Reinforcement learning

Episode 9 ¾, 2018

More policy gradients

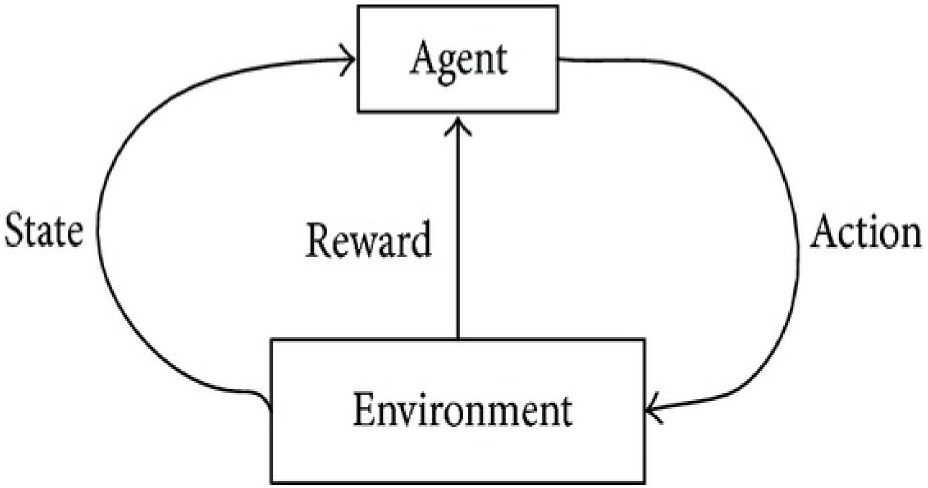
Yandex Data Factory

LAMBDA

British Hedgehog Preservation Society

Continuous action spaces

- Regular MDP
- $a \in \mathbb{R}^n$



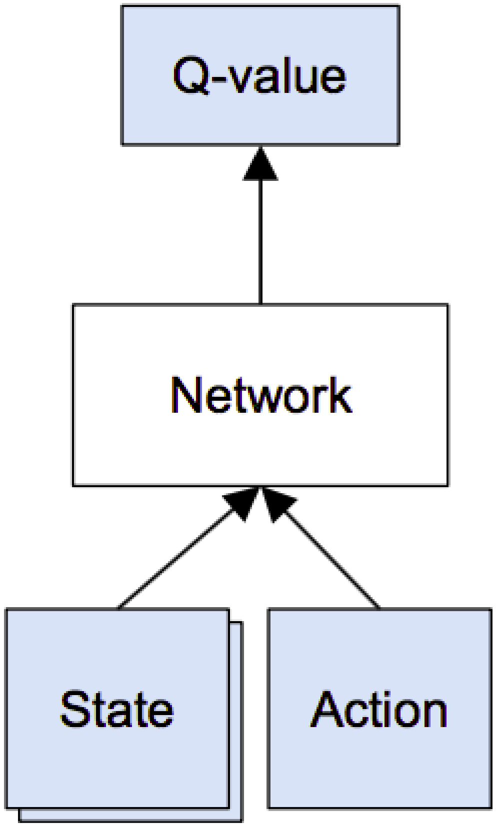
Which methods can we use?

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

Idea 1: 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?

Normalized advantage functions

Idea 1: 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

Idea 1: 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

Idea 1: restrict Q(s,a) so that optimization becomes trivial

For example, parabola (for 1d action space)

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

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

Where L(s) is a lower-triangular matrix

Normalized advantage functions

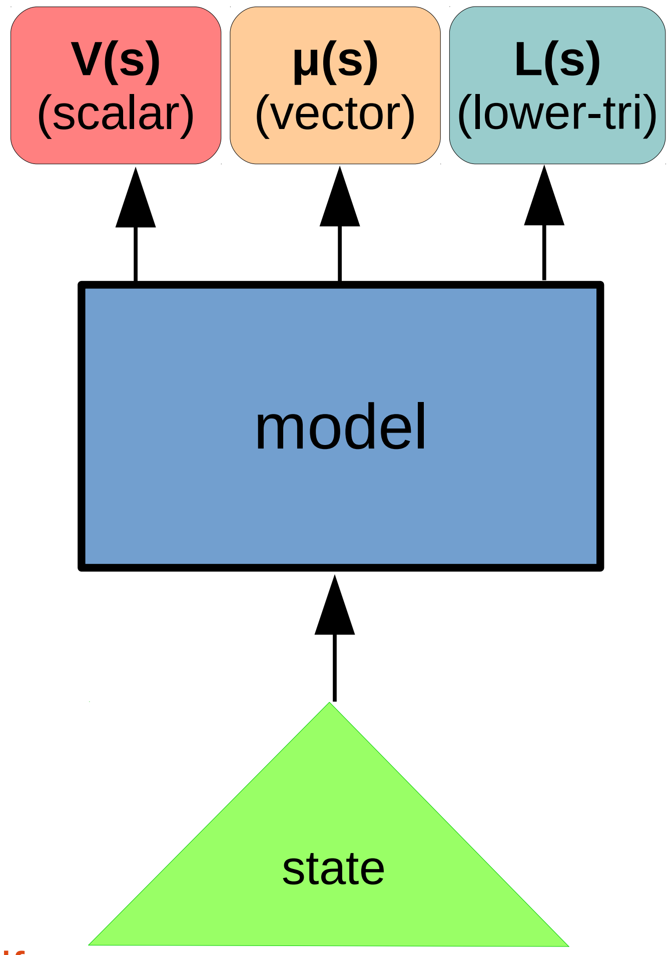
Network:

(trains end-to-end)

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

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

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



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

Deterministic policy gradient

- Idea2: 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 $a_{opt}(s) \approx \mu_\theta(s)$

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

Deterministic policy gradient

- Idea2: learn a separate network to find a_opt

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

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

How do we get V(s')?

- 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

- Idea2: learn a separate network to find a_opt

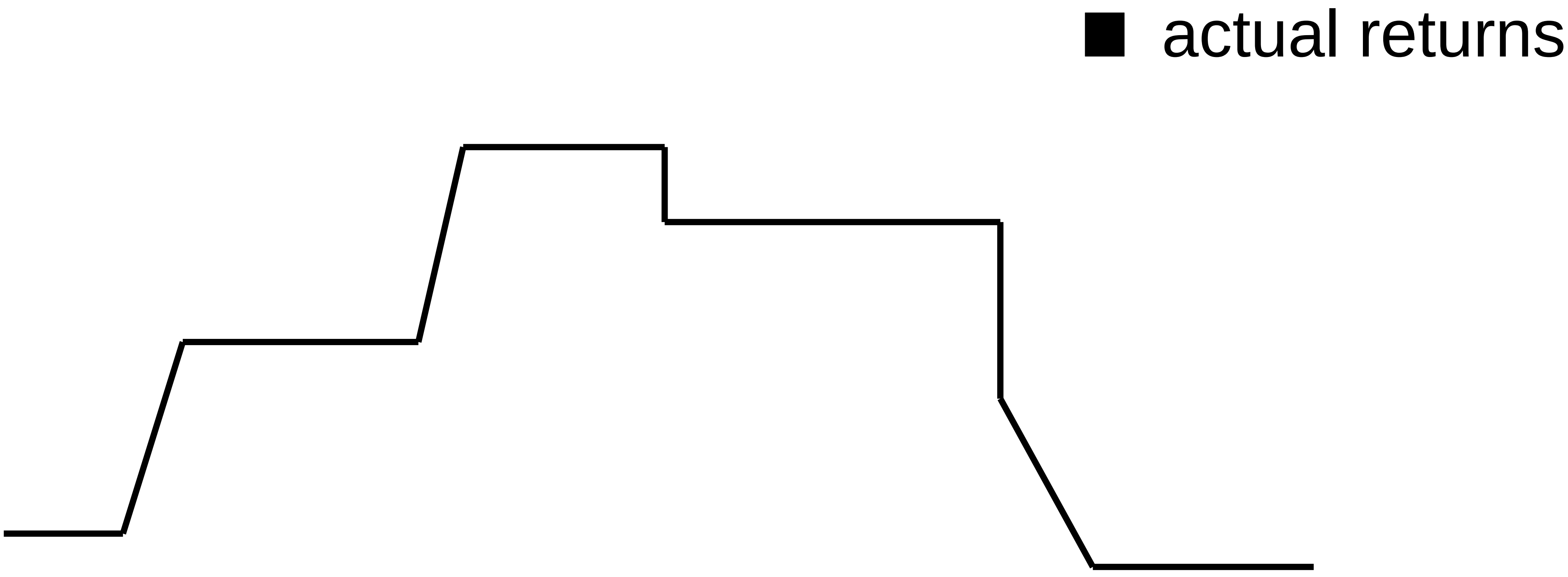
- Train critic $Q_\theta(s, a)$

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

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

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

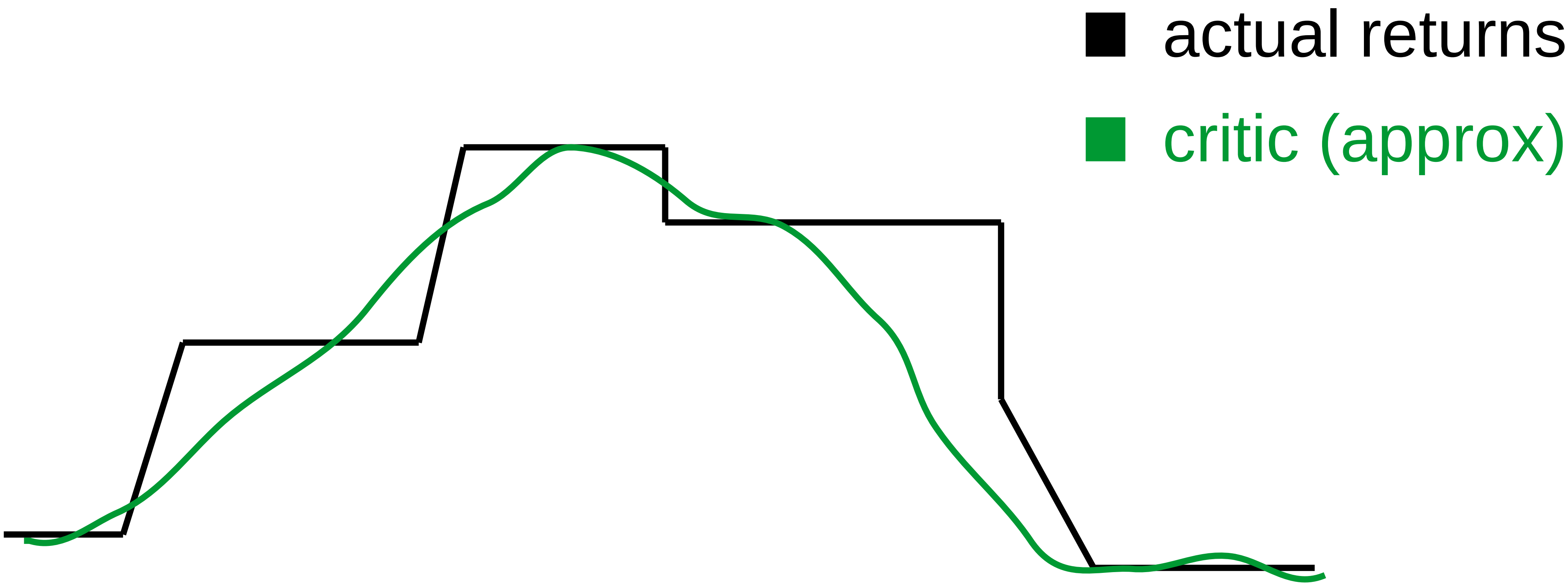
Deterministic policy gradient



Deterministic policy gradient

- Gradient approximation:

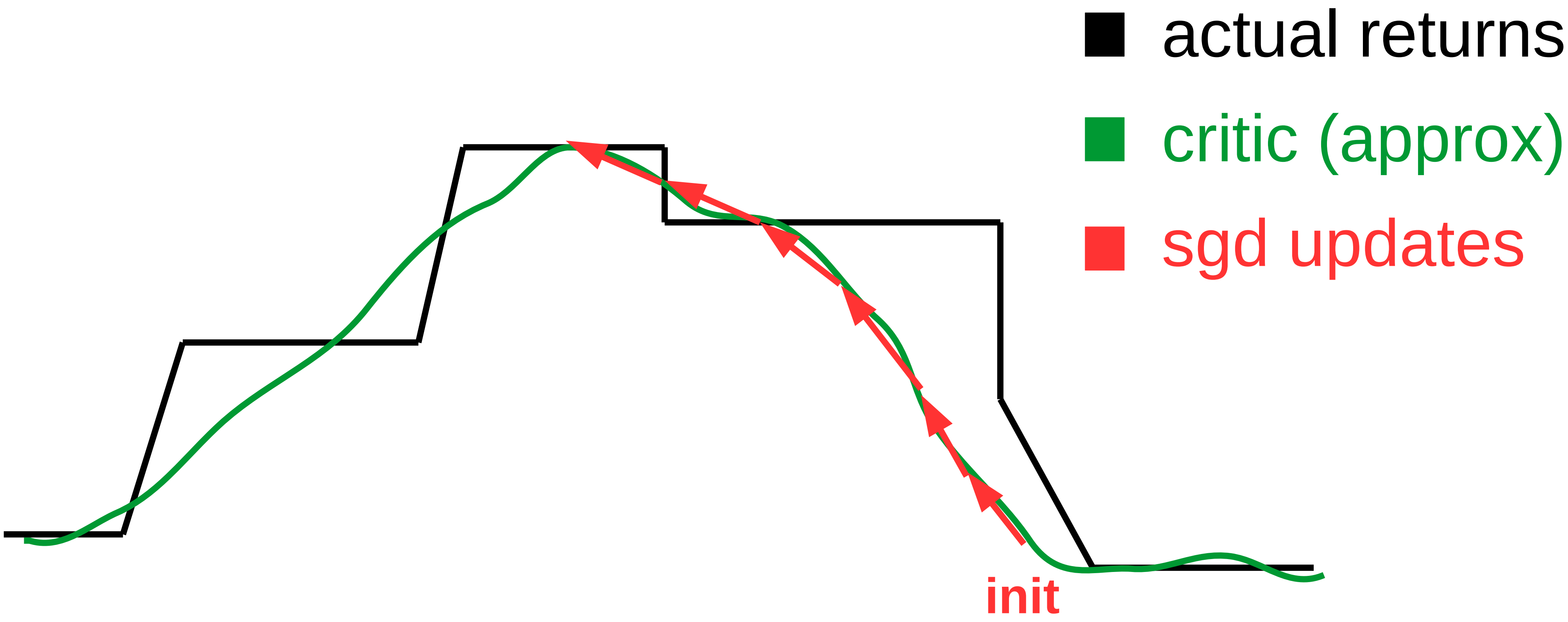
$$\nabla_\theta J = \frac{\partial Q^\theta(s,a)}{\partial a}\frac{\partial \mu(s|\theta)}{\partial \theta}$$



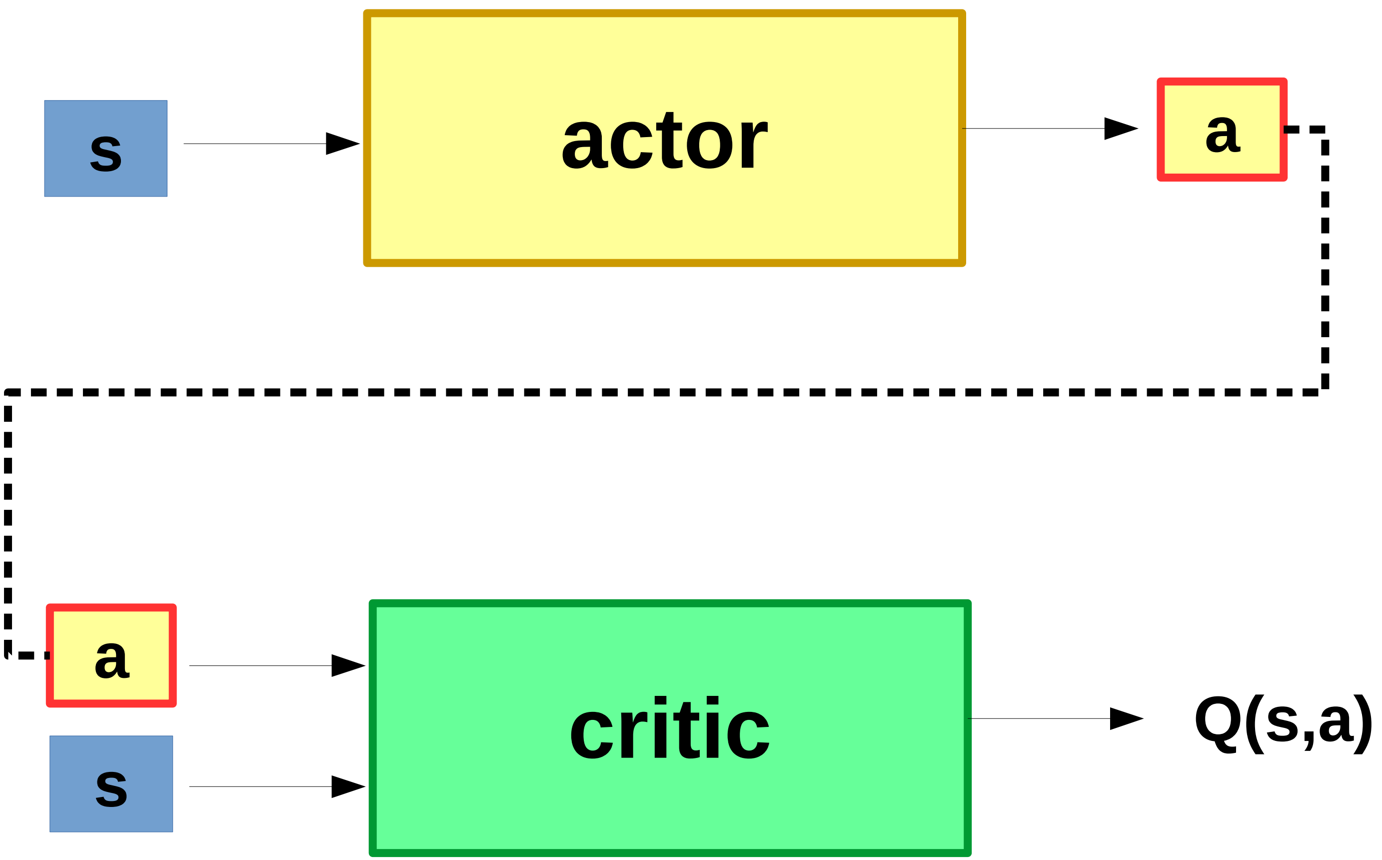
Deterministic policy gradient

- 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