Reinforcement Learning

Advanced Policy Gradient

Александр Костин
telegramm: @Ko3tin
LinkedIn: kostinalexander

Recap: Policy Gradient

$$\nabla J_{PG}(\theta) = \mathbb{E}_{p_\theta(\tau)}\Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(A_t \,|\, S_t) \sum_{k=t}^{T} \gamma^{k-t} R_k\Big]$$

or

$$\nabla J_{PG}(\theta) = \mathbb{E}_{p_\theta(\tau)}\Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(A_t \,|\, S_t)[Q_{\pi_\theta}(S_t, A_t) - b(S_t)]\Big]$$

or

$$\nabla J_{AC}(\theta) = \mathbb{E}_{p_\theta(\tau)}\Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(A_t \,|\, S_t)[A_{\pi_\theta}(S_t, A_t)]\Big]$$

Recap: A2C

- Generate trajectories $\{\tau_i\}$ following $\pi_\theta(a \mid s)$
- Policy improvement:

  Estimate gradient and make gradient ascent step:

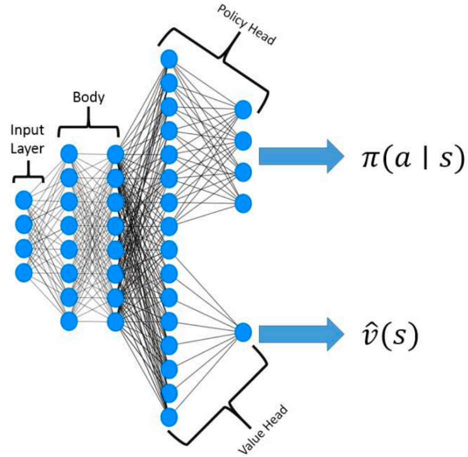
$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left[\sum_{t=0}^{T} \nabla \log \pi_\theta(a_{i,t} \mid s_{i,t}) A_{\pi_\theta}(s_{i,t}, a_{i,t}) \right]$$

- Policy evaluation:

  Estimate gradient and make gradient descent step:

$$\nabla_\phi L(\phi) \approx \frac{1}{N} \sum_{i=1}^{N} \left[\sum_{t=0}^{T} \nabla_\phi (r_{i,t} + \gamma \boxed{V_{\phi^-}}(s_{i,t+1}) - V_\phi(s_{i,t}))^2 \right]$$

Not target network, just frozen parameters



Source

Generalized Advantage Estimation (GAE)

Advantage function estimation:

$$A(s_t, a_t) = Q(s_t, a_t) - V(s_t) = r_t + \gamma V(s_{t+1}) - V(s) = -V(s) + r_t + \gamma V(s_{t+1})$$

Such advantage estimation might be biased

Also advantage function estimation:

$$A(s_t, a_t) = Q(s, a) - V(s) = -V(s) + r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots$$

Such advantage estimation might have high variance

GAE

$$
\begin{aligned}
\hat{A}_t^{(1)} &:= \delta_t^V &&= -V(s_t) + r_t + \gamma V(s_{t+1}) \\
\hat{A}_t^{(2)} &:= \delta_t^V + \gamma \delta_{t+1}^V &&= -V(s_t) + r_t + \gamma r_{t+1} + \gamma^2 V(s_{t+2}) \\
\hat{A}_t^{(3)} &:= \delta_t^V + \gamma \delta_{t+1}^V + \gamma^2 \delta_{t+2}^V &&= -V(s_t) + r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 V(s_{t+3})
\end{aligned}
$$

$$
\hat{A}_t^{(k)} := \sum_{l=0}^{k-1} \gamma^l \delta_{t+l}^V = -V(s_t) + r_t + \gamma r_{t+1} + \cdots + \gamma^{k-1} r_{t+k-1} + \gamma^k V(s_{t+k})
$$

$$
\hat{A}_t^{(\infty)} = \sum_{l=0}^{\infty} \gamma^l \delta_{t+l}^V = -V(s_t) + \sum_{l=0}^{\infty} \gamma^l r_{t+l}
$$

GAE

The generalized advantage estimator GAE(γ, λ) is defined as the exponentially-weighted average of these k-step estimators:

$$
\begin{aligned}
\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} &:= (1-\lambda)\left(\hat{A}_t^{(1)} + \lambda\hat{A}_t^{(2)} + \lambda^2\hat{A}_t^{(3)} + \ldots\right) \\
&= (1-\lambda)(\delta_t^V + \lambda(\delta_t^V + \gamma\delta_{t+1}^V) + \lambda^2(\delta_t^V + \gamma\delta_{t+1}^V + \gamma^2\delta_{t+2}^V) + \ldots) \\
&= (1-\lambda)(\delta_t^V(1+\lambda+\lambda^2+\ldots) + \gamma\delta_{t+1}^V(\lambda+\lambda^2+\lambda^3+\ldots) \\
&\qquad + \gamma^2\delta_{t+2}^V(\lambda^2+\lambda^3+\lambda^4+\ldots) + \ldots) \\
&= (1-\lambda)\left(\delta_t^V\left(\frac{1}{1-\lambda}\right) + \gamma\delta_{t+1}^V\left(\frac{\lambda}{1-\lambda}\right) + \gamma^2\delta_{t+2}^V\left(\frac{\lambda^2}{1-\lambda}\right) + \ldots\right) \\
&= \sum_{l=0}^{\infty}(\gamma\lambda)^l\delta_{t+l}^V
\end{aligned}
$$

Recap

Policy Gradients and Actor-Critic algorithms are on-policy algorithms so we can not use experience replay. Thus, our sample efficiency is quite low.

Policy Optimisation via Gradient Ascent

Several issues:

- We make gradient step in the space of parameters, get new parameters θ and policy π_θ from θ_{old} and old policy $\pi_{\theta_{old}}$. However, it's difficult to measure the impact of change in parameters on change in policy.
- Apply only first-order optimisation methods
- Low sample efficiency

$$\theta = \theta_{old} + \alpha \nabla J(\theta_{old})$$

Optimisation

$$J(\theta) \approx J(\theta_{old}) + \nabla J(\theta_{old})(\theta - \theta_{old})$$

$$J(\theta) \to \max_{\theta} \quad \Longleftrightarrow \quad \begin{array}{l} \nabla J(\theta_{old})(\theta - \theta_{old}) \to \max\limits_{\theta} \\ \text{s.t. } (\theta - \theta_{old})^T(\theta - \theta_{old}) \leq \delta \end{array}$$

Let’s $d = \theta - \theta_{old}$, then $d^* \propto \nabla J(\theta_{old})$

$$\theta = \theta_{old} + \alpha \nabla J(\theta_{old})$$

Optimisation

$$J(\theta_{old})(\theta - \theta_{old}) \rightarrow \max_{\theta} \text{ s.t.}$$

$$(\theta - \theta_{old})^T K(\theta - \theta_{old}) \leq \delta$$

K is symmetric, positive-definite matrix

Let's $d = \theta - \theta_{old}$, then $d^* \propto K^{-1} \nabla J(\theta_{old})$

$$\theta = \theta_{old} + \alpha K^{-1} \nabla J(\theta_{old})$$

Natural Gradient

$KL(\pi_{\theta_{old}} || \pi_\theta) \approx \frac{1}{2}(\theta - \theta_{old})^T K(\theta_{old})(\theta - \theta_{old})$, where $K(\theta_{old}) = \nabla_\theta^2 KL(\pi_{old} || \pi_\theta)|_{\theta_{old}}$

$\theta = \theta_{old} + \alpha s$, where $s = K^{-1} \nabla J(\theta_{old})$, α is a step size.

Natural Gradient

$KL(\pi_{\theta_{old}} || \pi_\theta) \approx \frac{1}{2}(\theta - \theta_{old})^T K(\theta_{old})(\theta - \theta_{old})$, where $K(\theta_{old}) = \nabla^2_\theta KL(\pi_{old} || \pi_\theta)|_{\theta_{old}}$

$\theta = \theta_{old} + \alpha s$, where $s = K^{-1} \nabla J(\theta_{old})$, α is a step size.

Choose the largest step:

$$\frac{1}{2}(\theta - \theta_{old})^T K(\theta_{old})(\theta - \theta_{old}) = \delta \iff \alpha = \sqrt{\frac{2\delta}{s^T K s}}$$

$$\theta = \theta_{old} + \alpha K^{-1} \nabla J(\theta_{old})$$

Natural Gradient

$KL(\pi_{\theta_{old}} || \pi_\theta) \approx \frac{1}{2}(\theta - \theta_{old})^T K(\theta_{old})(\theta - \theta_{old})$, where $K(\theta_{old}) = \nabla^2_\theta KL(\pi_{old} || \pi_\theta)|_{\theta_{old}}$

$\theta = \theta_{old} + \alpha s$, where $s = K^{-1} \nabla J(\theta_{old})$, α is a step size.

Choose the largest step:

$$\frac{1}{2}(\theta - \theta_{old})^T K(\theta_{old})(\theta - \theta_{old}) = \delta \iff \alpha = \sqrt{\frac{2\delta}{s^T K s}}$$

$$\theta = \theta_{old} + \alpha K^{-1} \nabla J(\theta_{old})$$

$K \in \mathbb{R}^{|\theta| \times |\theta|}$, K^{-1} computation takes $O(|\theta|^3)$

Conjugate Gradient Method

Paper

K is symmetric, positive-definite matrix

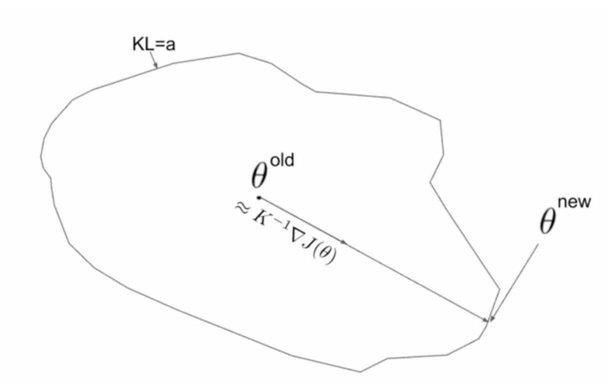
In order to find $K^{-1}\nabla J(\theta_{old})$ we can solve system $Ks = \nabla J(\theta)$ iteratively.

Conjugate Gradient Method

Paper

K is symmetric, positive-definite matrix

In order to find $K^{-1}\nabla J(\theta_{old})$ we can solve system $Ks = \nabla J(\theta)$ iteratively.



Source

Optimisation in Policy Space

Lemma:

$$J(\pi) = J(\pi_{old}) + \mathbb{E}_{\tau \sim \pi}[\sum_{t=0}^{T} \gamma^t A_{\pi_{old}}(S_t, A_t)], \text{ where } A_{\pi_{old}}(S_t, A_t) = Q_{\pi_{old}}(S_t, A_t) - V_{\pi_{old}}(S_t)$$

Let's rewrite it as a sum over states instead of timesteps:

$$J(\pi) = J(\pi_{old}) + \sum_{s} \rho_\pi(s) \sum_{a} \pi(a \mid s) A_{\pi_{old}}(s, a), \text{ where } \rho_\pi(s) = \mathbb{P}(s_0 = s) + \gamma \mathbb{P}(s_1 = s) + \ldots$$

Optimisation in Policy Space

Lemma:

$$J(\pi) = J(\pi_{old}) + \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{T}\gamma^t A_{\pi_{old}}(S_t, A_t)], \text{ where } A_{\pi_{old}}(S_t, A_t) = Q_{\pi_{old}}(S_t, A_t) - V_{\pi_{old}}(S_t)$$

Let's rewrite it as a sum over states instead of timesteps:

$$J(\pi) = J(\pi_{old}) + \sum_s \rho_\pi(s) \boxed{\sum_a \pi(a\,|\,s)A_{\pi_{old}}(s, a),} \text{ where } \rho_\pi(s) = \mathbb{P}(s_0 = s) + \gamma\mathbb{P}(s_1 = s) + \ldots$$

If this term is nonnegative than the policy improvement is guaranteed

Optimisation in Policy Space

Lemma:

$$J(\pi) = J(\pi_{old}) + \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{T}\gamma^t A_{\pi_{old}}(S_t, A_t)], \text{ where } A_{\pi_{old}}(S_t, A_t) = Q_{\pi_{old}}(S_t, A_t) - V_{\pi_{old}}(S_t)$$

Let's rewrite in sum over states instead of timesteps:

$$J(\pi) = J(\pi_{old}) + \sum_s \rho_\pi(s) \boxed{\sum_a \pi(a\,|\,s)A_{\pi_{old}}(s, a),} \text{ where } \rho_\pi(s) = \mathbb{P}(s_0 = s) + \gamma\mathbb{P}(s_1 = s) + \dots$$

If this term is nonnegative than the policy improvement is guaranteed

Since we don't know π this expression is intractable…

Optimisation in Policy Space

$$J(\pi) = J(\pi_{old}) + \sum_s \rho_\pi(s) \sum_a \pi(a \mid s) A_{\pi_{old}}(s, a)$$

$$J(\pi) \approx J(\pi_{old}) + \sum_s \rho_{\pi_{old}}(s) \sum_a \pi(a \mid s) A_{\pi_{old}}(s, a) = L_{\pi_{old}}(\pi)$$

Optimisation in Policy Space

$$J(\pi) = J(\pi_{old}) + \sum_s \rho_\pi(s) \sum_a \pi(a \mid s) A_{\pi_{old}}(s, a)$$

$$J(\pi) \approx J(\pi_{old}) + \sum_s \rho_{\pi_{old}}(s) \sum_a \pi(a \mid s) A_{\pi_{old}}(s, a) = L_{\pi_{old}}(\pi)$$

If π_θ is quite close to $\pi_{\theta_{old}}$ ($\mathbb{E}_{s \sim \rho_{old}}[KL(\pi_{\theta_{old}} \mid\mid \pi_\theta)] \leq \delta$), then

$$L_{\pi_{\theta_{old}}}(\pi_{\theta_{old}}) = J(\pi_{\theta_{old}})$$

$$\nabla_\theta L_{\pi_{\theta_{old}}}(\pi_\theta) \mid_{\theta_{old}} = \nabla_\theta J(\pi_\theta) \mid_{\theta_{old}}$$

Optimisation in Policy Space

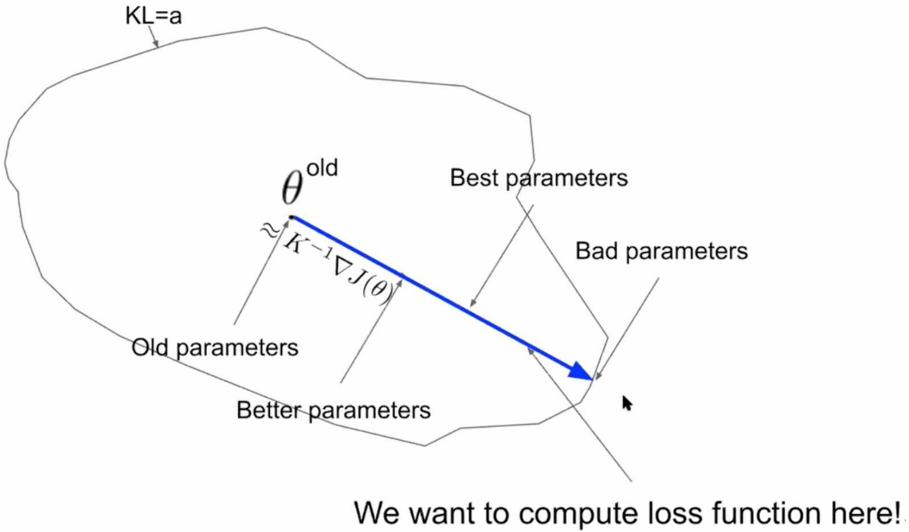
$$J(\pi) = J(\pi_{old}) + \sum_s \rho_\pi(s) \sum_a \pi(a \,|\, s) A_{\pi_{old}}(s, a)$$

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

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

$$= J(\pi_{old}) + \mathbb{E}_{\rho_{old}}\Big[\frac{\pi(a \,|\, s)}{\pi_{old}(a \,|\, s)} A_{\pi_{old}}(s, a)\Big]$$

Visualisation



Source

Trust Region Policy Optimisation (TRPO)

Original paper

We have to solve the following optimisation problem to generate a policy update:

$$\max_{\theta} \mathbb{E}_{s\sim\rho_{old},a\sim\pi_{\theta_{old}}}\left[\frac{\pi_\theta(a\,|\,s)}{\pi_{\theta_{old}}(a\,|\,s)}A_{\pi_{\theta_{old}}}(s,a)\right]$$

$$\text{s.t. } \mathbb{E}_{s\sim\rho_{old}}[KL(\pi_{\theta_{old}}\,||\,\pi_\theta)] \leq \delta$$

The authors change the advantage function by the Q-function.

TRPO Algorithm

Repeat until convergence:

1. Collect transitions following current policy $\pi_{\theta_{old}}$
2. Compute $g = \nabla_\theta \frac{1}{N} \sum_{i=1}^{N} \frac{\pi_\theta(a_i \mid s_i)}{\pi_{\theta_{old}}(a_i \mid s_i)} Q_{\pi_{\theta_{old}}}(s_i, a_i)$
3. Compute $K = \nabla^2_\theta \frac{1}{N} \sum_{i=1}^{N} KL(\pi_{\theta_{old}}(\,.\mid s_i) \,||\, \pi_\theta(\,.\mid s_i))$
4. Find optimal direction via Conjugate Gradients Method (find $s = K^{-1} g$)
5. Do linear search in optimal direction checking the KL constraint and objective value for each new parameter: $\theta_j = \theta_{old} + \alpha_j \sqrt{\frac{2\delta}{g^T s}} s$

TRPO

+ Extremely stable
+ Prominent results
- Computational expensive
- Require cheap sampling
- Difficult to implement

Conditional vs Unconditional Problem

TRPO problem

$$\max_{\theta} \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a \mid s)}{\pi_{\theta_{old}}(a \mid s)}\hat{A}_t\right]$$

s.t. $\hat{\mathbb{E}}_t[KL(\pi_{\theta_{old}} \mid\mid \pi_\theta)] \leq \delta$

Equivalent problem

$$\max_{\theta} \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a \mid s)}{\pi_{\theta_{old}}(a \mid s)}\hat{A}_t - \beta KL(\pi_{\theta_{old}} \mid\mid \pi_\theta)\right]$$

$\hat{A}$ is an estimator of the advantage function at timestep t. Here, the expectation $\hat{\mathbb{E}}_t$ indicates the empirical average over a finite batch of samples, in an algorithm that alternates between sampling and optimization.

PPO Objective

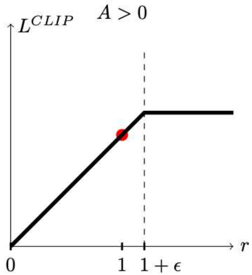
$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{old}}(a_t \mid s_t)}, \text{ so } r_t(\theta_{old}) = 1$$

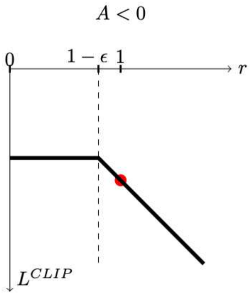
$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t\Big[\frac{\pi_\theta(a \mid s)}{\pi_{\theta_{old}}(a \mid s)}\hat{A}_t\Big] = \hat{\mathbb{E}}_t\Big[r_t(\theta)\hat{A}_t\Big]$$

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

PPO Objective

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



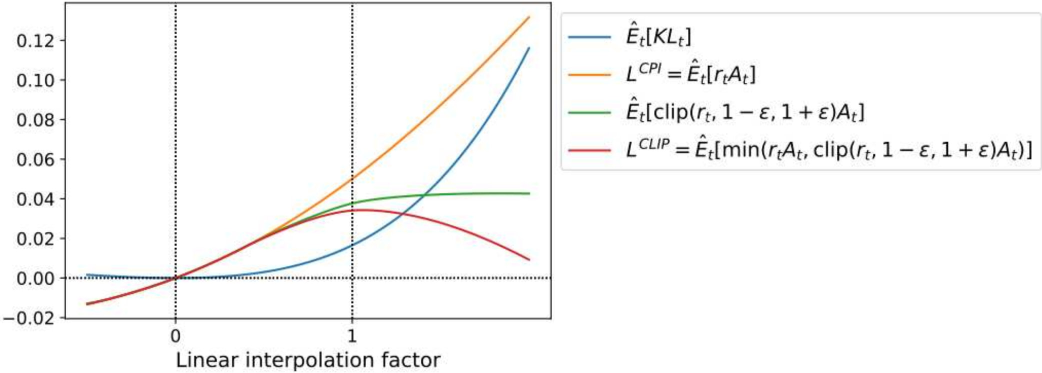


Source

$p_t(\theta) > 0$	A_t	**Return Value of** ***min***	**Objective is Clipped**	**Sign of Objective**	**Gradient**
$p_t(\theta) \in [1-\epsilon, 1+\epsilon]$	$+$	$p_t(\theta)A_t$	no	$+$	✓
$p_t(\theta) \in [1-\epsilon, 1+\epsilon]$	$-$	$p_t(\theta)A_t$	no	$-$	✓
$p_t(\theta) < 1-\epsilon$	$+$	$p_t(\theta)A_t$	no	$+$	✓
$p_t(\theta) < 1-\epsilon$	$-$	$(1-\epsilon)A_t$	yes	$-$	**0**
$p_t(\theta) > 1+\epsilon$	$+$	$(1+\epsilon)A_t$	yes	$+$	**0**
$p_t(\theta) > 1+\epsilon$	$-$	$p_t(\theta)A_t$	no	$-$	✓

Source

Surrogate Objectives



Source

Optimisation

Like in A2C, we obtain the following object, which is approximately maximised each iteration:

$$L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 H_{\pi_\theta}(s_t),$$

where c_1, c_2 are coefficients, H denotes an entropy bonus,

$L_t^{VF}(\theta)$ is a squared-error loss $\hat{\mathbb{E}}_t[V_\theta(s_t) - V_t^{target}]$

TRPO vs PPO

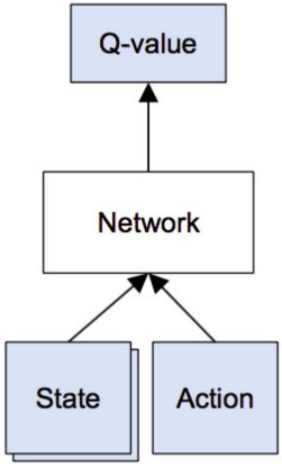
TRPO	PPO
- Works for smaller models	+ Works for big models
+ Second-order optimisation	- First-order optimisation

Continuous action spaces

- We can learn critic easily

- The problem is finding

$$a_{opt} = \arg\max_a Q(s, a)$$



DDPG

Idea: learn a separate network to find a_opt

- Train critic Q(s,a)

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

- Train actor a_opt(s) = mu(s)

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

DDPG

Idea: learn a separate network to find a_opt

- Train critic Q(s,a)

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

How to get V(s')?

- Train actor a_opt(s) = mu(s)

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

DDPG

Idea: learn a separate network to find a_opt

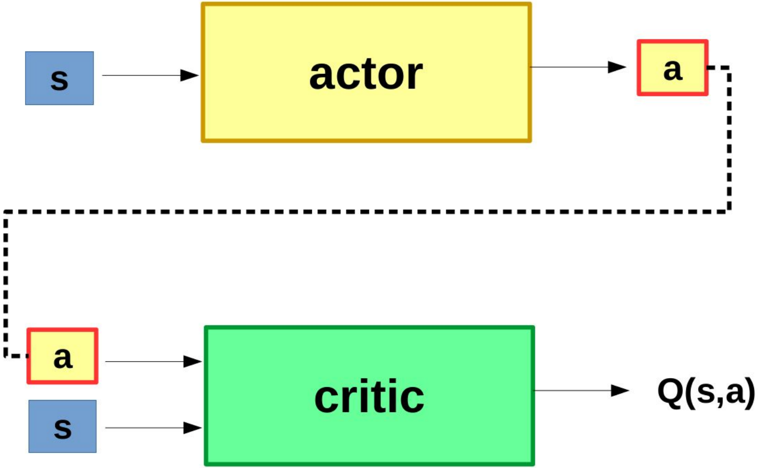
- Train critic Q(s,a)

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

- Train actor a_opt(s) = mu(s)

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

DDPG



DDPG

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