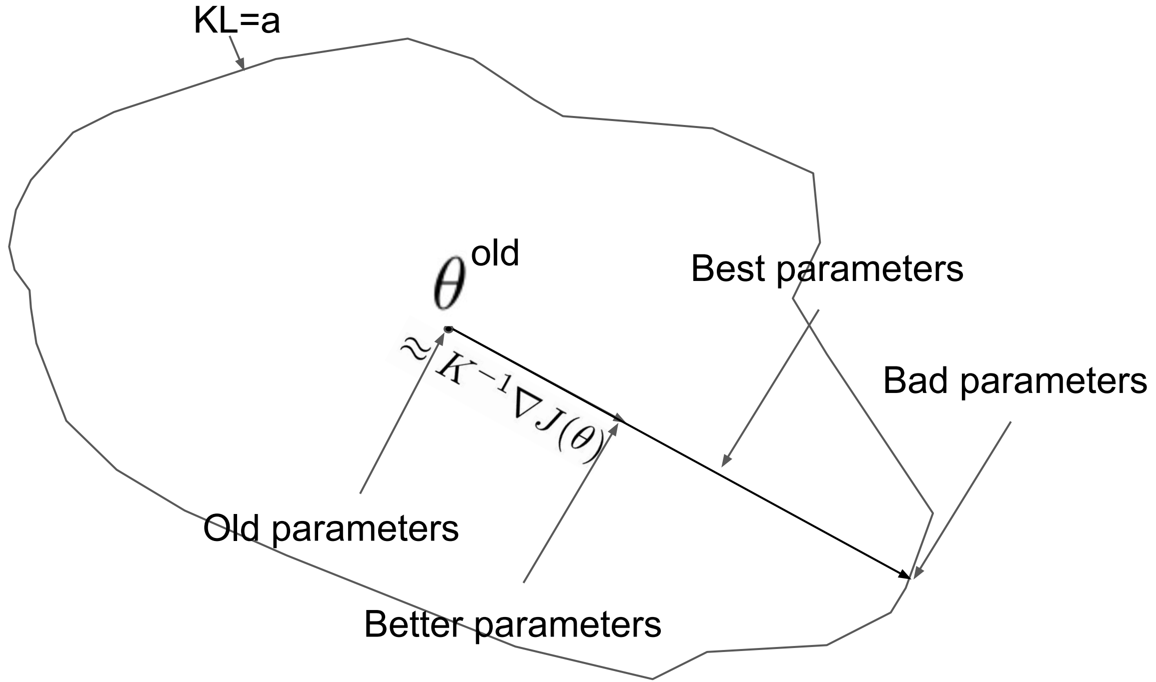


Linear search!

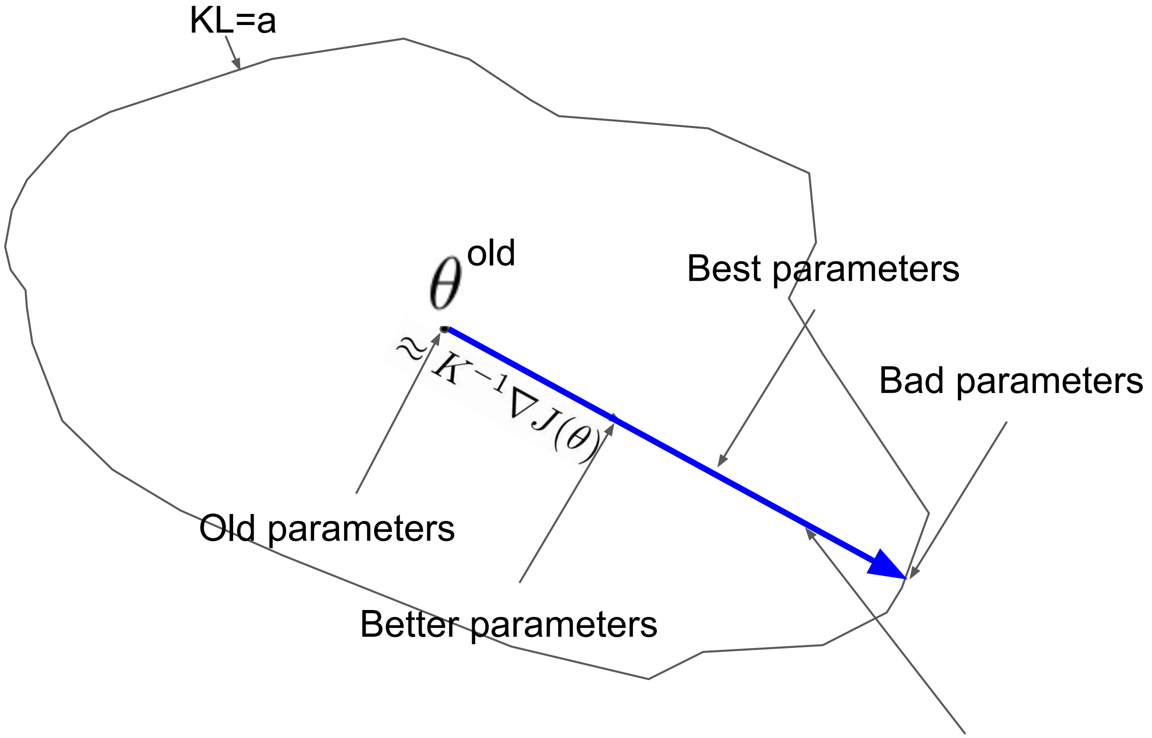


Problem?

Imagine this situation:



Imagine this situation:



We want to compute loss function here!

Recall: $J(\pi) = E_{\tau\sim\pi}\Big[\sum_{t=0}^{T}\gamma^t r(s_t)\Big]$

It can be proven that $J(\tilde{\pi}) = J(\pi) + E_{\tau\sim\tilde{\pi}}\Big[\sum_{t=0}^{T}\gamma^t A_\pi(s_t, a_t)\Big]$

Let's rewrite it this way $J(\tilde{\pi}) = J(\pi) + \sum_s \rho_{\tilde{\pi}}(s) \sum_a \tilde{\pi}(a|s) A_\pi(s, a)$

Trust Region trick:

If $E_s\Big[KL(\pi \,||\, \tilde{\pi})\Big]$ is small,

$$J(\tilde{\pi}) \approx J(\pi) + \sum_s \rho_{\pi}(s) \sum_a \tilde{\pi}(a|s) A_\pi(s, a)$$

Then:

$$J(\tilde{\pi}) \approx J(\pi) + \sum_s \rho_\pi(s) \sum_a \tilde{\pi}(a|s) A_\pi(s,a) =$$

$$= J(\pi) + \sum_s \rho_\pi(s) \sum_a \pi(a|s) * \frac{\tilde{\pi}(a|s)}{\pi(a|s)} * A_\pi(s,a) =$$

$$= J(\pi) + \boxed{E_{(s,a)\sim\pi}[\frac{\tilde{\pi}(a|s)}{\pi(a|s)} A_\pi(s,a)]}$$

Can be computed at every point!

Trust Region Policy Optimization

1) Sample state-action pairs from on-policy distribution

2) Compute $g = \nabla_{\theta'} \hat{J}(\tilde{\pi})$ = $\nabla_{\theta'} \frac{1}{N} \sum_{i=0}^{N} \frac{\tilde{\pi}(s_i, a_i)}{\pi(s_i, a_i)} A_\pi(s_i, a_i)$

$$K = \nabla_{\theta'}^2 \frac{1}{N} \sum_{i=0}^{N} KL(\pi(s_i) \,||\, \tilde{\pi}(s_i))$$

3) Find $\hat{d} = -1 * ConjGrad(Kx = g)$

4) Do linear search in direction of $\hat{d}$, constraint $\frac{1}{N} \sum_{i=0}^{N} KL(\pi(s_i) \,||\, \tilde{\pi}(s_i)) < \alpha$

and simultaneously check value of $\frac{1}{N} \sum_{i=0}^{N} \frac{\tilde{\pi}(s_i, a_i)}{\pi(s_i, a_i)} A_\pi(s_i, a_i)$

TRPO

Advantages

- Very stable training
- Good result

Disadvantages

- Cheap sampling is necessary
- Not easy to implement

KL constraint

Strict: TRPO

$$\underset{\theta}{\text{maximize}} \quad \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right]$$

$$\text{subject to} \quad \hat{\mathbb{E}}_t[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]] \leq \delta.$$

Penalty: PPO-decay

$$\underset{\theta}{\text{maximize}}\, \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t - \beta\,\text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]\right]$$

Different losses

Basic objective

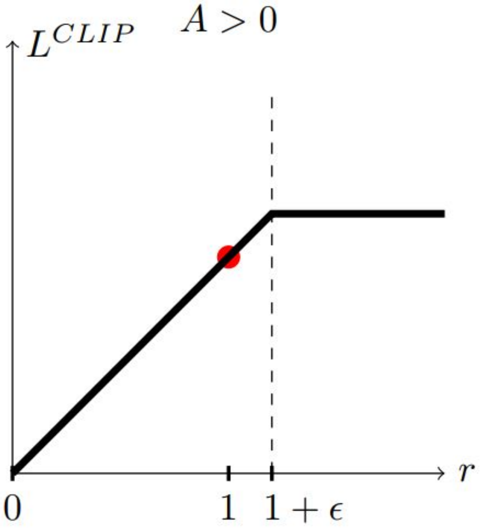
$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}, \text{ so } r(\theta_{\text{old}}) = 1$$

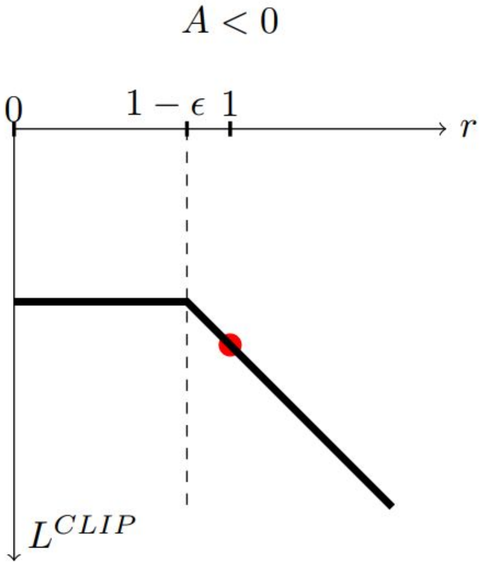
$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right] = \hat{\mathbb{E}}_t\left[r_t(\theta)\hat{A}_t\right]$$

Different losses

Clipped objective

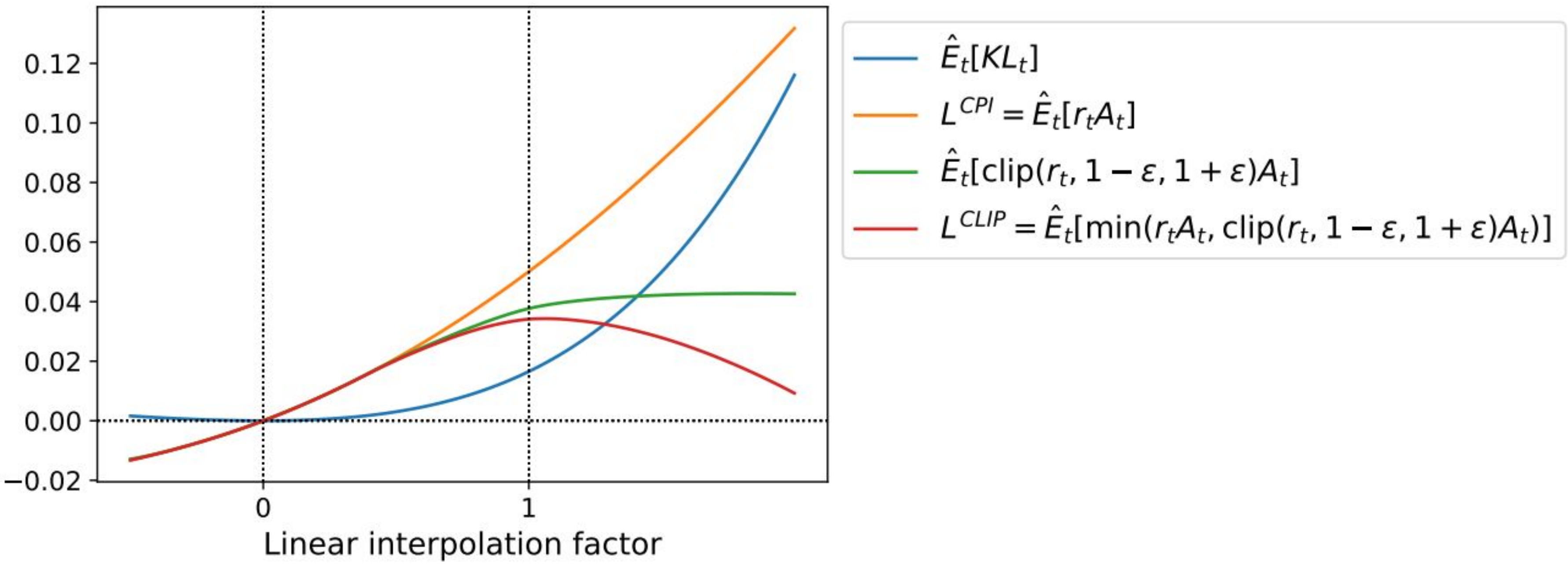
$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$





Behaviour of losses

Suppose we interpolate the parameters of the model between updates



PPO-clip

Repeat:

1) Collect data

2) Update policy a few times to maximize $L^{CLIP}(\theta)$

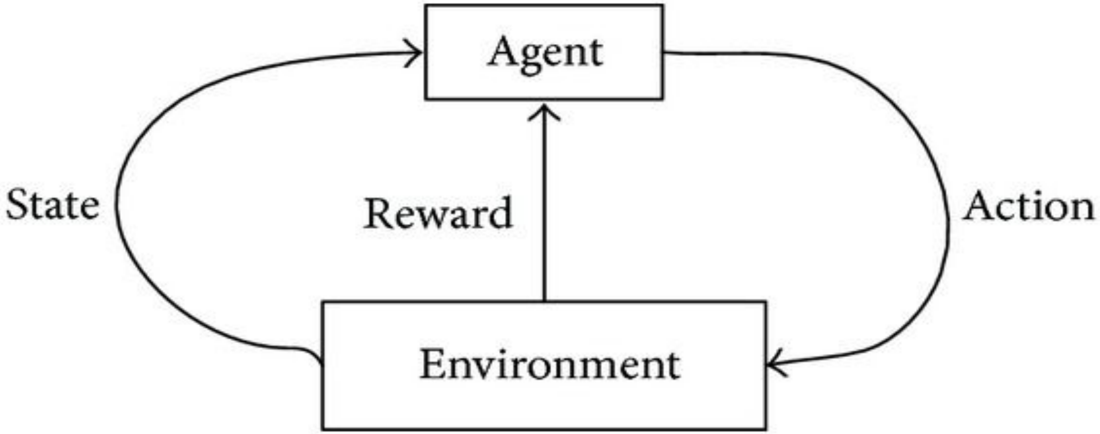
algorithm	avg. normalized score
No clipping or penalty	-0.39
Clipping, $\epsilon = 0.1$	0.76
Clipping, $\epsilon = 0.2$	**0.82**
Clipping, $\epsilon = 0.3$	0.70
Fixed KL, $\beta = 0.3$	0.62
Fixed KL, $\beta = 1.$	0.71
Fixed KL, $\beta = 3.$	0.72
Fixed KL, $\beta = 10.$	0.69

TRPO	PPO
• Works for smaller models • Second-order optimization	• Works for big models • First-order optimization

Continuous action spaces

Regular MDP

$a \in \mathbb{R}^n$



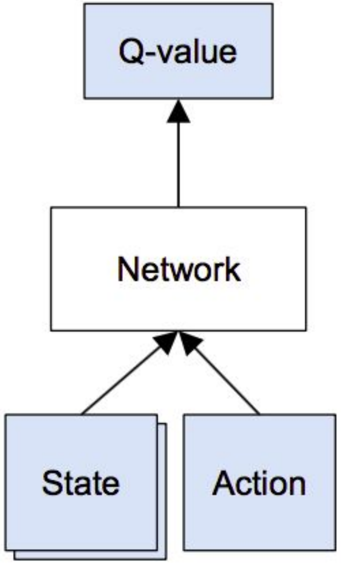
Which methods can we use?

Continuous action spaces

- We can learn critic easily
- The problem is finding

$$a_{opt}(s) = \underset{a}{argmax}\, Q(s,a)$$

Worst case: optimize over neural net!



Normalized advantage functions

- First idea: restrict Q(s,a) so that optimization becomes trivial
- For example, parabola (for 1d action space)

$$Q(s,a)=V(s)+A(s,a)$$

$$A(s,a)=-k_\theta(s)\cdot(a-\mu_\theta(s))^2$$

How to find optimal a?

a_opt = mu(s)

Normalized advantage functions

First idea: restrict Q(s,a) so that optimization becomes trivial
For example, parabola (for 1d action space)

$$Q(s,a)=V(s)+A(s,a)$$

$$A(s,a)=-k_\theta(s)\cdot(a-\mu_\theta(s))^2$$

Q: How does it generalize for n-dimensional **a**?

Normalized advantage functions

First idea: restrict Q(s,a) so that optimization becomes trivial

$$Q(s,a)=V(s)+A(s,a)$$

$$A(s,a)=-0.5\cdot(a-\mu_\theta(s))^T\cdot L(s)\cdot L(s)^T(a-\mu_\theta(s))$$

Where L(s) is a lower-triangular matrix

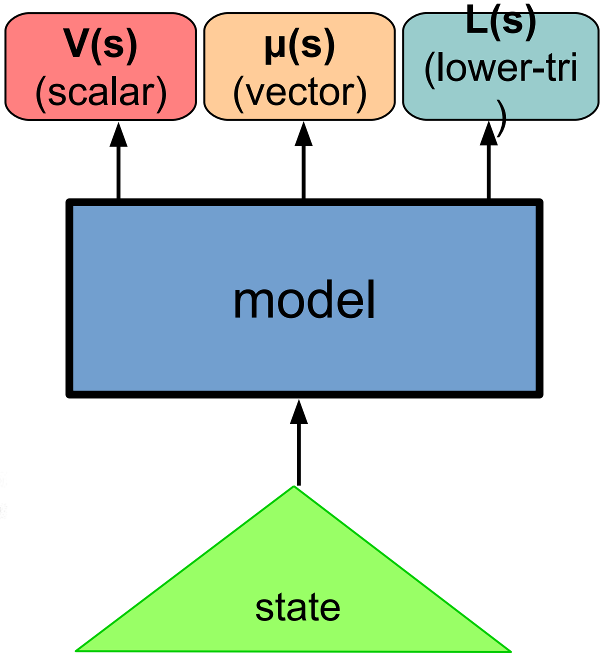
Normalized advantage functions

Network trains end-to-end

$$Q(s,a)=V(s)+A(s,a)$$

$$A(s,a)=\ldots$$

$$\underset{\theta}{argmin}\left(Q(s_t,a_t)-\left[r+\gamma\cdot V(s_{t+1})\right]\right)^2$$



Source: https://arxiv.org/pdf/1603.00748.pdf

Deterministic policy gradient

Second idea: learn a separate network to find a_opt

Train critic $Q_\theta(s, a)$

$$\underset{\theta}{argmin}\left(Q(s_t, a_t) - [r + \gamma \cdot V(s_{t+1})]\right)^2$$

Train actor $a_{opt}(s) \approx \mu_\theta(s)$

How do we get V(s')?

$$\nabla_\theta J = \frac{\partial Q^\theta(s, a)}{\partial a} \frac{\partial \mu(s|\theta)}{\partial \theta}$$

Deterministic policy gradient

Second idea: learn a separate network to find a_opt

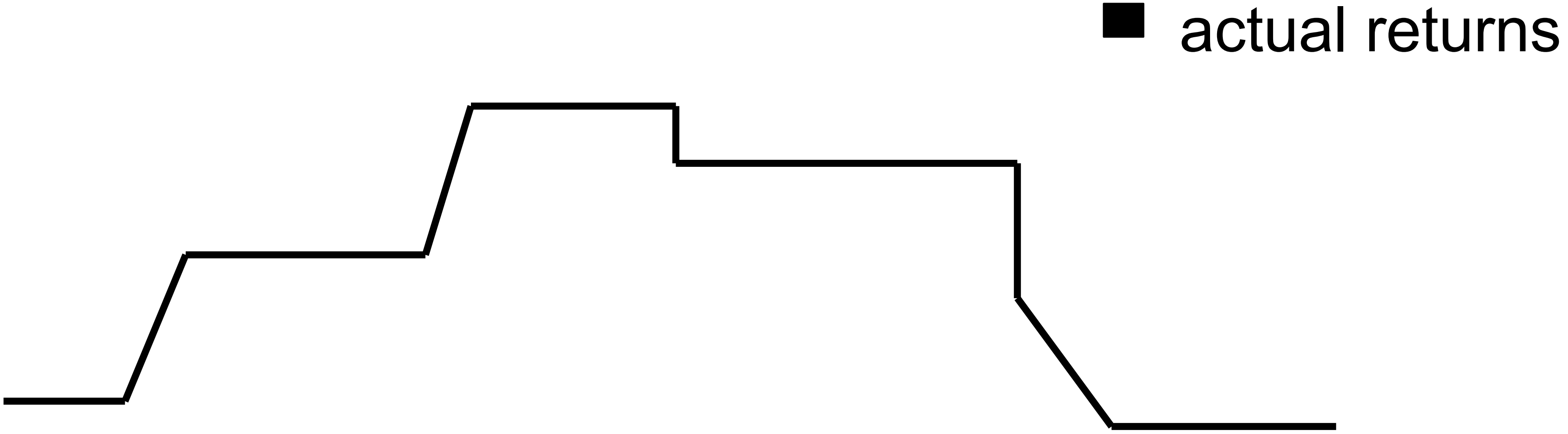
Train critic $Q_\theta(s, a)$

$$\underset{\theta}{argmin}\left(Q(s_t, a_t) - [r + \gamma \cdot Q(s_{t+1}, \mu_\theta(s_{t+1}))]\right)^2$$

Train actor $a_{opt}(s) \approx \mu_\theta(s)$

$$\nabla_\theta J = \frac{\partial Q^\theta(s, a)}{\partial a} \frac{\partial \mu(s|\theta)}{\partial \theta}$$

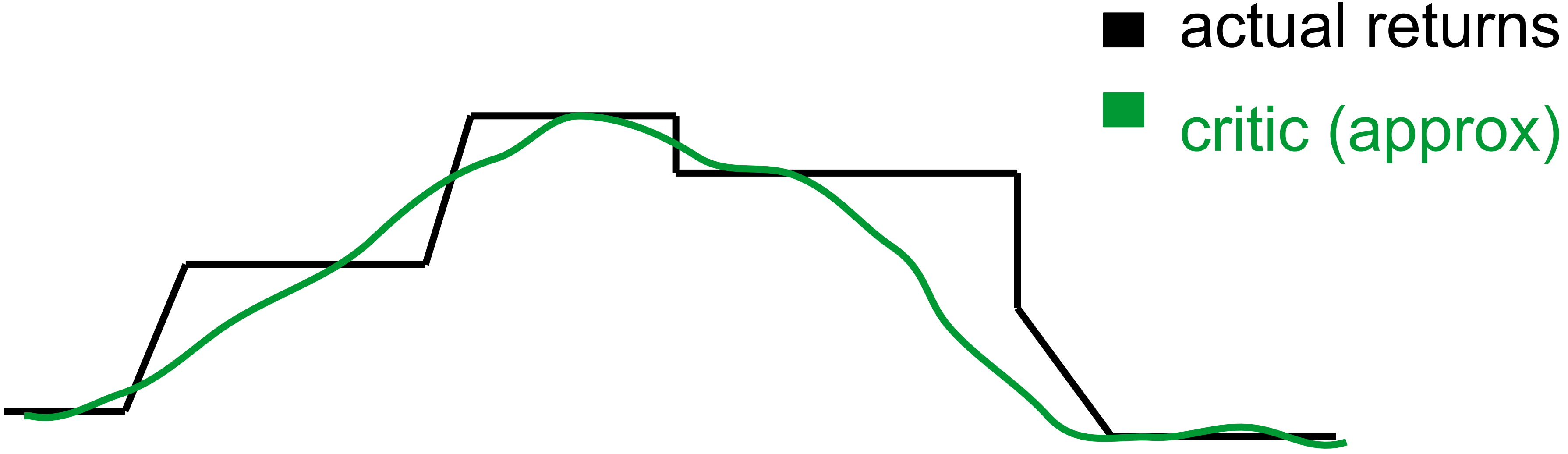
Deterministic policy gradient



Deterministic policy gradient

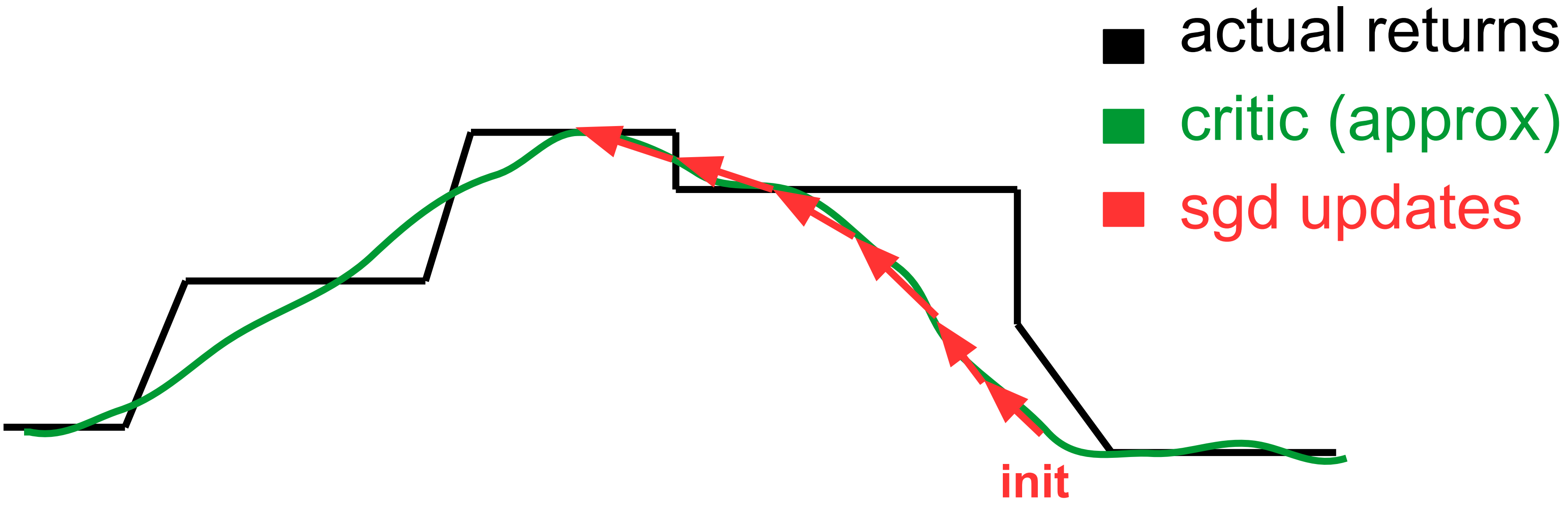
- Gradient approximation:

$$\nabla_\theta J = \frac{\partial Q^\theta(s,a)}{\partial a}\frac{\partial \mu(s|\theta)}{\partial \theta}$$

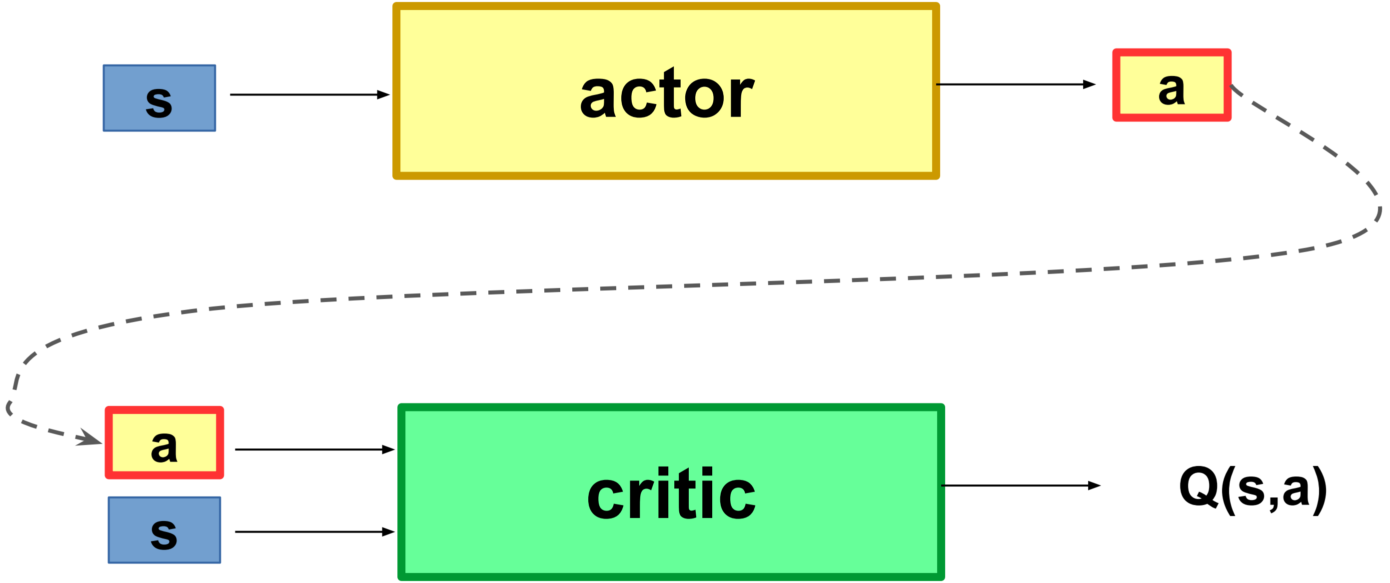


Deterministic policy gradient

- Gradient approximation: $$\nabla_\theta J = \frac{\partial Q^\theta(s,a)}{\partial a}\frac{\partial \mu(s|\theta)}{\partial \theta}$$



Going neural



Duct tape zone

- In general
  - “Natural” for continuous action spaces
    - Discrete: use gumbel-softmax, bit.ly/2v0Xfpz
  - Approximation is best around current policy
  - Weak critic can introduce bias

- vs. REINFORCE
  - Better off-policy
  - Less variance if reward is smooth
  - (subjectively) harder to tune

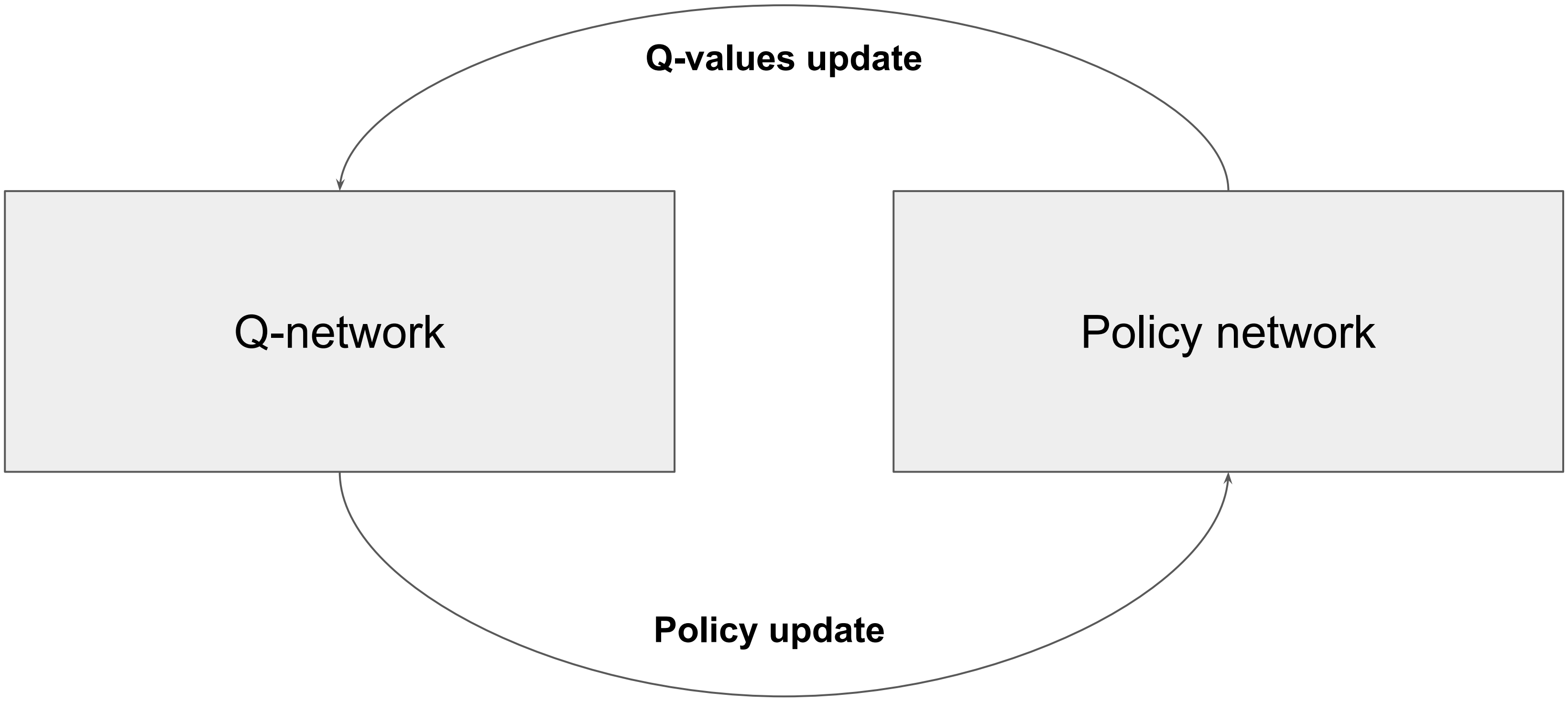
Deterministic policy gradient

- Demo with torcs http://bit.ly/2pXwdKa

Thank you for your attention!

Backup

Policy iteration for neural networks



Q-values update

$$J_Q(\theta) = \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \mathcal{D}} \left[\frac{1}{2} \left(Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - \hat{Q}(\mathbf{s}_t, \mathbf{a}_t) \right)^2 \right]$$

$$\hat{Q}(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \, \mathbb{E}_{\mathbf{s}_{t+1} \sim p} \left[V_{\bar{\psi}}(\mathbf{s}_{t+1}) \right]$$

$$\hat{\nabla}_\psi J_V(\psi) = \nabla_\psi V_\psi(\mathbf{s}_t) \left(V_\psi(\mathbf{s}_t) - Q_\theta(\mathbf{s}_t, \mathbf{a}_t) \right)$$

Policy update

$$\pi_{\text{new}} = \arg\min_{\pi' \in \Pi} \mathrm{D}_{\mathrm{KL}} \left(\pi'(\,\cdot\,|\mathbf{s}_t) \,\middle\|\, \frac{\exp\left(Q^{\pi_{\text{old}}}(\mathbf{s}_t,\,\cdot\,)\right)}{Z^{\pi_{\text{old}}}(\mathbf{s}_t)} \right)$$

Which can be rewritten as:

$$\mathbf{a}_t = f_\phi(\epsilon_t; \mathbf{s}_t)$$

$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}, \epsilon_t \sim \mathcal{N}} \left[\log \pi_\phi(f_\phi(\epsilon_t; \mathbf{s}_t)|\mathbf{s}_t) - Q_\theta(\mathbf{s}_t, f_\phi(\epsilon_t; \mathbf{s}_t)) \right]$$

Maximum entropy RL

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \rho_\pi} \left[r(\mathbf{s}_t, \mathbf{a}_t) + \alpha \mathcal{H}(\pi(\cdot | \mathbf{s}_t)) \right]$$

Soft Policy Iteration

$$\mathcal{T}^\pi Q(\mathbf{s}_t, \mathbf{a}_t) \triangleq r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \, \mathbb{E}_{\mathbf{s}_{t+1} \sim p} \left[V(\mathbf{s}_{t+1}) \right],$$

where

$$V(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t \sim \pi} \left[Q(\mathbf{s}_t, \mathbf{a}_t) - \log \pi(\mathbf{a}_t | \mathbf{s}_t) \right]$$

So what's new?

$$\hat{\nabla}_\psi J_V(\psi) = \nabla_\psi V_\psi(\mathbf{s}_t) \left(V_\psi(\mathbf{s}_t) - Q_\theta(\mathbf{s}_t, \mathbf{a}_t) + \log \pi_\phi(\mathbf{a}_t | \mathbf{s}_t) \right)$$

Automatic Tuning of Alpha

$$J(\alpha) = \mathbb{E}_{\mathbf{a}_t \sim \pi_t} \left[-\alpha \log \pi_t(\mathbf{a}_t | \mathbf{s}_t) - \alpha \bar{\mathcal{H}} \right]$$

Automatic Tuning of Alpha

$$J(\alpha) = \mathbb{E}_{\mathbf{a}_t \sim \pi_t} \left[-\alpha \log \pi_t(\mathbf{a}_t | \mathbf{s}_t) - \alpha \bar{\mathcal{H}} \right]$$

This algorithm is called **Soft Actor-Critic**