

Deep Reinforcement Learning and Control

Policy gradients

CMU 10-703

Katerina Fragkiadaki



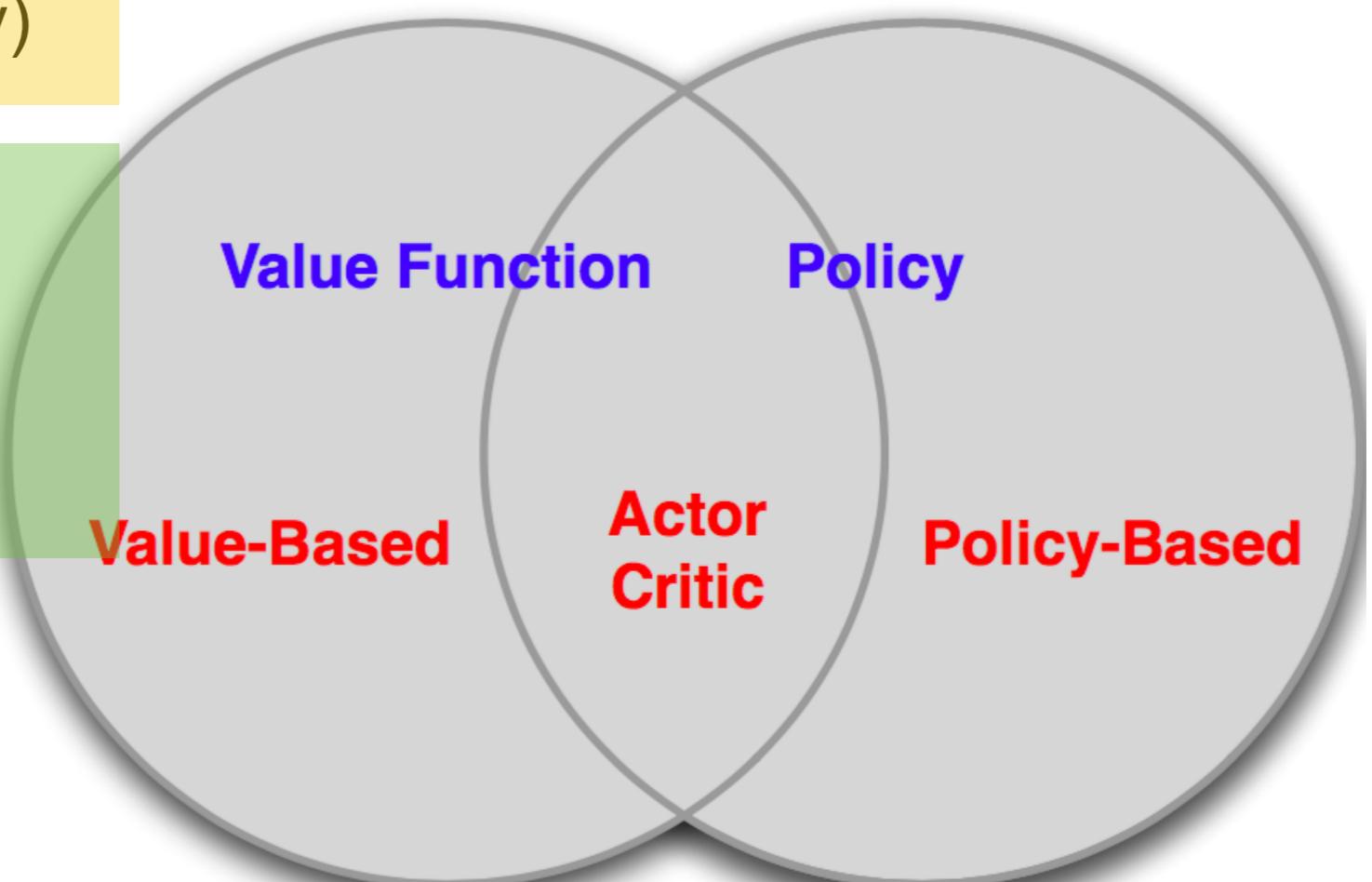
Value-Based and Policy-Based RL

- ▶ Value Based
 - Learned Value Function
 - Implicit policy (e.g. ϵ -greedy)

We have covered those

- ▶ Policy Based
 - No Value Function
 - Learned Policy
- ▶ Actor-Critic
 - Learned Value Function
 - Learned Policy

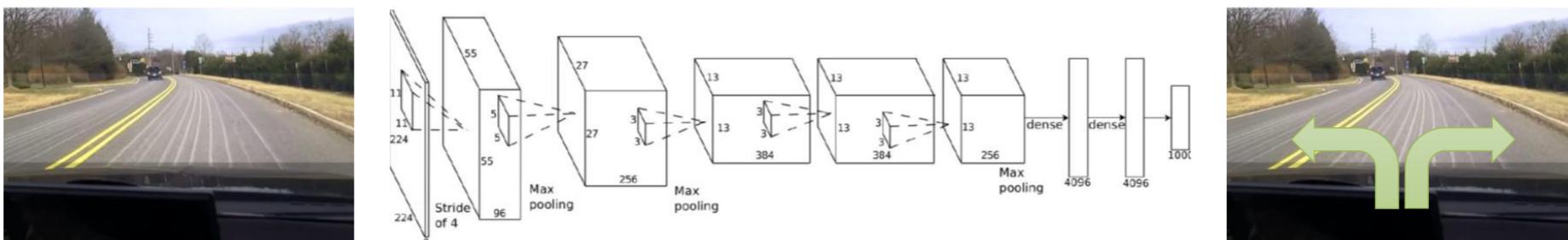
this lecture



Advantages of Policy-Based RL

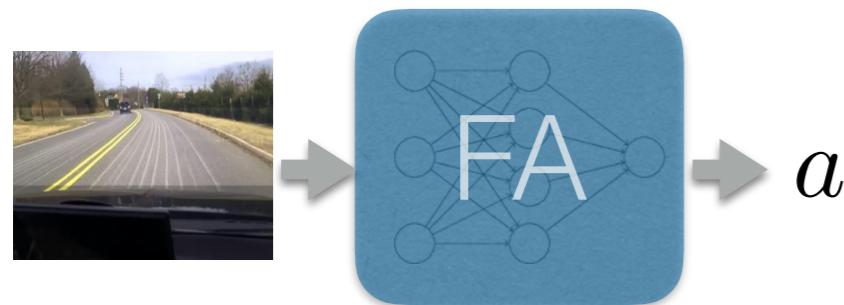
- ▶ Advantages
 - Effective in high-dimensional or **continuous** action spaces
 - Can learn **stochastic** policies

Policy function approximators



Policy function approximators

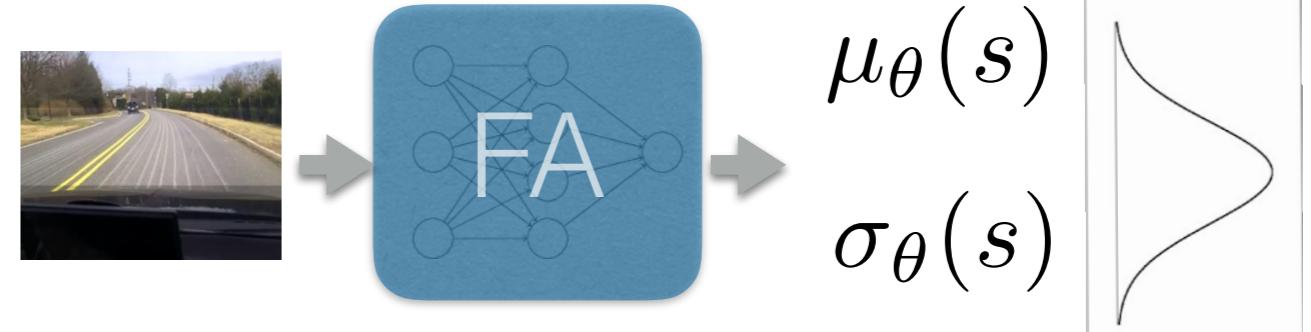
deterministic continuous policy



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

e.g. outputs a steering angle directly

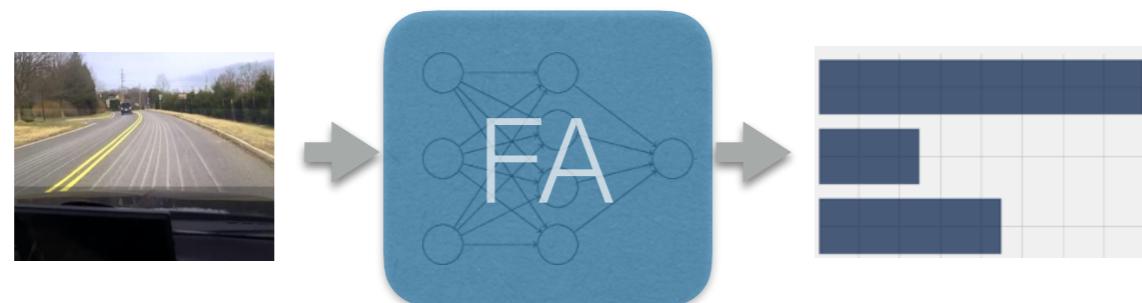
stochastic continuous policy



$$a \sim \mathcal{N}(\mu_\theta(s), \sigma_\theta^2(s))$$

FA for stochastic multimodal continuous policies is an active area of generative model research

(stochastic) policy over discrete actions

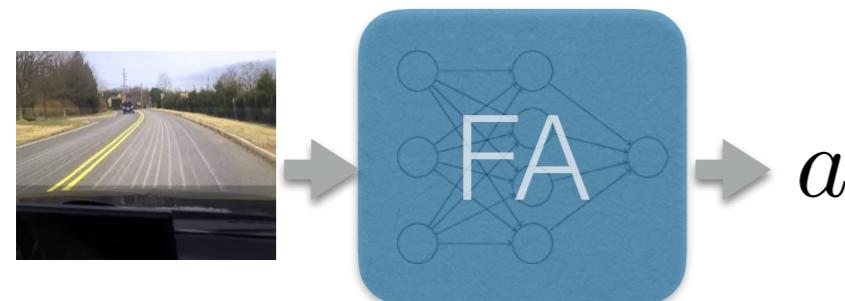


go left
go right
press brake

Outputs a distribution over a discrete set of actions

Policy function approximators - this lecture

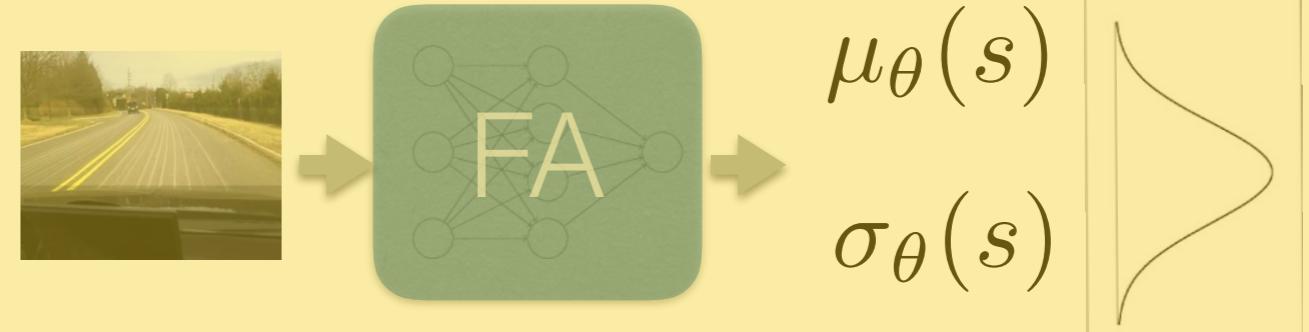
deterministic continuous policy



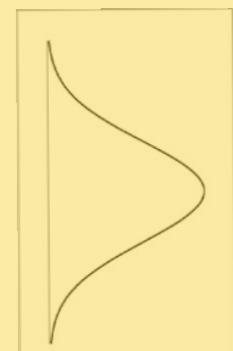
$$a = \pi_\theta(s)$$

e.g. outputs a steering angle directly

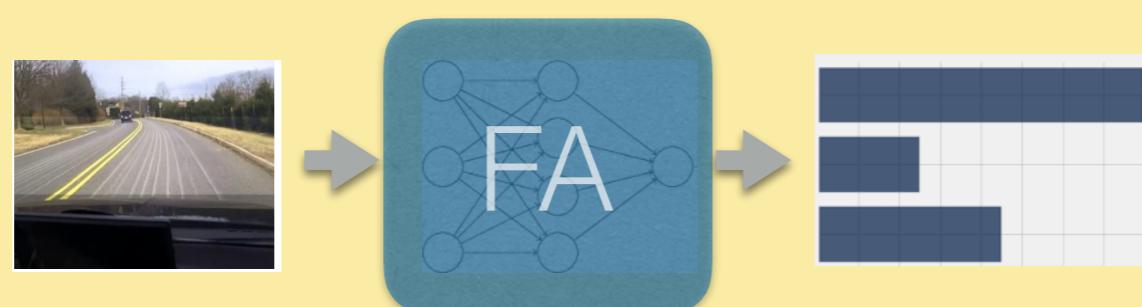
stochastic continuous policy



$$a \sim \mathcal{N}(\mu_\theta(s), \sigma_\theta^2(s))$$



(stochastic) policy over discrete actions



go left
go right
press brake

Outputs a distribution over a discrete set of actions

Policy Optimization

- › Let $U(\theta)$ be any policy **objective function**
- › Policy based reinforcement learning is an optimization problem
 - Find θ that maximizes $U(\theta)$
- › General alternatives: gradient free optimization
 - Hill climbing
 - Genetic algorithms (we have seen this)

Policy Gradient

- Let $U(\theta)$ be any policy **objective function**
- Policy gradient algorithms search for a **local** maximum in $U(\theta)$ by ascending the gradient of the policy, w.r.t. parameters θ

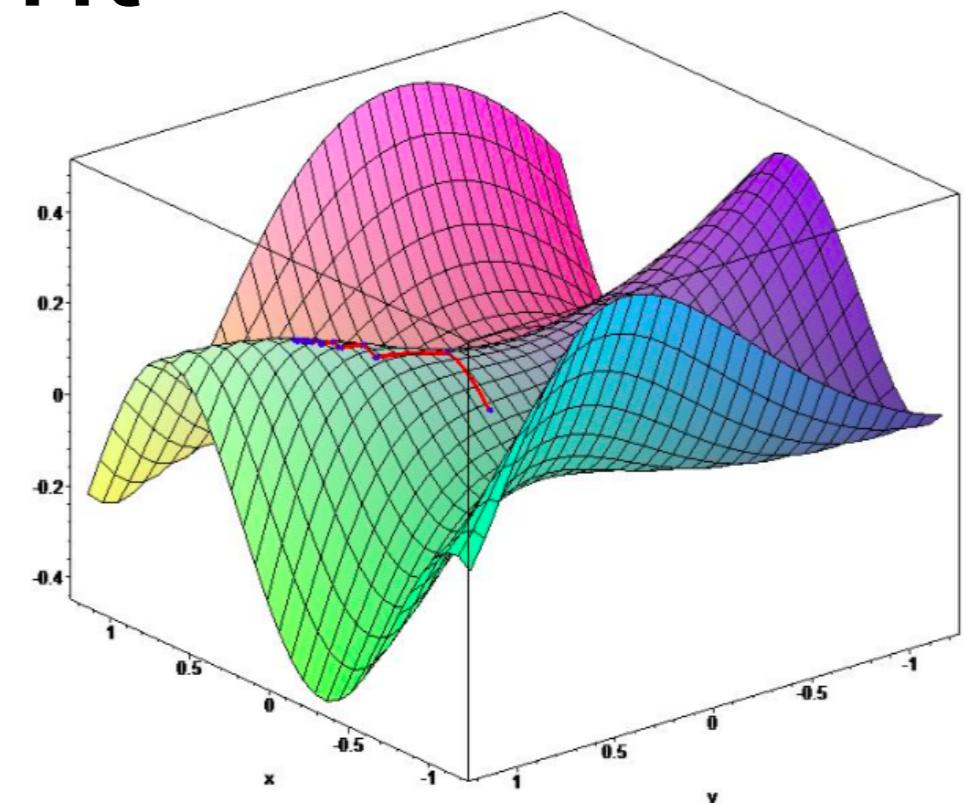
$$\theta_{new} = \theta_{old} + \Delta\theta$$

$$\Delta\theta = \alpha \nabla_{\theta} U(\theta)$$

α is a step-size parameter (learning rate)

is the **policy gradient**

$$\nabla_{\theta} U(\theta) = \begin{pmatrix} \frac{\partial U(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial U(\theta)}{\partial \theta_n} \end{pmatrix}$$



Policy gradient: the gradient of the policy objective w.r.t. the parameters of the policy

Computing Gradients By Finite Differences

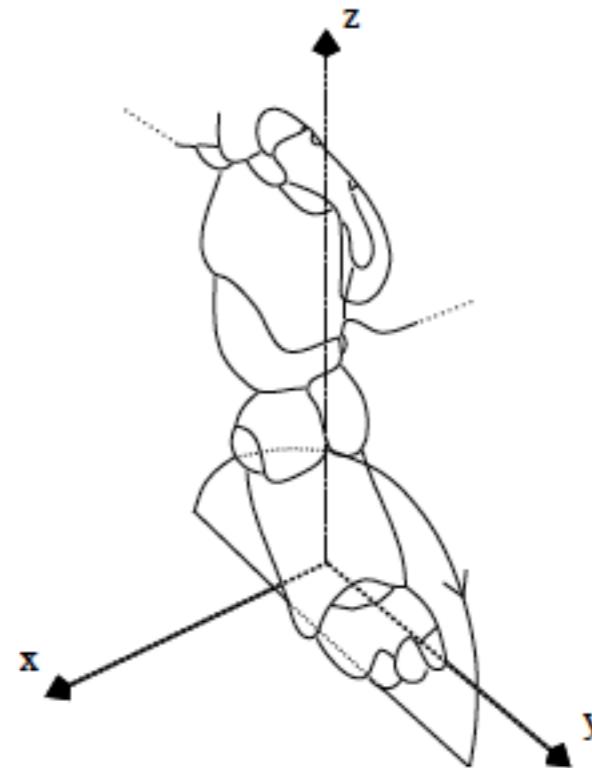
- ▶ Numerically approximating the policy gradient of $\pi_\theta(s, a)$
- ▶ For each dimension k in $[1, n]$
 - Estimate k^{th} partial derivative of objective function w.r.t. θ
 - By perturbing θ by small amount ϵ in k^{th} dimension

$$\frac{\partial U(\theta)}{\partial \theta_k} \approx \frac{U(\theta + \epsilon u_k) - U(\theta)}{\epsilon}$$

where u_k is a unit vector with 1 in k^{th} component, 0 elsewhere

- ▶ Uses n evaluations to compute policy gradient in n dimensions
- ▶ Works for arbitrary policies, even if the policy objective is not differentiable, similar to the evolutionary methods we saw in lecture 1.

Learning an AIBO running policy



- Goal: learn a fast AIBO walk (useful for Robocup)
- AIBO walk policy is controlled by 12 numbers (elliptical loci)
- Adapt these parameters by finite difference policy gradient
- Evaluate performance of policy by field traversal time

Learning an AIBO running policy



```
 $\pi \leftarrow InitialPolicy$ 
while !done do
     $\{R_1, R_2, \dots, R_t\} = t$  random perturbations of  $\pi$ 
    evaluate(  $\{R_1, R_2, \dots, R_t\}$  )
    for  $k = 1$  to  $N$  do
         $Avg_{+\epsilon, k} \leftarrow$  average score for all  $R_i$  that have a positive
            perturbation in dimension  $k$ 
         $Avg_{+0, k} \leftarrow$  average score for all  $R_i$  that have a zero
            perturbation in dimension  $k$ 
         $Avg_{-\epsilon, k} \leftarrow$  average score for all  $R_i$  that have a
            negative perturbation in dimension  $k$ 
        if  $Avg_{+0, k} > Avg_{+\epsilon, k}$  and  $Avg_{+0, k} > Avg_{-\epsilon, k}$  then
             $A_k \leftarrow 0$ 
        else
             $A_k \leftarrow Avg_{+\epsilon, k} - Avg_{-\epsilon, k}$ 
        end if
    end for
     $A \leftarrow \frac{A}{|A|} * \eta$ 
     $\pi \leftarrow \pi + A$ 
end while
```

What is the **score** here?

Learning an AIBO running policy



Initial



Training



Final

What general policy objective to use?

Our objective with value-based methods

- ▶ **Goal:** find parameter vector w minimizing mean-squared error between the true value function $v_\pi(S)$ and its approximation $\hat{v}(S, w)$

$$J(w) = \mathbb{E}_\pi [(v_\pi(S) - \hat{v}(S, w))^2]$$

Let $\mu(S)$ denote how much time we spend in each state s under policy π , then:

$$J(w) = \sum_{n=1}^{|S|} \mu(S) [v_\pi(S) - \hat{v}(S, w)]^2 \quad \sum_{s \in S} \mu(s) = 1$$

Very important choice: it is OK if we cannot learn the value of states we visit very few times, there are too many states, I should focus on the ones that matter: the RL solution to curse of dimensionality.

Our objective with value-based methods

- ▶ **Goal:** find parameter vector w minimizing mean-squared error between the true value function $v_\pi(S)$ and its approximation $\hat{v}(S, w)$

$$J(w) = \mathbb{E}_\pi [(v_\pi(S) - \hat{v}(S, w))^2]$$

Let $\mu(S)$ denote how much time we spend in each state s under policy π , then:

$$J(w) = \sum_{n=1}^{|S|} \mu(S) [v_\pi(S) - \hat{v}(S, w)]^2 \quad \sum_{s \in S} \mu(s) = 1$$

The on policy state distribution changes as we change the policy which changes as we change the value function estimate. But we did not consider this dependence, we did not differentiate $\mu(S)$ w.r.t. w .

Policy objective

Trajectory τ is a state action sequence $s_0, a_0, s_1, a_1, \dots, s_H, a_H$

Trajectory reward: $R(\tau) = \sum_{t=0}^H R(s_t, a_t)$

A reasonable policy objective then is:

$$\max_{\theta} . \quad U(\theta) = \mathbb{E}_{\tau \sim P(\tau; \theta)} [R(\tau)] = \sum_{\tau} P(\tau; \theta) R(\tau)$$

Probability of a trajectory: $P(\tau; \theta) = \prod_{t=0}^H \underbrace{P(s_{t+1} | s_t, a_t)}_{\text{dynamics}} \cdot \underbrace{\pi_{\theta}(a_t | s_t)}_{\text{policy}}$

Our problem is to compute $\nabla_{\theta} U(\theta) = \nabla_{\theta} \mathbb{E}_{\tau \sim P(\tau; \theta)} [R(\tau)]$

This lecture

$$U(\theta) = \mathbb{E}_{\tau \sim P(\tau; \theta)} [R(\tau)]$$

Computing derivatives of expectations w.r.t. variables that parametrize the distribution, not the quantity inside the expectation

$$\max_{\theta} . \mathbb{E}_{x \sim P(x; \theta)} f(x)$$

Assumptions:

- P is a probability density function that is continuous and differentiable
- P is easy to sample from

Derivatives of expectations

$$\nabla_{\theta} \mathbb{E}_x f(x) = \nabla_{\theta} \mathbb{E}_{x \sim P_{\theta}(x)} [f(x)]$$

Derivatives of expectations

$$\begin{aligned}\nabla_{\theta} \mathbb{E}_x f(x) &= \nabla_{\theta} \mathbb{E}_{x \sim P_{\theta}(x)} [f(x)] \\ &= \nabla_{\theta} \sum_x P_{\theta}(x) f(x)\end{aligned}$$

Derivatives of expectations

$$\begin{aligned}\nabla_{\theta} \mathbb{E}_x f(x) &= \nabla_{\theta} \mathbb{E}_{x \sim P_{\theta}(x)} [f(x)] \\&= \nabla_{\theta} \sum_x P_{\theta}(x) f(x) \\&= \sum_x \nabla_{\theta} P_{\theta}(x) f(x)\end{aligned}$$

Why?

Derivatives of expectations

$$\begin{aligned}\nabla_{\theta} \mathbb{E}_x f(x) &= \nabla_{\theta} \mathbb{E}_{x \sim P_{\theta}(x)} [f(x)] \\&= \nabla_{\theta} \sum_x P_{\theta}(x) f(x) \\&= \sum_x \nabla_{\theta} P_{\theta}(x) f(x)\end{aligned}$$

What is the problem here?

Derivatives of expectations

$$\begin{aligned}\nabla_{\theta} \mathbb{E}_x f(x) &= \nabla_{\theta} \mathbb{E}_{x \sim P_{\theta}(x)} [f(x)] \\&= \nabla_{\theta} \sum_x P_{\theta}(x) f(x) \\&= \sum_x \nabla_{\theta} P_{\theta}(x) f(x) \\&= \sum_x P_{\theta}(x) \frac{\nabla_{\theta} P_{\theta}(x)}{P_{\theta}(x)} f(x)\end{aligned}$$

Derivatives of expectations

$$\begin{aligned}\nabla_{\theta} \mathbb{E}_x f(x) &= \nabla_{\theta} \mathbb{E}_{x \sim P_{\theta}(x)} [f(x)] \\&= \nabla_{\theta} \sum_x P_{\theta}(x) f(x) \\&= \sum_x \nabla_{\theta} P_{\theta}(x) f(x) \\&= \sum_x P_{\theta}(x) \frac{\nabla_{\theta} P_{\theta}(x)}{P_{\theta}(x)} f(x) \\&= \sum_x P_{\theta}(x) \nabla_{\theta} \log P_{\theta}(x) f(x)\end{aligned}$$

Derivatives of expectations

$$\begin{aligned}\nabla_{\theta} \mathbb{E}_x f(x) &= \nabla_{\theta} \mathbb{E}_{x \sim P_{\theta}(x)} [f(x)] \\&= \nabla_{\theta} \sum_x P_{\theta}(x) f(x) \\&= \sum_x \nabla_{\theta} P_{\theta}(x) f(x) \\&= \sum_x P_{\theta}(x) \frac{\nabla_{\theta} P_{\theta}(x)}{P_{\theta}(x)} f(x) \\&= \sum_x P_{\theta}(x) \nabla_{\theta} \log P_{\theta}(x) f(x) \\&= \mathbb{E}_{x \sim P_{\theta}(x)} [\nabla_{\theta} \log P_{\theta}(x) f(x)]\end{aligned}$$

What have we achieved?

Derivatives of expectations

$$\begin{aligned}\nabla_{\theta} \mathbb{E}_x f(x) &= \nabla_{\theta} \mathbb{E}_{x \sim P_{\theta}(x)} [f(x)] \\&= \nabla_{\theta} \sum_x P_{\theta}(x) f(x) \\&= \sum_x \nabla_{\theta} P_{\theta}(x) f(x) \\&= \sum_x P_{\theta}(x) \frac{\nabla_{\theta} P_{\theta}(x)}{P_{\theta}(x)} f(x) \\&= \sum_x P_{\theta}(x) \nabla_{\theta} \log P_{\theta}(x) f(x) \\&= \mathbb{E}_{x \sim P_{\theta}(x)} [\nabla_{\theta} \log P_{\theta}(x) f(x)]\end{aligned}$$

I can obtain an unbiased estimator for the gradient $\nabla_{\theta} \mathbb{E}_x f(x)$ by sampling!

From the law of large numbers, it will converge to the right gradient with infinite number of samples

Derivatives of expectations

$$\begin{aligned}\nabla_{\theta} \mathbb{E}_x f(x) &= \nabla_{\theta} \mathbb{E}_{x \sim P_{\theta}(x)} [f(x)] \\&= \nabla_{\theta} \sum_x P_{\theta}(x) f(x) \\&= \sum_x \nabla_{\theta} P_{\theta}(x) f(x) \\&= \sum_x P_{\theta}(x) \frac{\nabla_{\theta} P_{\theta}(x)}{P_{\theta}(x)} f(x) \\&= \sum_x P_{\theta}(x) \nabla_{\theta} \log P_{\theta}(x) f(x) \\&= \mathbb{E}_{x \sim P_{\theta}(x)} [\nabla_{\theta} \log P_{\theta}(x) f(x)] \\&\approx \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log P_{\theta}(x^{(i)}) f(x^{(i)})\end{aligned}$$

I can obtain an unbiased estimator for the gradient $\nabla_{\theta} \mathbb{E}_x f(x)$ by sampling!

Gaussian Policy: Continuous Actions

- ▶ In continuous action spaces, a Gaussian policy is natural
- ▶ Mean is a linear combination of state features

$$\mu(s) = \phi(s)^\top \theta$$



Nonlinear extension: replace $\phi(s)$ with a deep neural network with trainable weights w

- ▶ Variance may be fixed σ_2 , or can also parameterized

$$\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- ▶ Policy is Gaussian $a \sim \mathcal{N}(\mu(s), \sigma^2)$

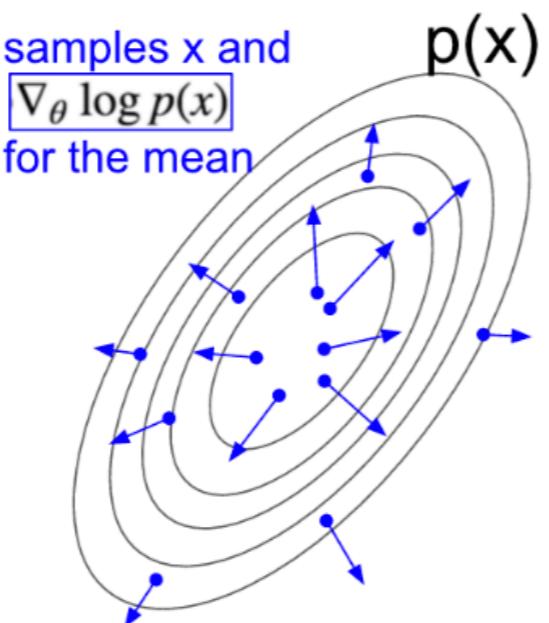
- ▶ The score function is

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

Derivatives of expectations

$$\begin{aligned}\nabla_{\theta} \mathbb{E}_x f(x) &= \nabla_{\theta} \mathbb{E}_{x \sim P_{\theta}(x)} [f(x)] \\&= \nabla_{\theta} \sum_x P_{\theta}(x) f(x) \\&= \sum_x \nabla_{\theta} P_{\theta}(x) f(x) \\&= \sum_x P_{\theta}(x) \frac{\nabla_{\theta} P_{\theta}(x)}{P_{\theta}(x)} f(x) \\&= \sum_x P_{\theta}(x) \nabla_{\theta} \log P_{\theta}(x) f(x) \\&= \mathbb{E}_{x \sim P_{\theta}(x)} [\nabla_{\theta} \log P_{\theta}(x) f(x)] \\&\approx \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log P_{\theta}(x^{(i)}) f(x^{(i)})\end{aligned}$$

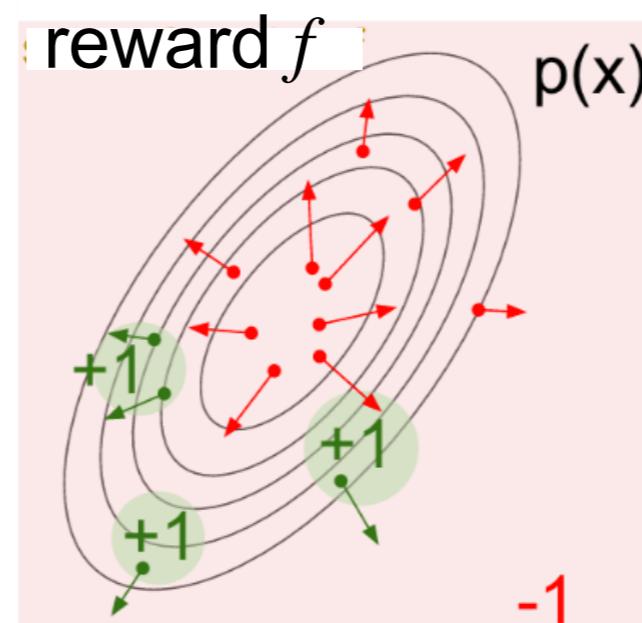
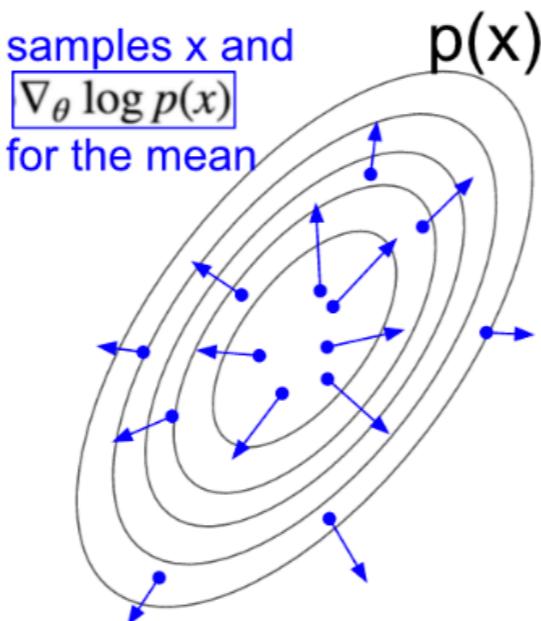
For Gaussian $P_{\theta}(x)$



Derivatives of expectations

$$\begin{aligned}\nabla_{\theta} \mathbb{E}_x f(x) &= \nabla_{\theta} \mathbb{E}_{x \sim P_{\theta}(x)} [f(x)] \\&= \nabla_{\theta} \sum_x P_{\theta}(x) f(x) \\&= \sum_x \nabla_{\theta} P_{\theta}(x) f(x) \\&= \sum_x P_{\theta}(x) \frac{\nabla_{\theta} P_{\theta}(x)}{P_{\theta}(x)} f(x) \\&= \sum_x P_{\theta}(x) \nabla_{\theta} \log P_{\theta}(x) f(x) \\&= \mathbb{E}_{x \sim P_{\theta}(x)} [\nabla_{\theta} \log P_{\theta}(x) f(x)] \\&\approx \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log P_{\theta}(x^{(i)}) f(x^{(i)})\end{aligned}$$

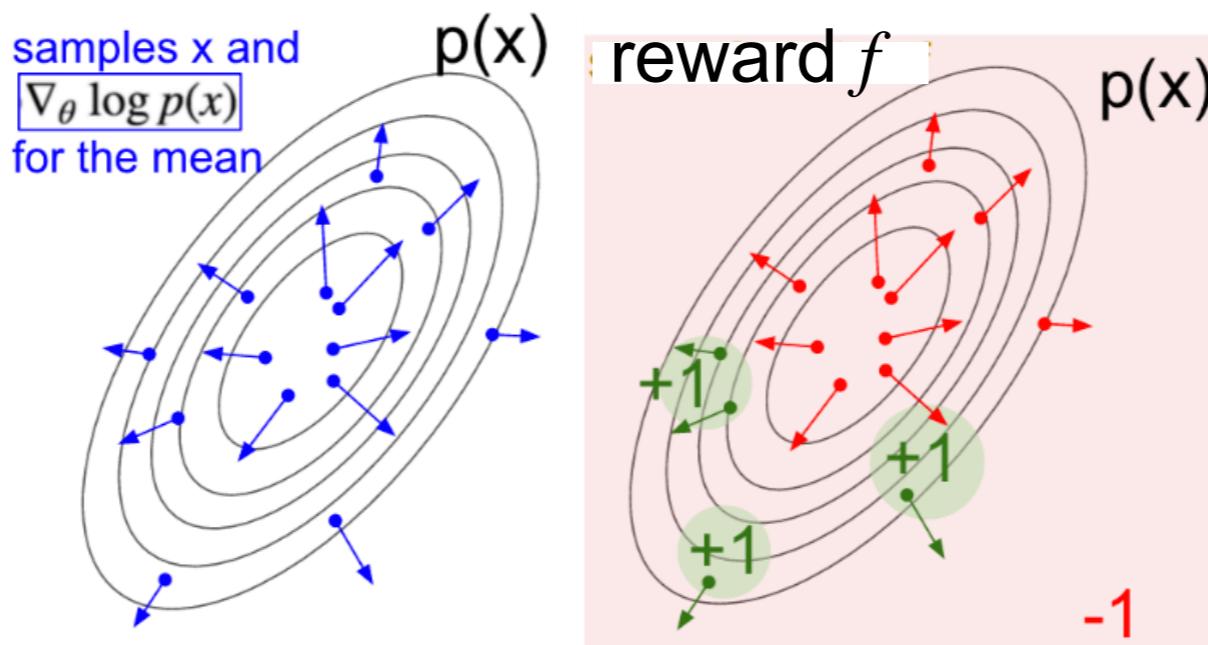
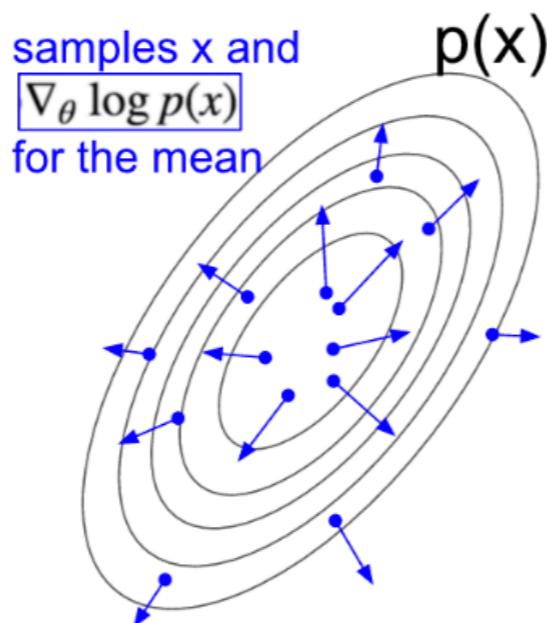
For Gaussian $P_{\theta}(x)$
What is θ here?



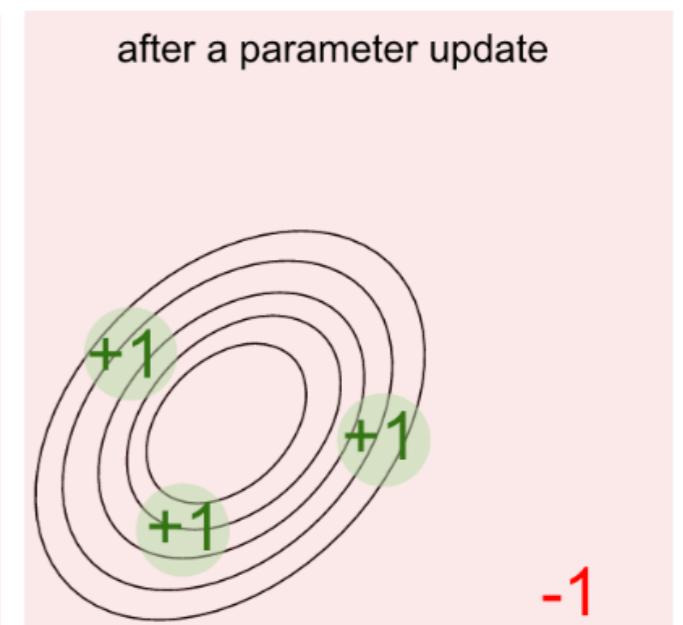
Derivatives of expectations

$$\begin{aligned}\nabla_{\theta} \mathbb{E}_x f(x) &= \nabla_{\theta} \mathbb{E}_{x \sim P_{\theta}(x)} [f(x)] \\&= \nabla_{\theta} \sum_x P_{\theta}(x) f(x) \\&= \sum_x \nabla_{\theta} P_{\theta}(x) f(x) \\&= \sum_x P_{\theta}(x) \frac{\nabla_{\theta} P_{\theta}(x)}{P_{\theta}(x)} f(x) \\&= \sum_x P_{\theta}(x) \nabla_{\theta} \log P_{\theta}(x) f(x) \\&= \mathbb{E}_{x \sim P_{\theta}(x)} [\nabla_{\theta} \log P_{\theta}(x) f(x)] \\&\approx \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log P_{\theta}(x^{(i)}) f(x^{(i)})\end{aligned}$$

For Gaussian $P_{\theta}(x)$



after a parameter update



Derivatives of the policy objective

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

Derivatives of the policy objective

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

$$\begin{aligned}\nabla_{\theta} U(\theta) &= \nabla_{\theta} \mathbb{E}_{\tau \sim P_{\theta}(\tau)} [R(\tau)] \\ &= \nabla_{\theta} \sum_{\tau} P_{\theta}(\tau) R(\tau) \\ &= \sum_{\tau} \nabla_{\theta} P_{\theta}(\tau) R(\tau)\end{aligned}$$

Derivatives of the policy objective

$$\max_{\theta} . \quad U(\theta) = \mathbb{E}_{\tau \sim P_{\theta}(\tau)} [R(\tau)]$$

$$\begin{aligned}\nabla_{\theta} U(\theta) &= \nabla_{\theta} \mathbb{E}_{\tau \sim P_{\theta}(\tau)} [R(\tau)] \\ &= \nabla_{\theta} \sum_{\tau} P_{\theta}(\tau) R(\tau) \\ &= \sum_{\tau} \nabla_{\theta} P_{\theta}(\tau) R(\tau) \\ &= \sum_{\tau} P_{\theta}(\tau) \frac{\nabla_{\theta} P_{\theta}(\tau)}{P_{\theta}(\tau)} R(\tau) \\ &= \sum_{\tau} P_{\theta}(\tau) \nabla_{\theta} \log P_{\theta}(\tau) R(\tau) \\ &= \mathbb{E}_{\tau \sim P_{\theta}(\tau)} [\nabla_{\theta} \log P_{\theta}(\tau) R(\tau)]\end{aligned}$$

Derivatives of the policy objective

$$\max_{\theta} . \quad U(\theta) = \mathbb{E}_{\tau \sim P_{\theta}(\tau)} [R(\tau)]$$

$$\begin{aligned}\nabla_{\theta} U(\theta) &= \nabla_{\theta} \mathbb{E}_{\tau \sim P_{\theta}(\tau)} [R(\tau)] \\ &= \nabla_{\theta} \sum_{\tau} P_{\theta}(\tau) R(\tau) \\ &= \sum_{\tau} \nabla_{\theta} P_{\theta}(\tau) R(\tau) \\ &= \sum_{\tau} P_{\theta}(\tau) \frac{\nabla_{\mu} P_{\theta}(\tau)}{P_{\theta}(\tau)} R(\tau) \\ &= \sum_{\tau} P_{\theta}(\tau) \nabla_{\theta} \log P_{\theta}(\tau) R(\tau) \\ &= \mathbb{E}_{\tau \sim P_{\theta}(\tau)} [\nabla_{\theta} \log P_{\theta}(\tau) R(\tau)]\end{aligned}$$

Approximate the gradient with empirical estimate from N sampled trajectories:

$$\nabla_{\theta} U(\theta) \approx \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log P_{\theta}(\tau^{(i)}) R(\tau^{(i)})$$

From trajectories to actions

$$\nabla_{\theta} \log P(\tau^{(i)}; \theta) = \nabla_{\theta} \log \left[\underbrace{\prod_{t=0}^T P(s_{t+1}^{(i)} | s_t^{(i)}, a_t^{(i)})}_{\text{dynamics}} \cdot \underbrace{\pi_{\theta}(a_t^{(i)} | s_t^{(i)})}_{\text{policy}} \right]$$

$$= \nabla_{\theta} \left[\underbrace{\sum_{t=0}^T \log P(s_{t+1}^{(i)} | s_t^{(i)}, a_t^{(i)})}_{\text{dynamics}} + \underbrace{\log \pi_{\theta}(a_t^{(i)} | s_t^{(i)})}_{\text{policy}} \right]$$

$$= \nabla_{\theta} \left[\sum_{t=0}^T \underbrace{\log \pi_{\theta}(a_t^{(i)} | s_t^{(i)})}_{\text{policy}} \right]$$

$$= \sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)})$$

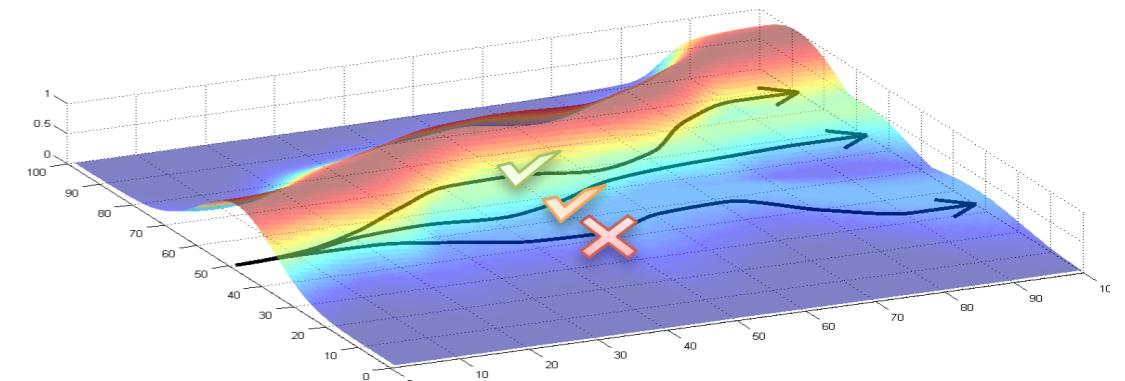
$$\nabla_{\theta} U(\theta) \approx \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log P_{\theta}(\tau^{(i)}) R(\tau^{(i)}) \quad \rightarrow \quad \nabla_{\theta} U(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)}) R(\tau^{(i)})$$

Intuition

$$\nabla_{\theta} U(\theta) \approx \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log P_{\theta}(\tau^{(i)}) R(\tau^{(i)})$$

- Gradient tries to:

- Increase probability of paths with positive R
- Decrease probability of paths with negative R



! Likelihood ratio changes probabilities of experienced paths,
does not try to change the paths (<-> Path Derivative)

Policy Gradients

$$\begin{aligned} \max_{\theta} . \quad U(\theta) &= \mathbb{E}_{\tau \sim P_{\theta}(\tau)} [R(\tau)] \\ \nabla_{\theta} U(\theta) &= \nabla_{\theta} \mathbb{E}_{\tau \sim P_{\theta}(\tau)} [R(\tau)] \\ &= \nabla_{\theta} \sum_{\tau} P_{\theta}(\tau) R(\tau) \\ &= \sum_{\tau} \nabla_{\theta} P_{\theta}(\tau) R(\tau) \\ &= \sum_{\tau} P_{\theta}(\tau) \frac{\nabla_{\theta} P_{\theta}(\tau)}{P_{\theta}(\tau)} R(\tau) \\ &= \sum_{\tau} P_{\theta}(\tau) \nabla_{\theta} \log P_{\theta}(\tau) R(\tau) \\ &= \mathbb{E}_{\tau \sim P_{\theta}(\tau)} [\nabla_{\theta} \log P_{\theta}(\tau) R(\tau)] \end{aligned}$$

Sample estimate:

$$\nabla_{\theta} U(\theta) \approx \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log P_{\theta}(\tau^{(i)}) R(\tau^{(i)})$$

Evolutionary Methods

$$\max_{\mu} . \ U(\mu) = \mathbb{E}_{\theta \sim P_\mu(\theta)} [F(\theta)]$$

$$\begin{aligned}\nabla_{\mu} U(\mu) &= \nabla_{\mu} \mathbb{E}_{\theta \sim P_\mu(\theta)} [F(\theta)] \\ &= \nabla_{\mu} \int P_\mu(\theta) F(\theta) d\theta \\ &= \int \nabla_{\mu} P_\mu(\theta) F(\theta) d\theta \\ &= \int P_\mu(\theta) \frac{\nabla_{\mu} P_\mu(\theta)}{P_\mu(\theta)} F(\theta) d\theta \\ &= \int P_\mu(\theta) \nabla_{\mu} \log P_\mu(\theta) F(\theta) d\theta \\ &= \mathbb{E}_{\theta \sim P_\mu(\theta)} [\nabla_{\mu} \log P_\mu(\theta) F(\theta)]\end{aligned}$$

Sample estimate:

$$\nabla_{\mu} U(\mu) \approx \frac{1}{N} \sum_{i=1}^N \nabla_{\mu} \log P_\mu(\theta^{(i)}) F(\theta^{(i)})$$

Policy gradients VS Evolutionary methods

Considers distribution over actions

$$\max_{\theta} . \quad U(\theta) = \mathbb{E}_{\tau \sim P_{\theta}(\tau)} [R(\tau)]$$

$$\begin{aligned}\nabla_{\theta} U(\theta) &= \nabla_{\theta} \mathbb{E}_{\tau \sim P_{\theta}(\tau)} [R(\tau)] \\ &= \nabla_{\theta} \sum_{\tau} P_{\theta}(\tau) R(\tau) \\ &= \sum_{\tau} \nabla_{\theta} P_{\theta}(\tau) R(\tau) \\ &= \sum_{\tau} P_{\theta}(\tau) \frac{\nabla_{\theta} P_{\theta}(\tau)}{P_{\theta}(\tau)} R(\tau) \\ &= \sum_{\tau} P_{\theta}(\tau) \nabla_{\theta} \log P_{\theta}(\tau) R(\tau) \\ &= \mathbb{E}_{\tau \sim P_{\theta}(\tau)} [\nabla_{\theta} \log P_{\theta}(\tau) R(\tau)]\end{aligned}$$

Sample estimate:

$$\nabla_{\theta} U(\theta) \approx \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log P_{\theta}(\tau^{(i)}) R(\tau^{(i)})$$

Considers distribution over policy parameters

$$\max_{\mu} . \quad U(\mu) = \mathbb{E}_{\theta \sim P_{\mu}(\theta)} [F(\theta)]$$

$$\begin{aligned}\nabla_{\mu} U(\mu) &= \nabla_{\mu} \mathbb{E}_{\theta \sim P_{\mu}(\theta)} [F(\theta)] \\ &= \nabla_{\mu} \int P_{\mu}(\theta) F(\theta) d\theta \\ &= \int \nabla_{\mu} P_{\mu}(\theta) F(\theta) d\theta \\ &= \int P_{\mu}(\theta) \frac{\nabla_{\mu} P_{\mu}(\theta)}{P_{\mu}(\theta)} F(\theta) d\theta \\ &= \int P_{\mu}(\theta) \nabla_{\mu} \log P_{\mu}(\theta) F(\theta) d\theta \\ &= \mathbb{E}_{\theta \sim P_{\mu}(\theta)} [\nabla_{\mu} \log P_{\mu}(\theta) F(\theta)]\end{aligned}$$

Sample estimate:

$$\nabla_{\mu} U(\mu) \approx \frac{1}{N} \sum_{i=1}^N \nabla_{\mu} \log P_{\mu}(\theta^{(i)}) F(\theta^{(i)})$$

Policy gradients VS Evolutionary methods

We are sampling in both cases...

- PG: sampling in action space
- ES: sampling in parameter space

Policy gradients VS Evolutionary methods

Considers distribution over actions

$$\max_{\theta} . \quad U(\theta) = \mathbb{E}_{\tau \sim P_{\theta}(\tau)} [R(\tau)]$$

$$\begin{aligned}\nabla_{\theta} U(\theta) &= \nabla_{\theta} \mathbb{E}_{\tau \sim P_{\theta}(\tau)} [R(\tau)] \\ &= \nabla_{\theta} \sum_{\tau} P_{\theta}(\tau) R(\tau) \\ &= \sum_{\tau} \nabla_{\theta} P_{\theta}(\tau) R(\tau) \\ &= \sum_{\tau} P_{\theta}(\tau) \frac{\nabla_{\theta} P_{\theta}(\tau)}{P_{\theta}(\tau)} R(\tau) \\ &= \sum_{\tau} P_{\theta}(\tau) \nabla_{\theta} \log P_{\theta}(\tau) R(\tau) \\ &= \mathbb{E}_{\tau \sim P_{\theta}(\tau)} [\nabla_{\theta} \log P_{\theta}(\tau) R(\tau)]\end{aligned}$$

Sample estimate:

$$\nabla_{\theta} U(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)}) R(s_t^{(i)}, a_t^{(i)})$$

Considers distribution over policy parameters

$$\max_{\mu} . \quad U(\mu) = \mathbb{E}_{\theta \sim P_{\mu}(\theta)} [F(\theta)]$$

$$\begin{aligned}\nabla_{\mu} U(\mu) &= \nabla_{\mu} \mathbb{E}_{\theta \sim P_{\mu}(\theta)} [F(\theta)] \\ &= \nabla_{\mu} \int P_{\mu}(\theta) F(\theta) d\theta \\ &= \int \nabla_{\mu} P_{\mu}(\theta) F(\theta) d\theta \\ &= \int P_{\mu}(\theta) \frac{\nabla_{\mu} P_{\mu}(\theta)}{P_{\mu}(\theta)} F(\theta) d\theta \\ &= \int P_{\mu}(\theta) \nabla_{\mu} \log P_{\mu}(\theta) F(\theta) d\theta \\ &= \mathbb{E}_{\theta \sim P_{\mu}(\theta)} [\nabla_{\mu} \log P_{\mu}(\theta) F(\theta)]\end{aligned}$$

Sample estimate:

$$\nabla_{\mu} U(\mu) \approx \frac{1}{N} \sum_{i=1}^N \nabla_{\mu} \log P_{\mu}(\theta^{(i)}) F(\theta^{(i)})$$

Pong from Pixels

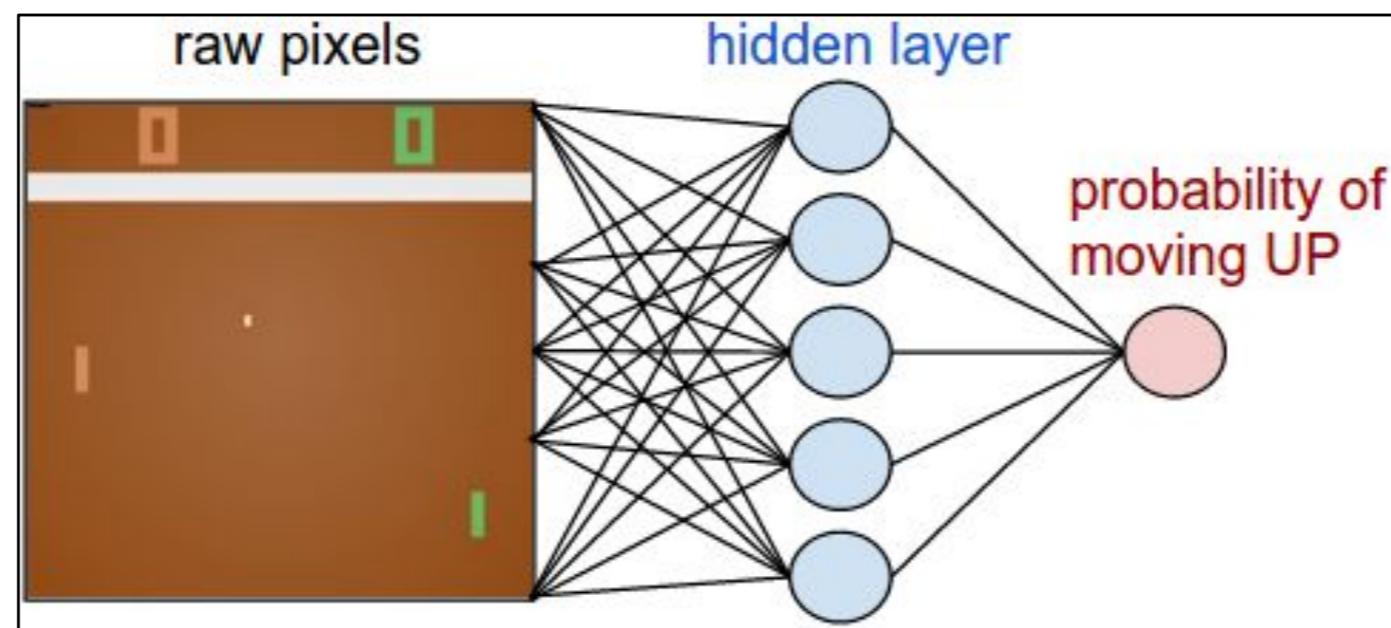
Slides from Andrei Karpathy

|



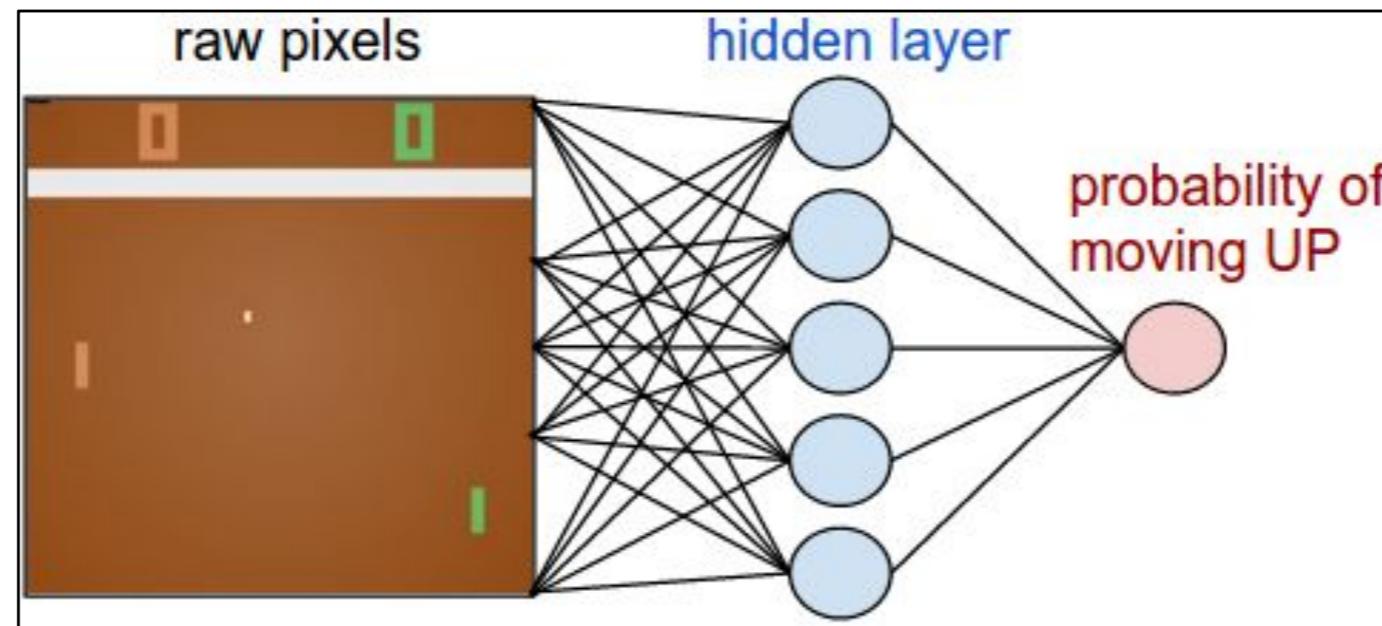
|

Policy network



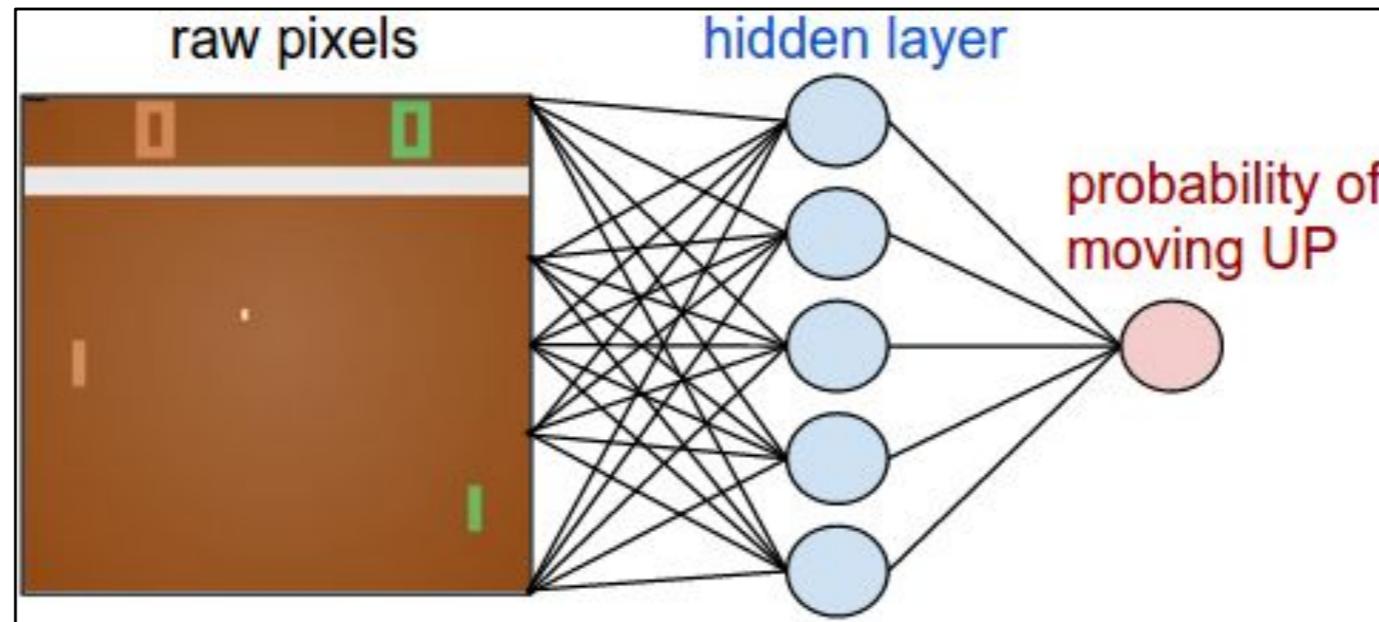
Policy network

e.g.,
height width
[80 x 80]
array of



Policy network

height width
[80 x 80]
array

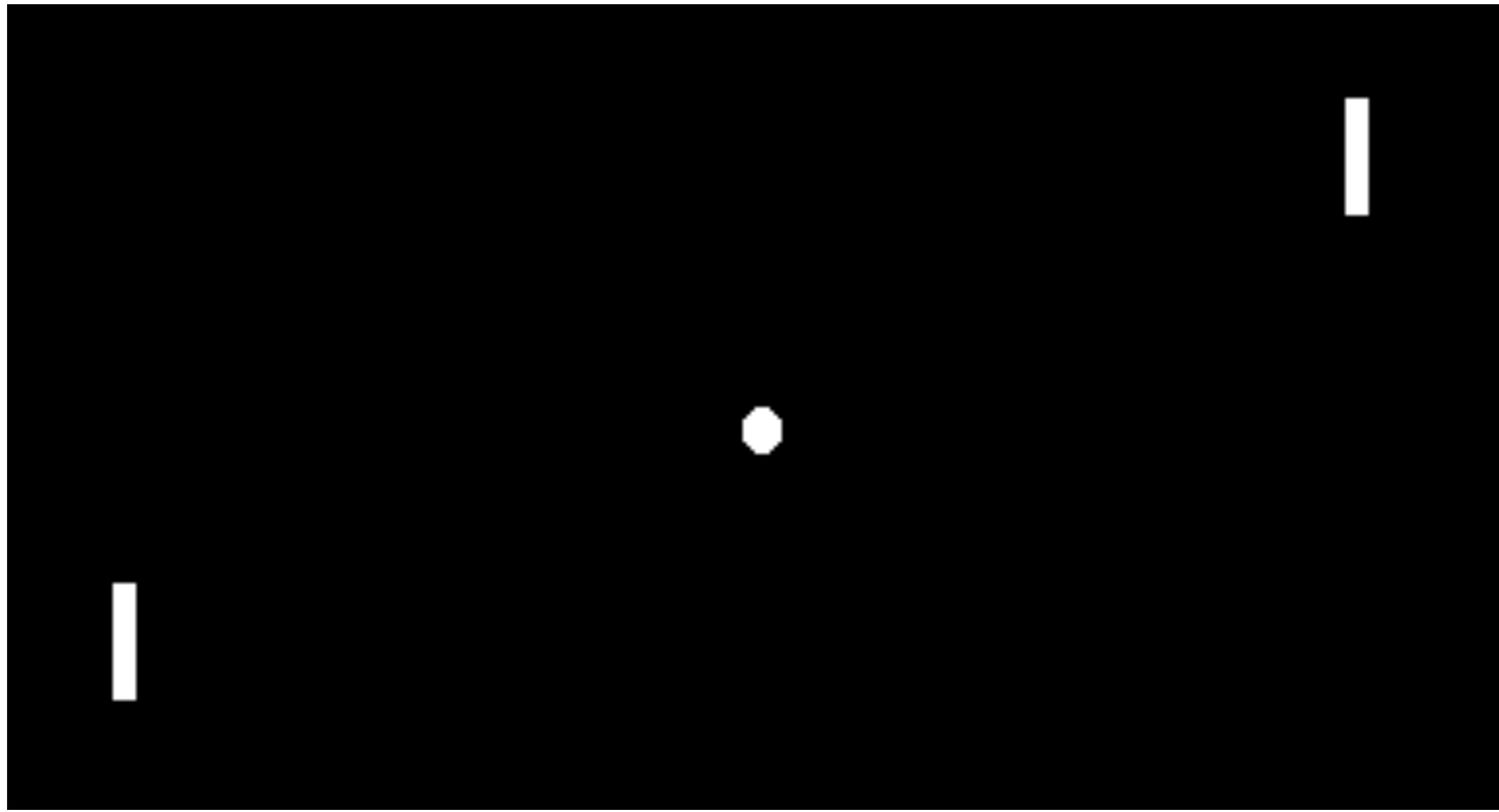


E.g. 200 nodes in the hidden network, so:

$$[(80*80)*200 + 200] + [200*1 + 1] = \sim 1.3M \text{ parameters}$$

Layer 1

Layer 2



Network does not see this. Network sees $80 \times 80 = 6,400$ numbers.
It gets a reward of +1 or -1, some of the time.
Q: How do we efficiently find a good setting of the 1.3M parameters?

Random search

Evolutionary methods

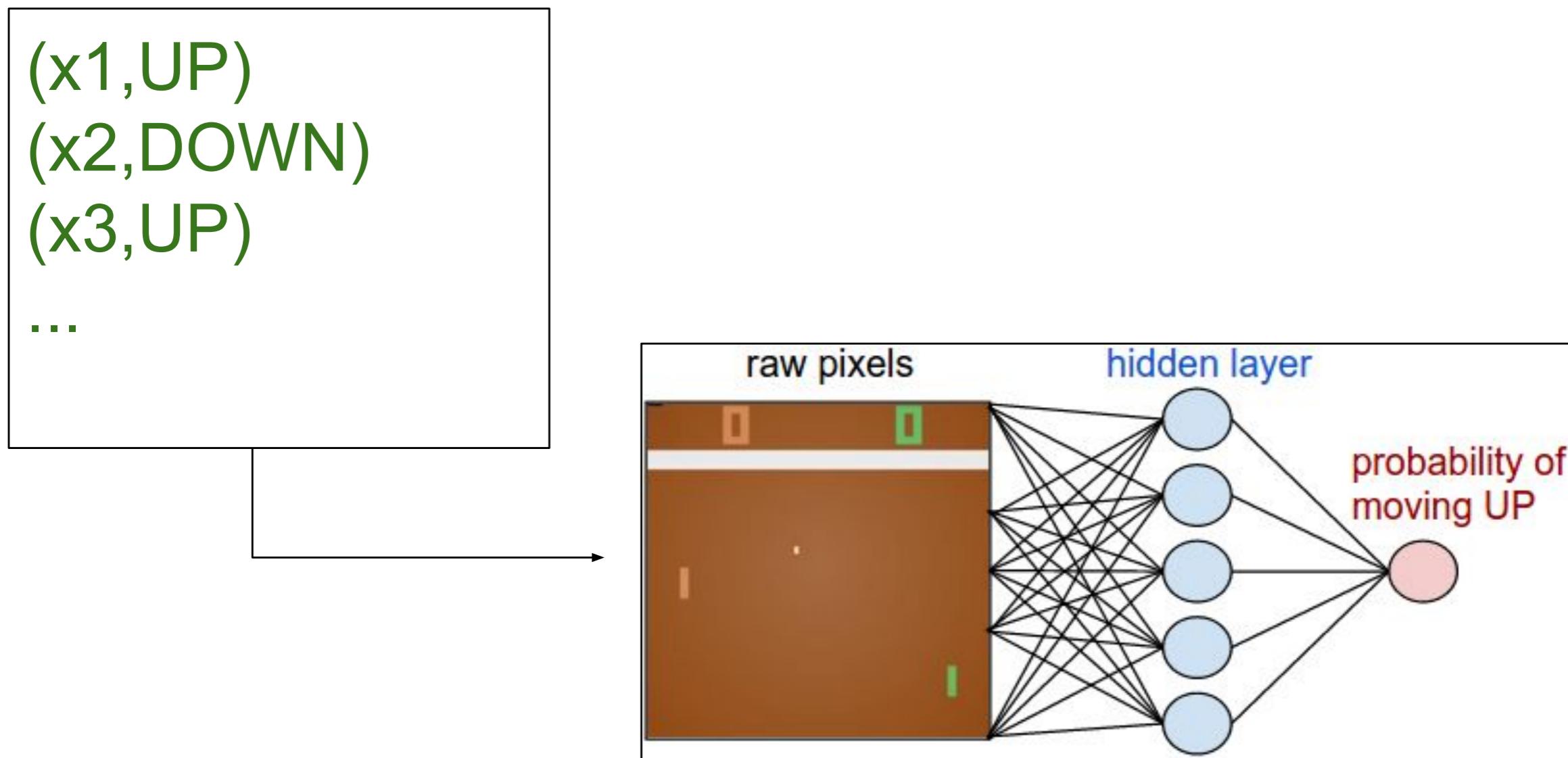
Approximation to the gradient via finite differences

Policy gradients

Suppose we had the training labels...
(we know what to do in any state)

(x1,UP)
(x2,DOWN)
(x3,UP)
...

Suppose we had the training labels...
(we know what to do in any state)

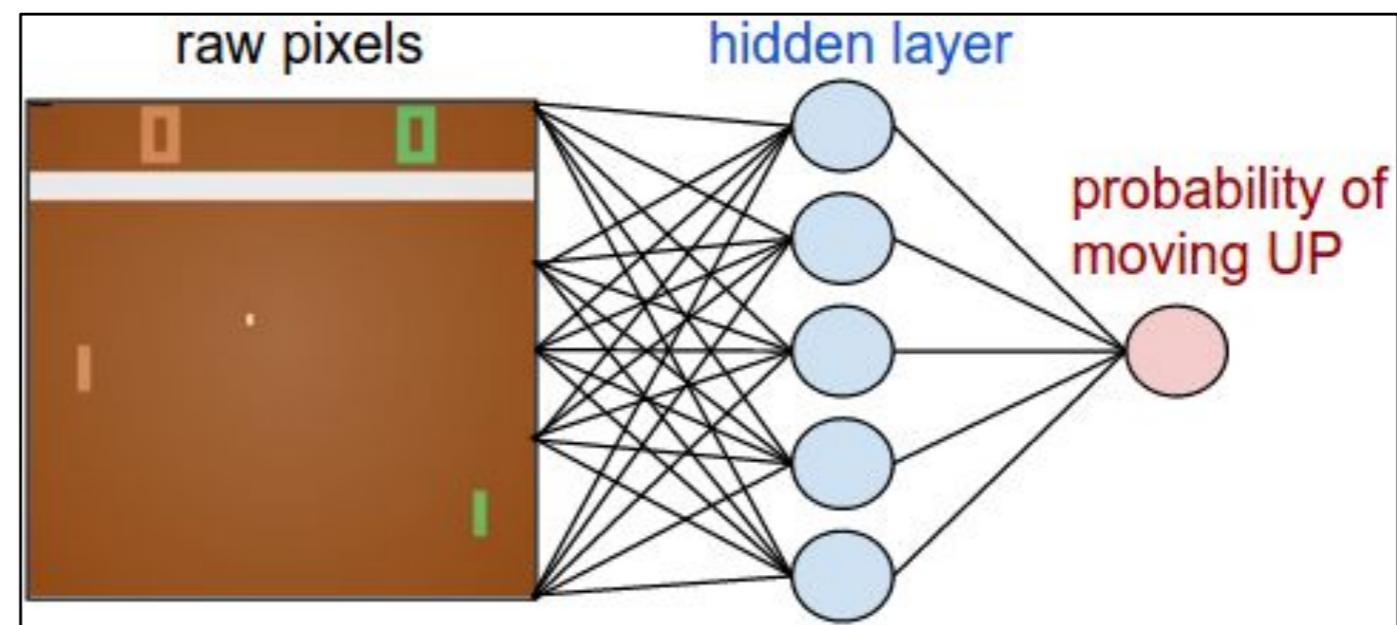


Suppose we had the training labels...
(we know what to do in any state)

(x_1, UP)
 (x_2, DOWN)
 (x_3, UP)
...

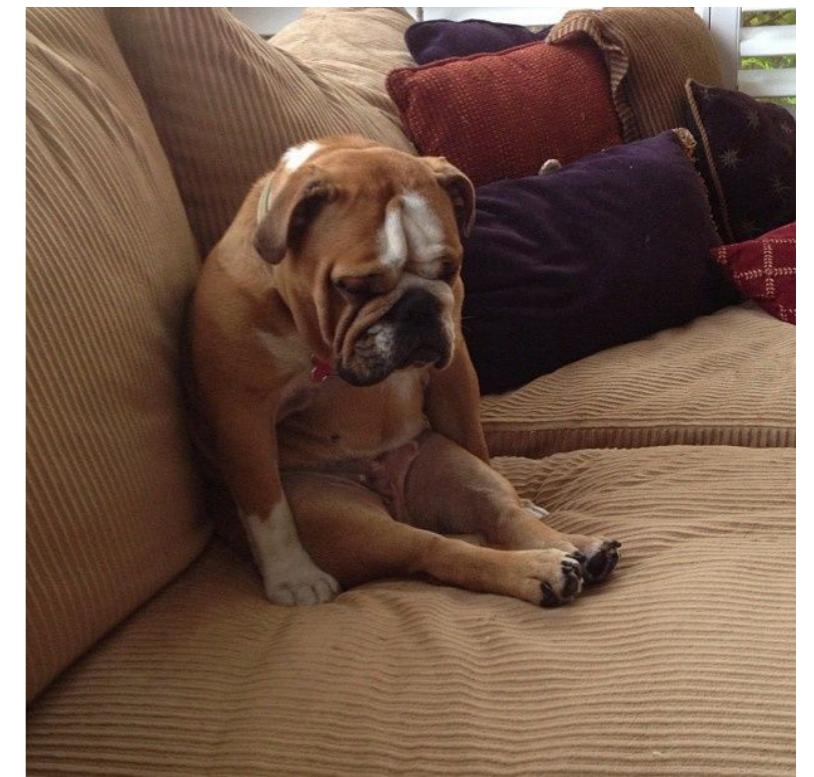
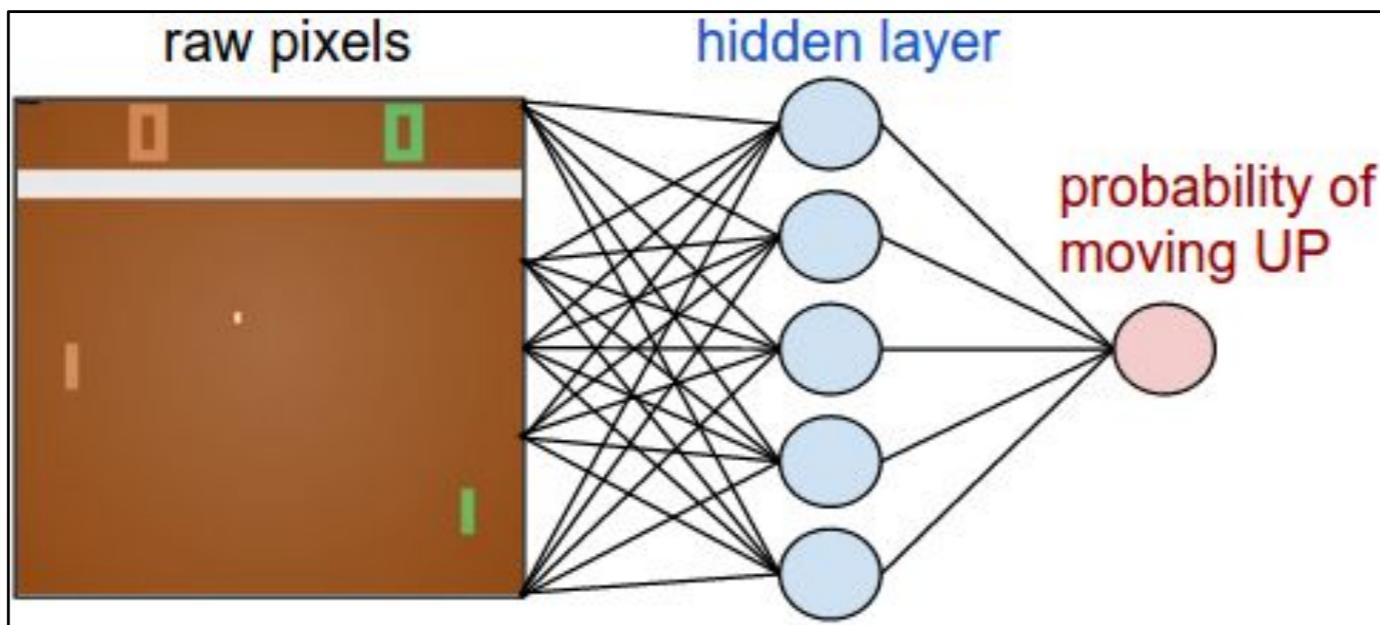
maximize:

$$\sum_i \log p(y_i | x_i)$$



supervised learning

Except, we don't have labels...



Should we go UP or DOWN?

Except, we don't have labels...

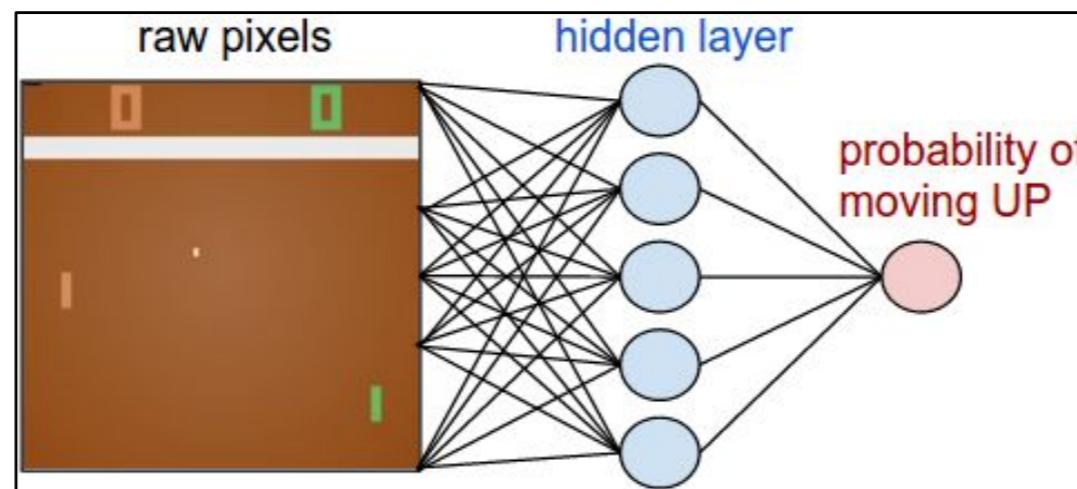


“Try a bunch of stuff and see what happens. Do more of the stuff that worked in the future.”

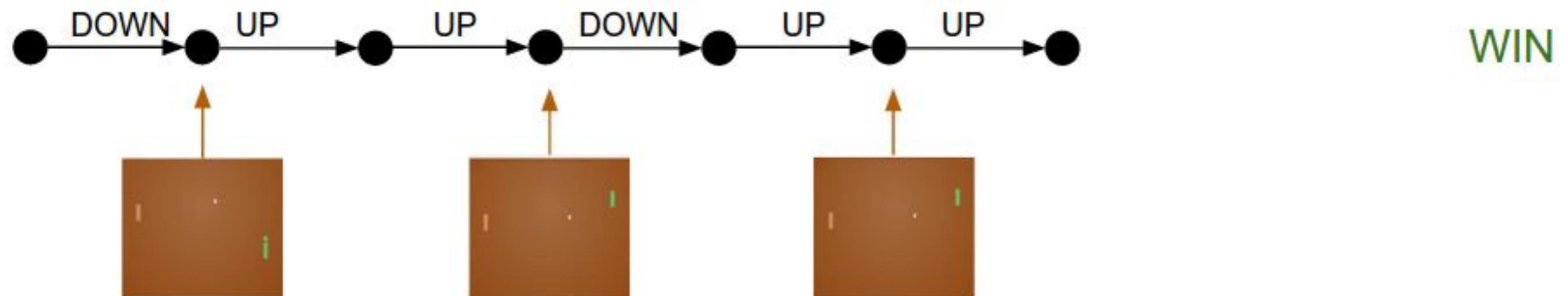
-RL

trial-and-error learning

Let's just act according to our current policy...

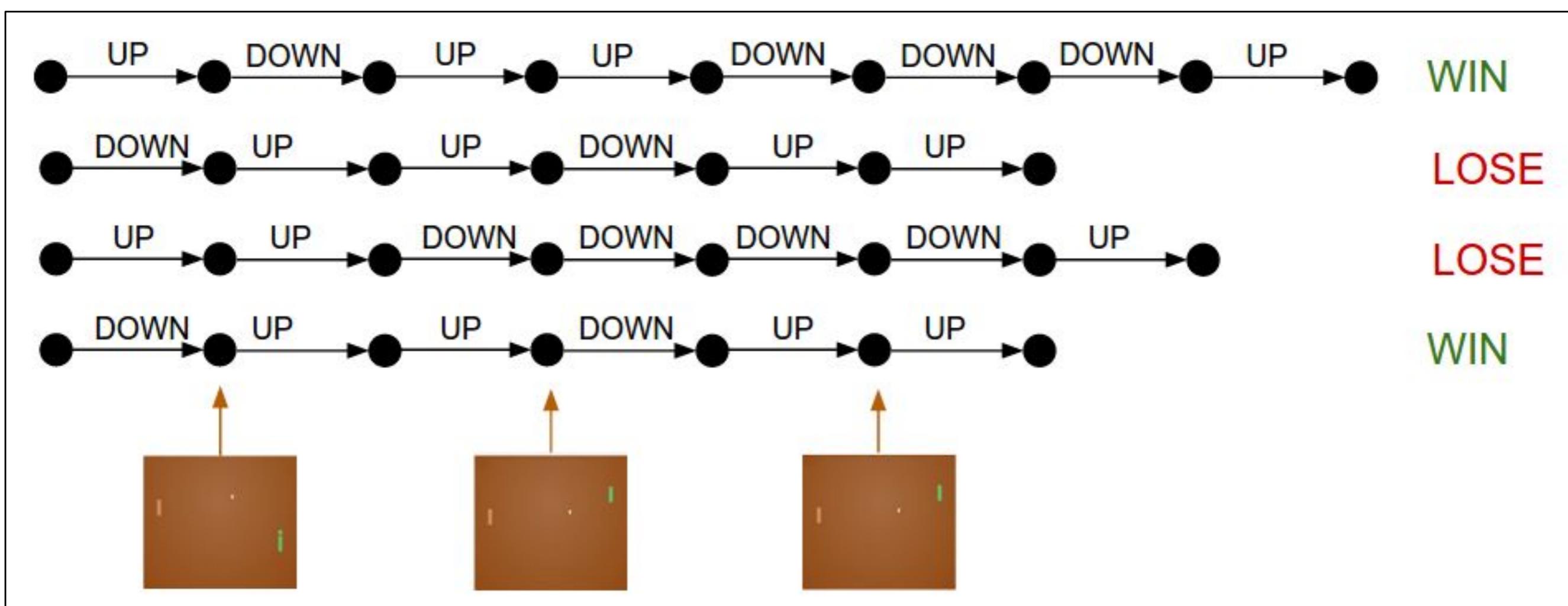


Rollout the policy
and collect an
episode

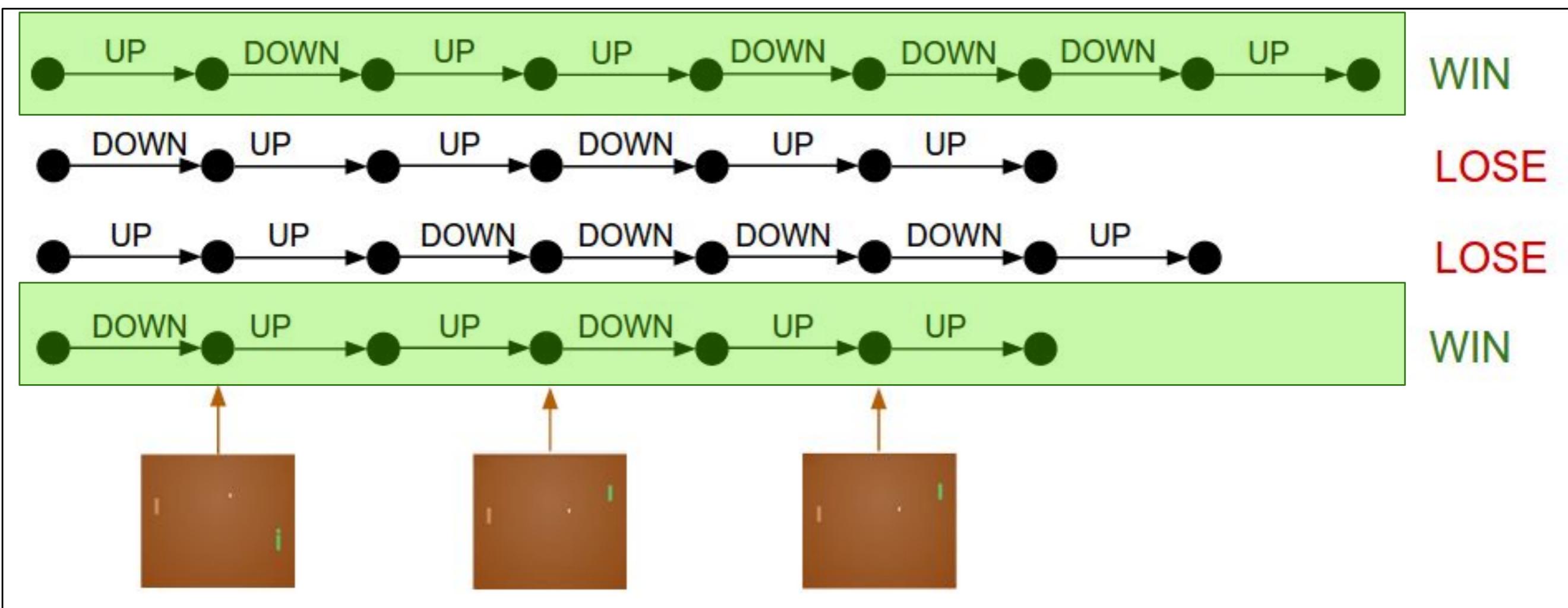


Collect many rollouts...

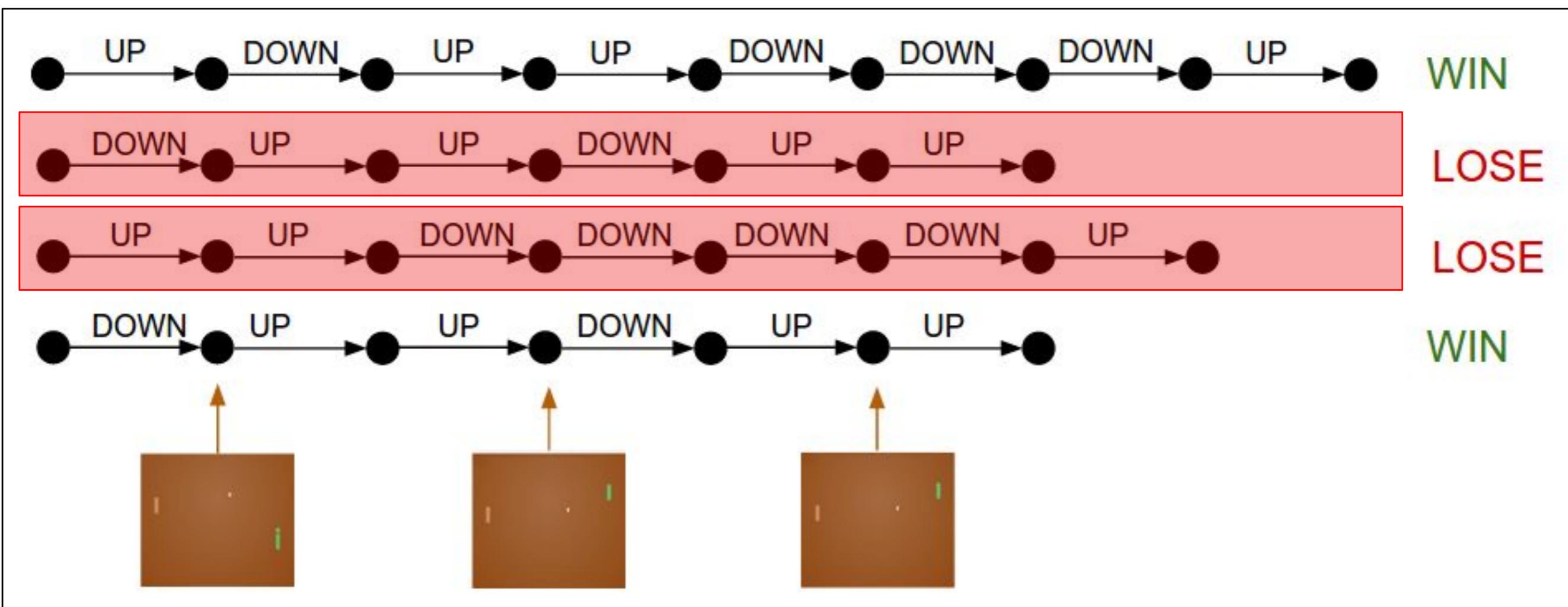
4 rollouts:



Not sure whatever we did here, but apparently it was good.



Not sure whatever we did here, but it was bad.

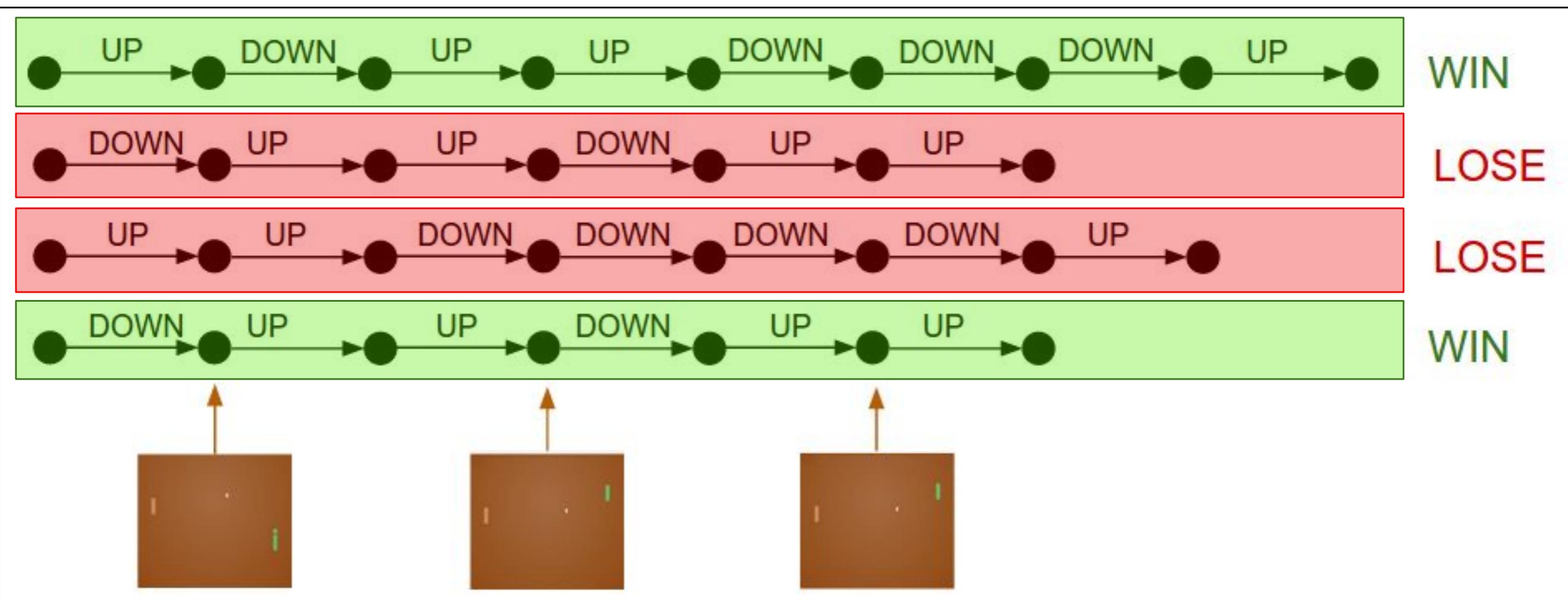


Pretend every action we took here was the correct label.

$$\text{maximize: } \log p(y_i | x_i)$$

Pretend every action we took here was the wrong label.

$$\text{maximize: } (-1) * \log p(y_i | x_i)$$



$$\nabla_{\theta} U(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)}) R(\tau^{(i)})$$

Supervised Learning

maximize:

$$\sum_i \log p(y_i | x_i)$$

For images x_i and their
labels y_i .

Supervised Learning

maximize:

$$\sum_i \log p(y_i | x_i)$$

For images x_i and their labels y_i .

Reinforcement Learning

Supervised Learning

maximize:

$$\sum_i \log p(y_i | x_i)$$

For images x_i and their labels y_i .

Reinforcement Learning

1) we have no labels so we sample:

$$y_i \sim p(\cdot | x_i)$$

Supervised Learning

maximize:

$$\sum_i \log p(y_i | x_i)$$

For images x_i and their labels y_i .

Reinforcement Learning

1) we have no labels so we sample:

$$y_i \sim p(\cdot | x_i)$$

2) once we collect a batch of rollouts:
maximize:

$$\sum_i A_i * \log p(y_i | x_i)$$

Supervised Learning

maximize:

$$\sum_i \log p(y_i | x_i)$$

For images x_i and their labels y_i .

Reinforcement Learning

1) we have no labels so we sample:

$$y_i \sim p(\cdot | x_i)$$

2) once we collect a batch of rollouts:
maximize:

$$\sum_i A_i * \log p(y_i | x_i)$$

We call this the **advantage**, it's a number, like +1.0 or -1.0 based on how this action eventually turned out.

Supervised Learning

maximize:

$$\sum_i \log p(y_i | x_i)$$

For images x_i and their labels y_i .

$$\nabla_{\theta} U(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\alpha_t^{(i)} | s_t^{(i)}) R(\tau^{(i)})$$

Reinforcement Learning

1) we have no labels so we sample:

$$y_i \sim p(\cdot | x_i)$$

2) once we collect a batch of rollouts:
maximize:

$$\sum_i A_i * \log p(y_i | x_i)$$

+ve advantage will make that action more likely in the future, for that state.

-ve advantage will make that action less likely in the future, for that state.

Can we do better than assigning the cumulative trajectory reward to every action in the trajectory?

$$\nabla_{\theta} U(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\alpha_t^{(i)} | s_t^{(i)}) R(\tau^{(i)})$$

Temporal structure

$$\begin{aligned}\hat{g} &= \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)}) R(\tau^{(i)}) \\ &= \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)}) \left(\sum_{k=0}^H R(s_k^{(i)}, a_k^{(i)}) \right)\end{aligned}$$

Each action takes the blame for the full trajectory!

Temporal structure

$$\hat{g} = \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)}) R(\tau^{(i)})$$

$$= \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)}) \left(\sum_{k=0}^H R(s_k^{(i)}, a_k^{(i)}) \right)$$

$$= \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)}) \left(\sum_{k=0}^{t-1} R(s_k^{(i)}, a_k^{(i)}) + \sum_{k=t}^H R(s_k^{(i)}, a_k^{(i)}) \right)$$

Each action takes the blame for the full trajectory!

These rewards are not caused by actions that come after t

Temporal structure

$$\hat{g} = \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)}) R(\tau^{(i)})$$

$$= \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)}) \left(\sum_{k=0}^H R(s_k^{(i)}, a_k^{(i)}) \right)$$

$$= \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)}) \left(\sum_{k=0}^{t-1} R(s_k^{(i)}, a_k^{(i)}) + \sum_{k=t}^H R(s_k^{(i)}, a_k^{(i)}) \right)$$

Each action takes the blame for the full trajectory!

Consider instead:

$$\hat{g} = \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)}) \left(\sum_{k=t}^H R(s_k^{(i)}, a_k^{(i)}) \right)$$

Each action takes the blame for the trajectory that comes after it

We can call this the return from t onwards G_t

Likelihood ratio gradient estimator

- Let's analyze the update:

$$\Delta\theta_t = \alpha G_t \nabla_{\theta} \log \pi_{\theta}(s_t, a_t)$$

- Let's us rewrite is as follows:

$$\theta_{t+1} \doteq \theta_t + \alpha \gamma^t G_t \frac{\nabla_{\theta} \pi(A_t | S_t, \theta)}{\pi(A_t | S_t, \theta)}$$

- Update is proportional to:
 - the product of a return G_t and
 - the gradient of the probability of taking the action actually taken,
 - divided by the probability of taking that action.

Likelihood ratio gradient estimator

- Let's analyze the update:

$$\Delta\theta_t = \alpha G_t \nabla_{\theta} \log \pi_{\theta}(s_t, a_t)$$

- Let's us rewrite is as follows:

move most in the directions that favor actions that yield **the highest return**

$$\theta_{t+1} \doteq \theta_t + \alpha \gamma^t G_t \frac{\nabla_{\theta} \pi(A_t | S_t, \theta)}{\pi(A_t | S_t, \theta)}$$

Update is **inversely proportional to the action probability** to fight the fact that actions that are selected frequently are at an advantage (the updates will be more often in their direction)

Variance

Here is our unbiased gradient estimator:

$$\hat{g} = \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\alpha_t^{(i)} | s_t^{(i)}) G_t^{(i)}$$

The trace of the covariance matrix:

$$\text{Var}(\hat{g}) = \text{tr} \left(\mathbb{E} [(\hat{g} - \mathbb{E}[\hat{g}])(\hat{g} - \mathbb{E}[\hat{g}])^T] \right) = \sum_{k=1}^n \mathbb{E} \left[(\hat{g}_k - \mathbb{E}[\hat{g}_k])^2 \right]$$

Our goal is to minimize the variance of our estimator

Remember that for random variable X: $\text{Var}(aX) = a^2 \text{Var}(x)$

Reducing variance with baseline

What if we subtract a constant b from the rewards:

$$\begin{aligned}\hat{g} &= \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log P_{\theta}(\tau^{(i)}) (R(\tau^{(i)}) - b) \\ \hat{g} &= \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log P_{\theta}(\tau^{(i)}) R(\tau^{(i)}) - \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log P_{\theta}(\tau^{(i)}) b\end{aligned}$$

$$\begin{aligned}&\mathbb{E} \nabla_{\theta} \log P_{\theta}(\tau) b \\ &= \sum_{\tau} P(\tau; \theta) \nabla_{\theta} \log P_{\theta}(\tau) b \\ &= \sum_{\tau} P(\tau; \theta) \frac{\nabla_{\theta} P_{\theta}(\tau)}{P(\tau; \theta)} b\end{aligned}$$

Reducing variance with baseline

What if we subtract a constant b from the rewards:

$$\begin{aligned}\hat{g} &= \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log P_{\theta}(\tau^{(i)}) (R(\tau^{(i)}) - b) \\ \hat{g} &= \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log P_{\theta}(\tau^{(i)}) R(\tau^{(i)}) - \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log P_{\theta}(\tau^{(i)}) b\end{aligned}$$

$$\begin{aligned}&\mathbb{E} \nabla_{\theta} \log P_{\theta}(\tau) b \\ &= \sum_{\tau} P(\tau; \theta) \nabla_{\theta} \log P_{\theta}(\tau) b \\ &= \sum_{\tau} P(\tau; \theta) \frac{\nabla_{\theta} P_{\theta}(\tau)}{P(\tau; \theta)} b \\ &= \sum_{\tau} \nabla_{\theta} P_{\theta}(\tau) b\end{aligned}$$

Reducing variance with baseline

What if we subtract a constant b from the rewards:

$$\begin{aligned}\hat{g} &= \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log P_{\theta}(\tau^{(i)}) (R(\tau^{(i)}) - b) \\ \hat{g} &= \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log P_{\theta}(\tau^{(i)}) R(\tau^{(i)}) - \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log P_{\theta}(\tau^{(i)}) b\end{aligned}$$

$$\begin{aligned}&\mathbb{E} \nabla_{\theta} \log P_{\theta}(\tau) b \\ &= \sum_{\tau} P(\tau; \theta) \nabla_{\theta} \log P_{\theta}(\tau) b \\ &= \sum_{\tau} P(\tau; \theta) \frac{\nabla_{\theta} P_{\theta}(\tau)}{P(\tau; \theta)} b \\ &= \sum_{\tau} \nabla_{\theta} P_{\theta}(\tau) b \\ &= b \left(\sum_{\tau} \nabla_{\theta} P_{\theta}(\tau) \right)\end{aligned}$$

Reducing variance with baseline

What if we subtract a constant b from the rewards:

$$\hat{g} = \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log P_{\theta}(\tau^{(i)}) (R(\tau^{(i)}) - b)$$
$$\hat{g} = \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log P_{\theta}(\tau^{(i)}) R(\tau^{(i)}) - \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log P_{\theta}(\tau^{(i)}) b$$

$$\begin{aligned}\mathbb{E} \nabla_{\theta} \log P_{\theta}(\tau) b \\ &= \sum_{\tau} P(\tau; \theta) \nabla_{\theta} \log P_{\theta}(\tau) b \\ &= \sum_{\tau} P(\tau; \theta) \frac{\nabla_{\theta} P_{\theta}(\tau)}{P(\tau; \theta)} b \\ &= \sum_{\tau} \nabla_{\theta} P_{\theta}(\tau) b \\ &= b \left(\sum_{\tau} \nabla_{\theta} P_{\theta}(\tau) \right) \\ &= b \left(\nabla_{\theta} \sum_{\tau} P_{\theta}(\tau) \right)\end{aligned}$$

Reducing variance with baseline

What if we subtract a constant b from the rewards:

$$\begin{aligned}\hat{g} &= \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log P_{\theta}(\tau^{(i)}) (R(\tau^{(i)}) - b) \\ \hat{g} &= \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log P_{\theta}(\tau^{(i)}) R(\tau^{(i)}) - \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log P_{\theta}(\tau^{(i)}) b\end{aligned}$$

$$\begin{aligned}&\mathbb{E} \nabla_{\theta} \log P_{\theta}(\tau) b \\ &= \sum_{\tau} P(\tau; \theta) \nabla_{\theta} \log P_{\theta}(\tau) b \\ &= \sum_{\tau} P(\tau; \theta) \frac{\nabla_{\theta} P_{\theta}(\tau)}{P(\tau; \theta)} b \\ &= \sum_{\tau} \nabla_{\theta} P_{\theta}(\tau) b \\ &= b \left(\sum_{\tau} \nabla_{\theta} P_{\theta}(\tau) \right) \\ &= b \left(\nabla_{\theta} \sum_{\tau} P_{\theta}(\tau) \right) \\ &= 0\end{aligned}$$

We still have an unbiased estimator of the gradient!

Baseline choices

$$\hat{g} = \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)}) \left(\sum_{k=t}^H R(s_k^{(i)}, a_k^{(i)}) - b \right)$$

- Constant baseline: $b = \mathbb{E}[R(\tau)]$
- Time-dependent baseline: $b_t = \sum_{i=1}^N \sum_{k=t}^H R(s_k^{(i)}, a_k^{(i)})$

REINFORCE

Algorithm 1 “Vanilla” policy gradient algorithm

Initialize policy parameter θ , baseline b

for iteration=1, 2, . . . **do**

 Collect a set of trajectories by executing the current policy

 At each timestep in each trajectory, compute

 the *return* $R_t = \sum_{t'=t}^{T-1} \gamma^{t'-t} r_{t'}$, and

 the *advantage estimate* $\hat{A}_t = R_t - b(s_t)$.

 Re-fit the baseline, by minimizing $\|b(s_t) - R_t\|^2$,
 summed over all trajectories and timesteps.

 Update the policy, using a policy gradient estimate \hat{g} ,
 which is a sum of terms $\nabla_\theta \log \pi(a_t | s_t, \theta) \hat{A}_t$

end for

Baseline choices

$$\hat{g} = \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)}) \left(\sum_{k=t}^H R(s_k^{(i)}, a_k^{(i)}) - b \right)$$

- Constant baseline: $b = \mathbb{E}[R(\tau)]$
- Time-dependent baseline: $b_t = \sum_{i=1}^N \sum_{k=t}^H R(s_k^{(i)}, a_k^{(i)})$
- State-dependent expected return:
$$b(s_t) = \mathbb{E} [r_t + r_{t+1} + r_{t+2} + \dots + r_{H-1}] = V_{\pi}(s_t)$$

$$\hat{g} = \frac{1}{N} \sum_{i=1}^N \sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)}) (G_t^i - b(s_t^{(i)}))$$

Variance

$$\hat{g} = \frac{1}{N} \sum_{i=1}^N \sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)}) (G_t^i - b(s_t^{(i)}))$$

$$\text{Var}(\hat{g}) = \text{tr} \left(\mathbb{E} [(\hat{g} - \mathbb{E}[\hat{g}])(\hat{g} - \mathbb{E}[\hat{g}])^T] \right) = \sum_{k=1}^n \mathbb{E} \left[(\hat{g}_k - \mathbb{E}[\hat{g}_k])^2 \right]$$

- Imagine in some state S_1 the rewards of all actions are ~ 3000 and in $S_2 \sim -4000$. The variance is large of this vector.
- Now imagine you have $b(S_1)=3000$ and $b(S_2)=-4000$. I have much decreased the variance.
- We want to encourage an action, not when it has high return, but when it have higher return than the state expected return $b(s)$, i.e., when making this action MORE probable would allow me to improve over what I have now.

Estimate $V_\pi(s_t)$

$$\hat{g} = \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_t^{(i)} | s_t^{(i)}) \left(\sum_{k=t}^H R(s_k^{(i)}, a_k^{(i)}) - V^\pi(s_k^{(i)}) \right)$$

MC estimation

Initialize ϕ

- Collect trajectories τ_1, \dots, τ_N
- Regress against empirical return:

$$\phi_{i+1} \leftarrow \arg \min_{\phi} \frac{1}{N} \sum_{i=1}^N \sum_{t=0}^{H-1} \left(V_\phi^\pi(s_t^{(i)}) - \left(\sum_{k=t}^{H-1} R(s_k^{(i)}, u_k^{(i)}) \right) \right)^2$$

Estimate $V_\pi(s_t)$

$$\hat{g} = \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_t^{(i)} | s_t^{(i)}) \left(\sum_{k=t}^H R(s_k^{(i)}, a_k^{(i)}) - V^\pi(s_k^{(i)}) \right)$$

TD estimation

Initialize ϕ

- Collect data $\{s, u, s', r\}$
- Fitted V iteration:

$$\phi_{i+1} \leftarrow \min_{\phi} \sum_{(s, u, s', r)} \|r + V_{\phi_i}^\pi(s') - V_\phi(s)\|_2^2 + \lambda \|\phi - \phi_i\|_2^2$$

Bootstrapping!

Better estimates for cumulative future reward

$$\hat{g} = \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)}) \left(\sum_{k=t}^H R(s_k^{(i)}, a_k^{(i)}) - V^{\pi}(s_k^{(i)}) \right)$$



- Estimation of Q from *single* roll-out

$$Q^{\pi}(s, u) = \mathbb{E}[r_0 + r_1 + r_2 + \dots | s_0 = s, a_0 = a]$$

- = high variance per sample based / no generalization
 - Reduce variance by discounting
 - Reduce variance by function approximation (=critic)

Actor-Critic

- › Monte-Carlo policy gradient still has **high variance**
- › We can use a **critic** to estimate the action-value function:

$$Q_w(s, a) \approx Q^{\pi_\theta}(s, a)$$

- › **Actor-critic algorithms** maintain two sets of parameters
 - **Critic Updates** action-value function parameters w
 - **Actor Updates** policy parameters θ , in direction suggested by critic
- › Actor-critic algorithms follow an approximate policy gradient

Actor-Critic

$$Q^{\pi,\gamma}(s,u) = \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots \mid s_0 = s, u_0 = u]$$

Actor-Critic

$$\begin{aligned} Q^{\pi, \gamma}(s, u) &= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots \mid s_0 = s, u_0 = u] \\ &= \mathbb{E}[r_0 + \gamma V^\pi(s_1) \mid s_0 = s, u_0 = u] \end{aligned}$$

Actor-Critic

$$\begin{aligned} Q^{\pi, \gamma}(s, u) &= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots \mid s_0 = s, u_0 = u] \\ &= \mathbb{E}[r_0 + \gamma V^\pi(s_1) \mid s_0 = s, u_0 = u] \\ &= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 V^\pi(s_2) \mid s_0 = s, u_0 = u] \end{aligned}$$

Actor-Critic

$$\begin{aligned} Q^{\pi, \gamma}(s, u) &= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots \mid s_0 = s, u_0 = u] \\ &= \mathbb{E}[r_0 + \gamma V^\pi(s_1) \mid s_0 = s, u_0 = u] \\ &= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 V^\pi(s_2) \mid s_0 = s, u_0 = u] \\ &= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 V^\pi(s_3) \mid s_0 = s, u_0 = u] \\ &= \cdots \end{aligned}$$

- **Async Advantage Actor Critic (A3C) [Mnih et al, 2016]**
 - \hat{Q} one of the above choices (e.g. k=5 step lookahead)

Asynchronous Deep RL

Asynchronous Methods for Deep Reinforcement Learning

Volodymyr Mnih¹

VMNIH@GOOGLE.COM

Adrià Puigdomènech Badia¹

ADRIAP@GOOGLE.COM

Mehdi Mirza^{1,2}

MIRZAMOM@IRO.UMONTREAL.CA

Alex Graves¹

GRAVESA@GOOGLE.COM

Tim Harley¹

THARLEY@GOOGLE.COM

Timothy P. Lillicrap¹

COUNTZERO@GOOGLE.COM

David Silver¹

DAVIDSILVER@GOOGLE.COM

Koray Kavukcuoglu¹

KORAYK@GOOGLE.COM

¹ Google DeepMind

² Montreal Institute for Learning Algorithms (MILA), University of Montreal

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

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 $\theta'_v = \theta_v$

$t_{start} = t$

 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}

$t \leftarrow t + 1$

$T \leftarrow T + 1$

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 $i \in \{t - 1, \dots, t_{start}\}$ **do**

$R \leftarrow r_i + \gamma R$

 Accumulate gradients wrt θ' : $d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi(a_i|s_i; \theta') (R - V(s_i; \theta'_v))$

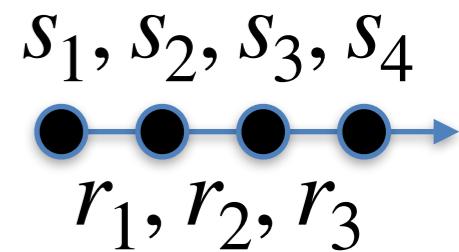
 Accumulate gradients wrt θ'_v : $d\theta_v \leftarrow d\theta_v + \partial (R - V(s_i; \theta'_v))^2 / \partial \theta'_v$

end for

 Perform asynchronous update of θ using $d\theta$ and of θ_v using $d\theta_v$.

until $T > T_{max}$

A3C



What is the approximation used for the advantage?

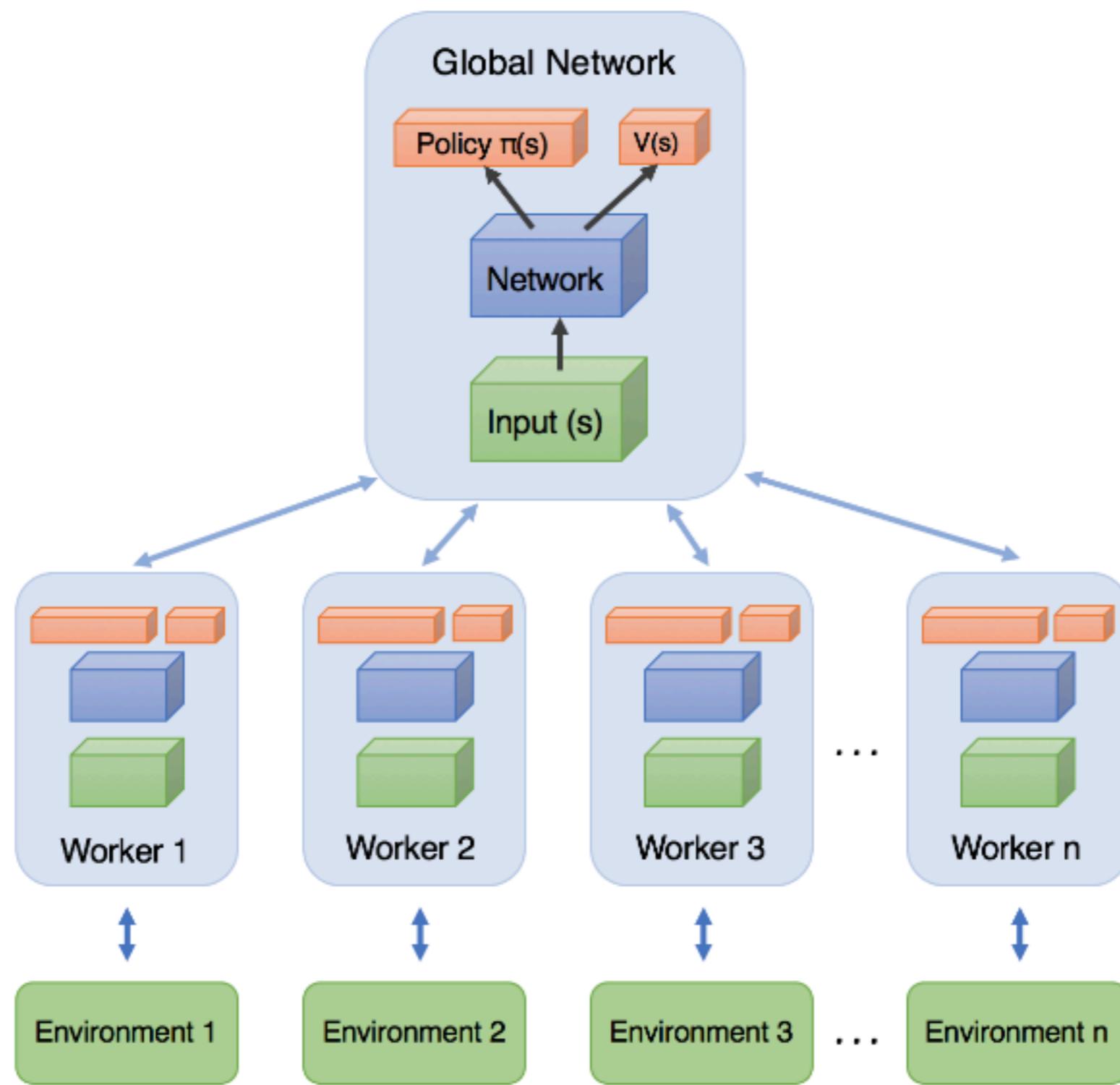
$$R_3 = r_3 + \gamma V(s_4, \theta'_v)$$

$$R_2 = r_2 + \gamma r_3 + \gamma^2 V(s_4, \theta'_v)$$

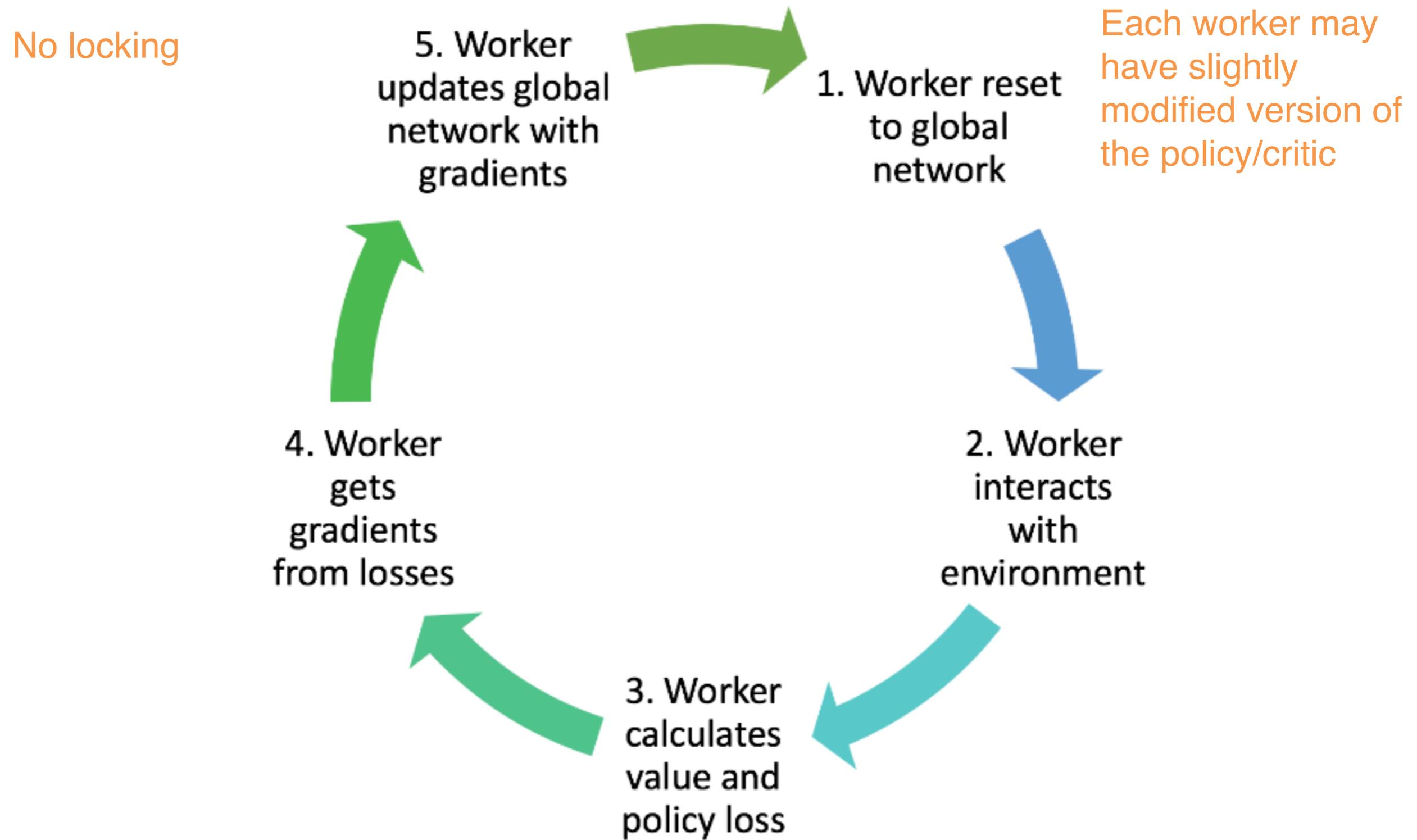
$$A_3 = R_3 - V(s_3; \theta'_v)$$

$$A_2 = R_2 - V(s_2; \theta'_v)$$

Distributed RL

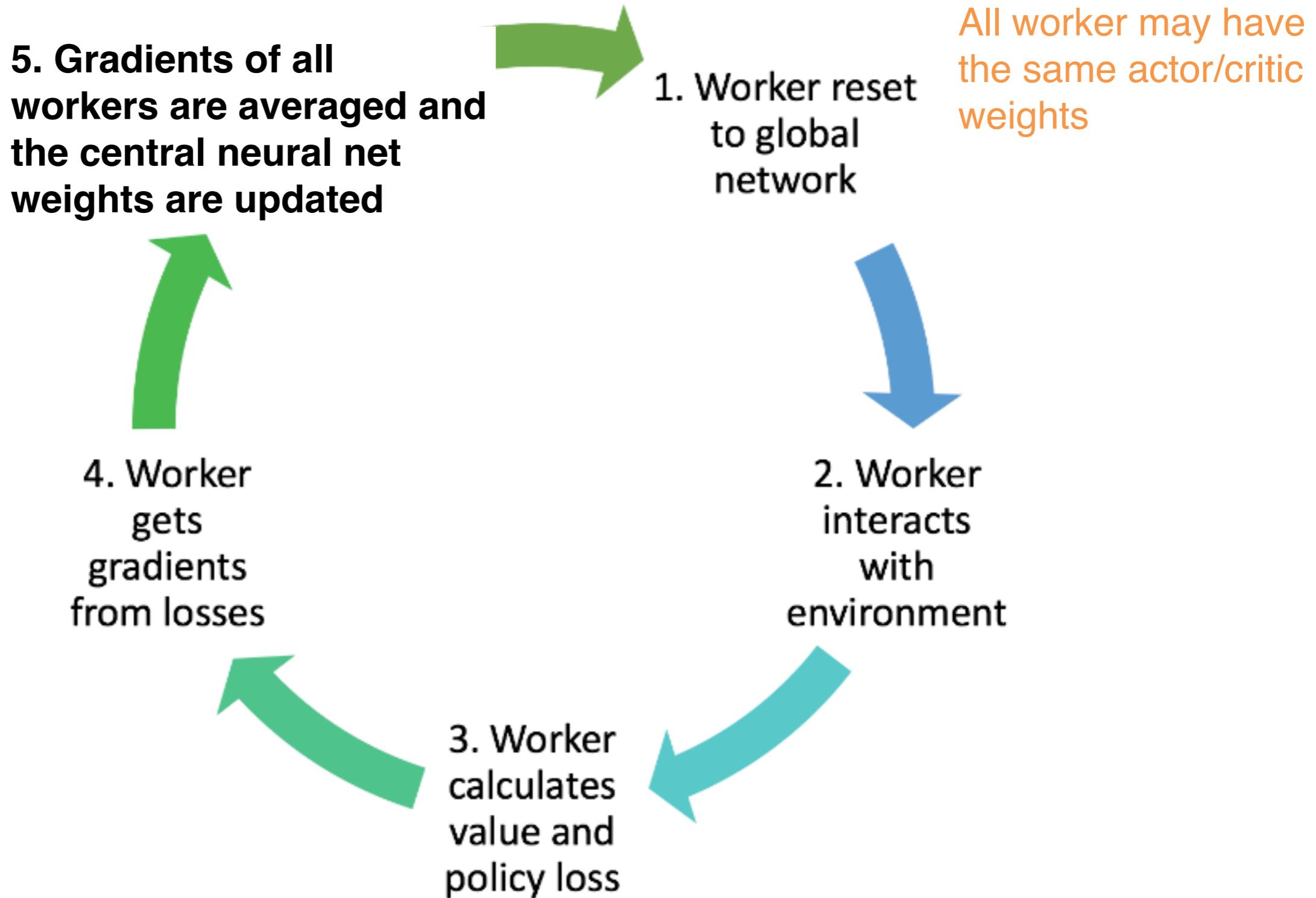


Distributed Asynchronous RL



The actor critic trained in such asynchronous way is known as A3C

Distributed Synchronous RL



The actor critic trained in such synchronous way is known as A2C

Advantages of Asynchronous (multi-threaded) RL

