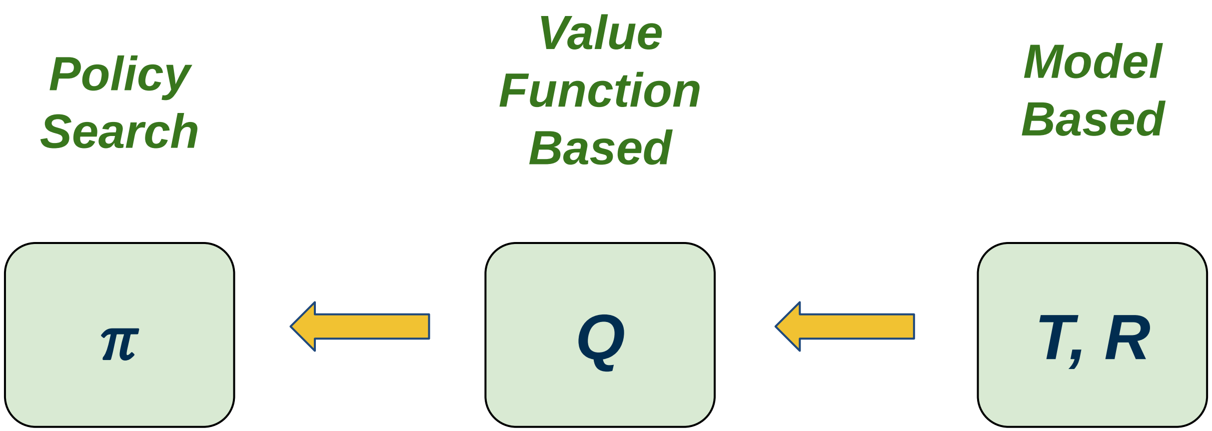
Reinforcement Learning – COMS4061A

Policy Gradient Methods

Prof. Benjamin Rosman
Benjamin.Rosman1@wits.ac.za / benjros@gmail.com

Based heavily on slides by Rich Sutton and Doina Precup

RL approaches



Policy search

- Learn policy directly:

$$\pi_\theta(s, a) = \pi(s, a; \theta)$$

- Parameterise policy: learn parameters of policy

- Why?
  - When might it be easier to learn a policy than a value function?
  - Learning a Q-function can be complicated
    - Policy may be much simpler than learning a value for each state-action
  - Injecting information?
  - Stochasticity?

Policy search

Trajectory τ

$$p(\tau|\theta)$$
$$= p_\theta(s_1, a_1, s_2, a_2, \dots, s_T, a_T)$$
$$= p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)$$

- Objective function?
  - Maximise return given θ:

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau|\theta)} \left[\sum_t \gamma^t r_t | \pi_\theta \right]$$
$$\theta^* = argmax_\theta J(\theta)$$

Note:
The return R, cost J, utility U are often used interchangeably here

- Always more efficient to follow the gradient!

Hill climbing

- What if you can't differentiate π?

- Sample-based optimisation:
  - Sample some θ values near your current best θ
  - Compute return
  - Approximate a gradient
    - Finite differences
  - Adjust your current best θ to give the highest value

- Other approaches, e.g. genetic algorithms

Aibo gait optimization

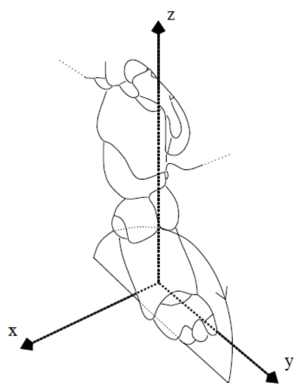


Fig. 2. The elliptical locus of the Aibo's foot. The half-ellipse is defined by length, height, and position in the x-y plane.

All told, the following set of 12 parameters define the Aibo's gait [10]:

- The front locus (3 parameters: height, x-pos., y-pos.)
- The rear locus (3 parameters)
- Locus length
- Locus skew multiplier in the x-y plane (for turning)
- The height of the front of the body
- The height of the rear of the body
- The time each foot takes to move through its locus
- The fraction of time each foot spends on the ground

from Kohl and Stone, ICRA 2004

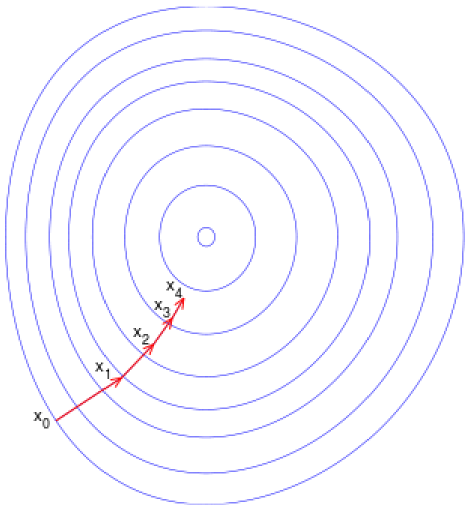
Using gradients

- If we can differentiate π
  - Compute and ascend $\partial R / \partial \theta$

  - This is the gradient of return w.r.t policy parameters

  - $\theta_{t+1} = \theta_t + \Delta\theta_t$
    $= \theta_t + \alpha \nabla J(\theta_t)$

  - These are called **policy gradient methods**



Policy Gradient Theorem

- Why does this work?

- Relate the **gradient of performance with respect to the policy parameter** to the **gradient of the policy**

- Policy gradient theorem:
  - $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a)\, Q^{\pi_\theta}(s,a)]$

- No explicit dependence on distribution of states (or model)

What is a good form for a parameterised function f?

- Simplest thing you can do?
  - *Linear value function approximation*
  - Use set of basis functions $\phi_1, \ldots, \phi_n$
  - f is a linear function of them:
  - $\hat{f} = \boldsymbol{w} \cdot \Phi(s, a) = \sum_{i=1}^{n} w_i \phi_i(s, a)$
  - We'll want to learn parameters $\boldsymbol{w}$
- Neural network:
  - $f = f(s, a; \boldsymbol{w})$

Basis functions $\phi(x)$:

- Could be polynomial in state vars:
  - 1st order: $[1, x, y]$
  - 2nd order: $[1, x, y, x^2, y^2, xy]$
  - This is a Taylor expansion
- Others:
  - Fourier basis
  - Wavelet basis
  - …

REINFORCE

REward Increment = Nonnegative Factor times Offset Reinforcement times Characteristic Eligibility

(Monte-Carlo Policy Gradient)

- REINFORCE: one particularly popular sample-based estimate of the gradient
  - Based on the policy gradient theorem, but approximate the $\mathbb{E}$ with sampled trajectories (Monte-Carlo samples)

$$\Delta\theta_t = \alpha R_t \frac{\nabla_\theta \pi_\theta(a_t|s_t)}{\pi_\theta(a_t|s_t)} = \alpha R_t \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Log derivative trick: $\frac{\nabla_\theta x}{x} = \nabla_\theta \log x$

The return R_t acts as an estimate of $Q^{\pi_\theta}(s,a)$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a)\; Q^{\pi_\theta}(s,a)]$$

REINFORCE algorithm

- Initialise θ
- For each episode:
  - Choose actions according to π_θ: $a \sim \pi_\theta(a|s)$
  - Gather samples $\{s_1, a_1, r_1, \ldots, s_T, a_T, r_T\}$
  - For $t = 1\ to\ T$
    - $\theta \leftarrow \theta + \alpha R_t \nabla_\theta \log \pi_\theta(a_t|s_t)$

(that's it)

Deriving REINFORCE

- Cost:

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)} \left[r(\tau) \right] \\ = \int_\tau r(\tau) p(\tau;\theta) \mathrm{d}\tau$$

$$\tau = (s_0, a_0, r_0, s_1, \ldots)$$

- Derivative:

$$\nabla_\theta J(\theta) = \int_\tau r(\tau) \nabla_\theta p(\tau;\theta) \mathrm{d}\tau$$

- Transformation:

Log derivative trick: $\frac{\nabla_\theta x}{x} = \nabla_\theta \log x$

$$\nabla_\theta p(\tau;\theta) = \boxed{p(\tau;\theta)} \frac{\nabla_\theta p(\tau;\theta)}{\boxed{p(\tau;\theta)}} = p(\tau;\theta) \nabla_\theta \log p(\tau;\theta)$$

- Substitute:

$$\nabla_\theta J(\theta) = \int_\tau \left(r(\tau) \nabla_\theta \log p(\tau;\theta) \right) p(\tau;\theta) \mathrm{d}\tau \\ = \mathbb{E}_{\tau \sim p(\tau;\theta)} \left[r(\tau) \nabla_\theta \log p(\tau;\theta) \right]$$

Deriving REINFORCE

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)} \left[r(\tau) \nabla_\theta \log p(\tau;\theta) \right]$$

- Recall:

$$p(\tau;\theta) = \prod_{t \geq 0} p(s_{t+1}|s_t, a_t) \pi_\theta(a_t|s_t)$$

- So:

$$\log p(\tau;\theta) = \sum_{t \geq 0} \log p(s_{t+1}|s_t, a_t) + \log \pi_\theta(a_t|s_t)$$

- Derivative:

$$\nabla_\theta \log p(\tau;\theta) = \sum_{t \geq 0} \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Note we lose dependence on dynamics

- And so estimate:

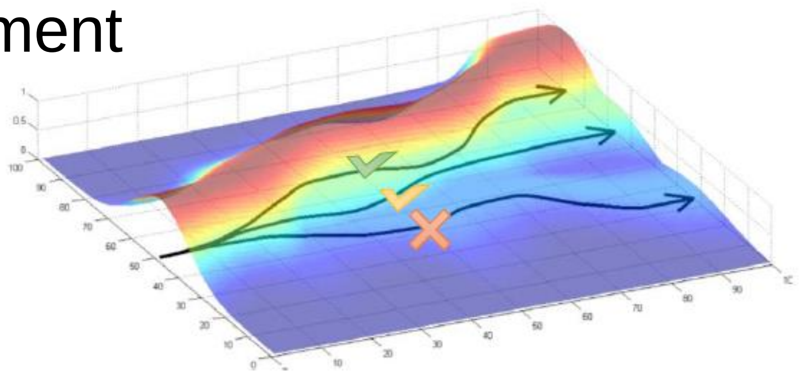
$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} r(\tau) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Interpretation

- Gradient estimator: $\nabla_\theta J(\theta) \approx \sum_{t \geq 0} \boxed{r(\tau)} \boxed{\nabla_\theta \log \pi_\theta(a_t|s_t)}$

- Think of this as saying:
  - If $r(\tau)$ is high: push up probabilities of seen actions
  - If $r(\tau)$ is low: push down probabilities of seen actions

- Simple version of credit assignment



Variance

- This gradient estimator (MC) turns out to have a high variance!!

$$\nabla_\theta J(\theta) \approx \sum_{t\geq 0} r(\tau) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

  - Why?
    - These are all samples of a run of a policy!
  - Slow convergence
- Correct with a baseline:
- $\nabla_\theta J(\theta) \approx \sum_{t\geq 0} (r(\tau) - b(s_t)) \nabla_\theta \log \pi_\theta(a_t|s_t)$
- Because b could be 0, this is a generalisation of REINFORCE
  - Will converge asymptotically to a local minimum

Why can we use a baseline?

- $\nabla_\theta J(\theta) \approx \sum_{t\geq 0}(r(\tau) - b(s_t))\, \nabla_\theta \log \pi_\theta(a_t|s_t)$
- Intuition: can add or subtract b from r without biasing algorithm
  - As long as b not a function of a_t
- Mathematically (with some notational abuse):

$$\mathbb{E}_{\pi_\theta}\left[\sum_t b\nabla_\theta \log \pi_\theta(a_t|s_t)\right]$$

Keeps the gradient unbiased, i.e. doesn't change the expected value, but can change the variance!

$$= \int \left[\sum_t b\pi_\theta(a_t|s_t)\nabla_\theta \log \pi_\theta(a_t|s_t)\right] d\tau$$

$$= \int \mathrm{b}\nabla_\theta \pi_\theta(\tau)\, d\tau$$

Log derivative trick: $\frac{\nabla_\theta x}{x} = \nabla_\theta \log x$

Try make cumulative reward smaller ➔ smaller variance

$$= b\nabla_\theta \int \pi_\theta(\tau)\, d\tau = b\nabla_\theta 1 = 0$$

Choice of baseline

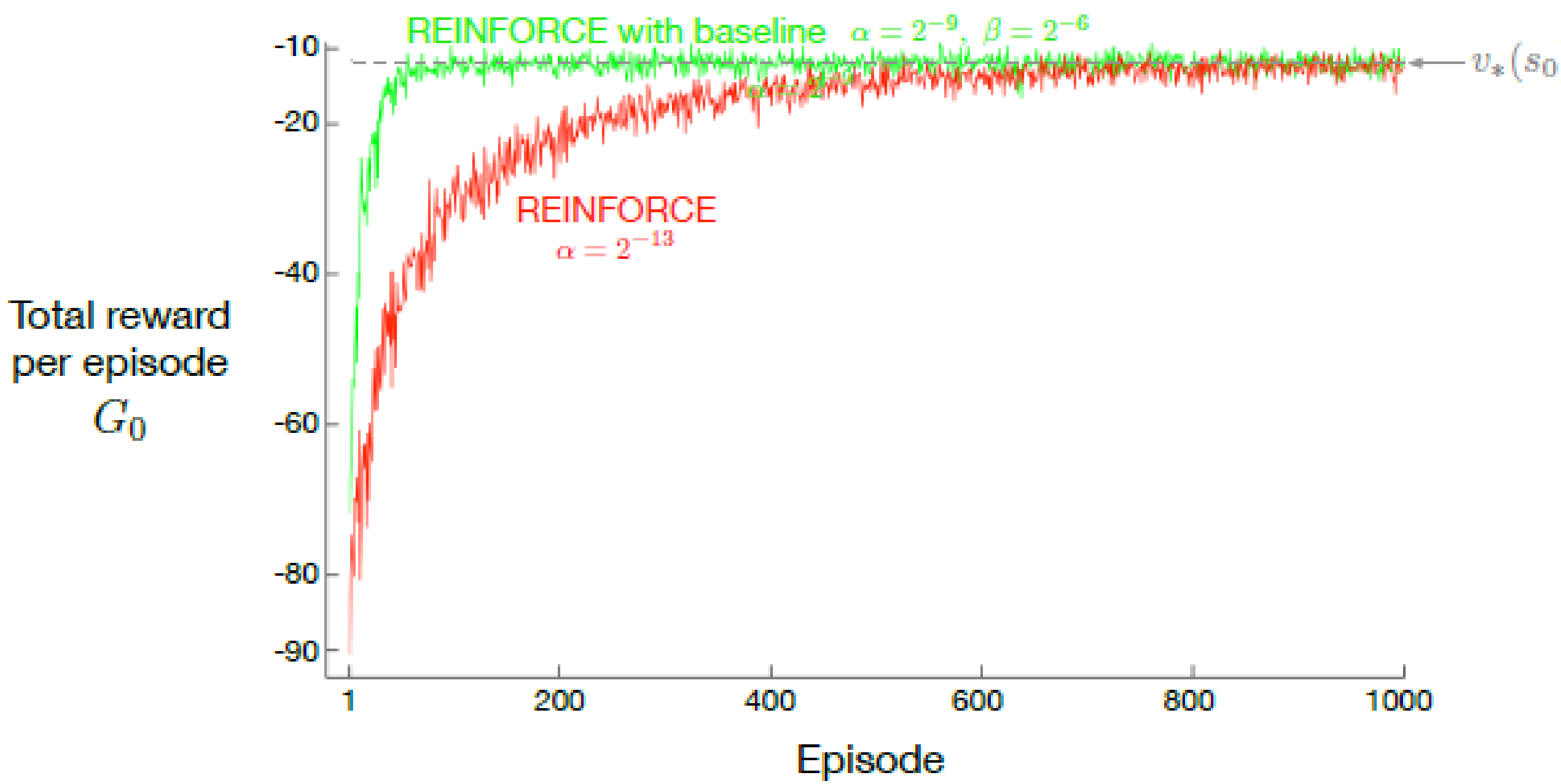
- What baseline to use?

- $b(s_t) = V(s_t)$
  - Change based on whether or not reward was better than expected

  - Term $r(\tau) - b(s_t)$ resembles advantage function:
    - $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$
    - Measures how much better a is than whatever π would have done

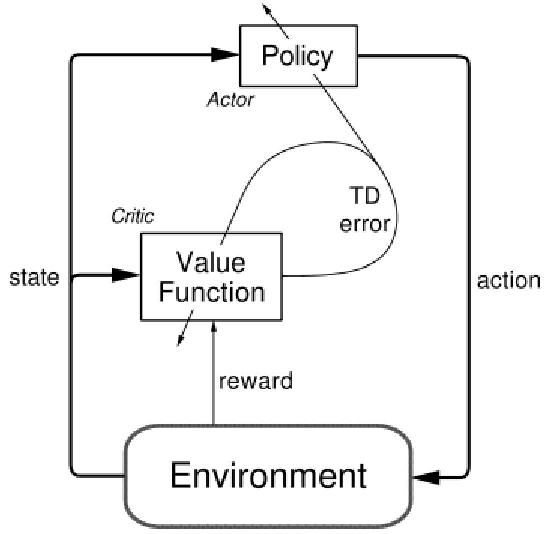
- Suggests we should be learning π and V!

Learning with a baseline



Actor-Critic

- Combine ideas from policy and value function methods
  - Approximate both the policy and the value function
- Actor improvement
  - Policy parameterised by θ
- Critic evaluation
  - Value function parameterised by ω
  - Either $V(s;\omega)$ or $Q(s,a;\omega)$

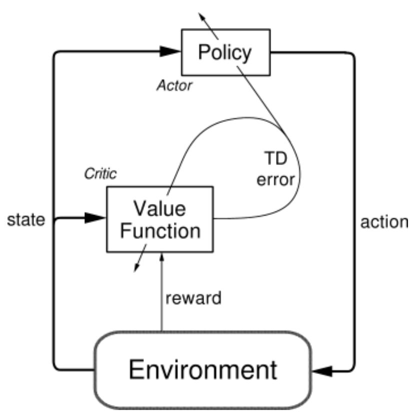


- Keep track of two sets of parameters

Actor-Critic pseudocode

What forms could we use?

- Input: parameterised forms for $\pi_\theta(a|s)$ and $V_\omega(s)$
- Input: learning rates $\alpha_\omega > 0$ and $\alpha_\theta > 0$
- For each episode:
  - Initialise s
  - For each time step:
    - Choose $a \sim \pi_\theta(a|s)$ Policy is stochastic: this is a random draw (actor)
    - Take a, observe s', r
    - $\delta \leftarrow r + \gamma V_\omega(s') - V_\omega(s)$ Compute TD error (critic)
    - $\omega \leftarrow \omega + \alpha_\omega \delta \nabla_\omega V_\omega(s)$
    - $\theta \leftarrow \theta + \alpha_\theta \delta \nabla_\theta \log \pi_\theta\,(a|s)$
    - $s \leftarrow s'$ Update parameters by gradient ascent



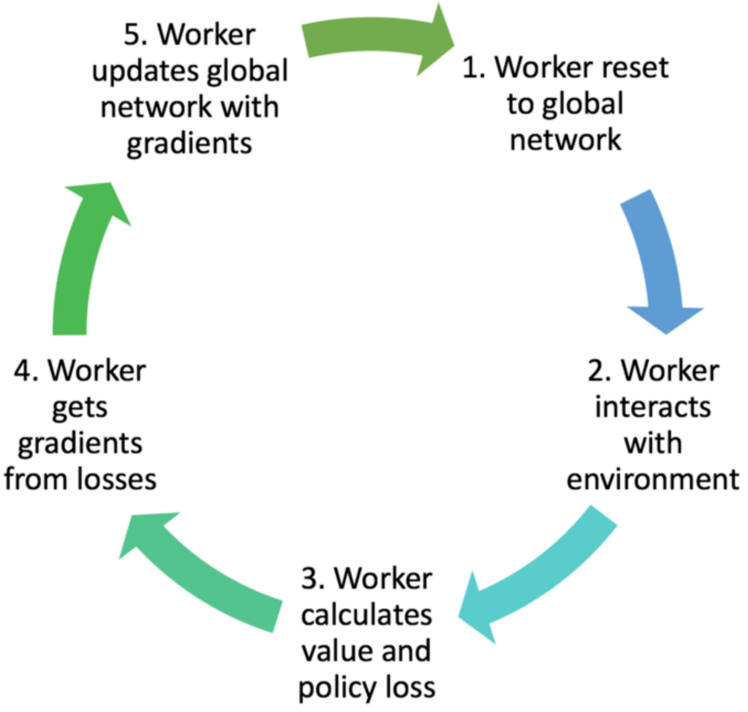
Many ways to do the updates

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ v_t\right] && \text{REINFORCE} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ Q^w(s,a)\right] && \text{Q Actor-Critic} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ A^w(s,a)\right] && \text{Advantage Actor-Critic} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ \delta\right] && \text{TD Actor-Critic} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ \delta e\right] && \text{TD}(\lambda)\ \text{Actor-Critic} \\
G_\theta^{-1} \nabla_\theta J(\theta) &= w && \text{Natural Actor-Critic}
\end{aligned}
$$

Asynchronous Advantage Actor-Critic (A3C)

- Actor-Critic can be easily parallelised
- Why is this useful?
  - Speed up exploration of state space

- Have multiple agents training with shared parameters



[Mnih et al., 2016]

Algorithm S3 Asynchronous advantage actor-critic - pseudocode for each actor-learner thread.

// Assume global shared parameter vectors θ and θ_v and global shared counter $T = 0$
// Assume thread-specific parameter vectors θ' and θ'_v Spin up a new agent/thread
Initialize thread step counter $t \leftarrow 1$
repeat
    Reset gradients: $d\theta \leftarrow 0$ and $d\theta_v \leftarrow 0$.
    Synchronize thread-specific parameters $\theta' = \theta$ and $\theta'_v = \theta_v$
    $t_{start} = t$ Use global parameters
    Get state s_t
    repeat
        Perform a_t according to policy $\pi(a_t|s_t;\theta')$
        Receive reward r_t and new state s_{t+1} Act
        $t \leftarrow t + 1$
        $T \leftarrow T + 1$
    until terminal s_t **or** $t - t_{start} == t_{max}$
    $R = \begin{cases} 0 & \text{for terminal } s_t \\ V(s_t, \theta'_v) & \text{for non-terminal } s_t \text{// Bootstrap from last state} \end{cases}$
    for $i \in \{t-1, \ldots, t_{start}\}$ **do** Update local parameters using advantage functions
        $R \leftarrow r_i + \gamma R$
        Accumulate gradients wrt θ': $d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi(a_i|s_i;\theta')(R - V(s_i;\theta'_v))$
        Accumulate gradients wrt θ'_v: $d\theta_v \leftarrow d\theta_v + \partial\left(R - V(s_i;\theta'_v)\right)^2 / \partial\theta'_v$
    end for
    Perform asynchronous update of θ using $d\theta$ and of θ_v using $d\theta_v$. Update global parameters
until $T > T_{max}$

[Mnih et al., 2016]

Deep policy search

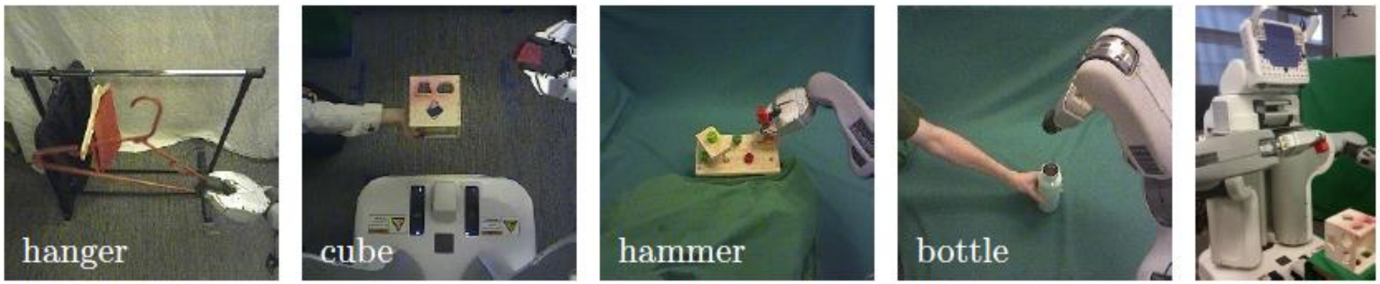
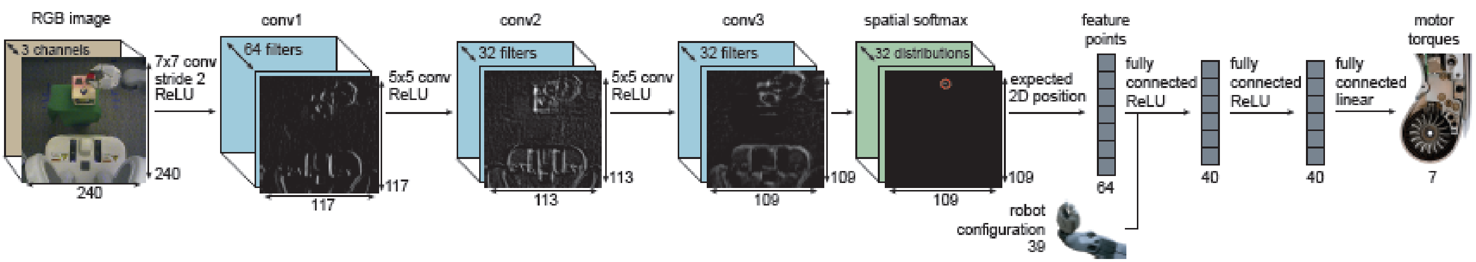
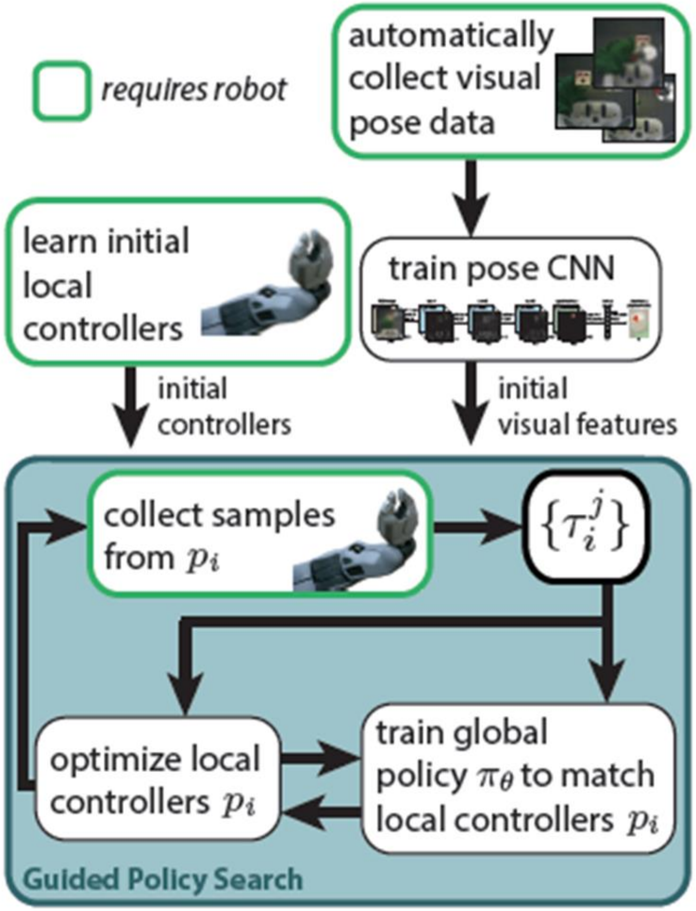


Figure 1: Our method learns visuomotor policies that directly use camera image observations (left) to set motor torques on a PR2 robot (right).



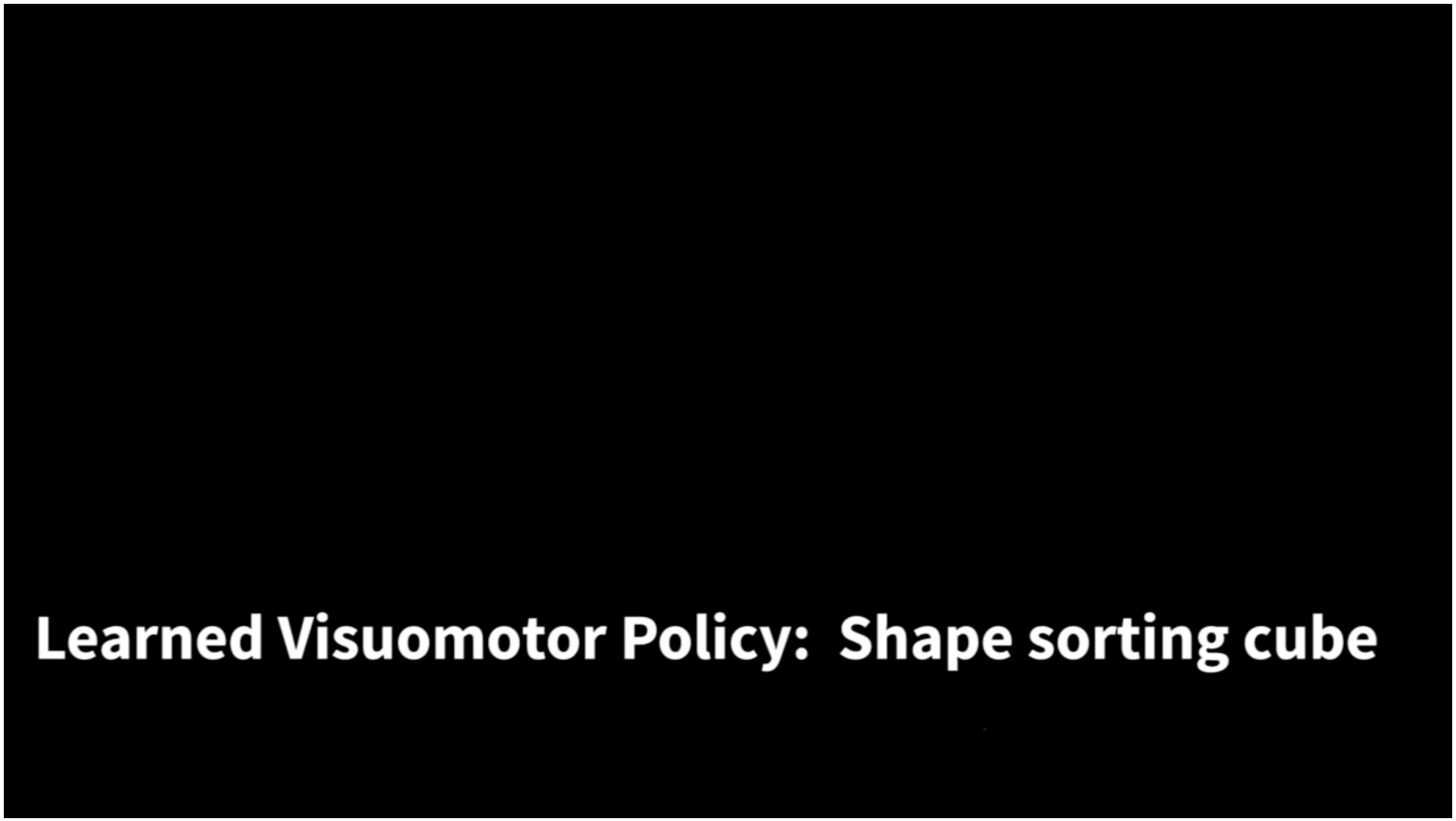
[Levine et al., 2016]

Deep policy search



[Levine et al., 2016]

Robotics



[Levine et al., 2016]

https://www.youtube.com/watch?v=Q4bMcUk6pcw

Homework

1. Read this paper (with all supplementary material) on reproducibility in deep RL:

   Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., & Meger, D. (2018, April). Deep reinforcement learning that matters. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

   https://arxiv.org/pdf/1709.06560.pdf

   What implications does this have for how you should conduct your experiments?

2. Work on your assignment!

3. Quiz next week!