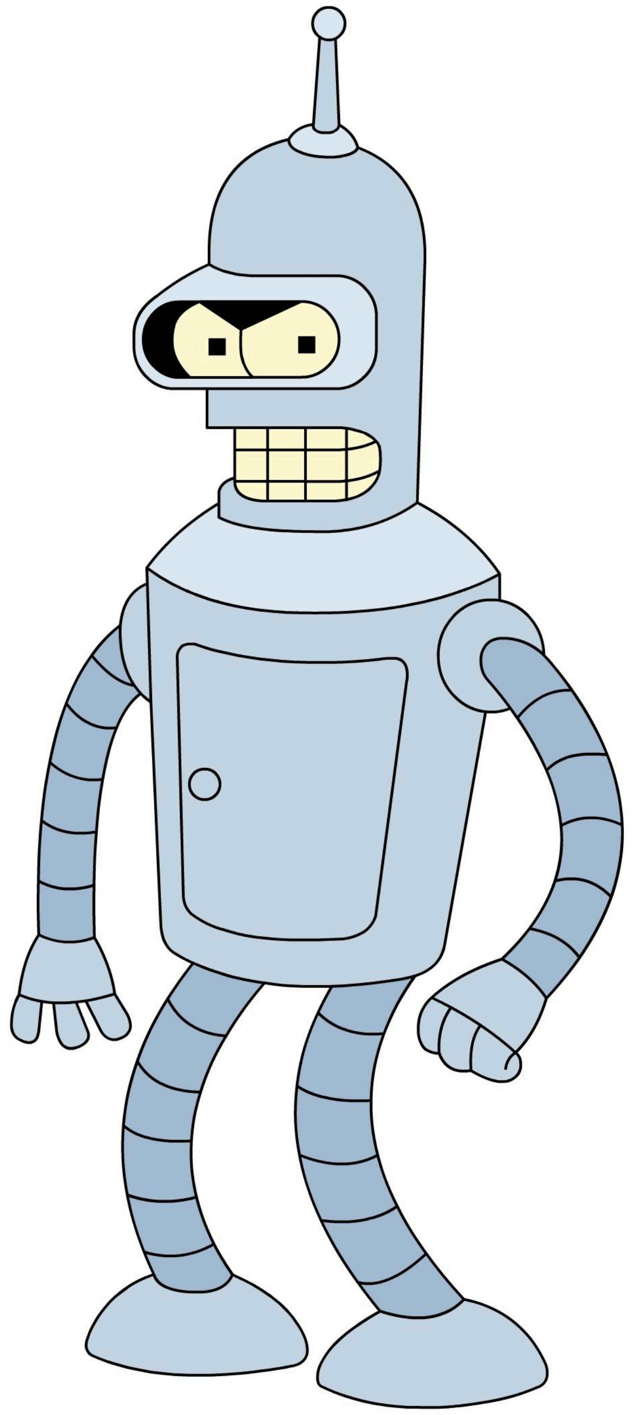
Reinforcement Learning

HSE, autumn - winter 2022

Lecture 6: Advanced RL Algorithms



Sergei Laktionov
slaktionov@hse.ru
LinkedIn

Background

1. Practical RL course by YSDA, week 8 + 9

Part 1: POMDP

Recap: MDP

MDP is a 4-tuple $(\mathcal{S}, \mathcal{A}, p, r)$:

1. $\mathcal{A}$ is an action space

2. $\mathcal{S}$ is a state space

3. $p(s' \mid s, a) = \mathbb{P}(S_{t+1} = s' \mid S_t = s, A_t = a)$ is a state-transition function

4. $r : \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is a reward function giving an expected reward:
   $r(s, a) = \mathbb{E}[R \mid s, a]$

Markov property:

(the future is independent of the past given the present)

$$p(R_t, S_{t+1} \mid S_t, A_t, R_{t-1}, S_{t-1}, A_{t-1}, \ldots) = p(R_t, S_{t+1} \mid S_t, A_t)$$

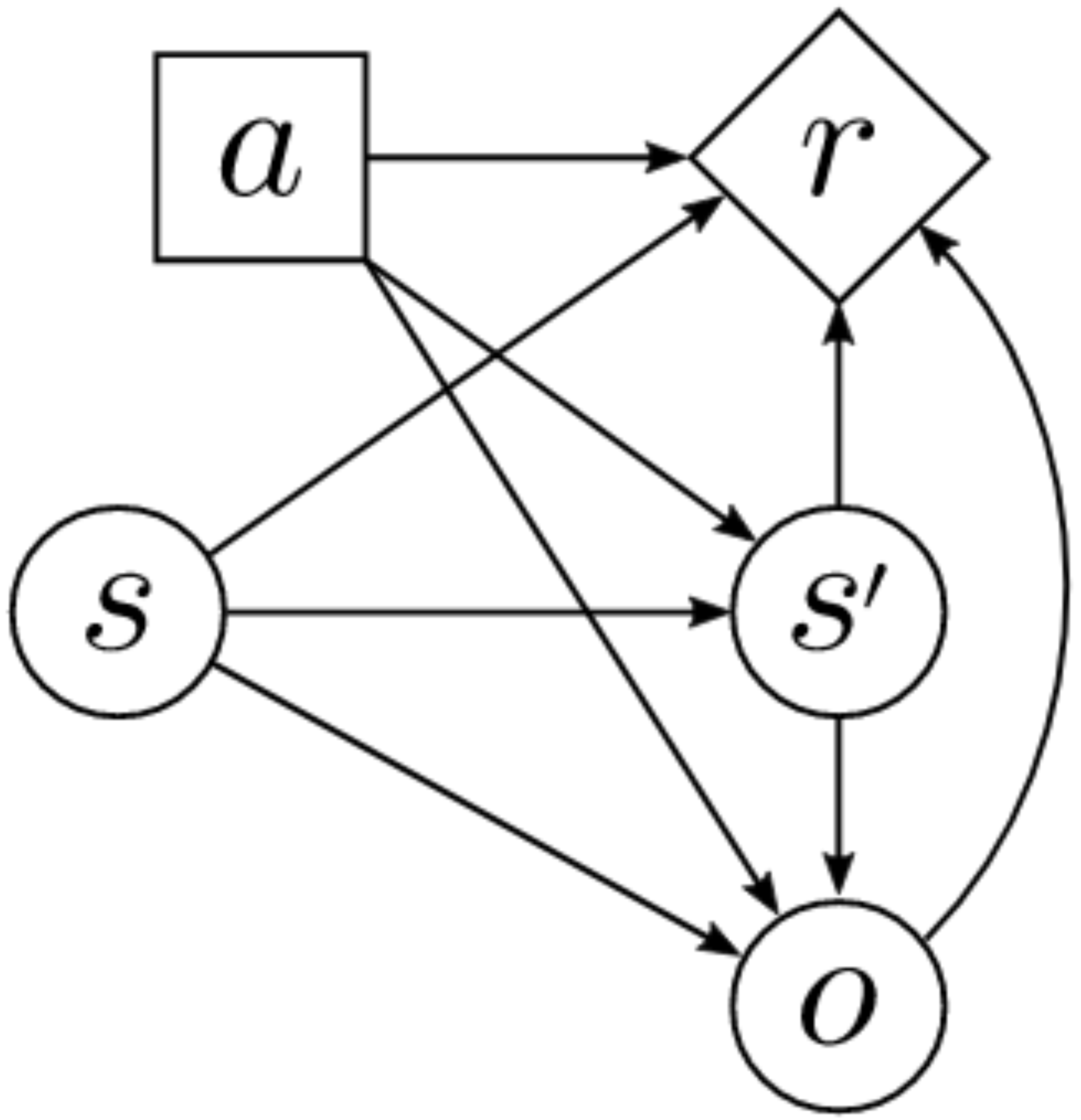
POMDP

1. $(\mathcal{S}, \mathcal{A}, r)$ are the same as in MDP

2. $\mathcal{O}$ is a set of possible observations

3. $p(s' \mid s, a) = \mathbb{P}(S_{t+1} = s' \mid S_t = s, A_t = a)$
   is a state-transition function

4. $p(o \mid s', a) = \mathbb{P}(O_t = o \mid S_{t+1} = s', A_t = a)$
   is a state-transition function

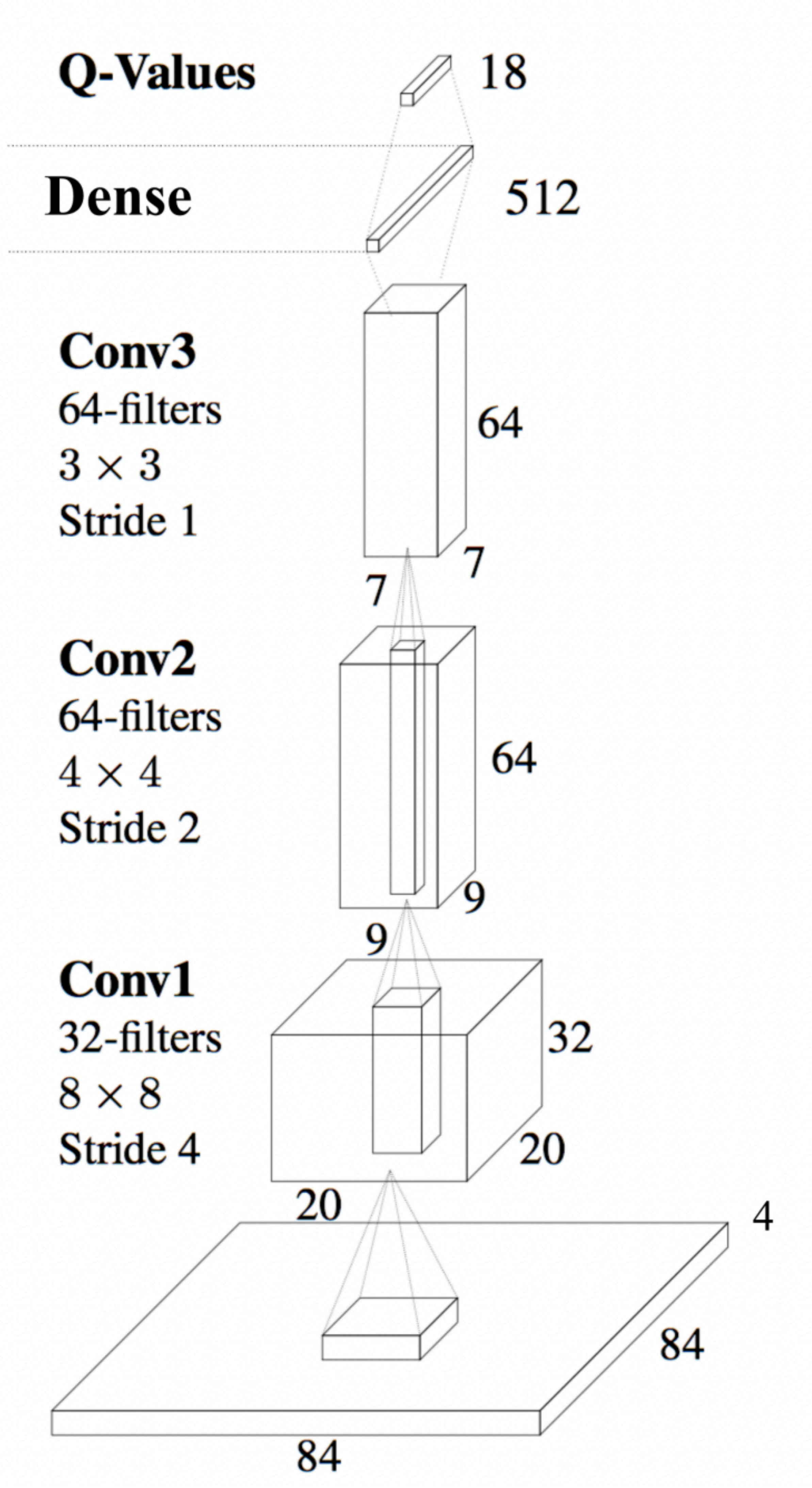


DRQN

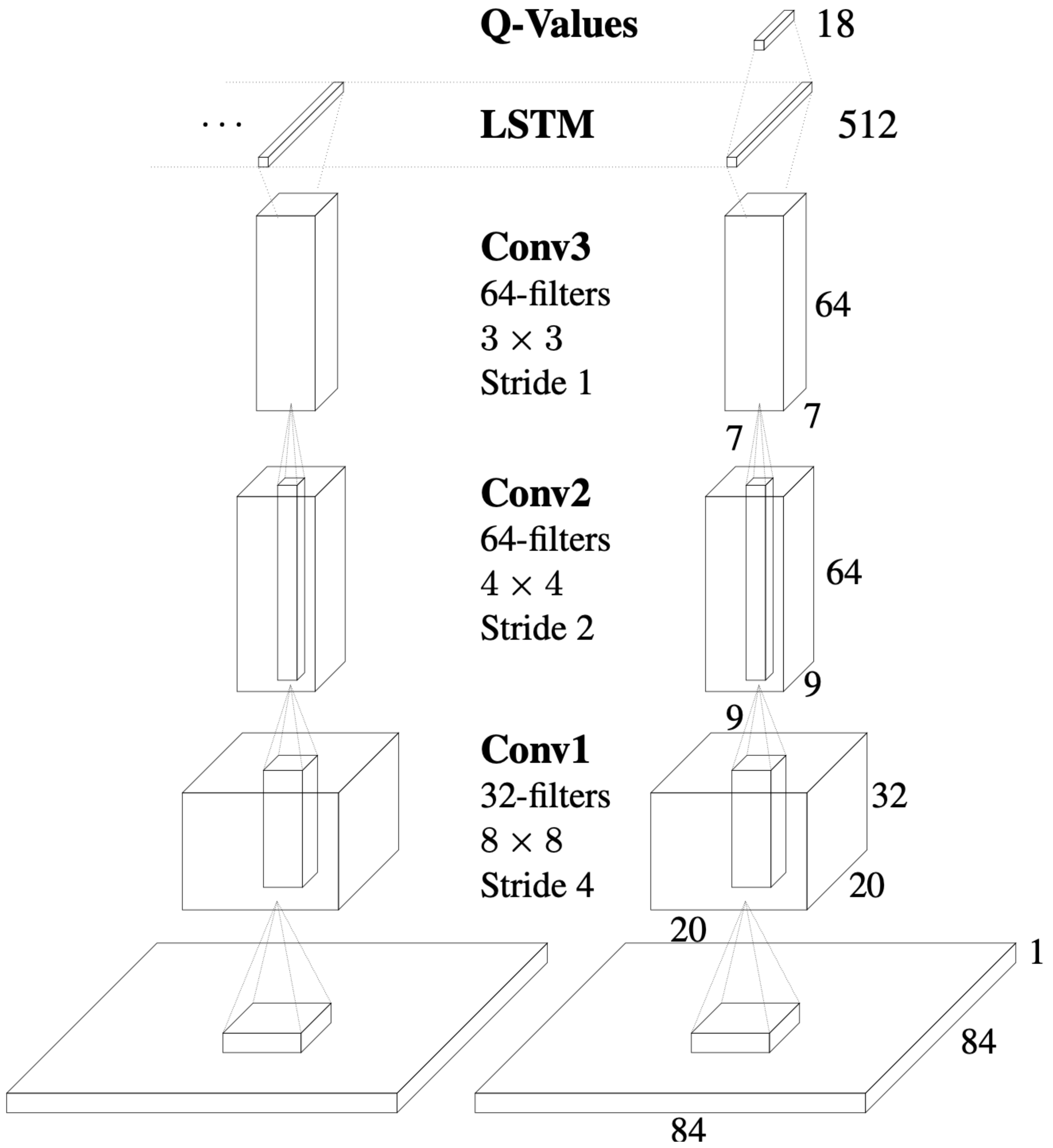
Vanilla Deep Q-Learning has no explicit mechanisms for deciphering the underlying state of the POMDP and is only effective if the observations are reflective of underlying system states. In the general case, estimating a Q-value from an observation can be arbitrarily bad since $Q(o, a; \theta) \neq Q(s, a; \theta)$.

- Let's equip agent with memory h_t
- $Q(s_t, a_t) \approx Q(o_t, h_{t-1}, a_t)$
- $h_t = LSTM(o_t, h_{t-1})$

DQN vs DRQN



Original paper

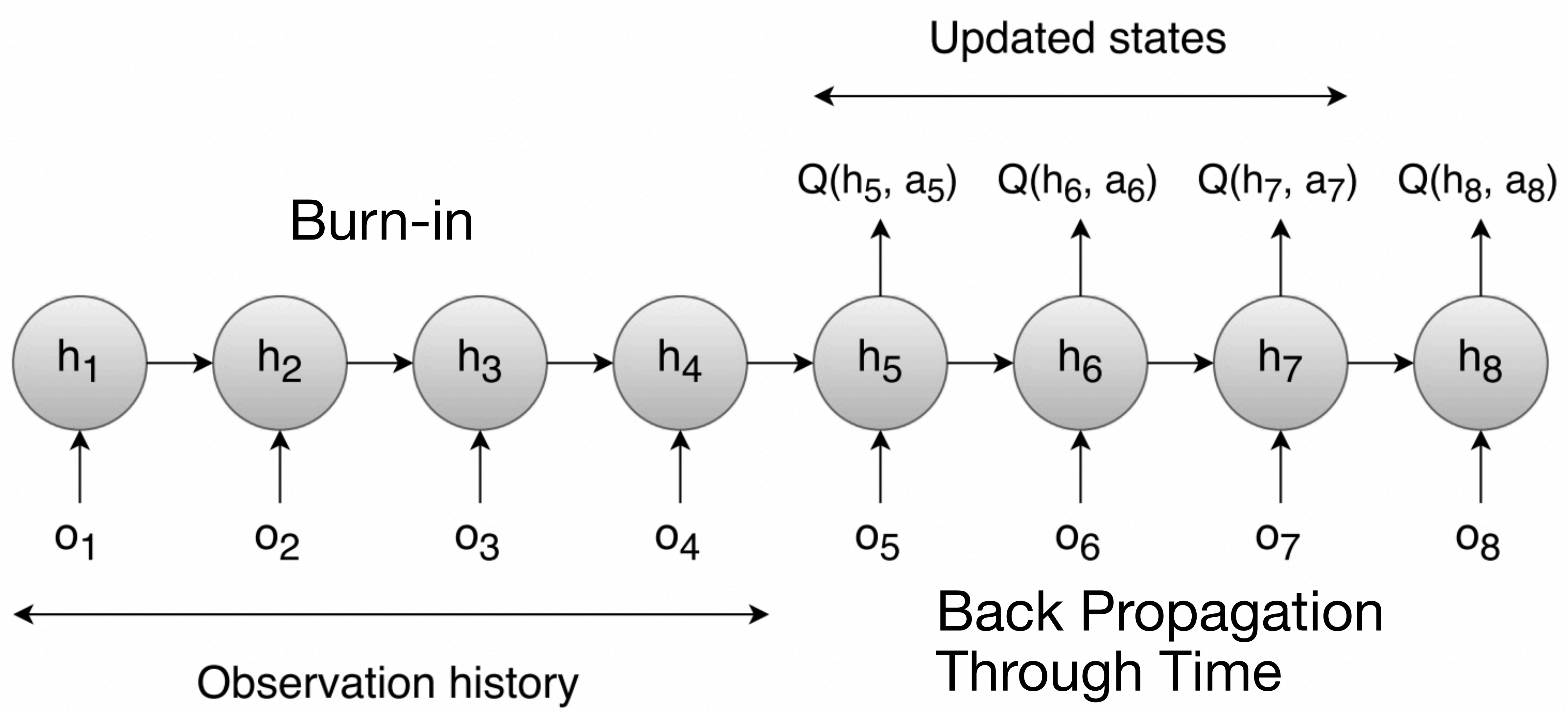


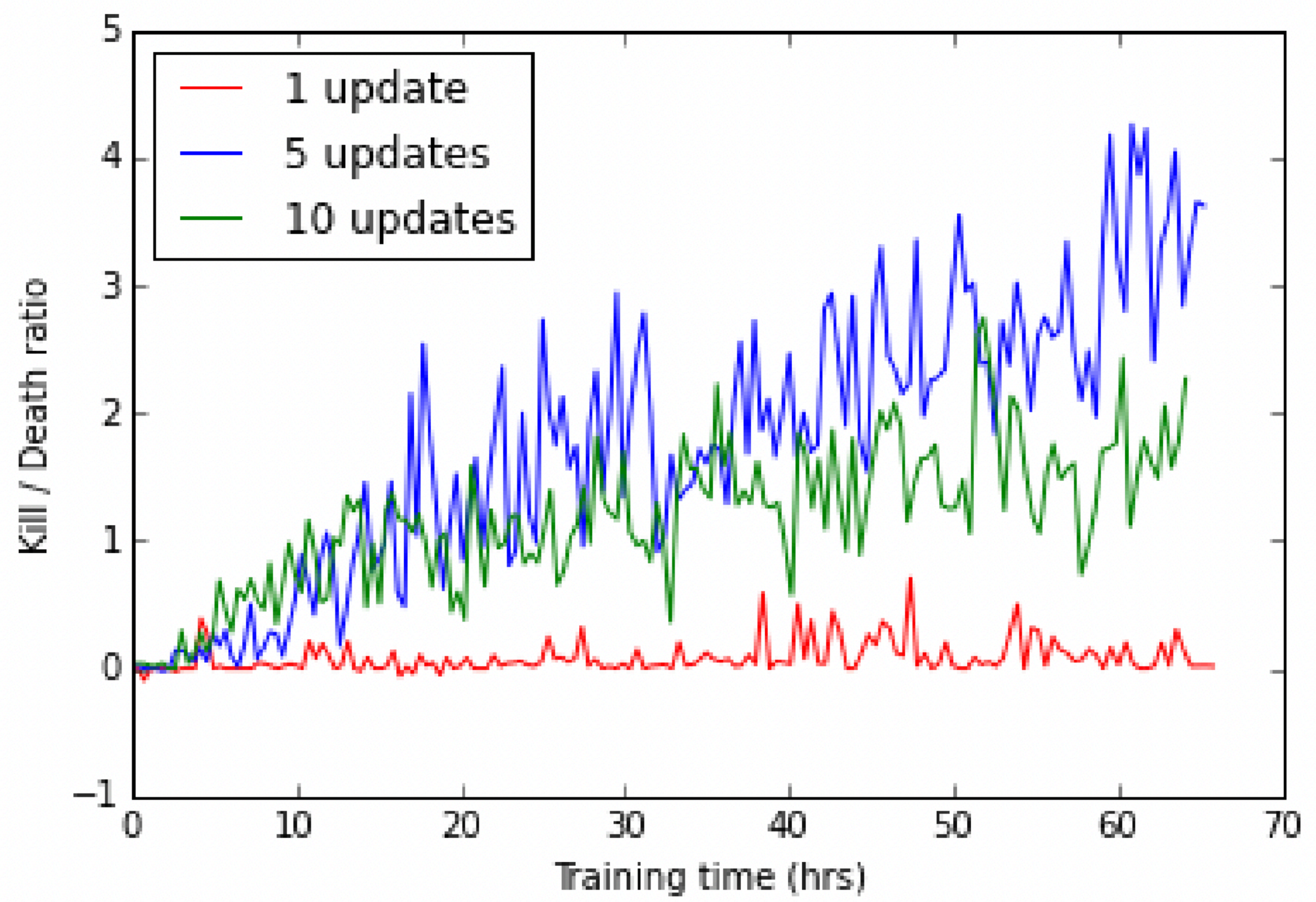
Original paper

DRQN Experience Replay

Original paper

- Sample random time step
- Consider N consecutive transitions
- Update only last M





Part 2: Advanced Policy Optimisation

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:

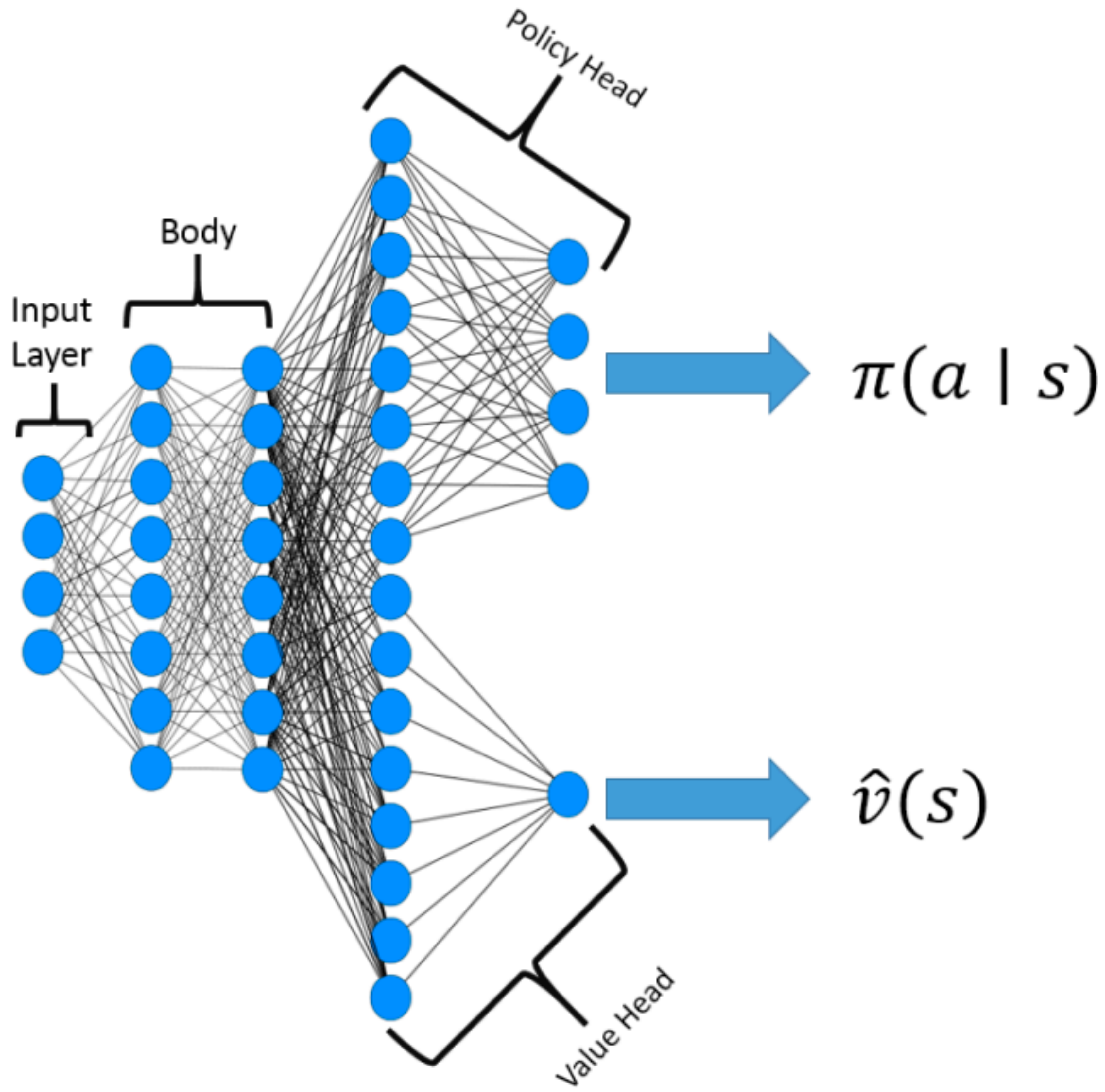
$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \Big[\sum_{t=0}^{T} \nabla \log \pi_\theta(a_{i,t} \mid s_{i,t}) A_{\pi_\theta}(s_{i,t}, a_{i,t})\Big]$$

- Policy evaluation:

  Estimate gradient and make gradient descent step:

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

Not target network, just fixate parameters from the previous step



Source

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

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

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

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

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

Optimisation

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

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

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

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

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

Natural Gradient

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

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

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

Conjugate Gradient Method

Paper

K is symmetric, positive-definite matrix

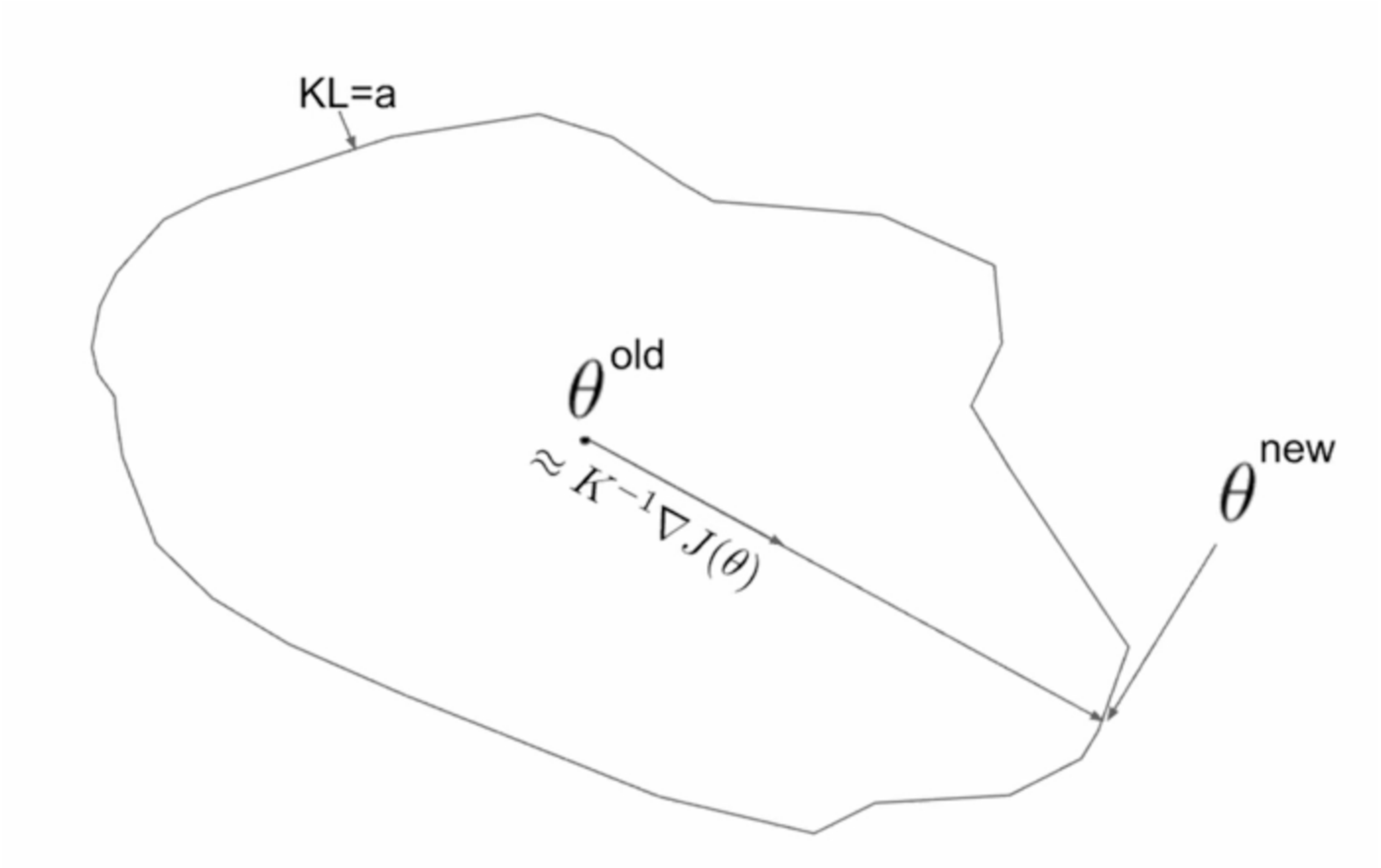
In order to find $K^{-1}\nabla J(\theta_{old})$ we can solve system $Kx = \nabla J(\theta)$ iteratively.

Conjugate Gradient Method

Paper

K is symmetric, positive-definite matrix

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



Source

Optimisation in Policy Space

<u>Lemma</u>:

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

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

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

Optimisation in Policy Space

<u>Lemma</u>:

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

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

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

If this term is nonnegative than the policy improvement is guaranteed

Optimisation in Policy Space

<u>Lemma</u>:

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

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

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

If this term is nonnegative than the policy improvement is guaranteed

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

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

$$\nabla_\theta L_{\pi_{\theta_{old}}}(\pi_\theta) \,|_{\theta_{old}} = \nabla_\theta J(\pi_\theta) \,|_{\theta_{old}}$$

Optimisation in Policy Space

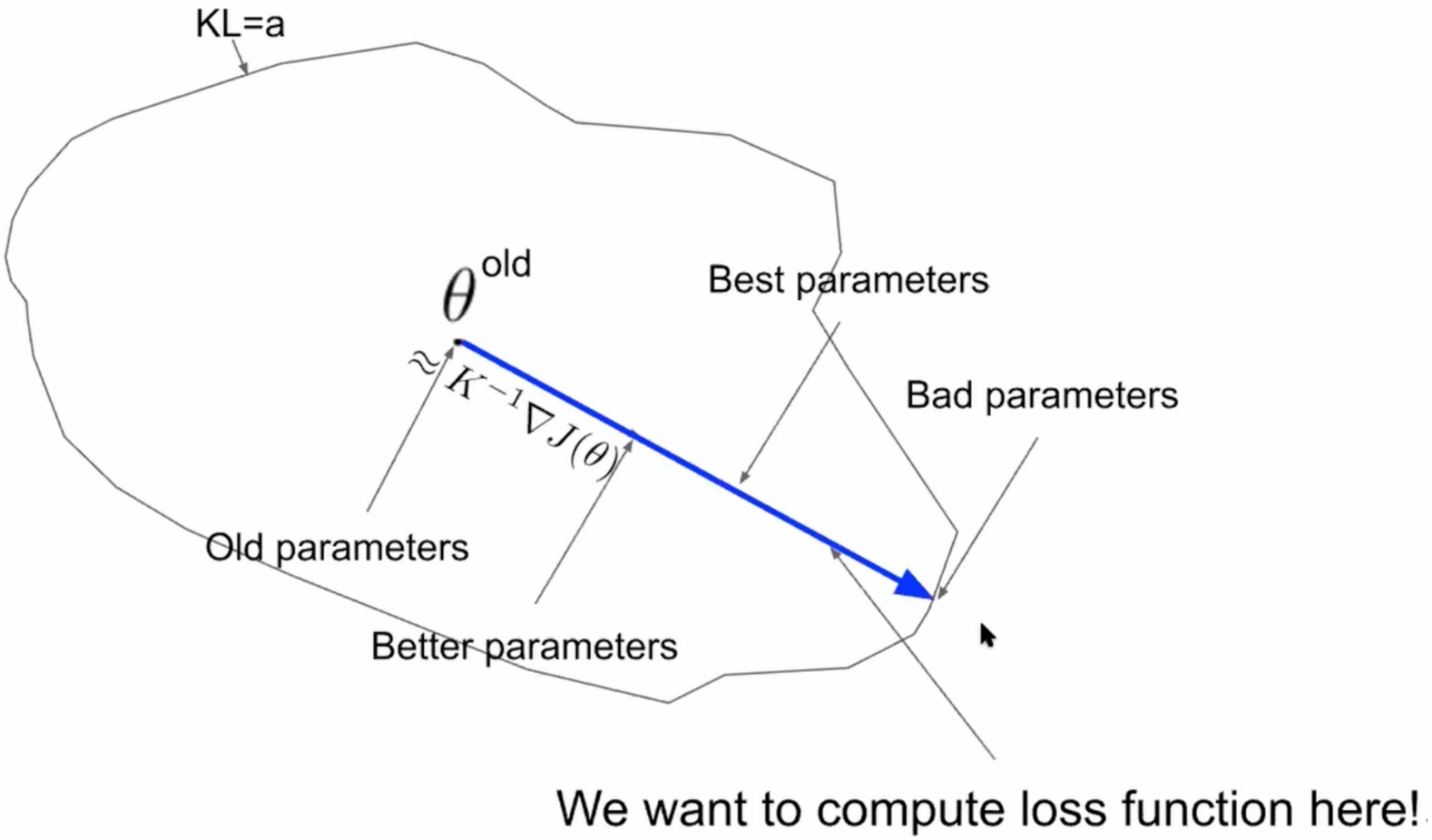
$$J(\pi) = J(\pi_{old}) + \sum_s \rho_\pi(s) \sum_a \pi(a \mid s) A_{\pi_{old}}(s, a)$$

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

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

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

Visualisation



Source

Trust Region Policy Optimisation (TRPO)

Original paper

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

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

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

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

TRPO Algorithm

Repeat until convergence:

1. Collect transitions following current policy $\pi_{\theta_{old}}$
2. Compute $g = \nabla_\theta \frac{1}{N} \sum_{i=1}^{N} \frac{\pi(a_i \mid s_i)}{\pi_{old}(a_i \mid s_i)} Q_{\pi_{old}}(s_i, a_i)$
3. Compute $K = \nabla_\theta^2 \frac{1}{N} \sum_{i=1}^{N} KL(\pi_{\theta_{old}}(. \mid s_i) \mid\mid \pi_\theta(. \mid s_i))$
4. Find optimal direction via Conjugate Gradients Method
5. Do linear search in optimal direction checking the KL constraint and objective value

TRPO

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

Conditional vs Unconditional Problem

TRPO problem

$$\max_{\theta} \hat{\mathbb{E}}_t\Big[\frac{\pi(a\,|\,s)}{\pi_{old}(a\,|\,s)}\hat{A}_t\Big]$$

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

Equivalent problem

$$\max_{\theta} \hat{\mathbb{E}}_t\Big[\frac{\pi(a\,|\,s)}{\pi_{old}(a\,|\,s)}\hat{A}_t - \beta KL(\pi_{\theta_{old}}\,||\,\pi_\theta)\Big]$$

$\hat{A}$ 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

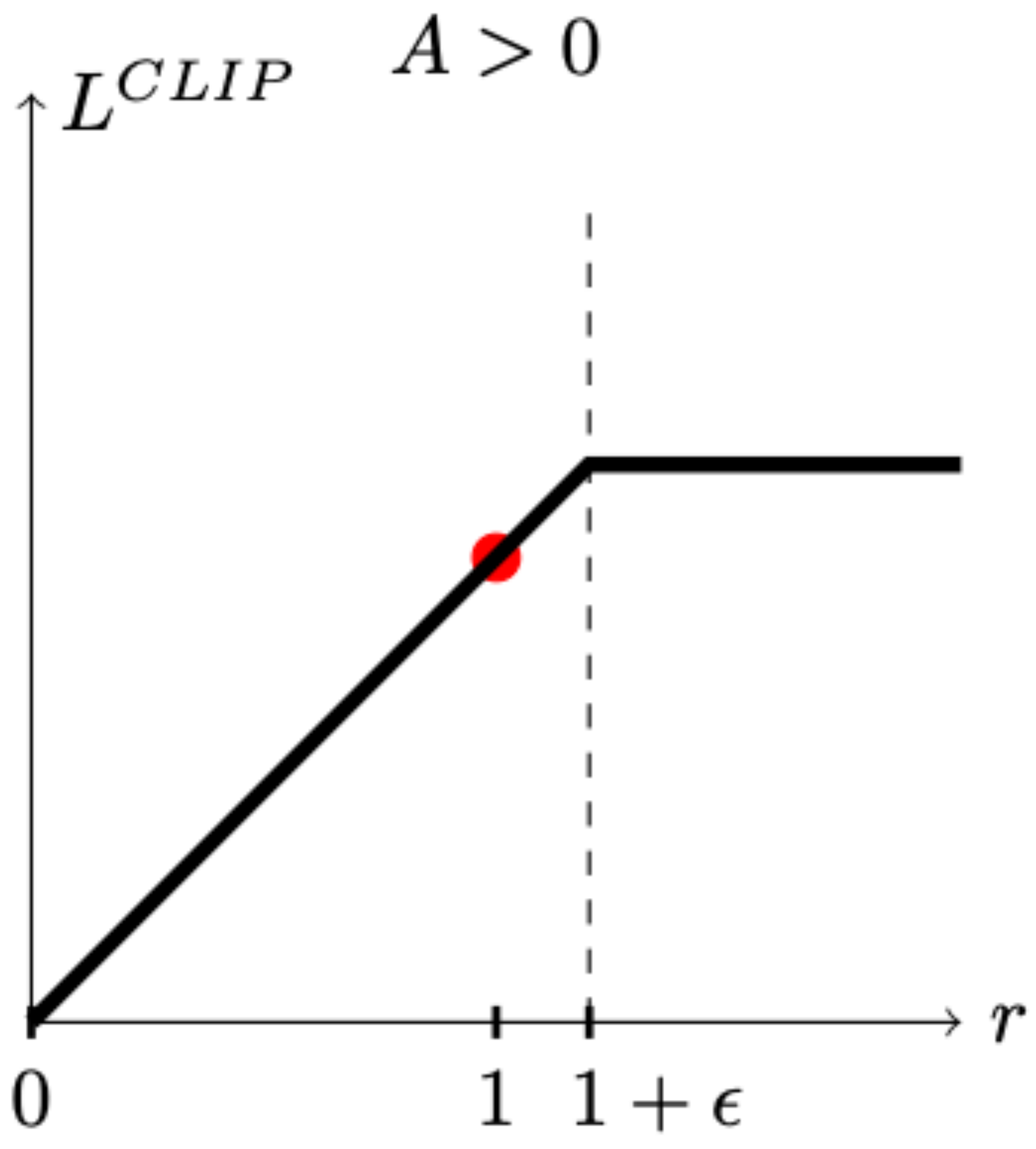
$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{old}}(a_t \mid s_t)}, \text{ so } r_t(\theta_{old}) = 1$$

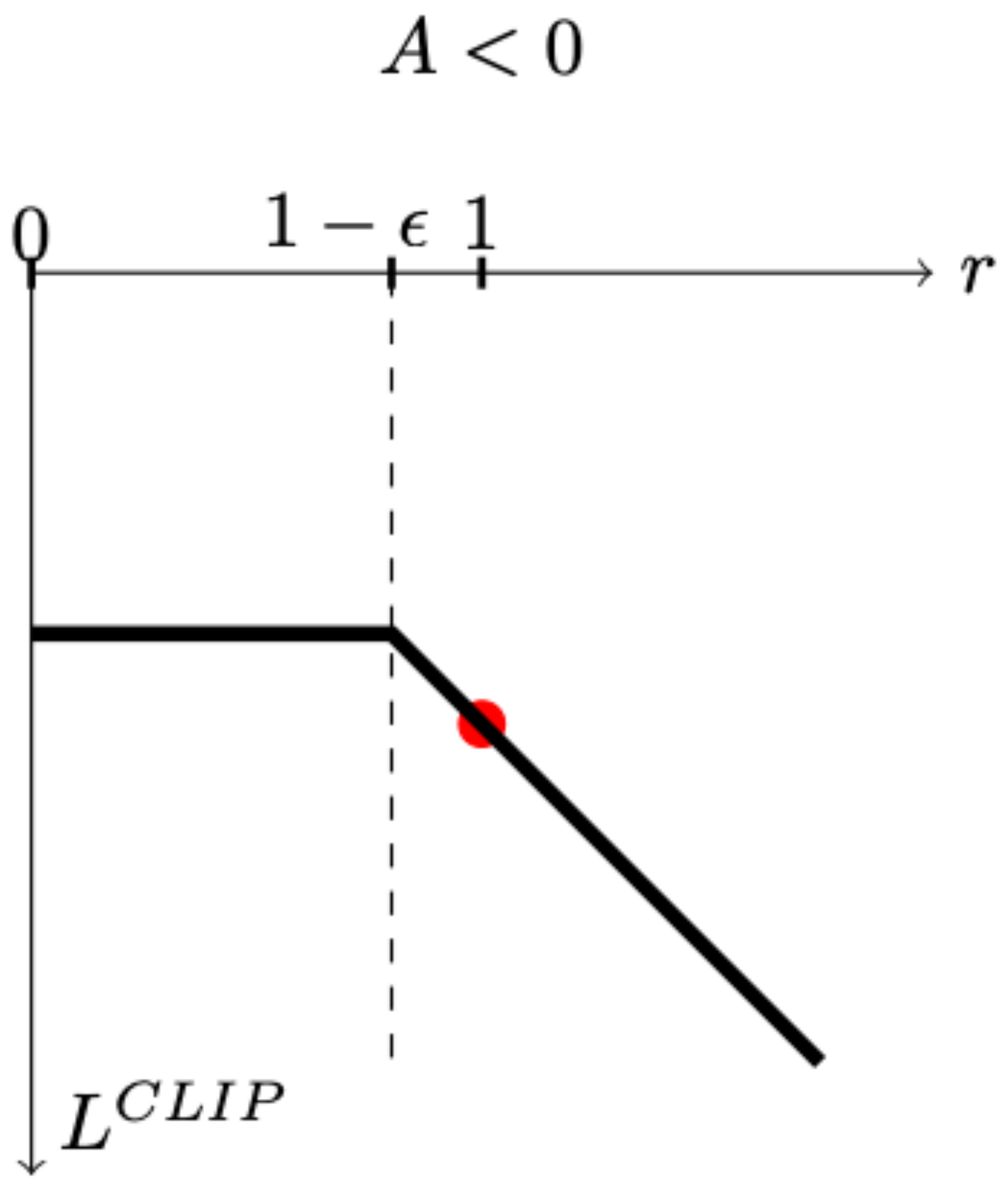
$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t\Big[\frac{\pi(a \mid s)}{\pi_{old}(a \mid s)}\hat{A}_t\Big] = \hat{\mathbb{E}}_t\big[r_t(\theta)\hat{A}_t\big]$$

PPO Objective

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

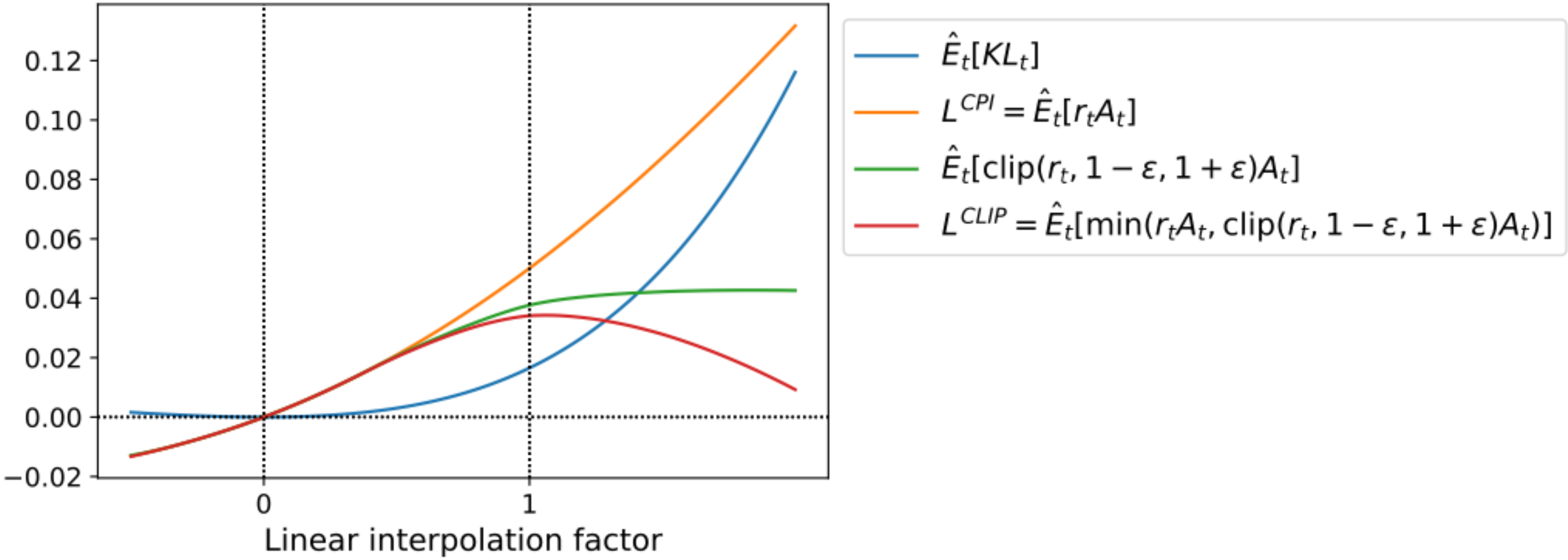
$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right]$$





Source

Surrogate Objectives



Source

TRPO vs PPO

- Works for smaller models

\+ Second-order optimisation

\+ Works for big models

- First-order optimisation

Bonus

Published as a conference paper at ICLR 2020

IMPLEMENTATION MATTERS IN DEEP POLICY GRADIENTS: A CASE STUDY ON PPO AND TRPO

Logan Engstrom[1*], Andrew Ilyas[1*], Shibani Santurkar[1], Dimitris Tsipras[1],
Firdaus Janoos[2], Larry Rudolph[1,2], and Aleksander Mądry[1]

[1]MIT [2]Two Sigma
{engstrom,ailyas,shibani,tsipras,madry}@mit.edu
rudolph@csail.mit.edu, firdaus.janoos@twosigma.com

ABSTRACT

We study the roots of algorithmic progress in deep policy gradient algorithms through a case study on two popular algorithms: Proximal Policy Optimization (PPO) and Trust Region Policy Optimization (TRPO). Specifically, we investigate the consequences of "code-level optimizations:" algorithm augmentations found only in implementations or described as auxiliary details to the core algorithm. Seemingly of secondary importance, such optimizations turn out to have a major impact on agent behavior. Our results show that they (a) are responsible for most of PPO's gain in cumulative reward over TRPO, and (b) fundamentally change how RL methods function. These insights show the difficulty and importance of attributing performance gains in deep reinforcement learning.

Original paper

Thank you for your attention!