Carnegie Mellon

School of Computer Science

Deep Reinforcement Learning and Control

Determinist PG, Re-parametrized PG

Fall 2020, CMU 10-703

Katerina Fragkiadaki

Advantage Actor-Critic

0. Initialize policy parameters θ and critic parameters ϕ .

1. Sample trajectories $\{\tau_i = \{s_t^i, a_t^i\}_{i=0}^T\}$ by deploying the current policy $\pi_\theta(a_t \mid s_t)$.

2. Fit value function $V_\phi^\pi(s)$ by MC or TD estimation (update ϕ)

3. Compute action advantages $A^\pi(s_t^i, a_t^i) = R(s_t^i, a_t^i) + \gamma V_\phi^\pi(s_{t+1}^i) - V_\phi^\pi(s_t^i)$

4. $\nabla_\theta U(\theta) \approx \hat{g} = \dfrac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\alpha_t^i \mid s_t^i) A^\pi(s_t^i, a_t^i)$

5. $\theta \leftarrow \theta + \alpha \nabla_\theta U(\theta)$

Policy gradients so far

Policy objective:

$$\max_{\theta} . \ \mathbb{E}_{\tau \sim P_\theta(\tau)} \left[R(\tau) \right]$$

Advantage actor critic policy gradient:

$$\mathbb{E}_{s \sim d^{\pi_\theta}(s),\ a \sim \pi_\theta(a|s)} \nabla_\theta \log \pi_\theta(a \mid s) \left[A(s, a; \phi) \right]$$

Another policy objective

Previous policy objective:

$$\max_{\theta} . \ \mathbb{E}_{\tau \sim P_\theta(\tau)} \left[R(\tau) \right]$$

New policy objective:

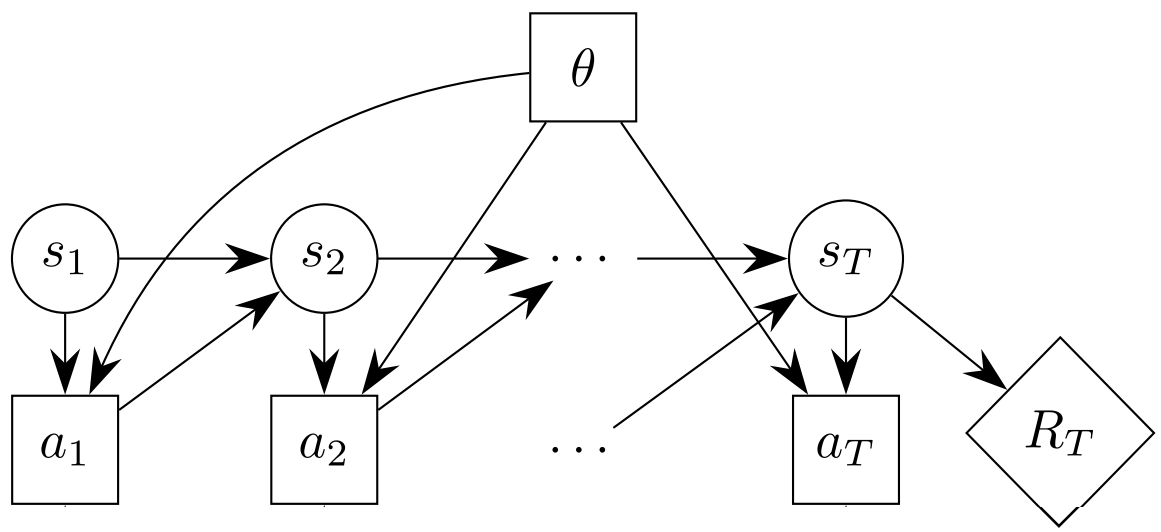
$$\max_{\theta} . \ \mathbb{E}_{\tau \sim P_\theta(\tau)} \left[\sum_{t=1}^{T} Q(s_t, a_t) \right]$$

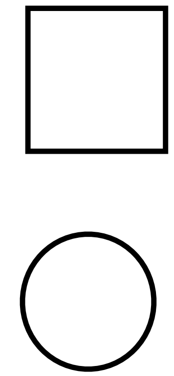
Qs:

- Can we backpropagate through the Q function approximator?
- If the policy is deterministic we already know how to do it via the chain rule!

$$\mathbb{E} \sum_t \frac{dQ(s_t, a_t)}{d\theta} = \mathbb{E} \sum_t \frac{dQ(s_t, a_t)}{da_t} \frac{da_t}{d\theta}$$

Deep Deterministic Policy Gradients





deterministic node: the value is a deterministic function of its input

stochastic node: the value is sampled based on its input (which parametrizes the distribution to sample from)

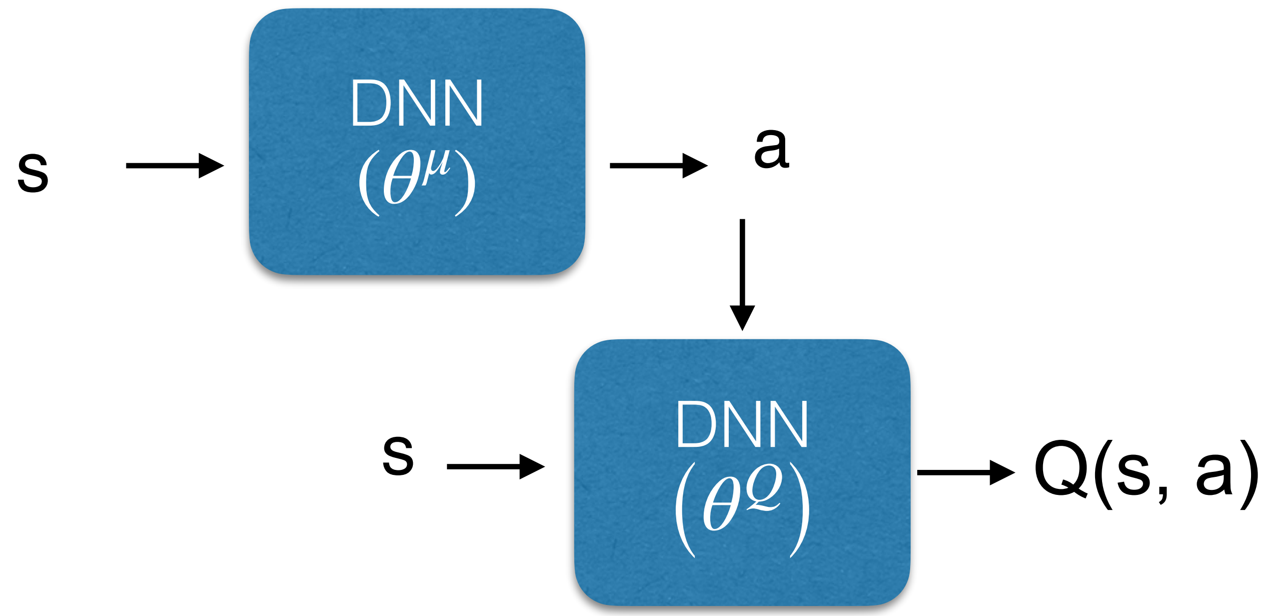
$$a = \pi_\theta(s)$$

$$\mathbb{E}\sum_t \frac{dQ(s_t, a_t)}{d\theta} = \mathbb{E}\sum_{t=1}^{T} \frac{dQ(s_t, a_t)}{da_t}\frac{da_t}{d\theta}$$

Continuous control with deep reinforcement learning, *Lilicrap et al. 2016*

Deep Deterministic Policy Gradients

The computational graph:



We are following a stochastic behavior policy to collect data.
DDPG :Deep Q learning for continuous actions

Continuous control with deep reinforcement learning, *Lilicrap et al. 2016*

Deep Deterministic Policy Gradients

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$ Fitting the Q function
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

Another policy objective

Previous policy objective:

$$\max_{\theta} . \ \mathbb{E}_{\tau \sim P_\theta(\tau)} \left[R(\tau) \right]$$

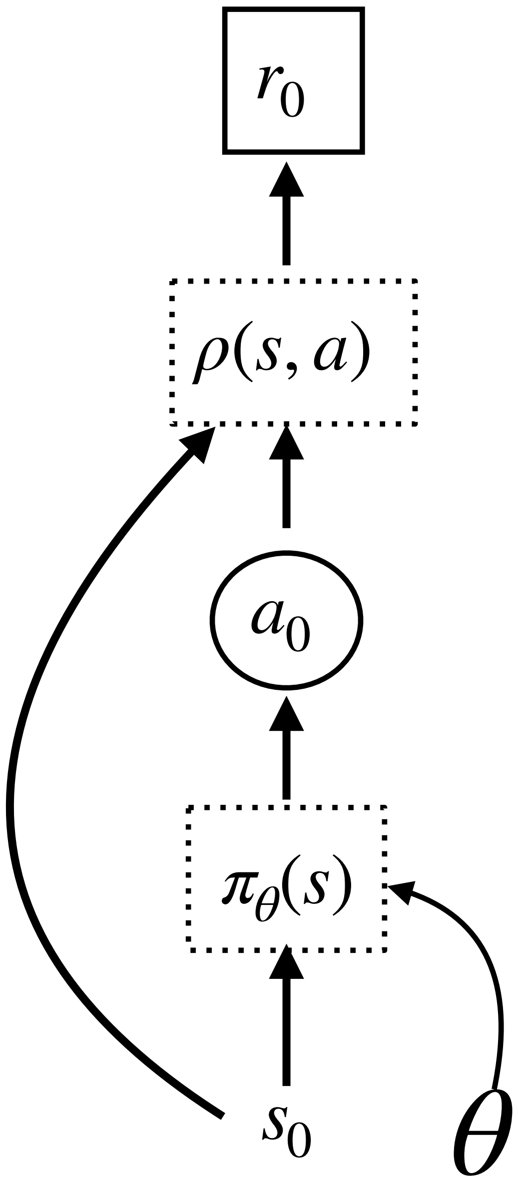
New policy objective:

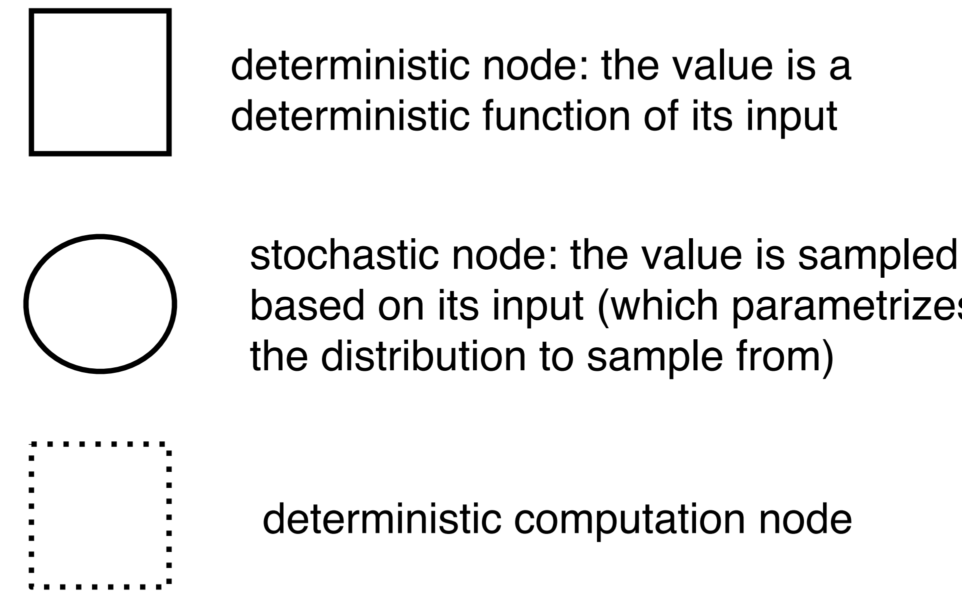
$$\max_{\theta} . \ \mathbb{E}_{\tau \sim P_\theta(\tau)} \left[\sum_{t=1}^{T} Q(s_t, a_t) \right]$$

Qs:

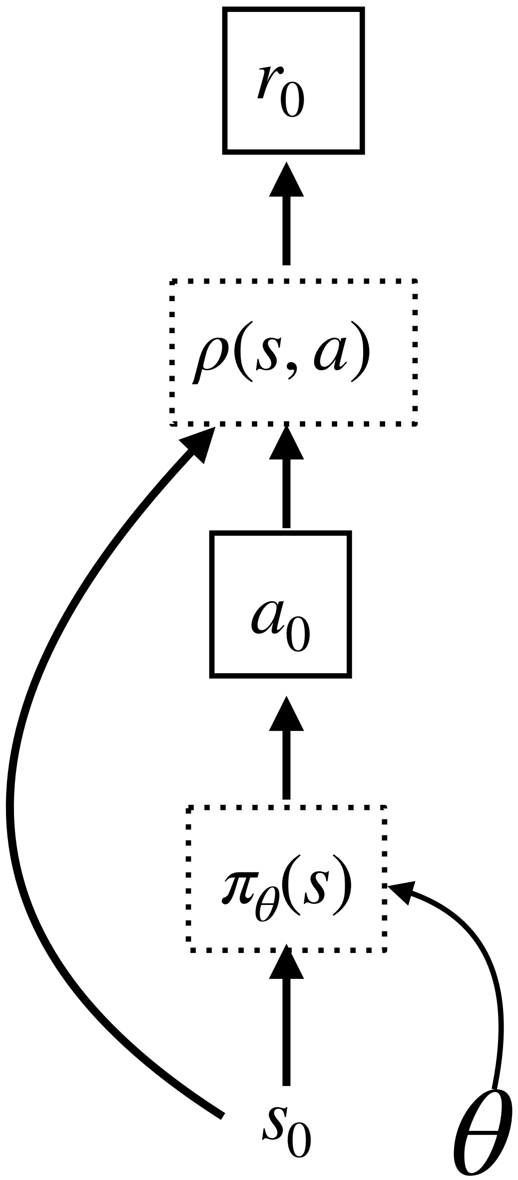
- Can we backpropagate through the Q function approximator?
- If the policy is deterministic we already know how to do it via the chain rule!
- What if the policy is a parametrized Gaussian distribution?

Imagine we knew the reward function $\rho(s, a)$

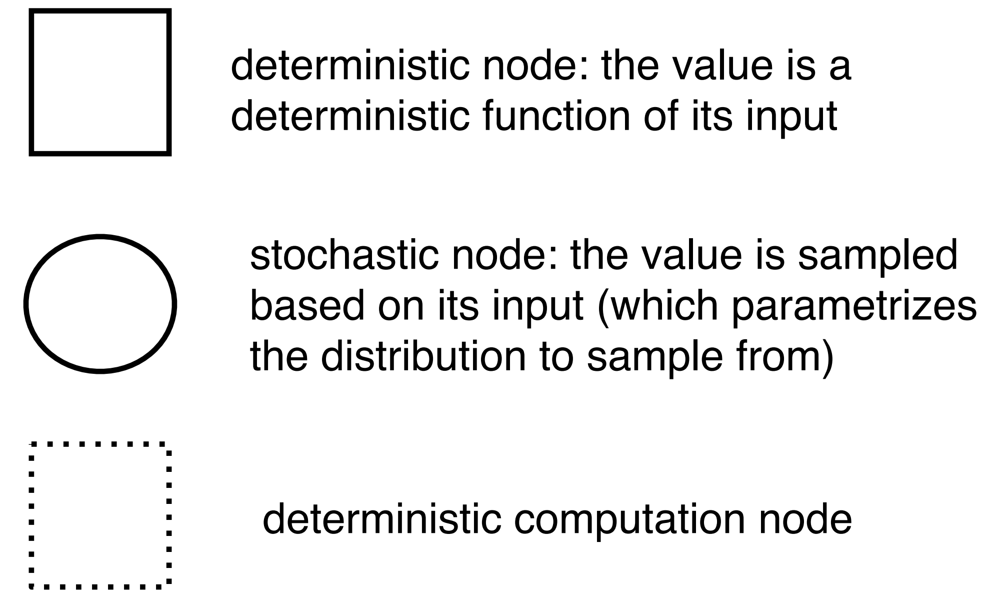




Deterministic policy



$a = \pi_\theta(s)$



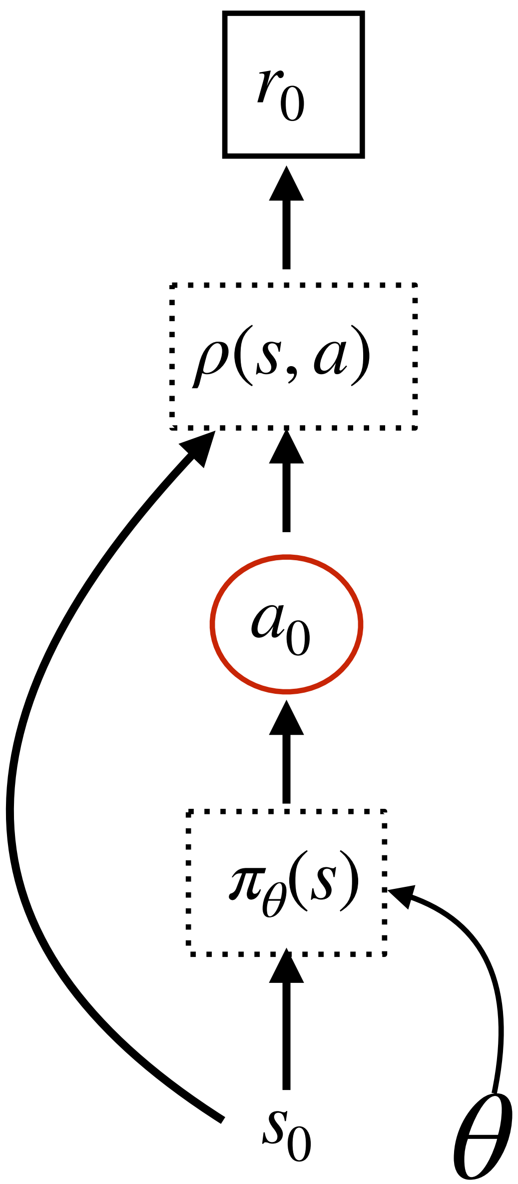
I want to learn θ to maximize the average reward obtained.

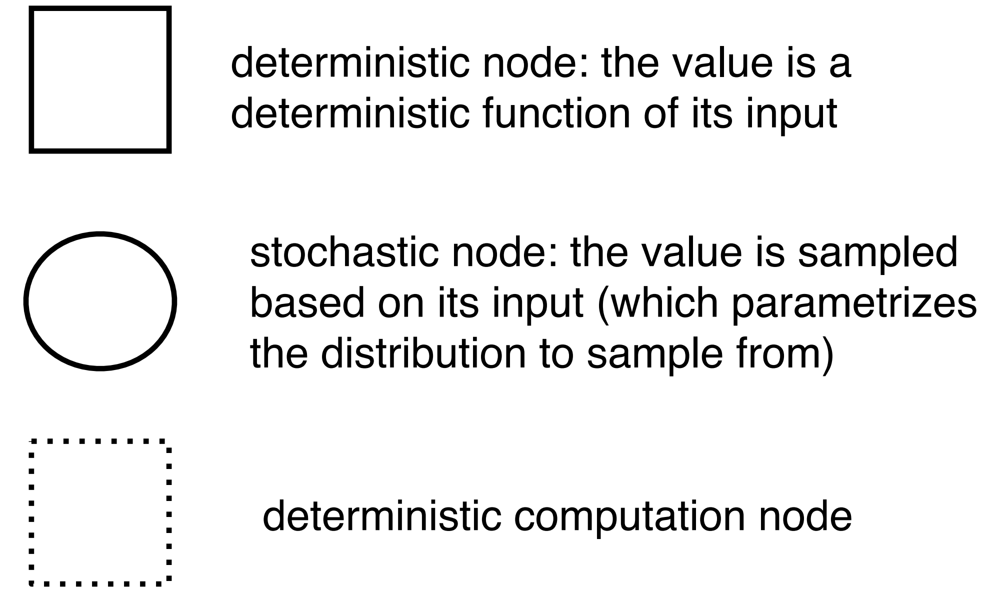
$$\max_{\theta}. \quad \rho(s_0, a)$$

I can compute the gradient with the chain rule.

$$\nabla_\theta \rho(s, a) = \frac{d\rho}{da}\frac{da}{d\theta}$$

Stochastic policy



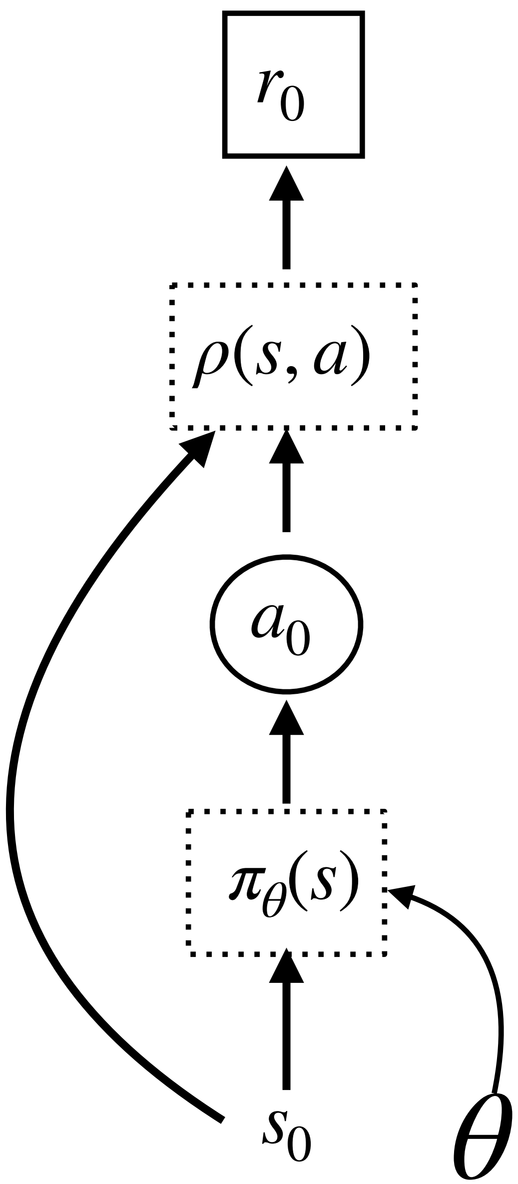


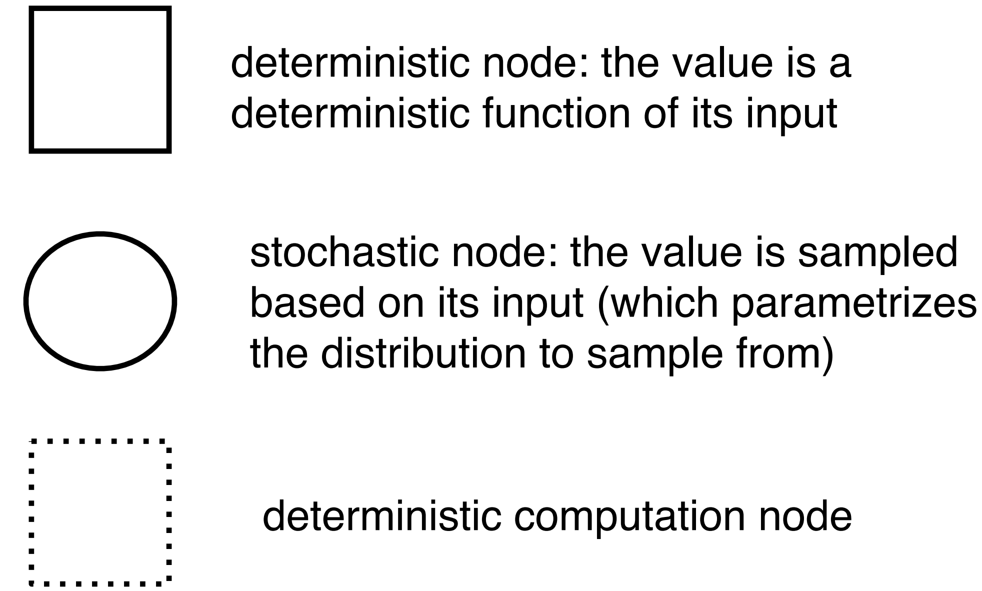
I want to learn θ to maximize the average reward obtained.

$$\max_{\theta} . \quad \mathbb{E}_a \rho(s_0, a)$$

$$\nabla_\theta \mathbb{E}_a \rho(s_0, a)$$

Stochastic policy





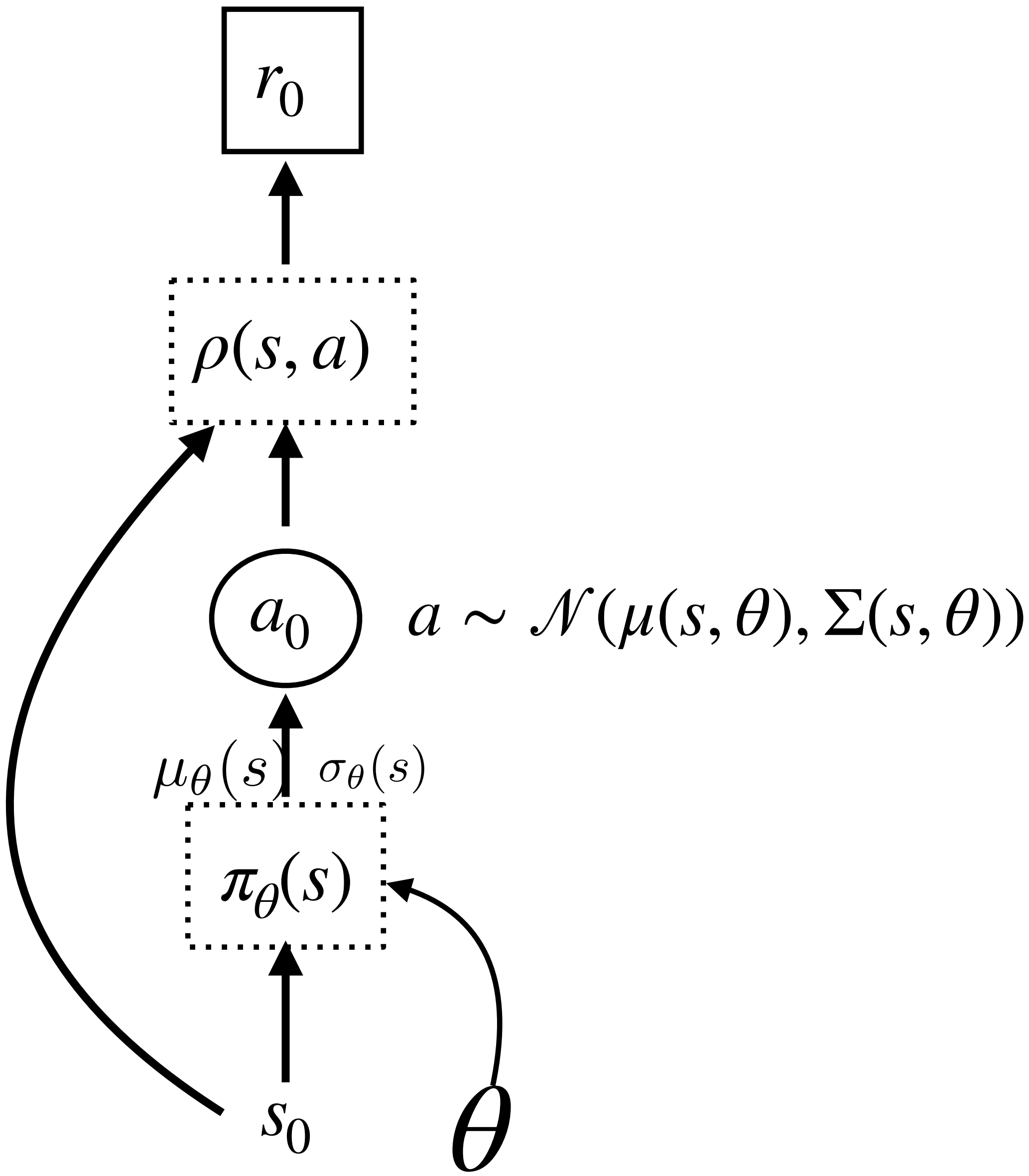
I want to learn θ to maximize the average reward obtained.

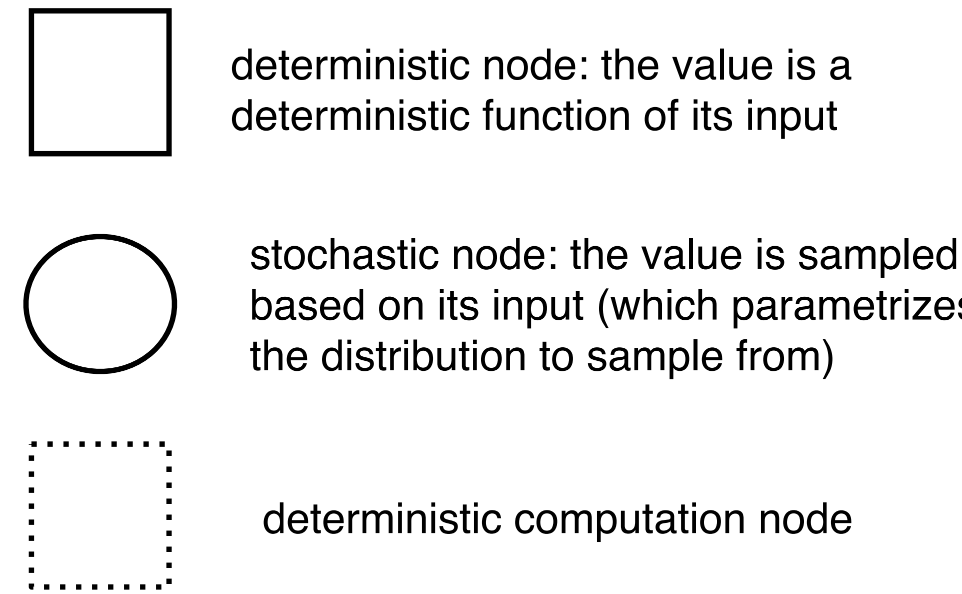
$$\max_\theta . \quad \mathbb{E}_a \rho(s_0, a)$$

Likelihood ratio estimator, works for both continuous and discrete actions

$$\mathbb{E}_a \nabla_\theta \log \pi_\theta(s) \rho(s_0, a)$$

Example: Gaussian policy





I want to learn θ to maximize the average reward obtained.

$$\max_{\theta} . \quad \mathbb{E}_a \rho(s_0, a)$$

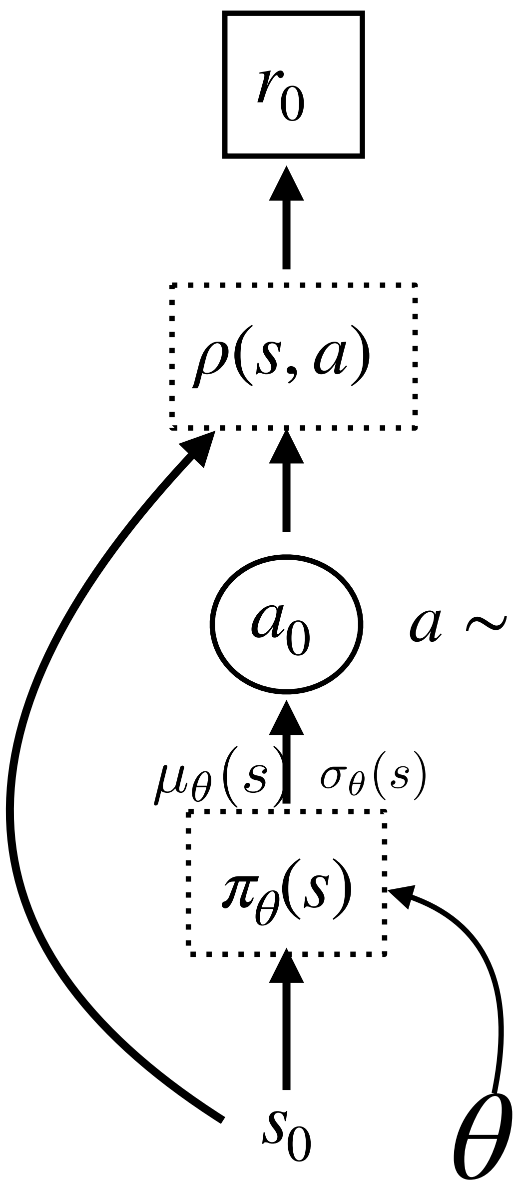
Likelihood ratio estimator, works for both continuous and discrete actions

$$\mathbb{E}_a \nabla_\theta \log \pi_\theta(s) \rho(s_0, a)$$

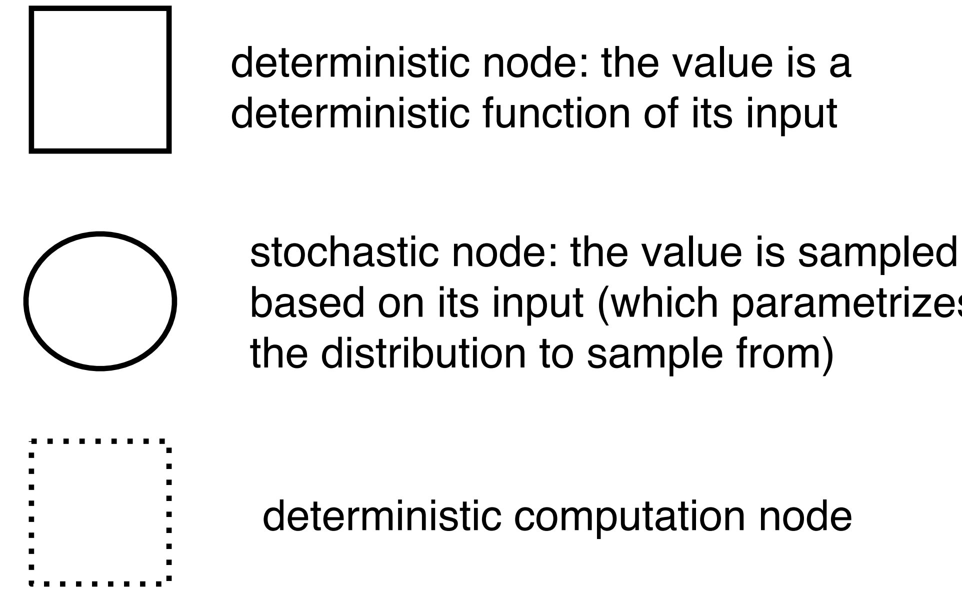
If σ^2 is constant:

$$\nabla_\theta \log \pi_\theta(s, a) = \frac{(a - \mu(s; \theta)) \frac{\partial \mu(s; \theta)}{\partial \theta}}{\sigma^2}$$

Example: Gaussian policy



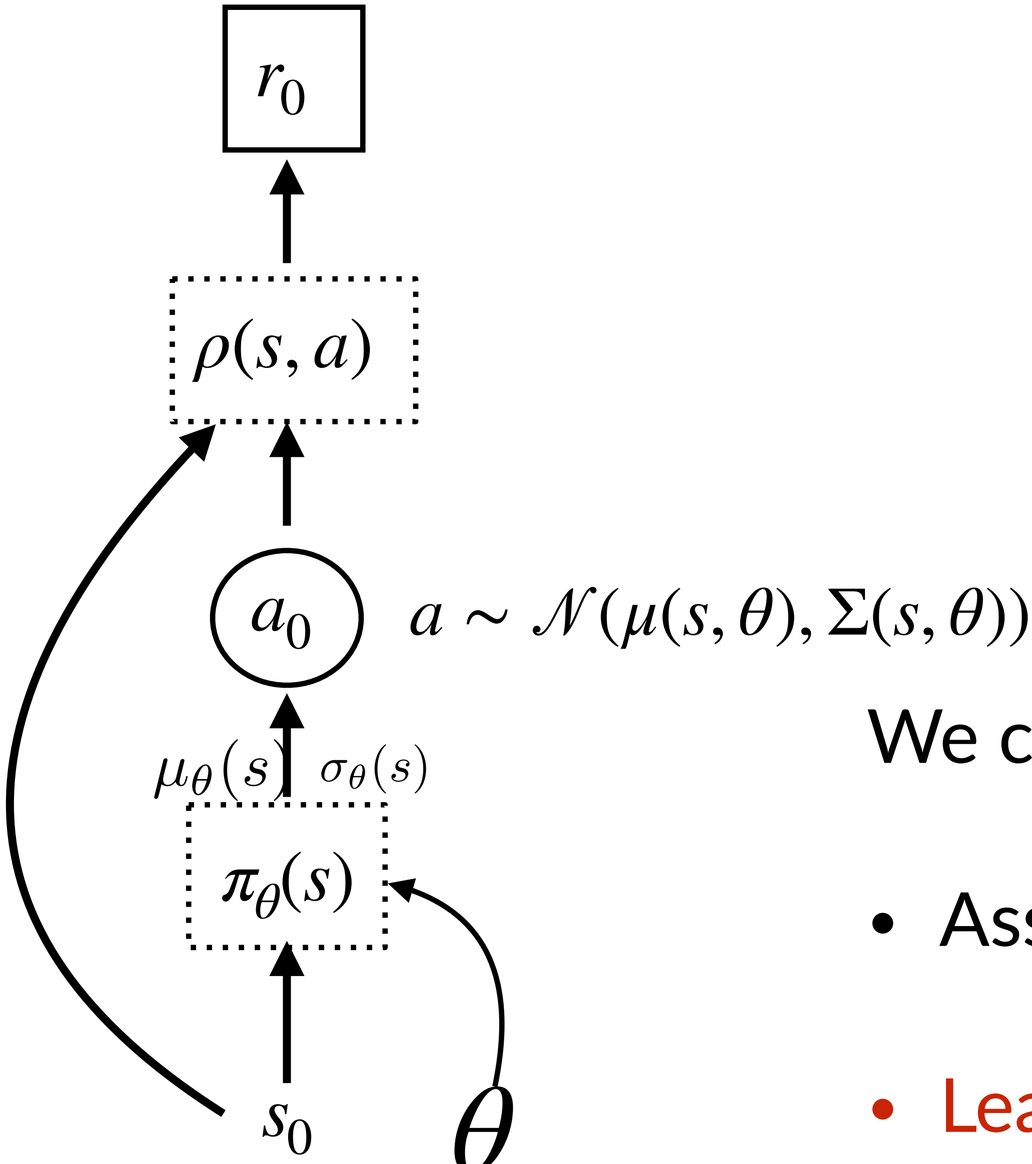
$a \sim \mathcal{N}(\mu(s, \theta), \Sigma(s, \theta))$

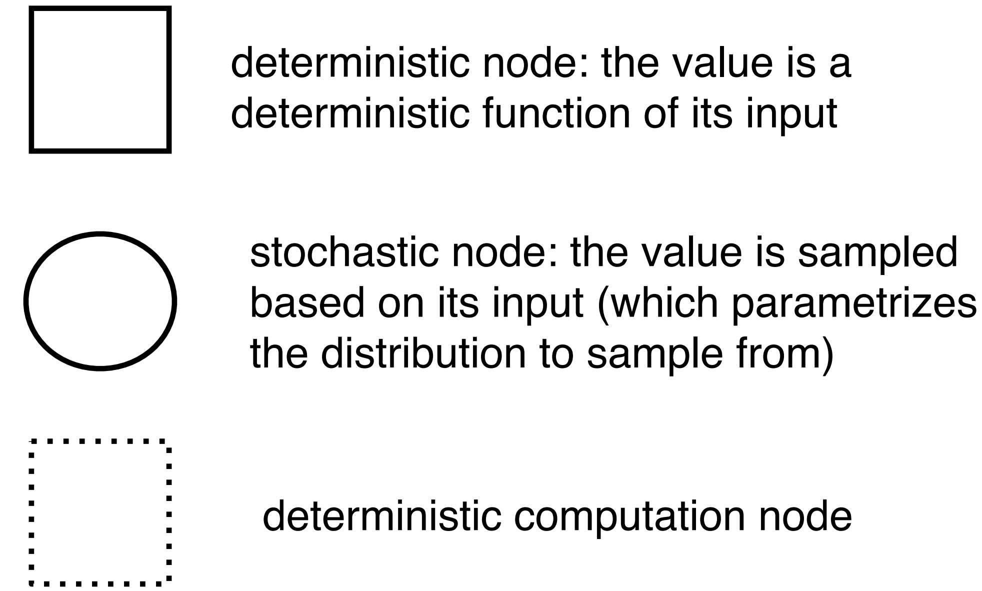


We can either:

- Assume σ fixed (not learned)
- Learn $\sigma(s, \theta)$ one value for all action coordinates (spherical or isotropic Gaussian)
- Learn $\sigma^i(s, \theta), i = 1 \cdots n$ (diagonal covariance)
- Learn a full covariance matrix $\Sigma(s, \theta)$

Example: Gaussian policy

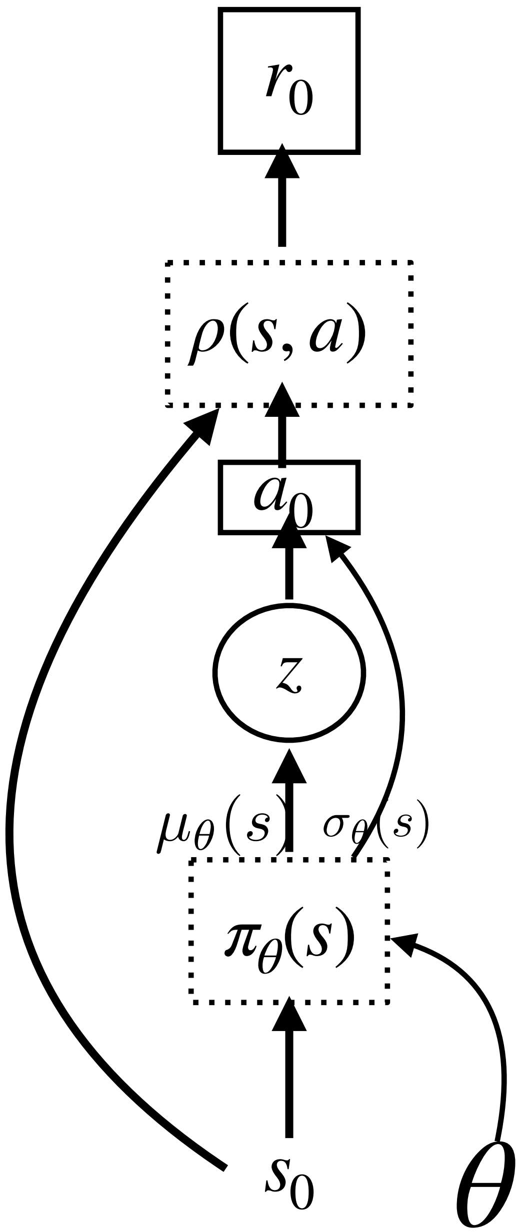




We can either:

- Assume σ fixed (not learned)
- Learn $\sigma(s, \theta)$ one value for all action coordinates (spherical or isotropic Gaussian)
- Learn $\sigma^i(s, \theta), i = 1 \cdots n$ (diagonal covariance)
- Learn a full covariance matrix $\Sigma(s, \theta)$

Re-parametrization for Gaussian



Instead of: $a \sim \mathcal{N}(\mu(s,\theta), \Sigma(s,\theta))$

We can write: $a = \mu(s,\theta) + z\sigma(s,\theta) \quad z \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{n\times n})$

Because: $\mathbb{E}_z(\mu(s,\theta) + z\sigma(s,\theta)) = \mu(s,\theta)$

$\mathrm{Var}_z(\mu(s,\theta) + z\sigma(s,\theta)) = \sigma(s,\theta)^2 \mathbf{I}_{n\times n}$

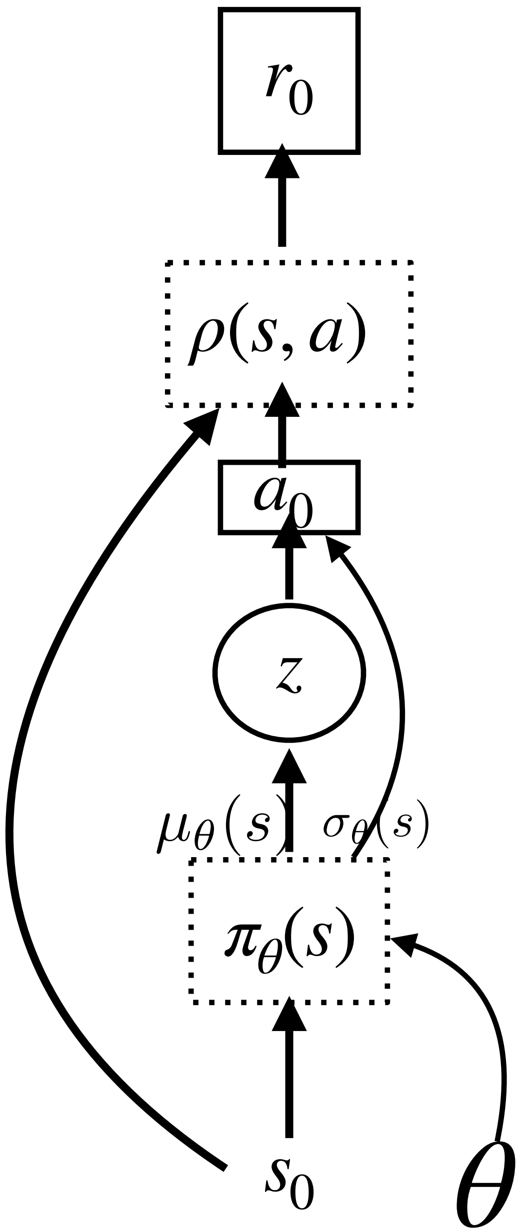
$$\max_\theta . \quad \mathbb{E}_a \rho(s_0, a)$$

$$\max_\theta . \quad \mathbb{E}_z \rho(s_0, a(z))$$

Qs:

- Does a depend on θ ?
- Does z depend on θ ?

Re-parametrization for Gaussian



Instead of: $a \sim \mathcal{N}(\mu(s,\theta), \Sigma(s,\theta))$

We can write: $a = \mu(s,\theta) + z\sigma(s,\theta) \quad z \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{n\times n})$

What do we gain?

$$\nabla_\theta \mathbb{E}_z \left[\rho\left(a(\theta,z), s\right) \right] = \mathbb{E}_z \frac{d\rho\left(a(\theta,z), s\right)}{da} \frac{da(\theta,z)}{d\theta}$$

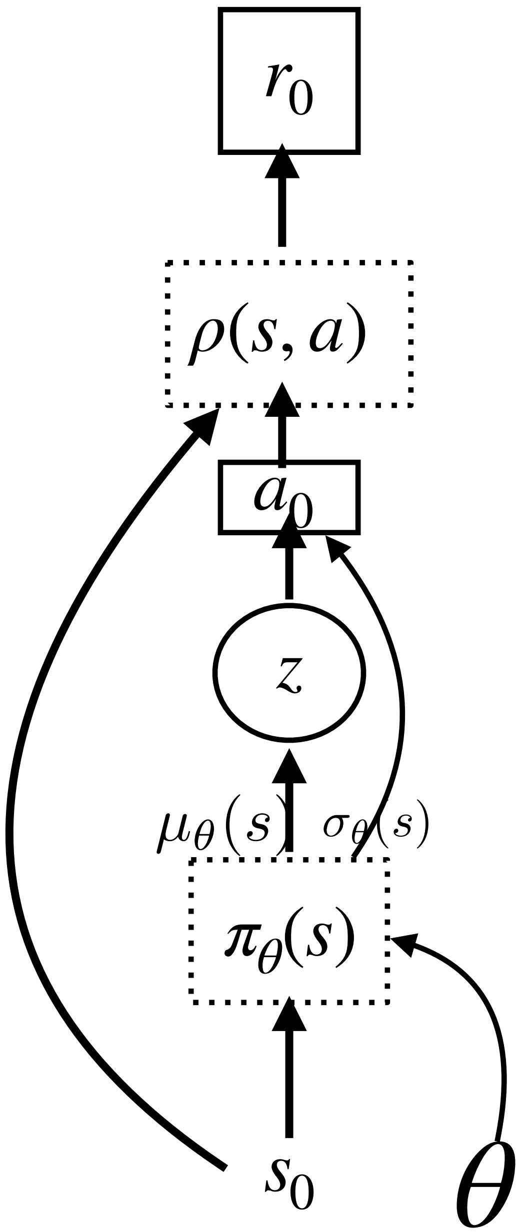
$$\frac{da(\theta,z)}{d\theta} = \frac{d\mu(s,\theta)}{d\theta} + z\frac{d\sigma(s,\theta)}{d\theta}$$

$$\max_\theta . \quad \mathbb{E}_a \rho(s_0, a)$$

$$\Downarrow$$

$$\max_\theta . \quad \mathbb{E}_z \rho(s_0, a(z))$$

Re-parametrization for Gaussian



Instead of: $a \sim \mathcal{N}(\mu(s,\theta), \Sigma(s,\theta))$

We can write: $a = \mu(s,\theta) + z\sigma(s,\theta) \quad z \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{n\times n})$

What do we gain?

$$\nabla_\theta \mathbb{E}_z\left[\rho\left(a(\theta,z),s\right)\right] = \mathbb{E}_z \frac{d\rho\left(a(\theta,z),s\right)}{da}\frac{da(\theta,z)}{d\theta}$$

$$\frac{da(\theta,z)}{d\theta} = \frac{d\mu(s,\theta)}{d\theta} + z\frac{d\sigma(s,\theta)}{d\theta}$$

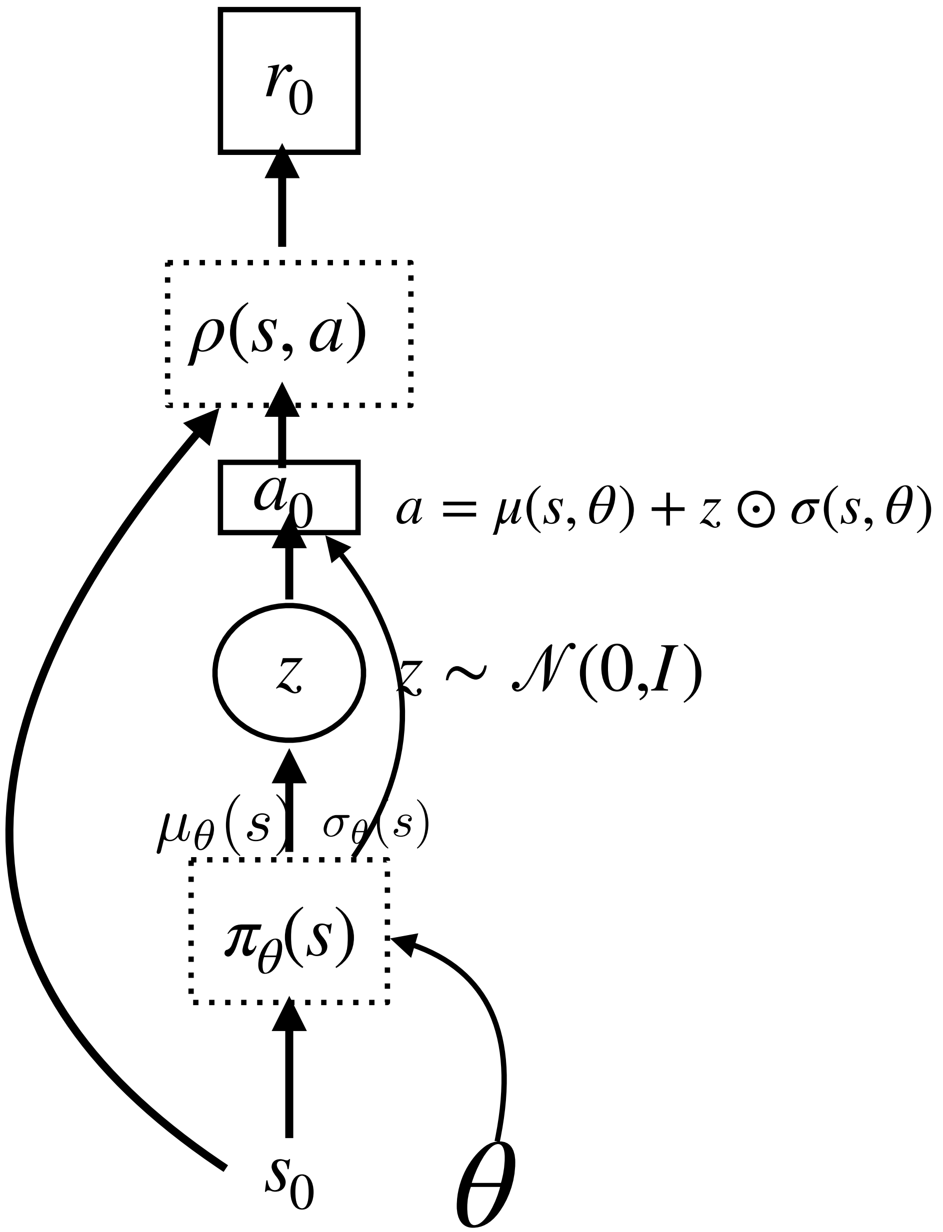
$$\max_\theta . \quad \mathbb{E}_a \rho(s_0, a)$$

$$\max_\theta . \quad \mathbb{E}_z \rho(s_0, a(z))$$

Sample estimate:

$$\nabla_\theta \frac{1}{N}\sum_{i=1}^{N}\left[\rho\left(a(\theta,z_i),s\right)\right] = \frac{1}{N}\sum_{i=1}^{N}\frac{d\rho\left(a(\theta,z),s\right)}{da}\frac{da(\theta,z)}{d\theta}\Big|_{z=z_i}$$

Re-parametrization for Gaussian



Likelihood ratio grad estimator:

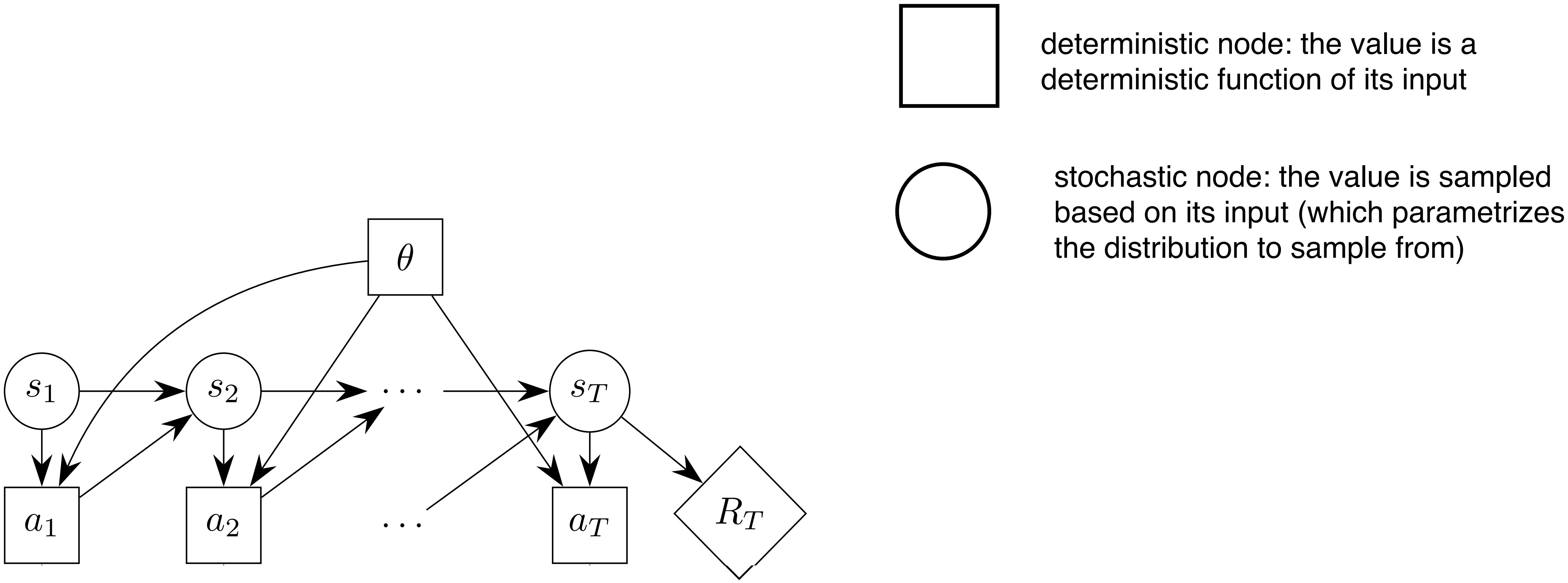
$$\boxed{\mathbb{E}_a \nabla_\theta \log \pi_\theta(s, a) \rho(s, a)}$$

Pathwise derivative:

$$\boxed{\mathbb{E}_z \frac{d\rho\left(a(\theta, z), s\right)}{da} \frac{da(\theta, z)}{d\theta}}$$

The pathwise derivative uses the derivative of the reward w.r.t. the action!

Deep Deterministic Policy Gradients

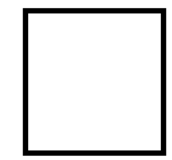


$$a = \pi_\theta(s)$$

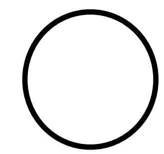
$$\mathbb{E} \sum_t \frac{dQ(s_t, a_t)}{d\theta} = \mathbb{E} \sum_{t=1}^{T} \frac{dQ(s_t, a_t)}{da_t} \frac{da_t}{d\theta}$$

Continuous control with deep reinforcement learning, *Lilicrap et al. 2016*

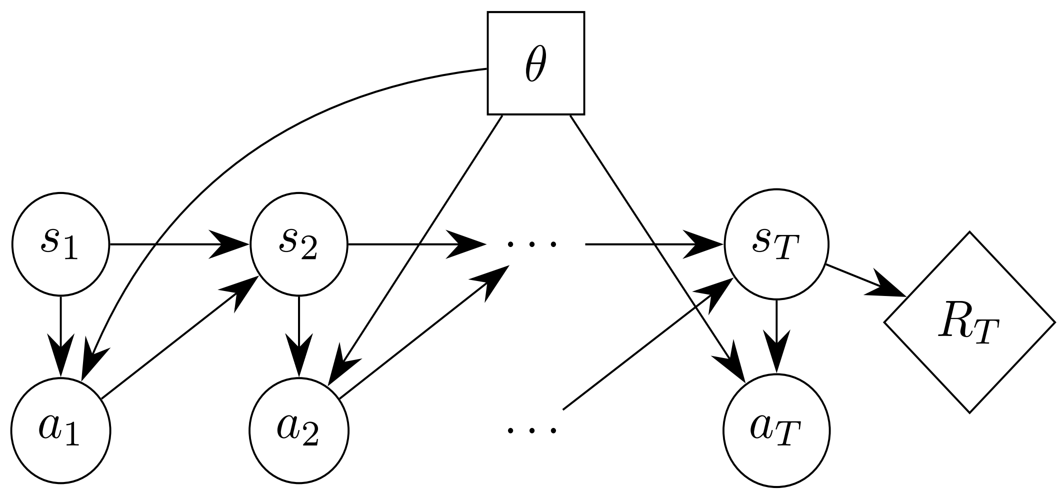
Re-parametrized Policy Gradients



deterministic node: the value is a deterministic function of its input

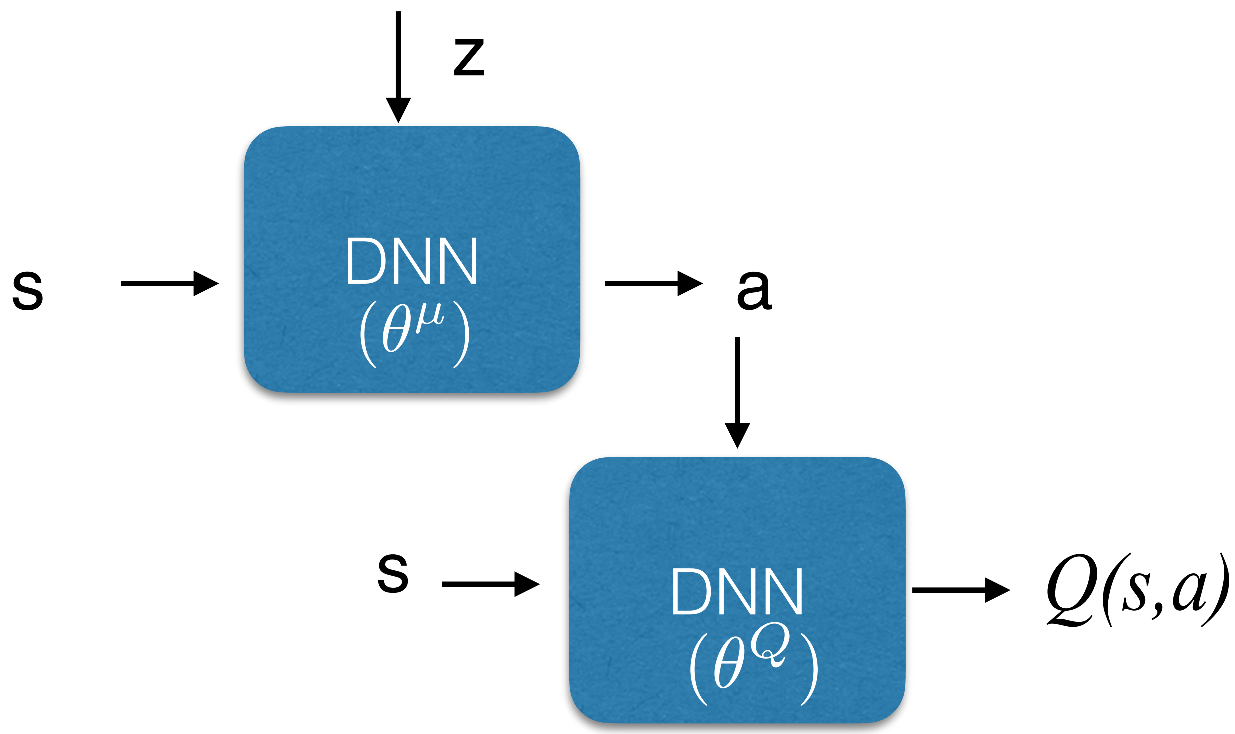


stochastic node: the value is sampled based on its input (which parametrizes the distribution to sample from)



Continuous control with deep reinforcement learning, *Lilicrap et al. 2016*

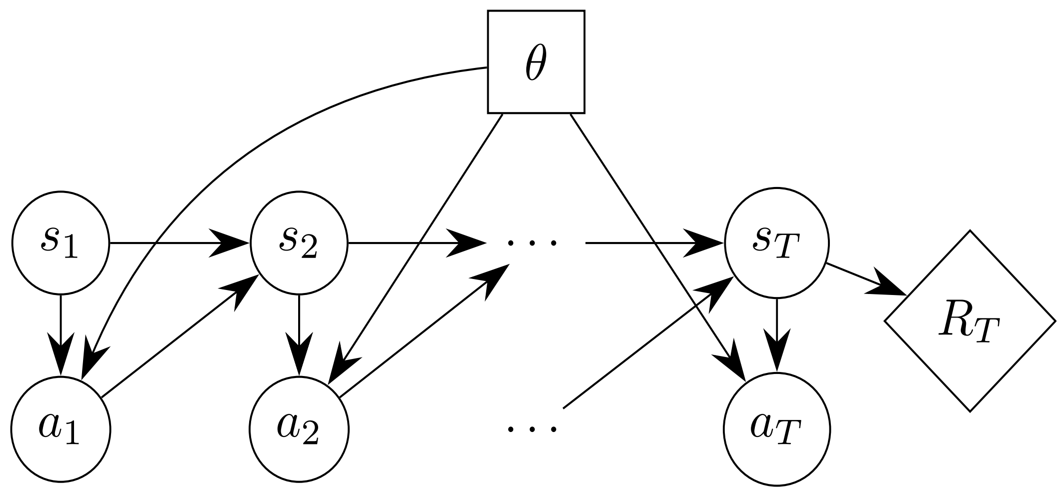
$$z \sim \mathcal{N}(0, 1)$$

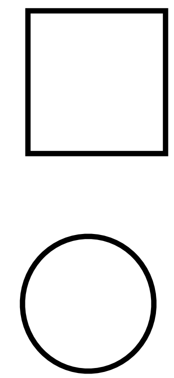


$$a = \mu(s; \theta) + z\sigma(s; \theta)$$

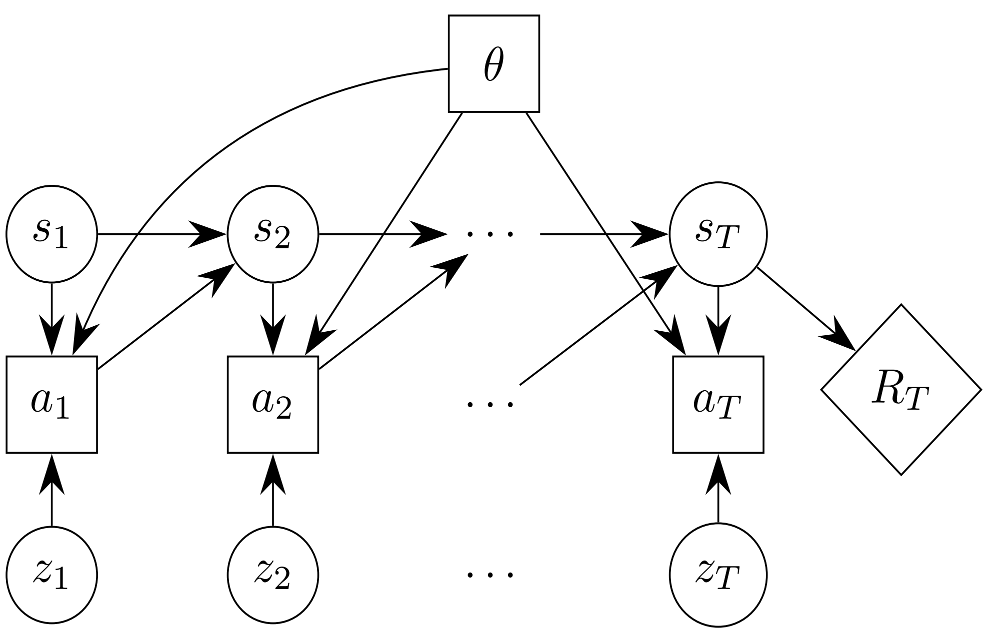
Learning continuous control by stochastic value gradients, Hees et al.

Re-parametrized Policy Gradients





- Reparameterize: $a_t = \pi(s_t, z_t, \theta)$. z_t is noise from fixed distribution



$$a = \mu(s;\theta) + z\sigma(s;\theta)$$

$$\mathbb{E}\sum_t \frac{dQ(s_t, a_t)}{d\theta} = \mathbb{E}\sum_{t=1}^{T} \frac{dQ(s_t, a_t)}{da_t}\frac{da_t}{d\theta} = \mathbb{E}\sum_{t=1}^{T} \frac{dQ(s_t, a_t)}{da_t}\left(\frac{d\mu(s_t;\theta)}{d\theta} + z_t \frac{d\sigma(s_t;\theta)}{d\theta}\right)$$

Stochastic Value Gradients V0

for iteration$=1, 2, \dots$ **do**

Execute policy π_θ to collect T timesteps of data

Update π_θ using $g \propto \nabla_\theta \sum_{t=1}^{T} Q(s_t, \pi(s_t, z_t; \theta))$

Update Q_ϕ using $g \propto \nabla_\phi \sum_{t=1}^{T} (Q_\phi(s_t, a_t) - \hat{Q}_t)^2$, e.g. with TD($\lambda$)

end for

Learning continuous control by stochastic value gradients, Hees et al.

Computing Gradients of Expectations

- When the variable w.r.t. which we are differentiating appears inside the expectation:

$$\nabla_\theta \mathbb{E}_{x\sim P(x)} f(x(\theta)) = \mathbb{E}_{x\sim P(x)} \nabla_\theta f(x(\theta)) = \mathbb{E}_{x\sim P(x)} \frac{df(x(\theta))}{dx}\frac{dx}{d\theta}$$

- When the variable w.r.t. which we are differentiating appears in the distribution: $\nabla_\theta \mathbb{E}_{x\sim P_\theta(x)} f(x)$

Likelihood ratio gradient estimator:

$$\mathbb{E}_{x\sim P_\theta(x)} \nabla_\theta \log P_\theta(x) f(x)$$

Re-parametrized gradient for Gaussian distributions:

$$\nabla_\theta \mathbb{E}_{z\sim\mathcal{N}(0,I)} f(x(z,\theta)) = \mathbb{E}_{z\sim\mathcal{N}(0,I)} \frac{df}{dx}\left(\frac{d\mu(\theta)}{d\theta} + z\frac{d\sigma(\theta)}{d\theta}\right)$$

Carnegie Mellon

School of Computer Science

Deep Reinforcement Learning and Control

Goal Relabeling

Fall 2020, CMU 10-703

Katerina Fragkiadaki

Universal value function Approximators

$$V(s;\theta) \rightarrow V(s,g;\theta)$$

$$\pi(s;\theta) \rightarrow \pi(s,g;\theta)$$

- All methods we have learnt so far can be used.
- At the beginning of an episode, we sample not only a start state but also a goal g, which stays constant throughout the episode
- The experience tuples should contain the goal.

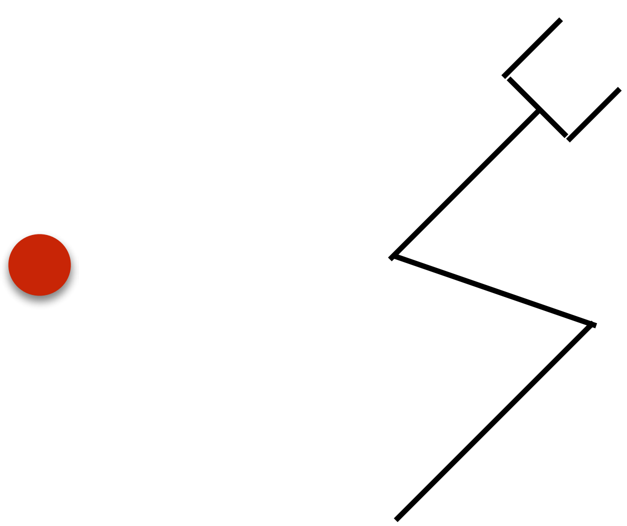
$$(s,a,r,s') \rightarrow \left(s,g,a,r,s'\right)$$

Universal Value Function Approximators, Schaul et al.

Hindsight Experience Replay

Marcin Andrychowicz[*], Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong,
Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel[†], Wojciech Zaremba[†]
OpenAI

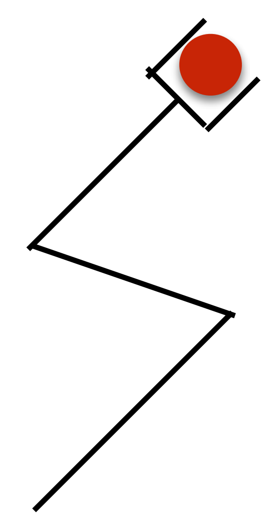
Main idea: use failed executions under one goal g, as successful executions under an alternative goal g' (which is where we ended at the end of the episode).



No reward :-(

Goal g

Our reacher at the end of the episode $(s, g, a, 0, s')$



Goal g'

reward :-)

Our reacher at the end of the episode $(s, g', a, 1, s')$

Hindsight Experience Replay

Algorithm 1 Hindsight Experience Replay (HER)

Given:

- an off-policy RL algorithm $\mathbb{A}$, ▷ e.g. DQN, DDPG, NAF, SDQN
- a strategy $\mathbb{S}$ for sampling goals for replay, ▷ e.g. $\mathbb{S}(s_0, \dots, s_T) = m(s_T)$
- a reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \to \mathbb{R}$. ▷ e.g. $r(s, a, g) = -[f_g(s) = 0]$

Initialize $\mathbb{A}$ ▷ e.g. initialize neural networks
Initialize replay buffer R
for episode $= 1, M$ **do**
  Sample a goal g and an initial state s_0.
  for $t = 0, T - 1$ **do**
    Sample an action a_t using the behavioral policy from $\mathbb{A}$:
      $a_t \leftarrow \pi_b(s_t || g)$ ▷ $||$ denotes concatenation
    Execute the action a_t and observe a new state s_{t+1}
  end for
  for $t = 0, T - 1$ **do**
    $r_t := r(s_t, a_t, g)$
    Store the transition $(s_t || g, a_t, r_t, s_{t+1} || g)$ in R ▷ standard experience replay
    Sample a set of additional goals for replay $G := \mathbb{S}($**current episode**$)$
    for $g' \in G$ **do**
      $r' := r(s_t, a_t, g')$
      Store the transition $(s_t || g', a_t, r', s_{t+1} || g')$ in R ▷ HER
    end for
  end for
  for $t = 1, N$ **do**
    Sample a minibatch B from the replay buffer R
    Perform one step of optimization using $\mathbb{A}$ and minibatch B
  end for
end for

G : the states of the current episode

Usually as additional goal we pick the goal that this episode achieved, and the reward becomes non zero

Hindsight Experience Replay