Carnegie Mellon

School of Computer Science

Deep Reinforcement Learning and Control

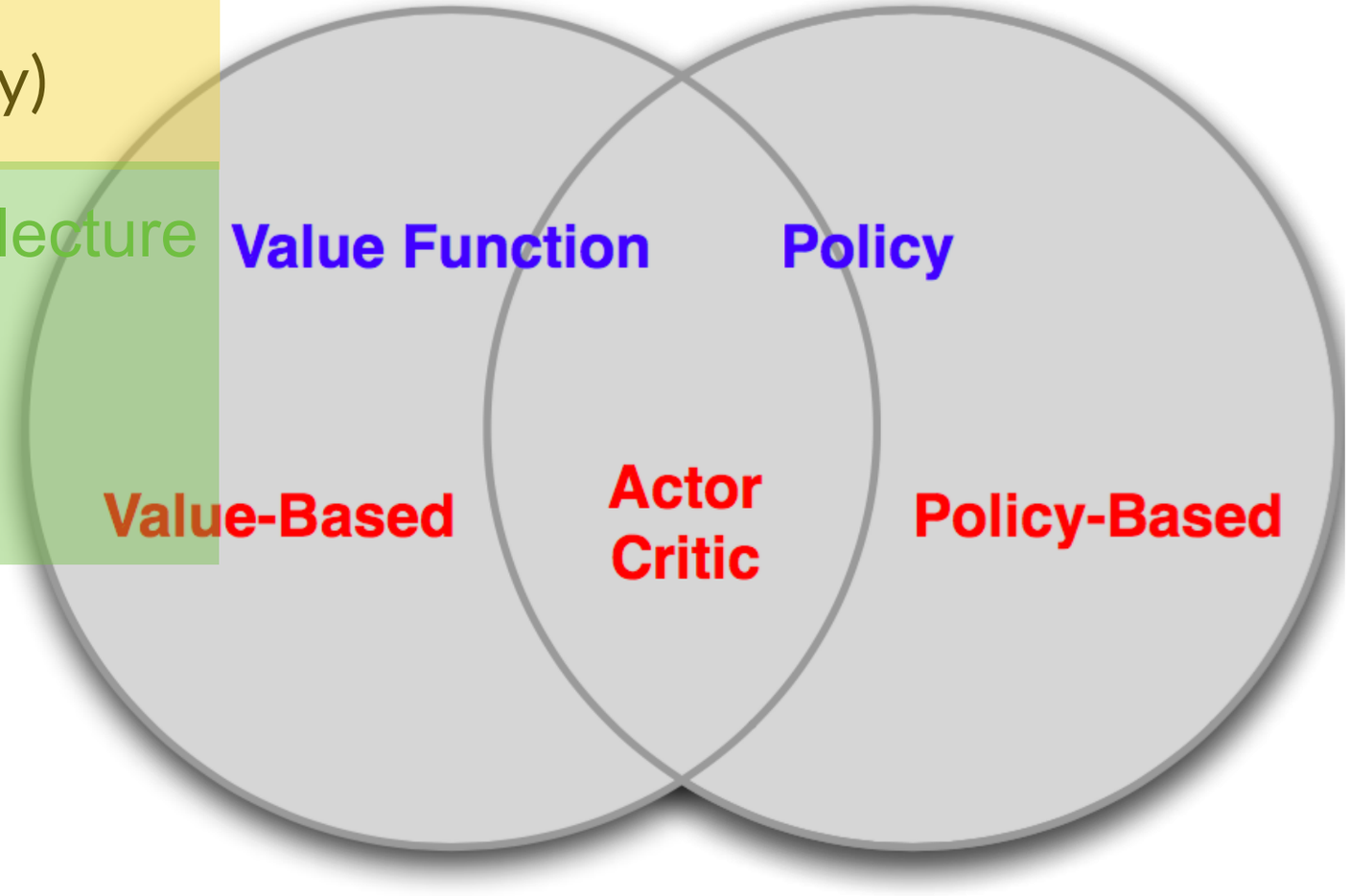
# Policy gradients

Spring 2020, CMU 10-403

Katerina Fragkiadaki

# Value-Based and Policy-Based RL

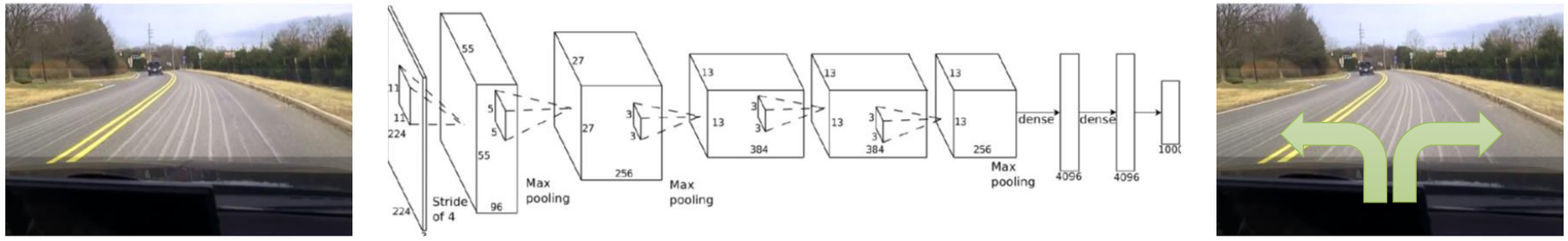
- Value Based We have covered those
  - Learned Value Function
  - Implicit policy (e.g. ε-greedy)
- Policy Based this lecture
  - No Value Function
  - Learned Policy
- Actor-Critic
  - Learned Value Function
  - Learned Policy



# Advantages of Policy-Based RL

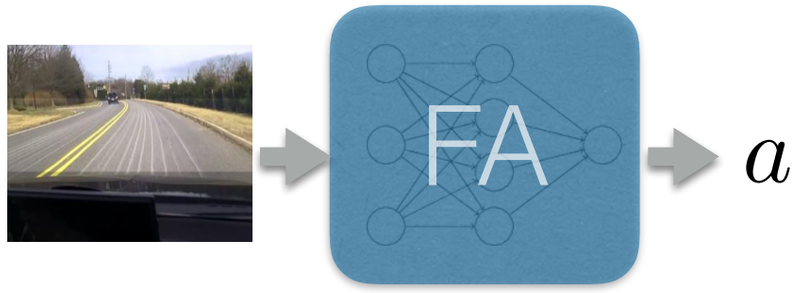
- Advantages
  - Effective in high-dimensional or continuous action spaces
  - Can learn stochastic policies

# Policy function approximators



# Policy function approximators

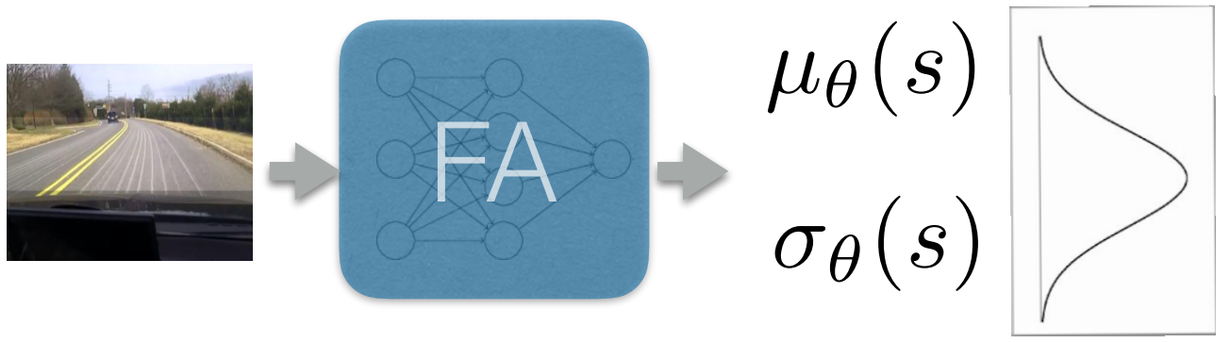
deterministic continuous policy




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

e.g. outputs  a steering angle directly

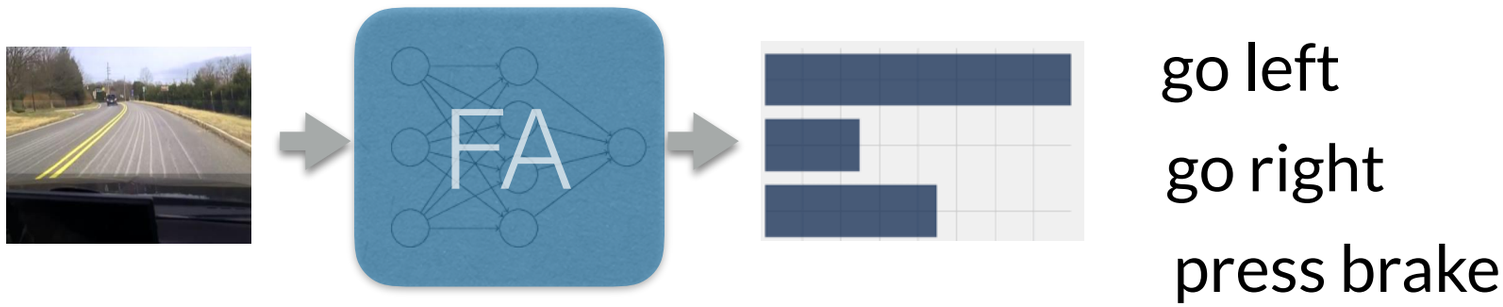
stochastic continuous policy




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

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

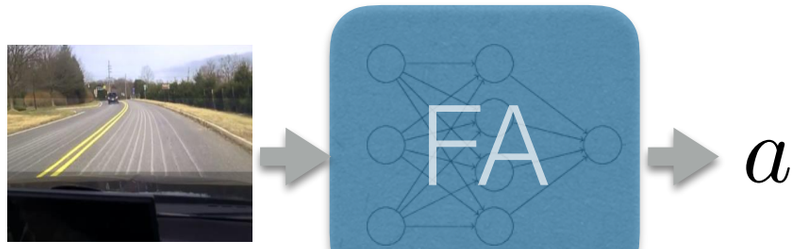
(stochastic) policy over discrete actions




Outputs a distribution over a discrete set of actions

# Policy function approximators

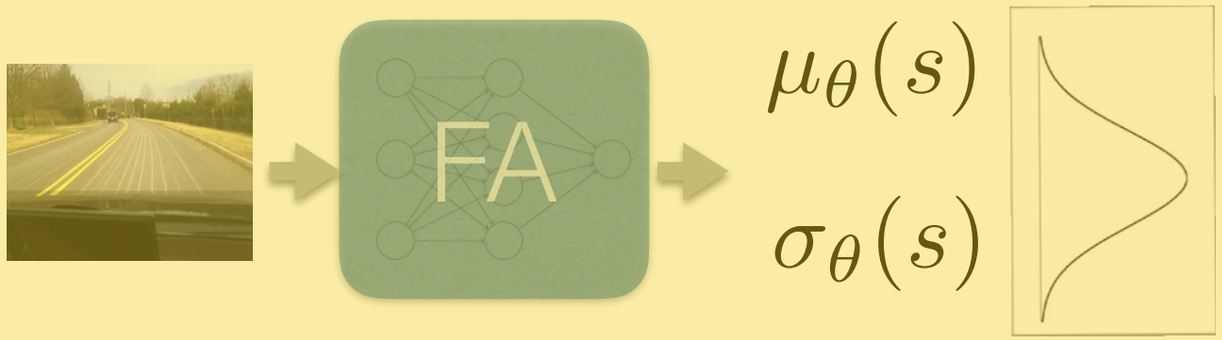
deterministic continuous policy




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

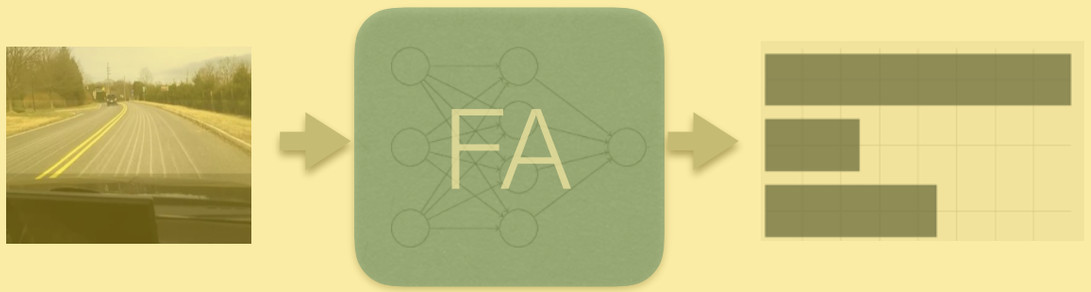
e.g. outputs  a steering angle directly

stochastic continuous policy




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

(stochastic) policy over discrete actions



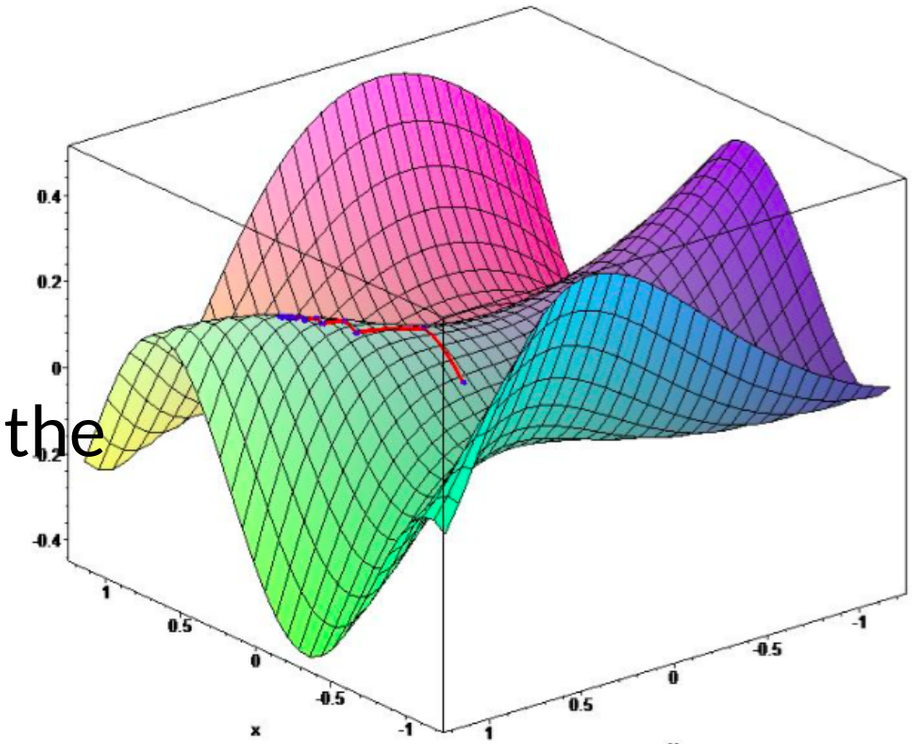

Outputs a distribution over a discrete set of actions

# Policy Optimization

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

# Policy Gradient

- Let U(θ) be any policy objective function
- Policy gradient algorithms search for a local maximum in U(θ) 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

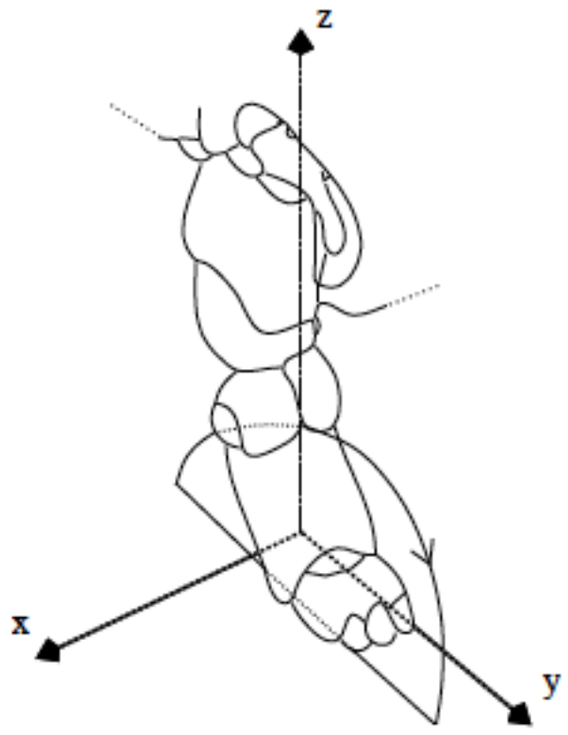
- Numerically approximating the policy gradient of $\pi_\theta(s)$
- For each dimension k in [1, n]
  - Estimate kth partial derivative of objective function w.r.t. θ
  - By perturbing θ by small amount ε in kth dimension

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

where $u_k$ is a unit vector with 1 in kth 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 the ES lecture.

# 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

Initial

Training

Final

# Learning an AIBO running policy


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

What is the score here?

Policy Gradient Reinforcement Learning for Fast Quadrupedal Locomotion, *Kohl and Stone, 2004*

# Policy objective

Trajectory $\tau$ is a state action sequence $s_0, a_0, s_1, a_1, \ldots s_H, a_H$

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

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

# Policy objective

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

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

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

Assumptions:

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

# Derivatives of expectations

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

# Derivatives of expectations

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

# Derivatives of expectations

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

Why?

# Derivatives of expectations

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

What is the problem here?

# Derivatives of expectations

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

# Derivatives of expectations

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

# Derivatives of expectations

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

What have we achieved?

# Derivatives of expectations

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

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 the policy objective

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

# Derivatives of the policy objective

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

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

# Derivatives of the policy objective

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

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

# Derivatives of the policy objective

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

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

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

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

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

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

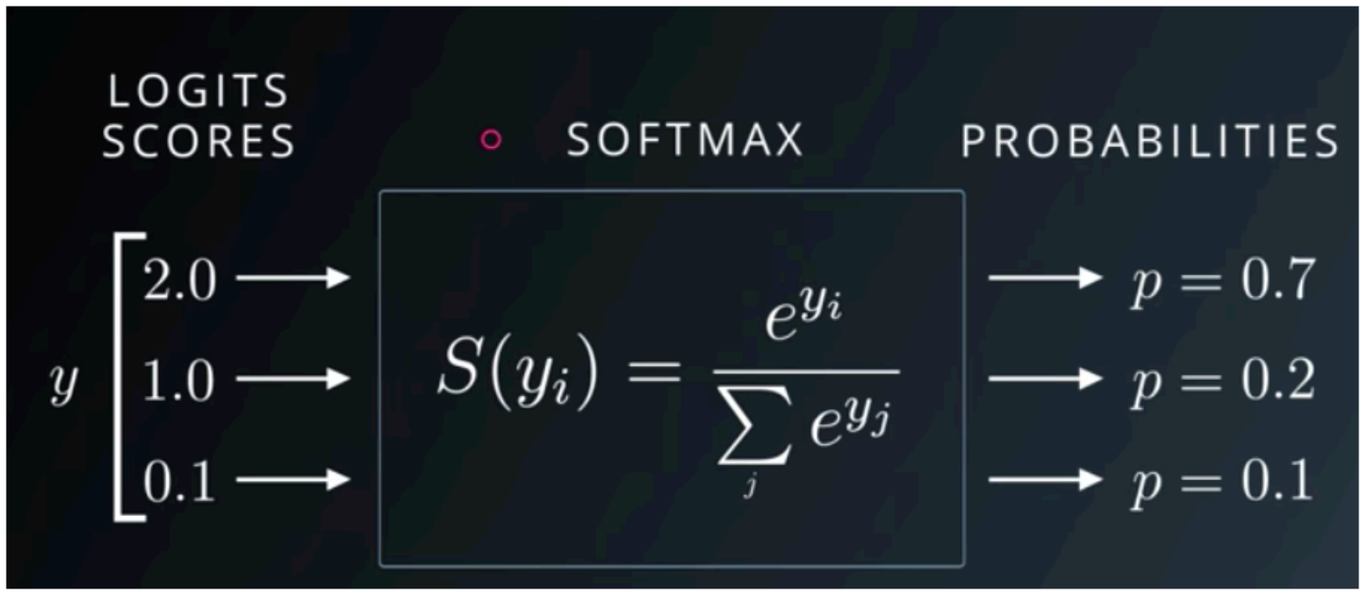
$$= \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^{(i)} \mid s_t^{(i)})$$

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

# $\nabla_\theta \log \pi_\theta(a)$ for Gaussian policy

- Policy is Gaussian $a \sim \mathcal{N}(\mu(s, \theta), \sigma^2 I)$
- Variance may be fixed $\sigma^2$, or can also parameterized
- Remember: univariate Gaussian $\mathcal{N}(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(a-\mu)^2}{\sigma^2}}$
- $\nabla_\theta \log \pi_\theta(a \mid s) = \text{const.} \cdot \frac{(\mu_\theta(s) - a)}{\sigma^2} \nabla_\theta \mu_\theta(s)$

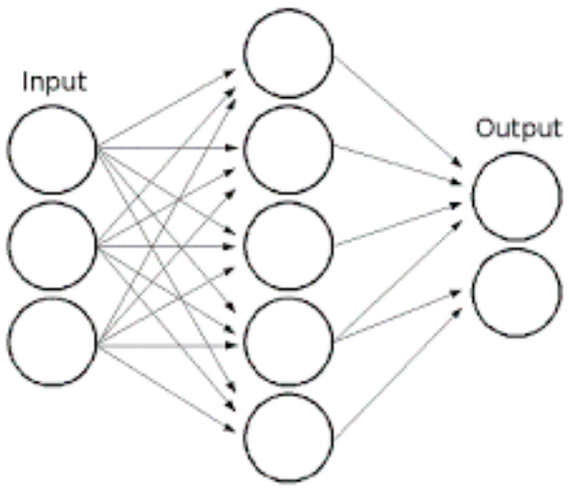
# Softmax policy




$$\pi(a \mid s, \theta) \doteq \frac{e^{h(s,a,\theta)}}{\sum_b e^{h(s,b,\theta)}}$$

discrete actions

$s$




go left

go right

Output is a distribution over a discrete set of actions

# $\nabla_\theta \log \pi_\theta(a)$ for softmax policy

- Policy is  the output of a softmax

$$\pi_\theta(a \mid s) = \frac{e^{h_\theta(s,a)}}{\sum_b e^{h_\theta(s,b)}}$$

$$\nabla_\theta \log \pi_\theta(a) =$$

# $\nabla_\theta \log \pi_\theta(a)$ for softmax policy

- Policy is  the output of a softmax

$$\pi_\theta(a \mid s) = \frac{e^{h_\theta(s,a)}}{\sum_b e^{h_\theta(s,b)}}$$

$$\nabla_\theta \log \pi_\theta(a) = \nabla_\theta h_\theta(s,a) - \nabla_\theta \log \sum_b e^{h_\theta(s,b)}$$

# $\nabla_\theta \log \pi_\theta(a)$ for softmax policy

- Policy is  the output of a softmax

$$\pi_\theta(a \mid s) = \frac{e^{h_\theta(s,a)}}{\sum_b e^{h_\theta(s,b)}}$$

$$\nabla_\theta \log \pi_\theta(a) = \nabla_\theta h_\theta(s,a) - \nabla_\theta \log \sum_b e^{h_\theta(s,b)}$$

$$\nabla_\theta h_\theta(s,a) - \frac{1}{\sum_b e^{h_\theta(s,b)}} \nabla_\theta \sum_b e^{h_\theta(s,b)}$$

# $\nabla_\theta \log \pi_\theta(a)$ for softmax policy

- Policy is  the output of a softmax

$$\pi_\theta(a \mid s) = \frac{e^{h_\theta(s,a)}}{\sum_b e^{h_\theta(s,b)}}$$

$$\begin{aligned}
\nabla_\theta \log \pi_\theta(a) = \; & \nabla_\theta h_\theta(s,a) - \nabla_\theta \log \sum_b e^{h_\theta(s,b)} \\
& \nabla_\theta h_\theta(s,a) - \frac{1}{\sum_b e^{h_\theta(s,b)}} \nabla_\theta \sum_b e^{h_\theta(s,b)} \\
& \nabla_\theta h_\theta(s,a) - \frac{1}{\sum_b e^{h_\theta(s,b)}} \sum_b \nabla_\theta e^{h_\theta(s,b)}
\end{aligned}$$

# $\nabla_\theta \log \pi_\theta(a)$ for softmax policy

- Policy is  the output of a softmax

$$\pi_\theta(a \mid s) = \frac{e^{h_\theta(s,a)}}{\sum_b e^{h_\theta(s,b)}}$$

$$\begin{aligned}
\nabla_\theta \log \pi_\theta(a) = \; & \nabla_\theta h_\theta(s,a) - \nabla_\theta \log \sum_b e^{h_\theta(s,b)} \\
& \nabla_\theta h_\theta(s,a) - \frac{1}{\sum_b e^{h_\theta(s,b)}} \nabla_\theta \sum_b e^{h_\theta(s,b)} \\
& \nabla_\theta h_\theta(s,a) - \frac{1}{\sum_b e^{h_\theta(s,b)}} \sum_b \nabla_\theta e^{h_\theta(s,b)} \\
& \nabla_\theta h_\theta(s,a) - \frac{1}{\sum_b e^{h_\theta(s,b)}} \sum_b e^{h_\theta(s,b)} \nabla_\theta h_\theta(s,b)
\end{aligned}$$

# $\nabla_\theta \log \pi_\theta(a)$ for softmax policy

- Policy is  the output of a softmax

$$\pi_\theta(a \mid s) = \frac{e^{h_\theta(s,a)}}{\sum_b e^{h_\theta(s,b)}}$$

$$\begin{aligned}
\nabla_\theta \log \pi_\theta(a) = & \nabla_\theta h_\theta(s,a) - \nabla_\theta \log \sum_b e^{h_\theta(s,b)} \\
& \nabla_\theta h_\theta(s,a) - \frac{1}{\sum_b e^{h_\theta(s,b)}} \nabla_\theta \sum_b e^{h_\theta(s,b)} \\
& \nabla_\theta h_\theta(s,a) - \frac{1}{\sum_b e^{h_\theta(s,b)}} \sum_b \nabla_\theta e^{h_\theta(s,b)} \\
& \nabla_\theta h_\theta(s,a) - \frac{1}{\sum_b e^{h_\theta(s,b)}} \sum_b e^{h_\theta(s,b)} \nabla_\theta h_\theta(s,b) \\
& \nabla_\theta h_\theta(s,a) - \sum_b \frac{e^{h_\theta(s,b)}}{\sum_b e^{h_\theta(s,b)}} \nabla_\theta h_\theta(s,b)
\end{aligned}$$

# $\nabla_\theta \log \pi_\theta(a)$ for softmax policy

- Policy is the output of a softmax

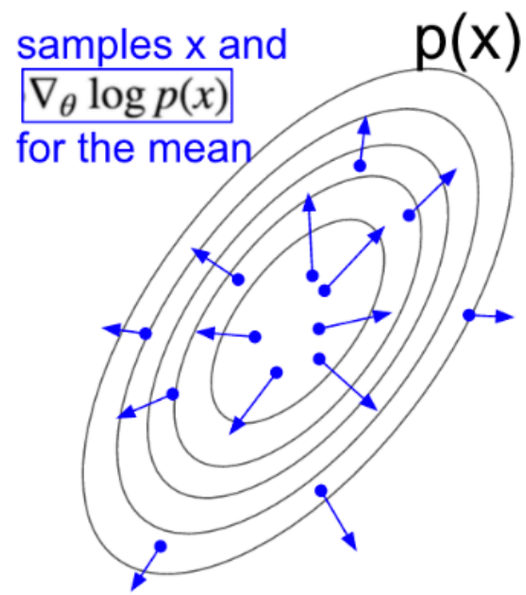
$$\pi_\theta(a \mid s) = \frac{e^{h_\theta(s,a)}}{\sum_b e^{h_\theta(s,b)}}$$

$$\begin{aligned}
\nabla_\theta \log \pi_\theta(a) = \; & \nabla_\theta h_\theta(s,a) - \nabla_\theta \log \sum_b e^{h_\theta(s,b)} \\
& \nabla_\theta h_\theta(s,a) - \frac{1}{\sum_b e^{h_\theta(s,b)}} \nabla_\theta \sum_b e^{h_\theta(s,b)} \\
& \nabla_\theta h_\theta(s,a) - \frac{1}{\sum_b e^{h_\theta(s,b)}} \sum_b \nabla_\theta e^{h_\theta(s,b)} \\
& \nabla_\theta h_\theta(s,a) - \frac{1}{\sum_b e^{h_\theta(s,b)}} \sum_b e^{h_\theta(s,b)} \nabla_\theta h_\theta(s,b) \\
& \nabla_\theta h_\theta(s,a) - \sum_b \frac{e^{h_\theta(s,b)}}{\sum_b e^{h_\theta(s,b)}} \nabla_\theta h_\theta(s,b) \\
& \nabla_\theta h_\theta(s,a) - \sum_b \pi_\theta(s,b) \nabla_\theta h_\theta(s,b)
\end{aligned}$$

# Derivatives of expectations

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

For Gaussian $P_\theta(x)$

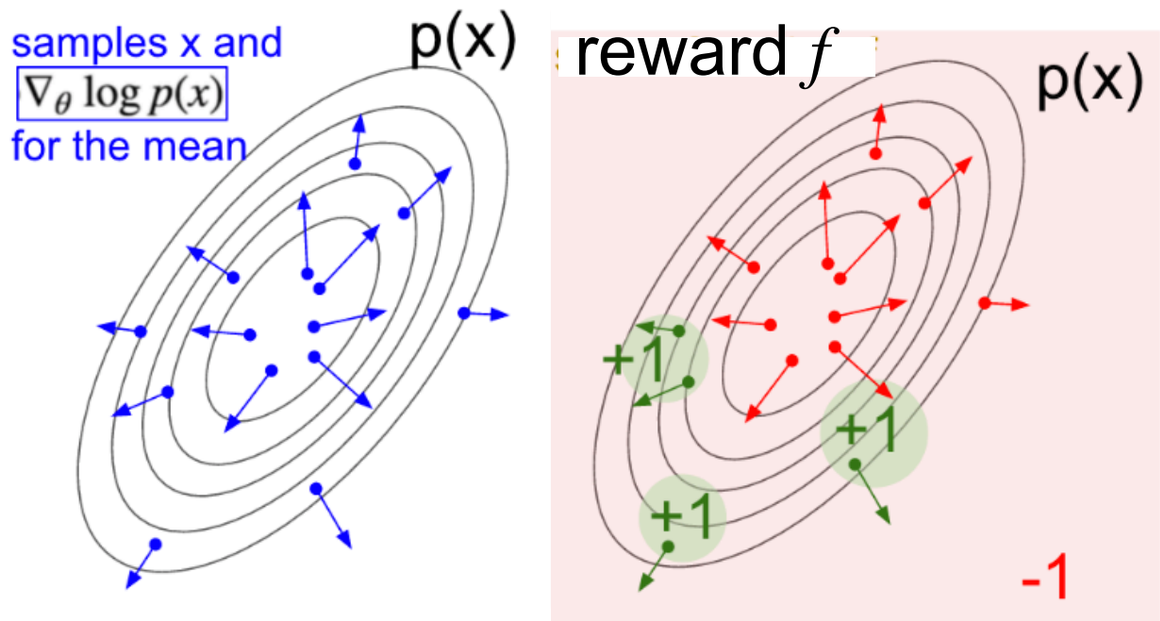


# Derivatives of expectations

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

For Gaussian $P_\theta(x)$

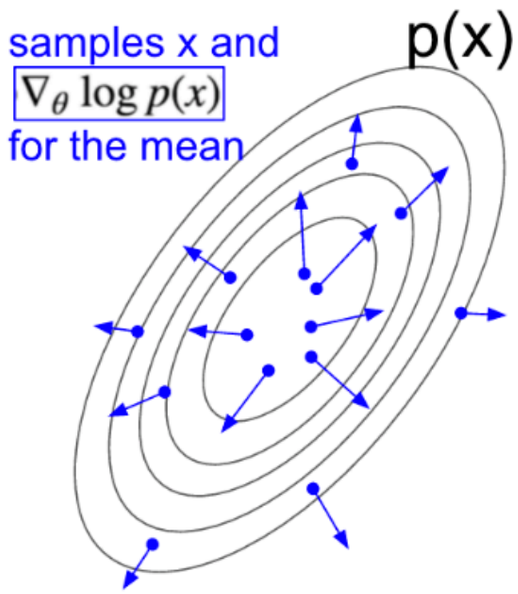
What is $\theta$ here?

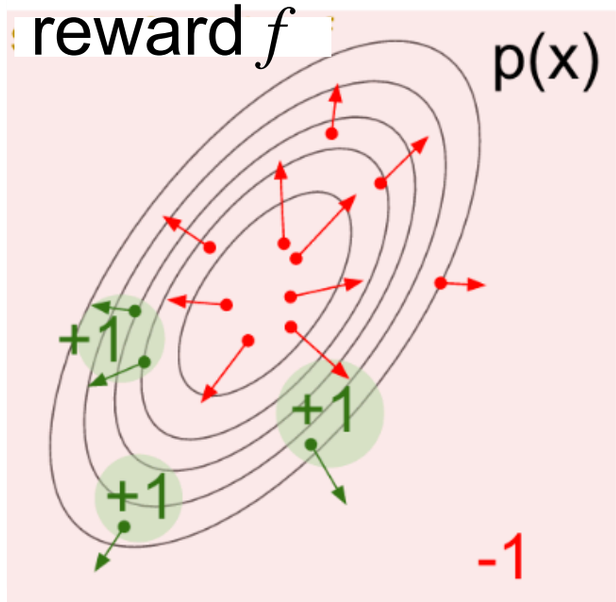


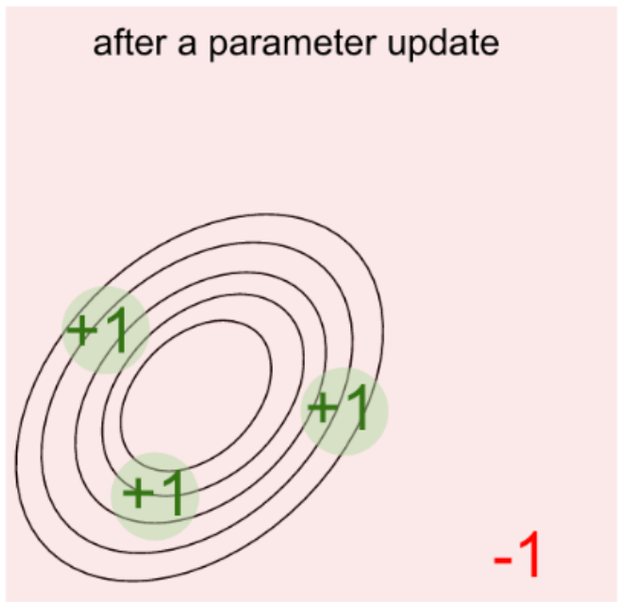
# Derivatives of expectations

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

For Gaussian $P_\theta(x)$



# Policy Gradients

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Sample estimate:

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

Considers distribution over policy paran

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

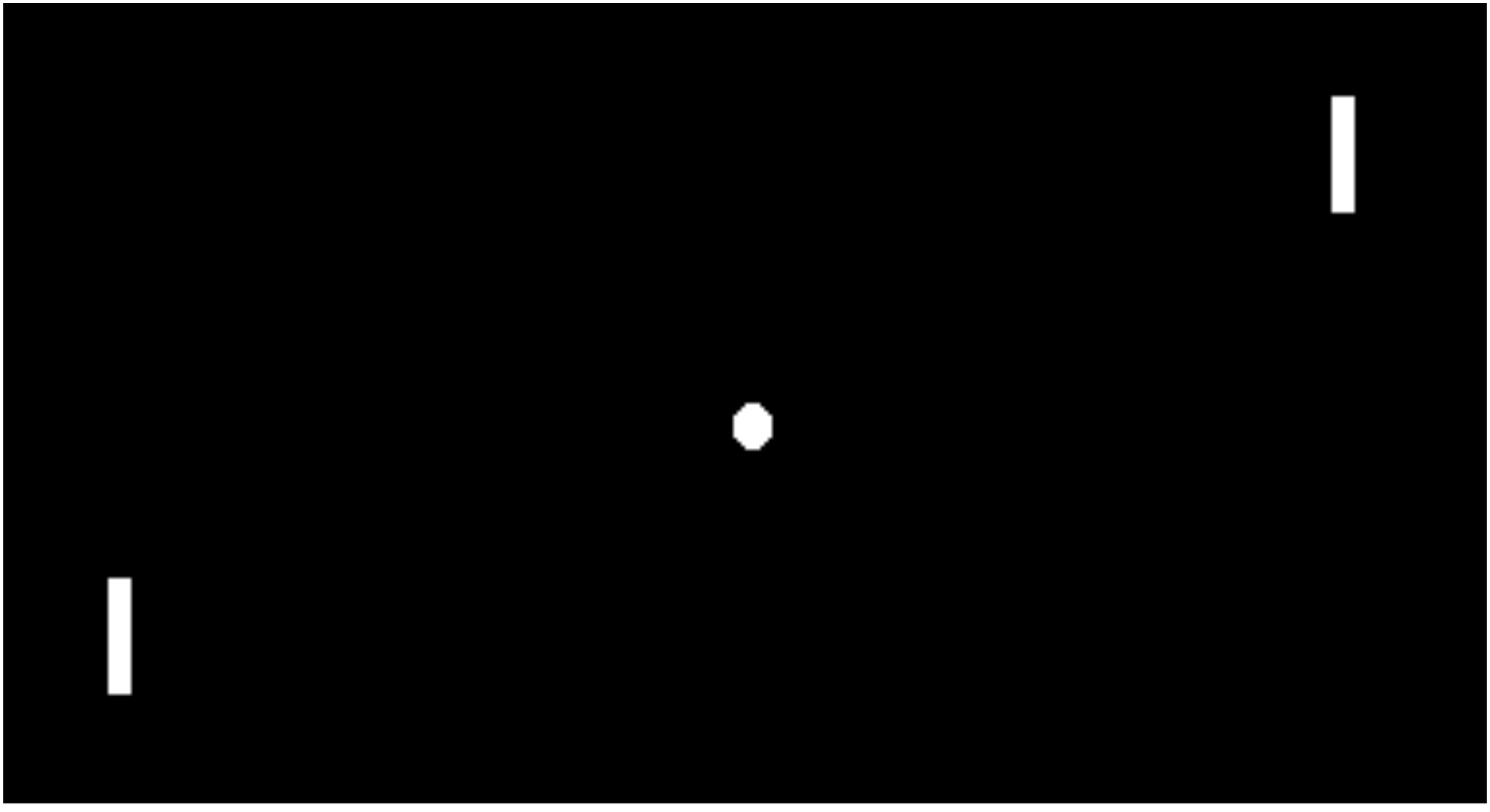
$$\begin{aligned}
\nabla_\mu U(\mu) &= \nabla_\mu \mathbb{E}_{\theta \sim P_\mu(\theta)} \left[ F(\theta) \right] \\
&= \nabla_\mu \int P_\mu(\theta) F(\theta) d\theta \\
&= \int \nabla_\mu P_\mu(\theta) F(\theta) d\theta \\
&= \int P_\mu(\theta) \frac{\nabla_\mu P_\mu(\theta)}{P_\mu(\theta)} F(\theta)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

Sample estimate:

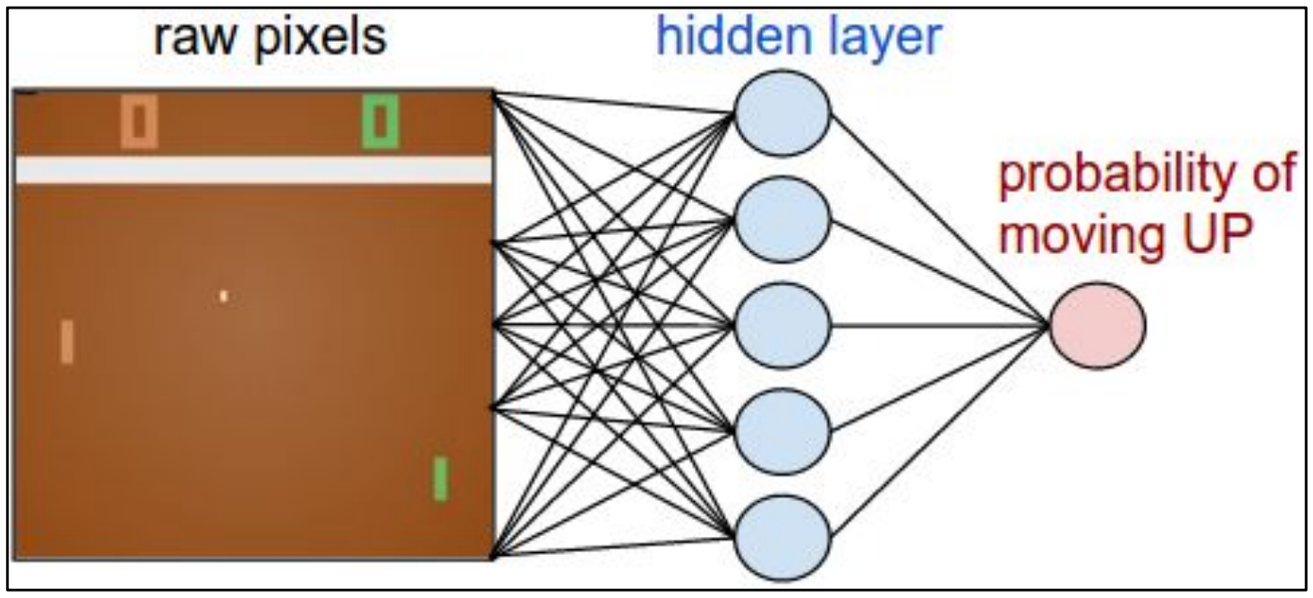
$$\nabla_\mu U(\mu) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\mu \log P_\mu(\theta^{(i)}) F(\theta^{(i)})$$

# Pong from Pixels

Slides from Andrei Karpathy



# Policy network

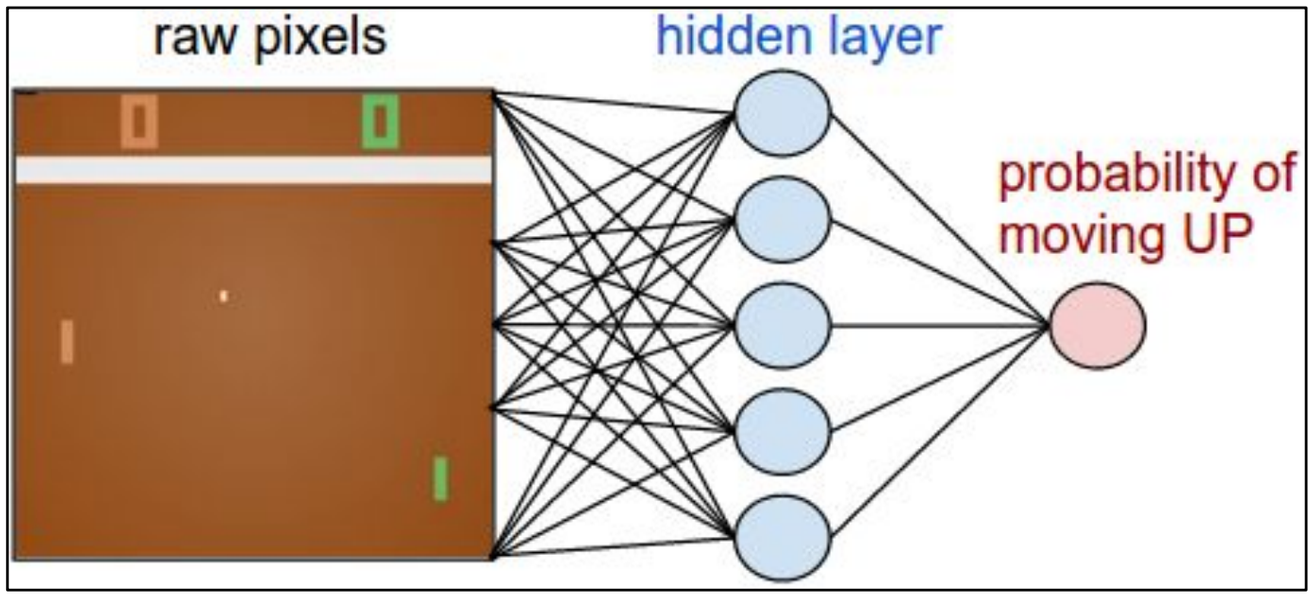


# Policy network

e.g.,

height width

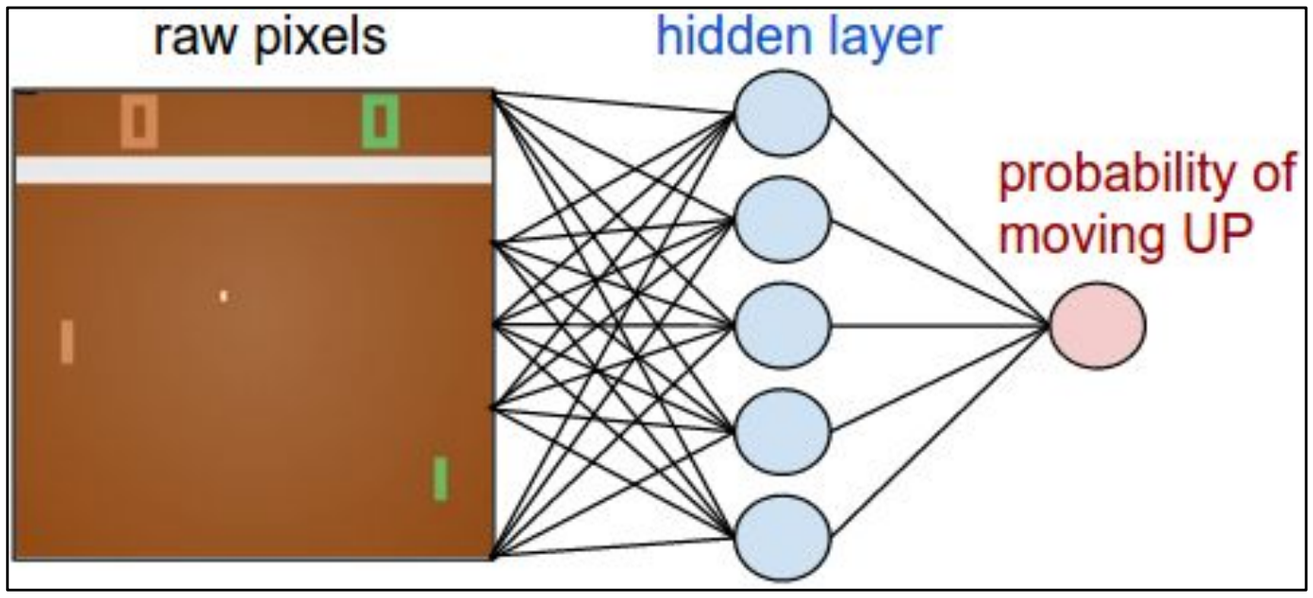
[**80** x **80**]
array of



# Policy network

height width

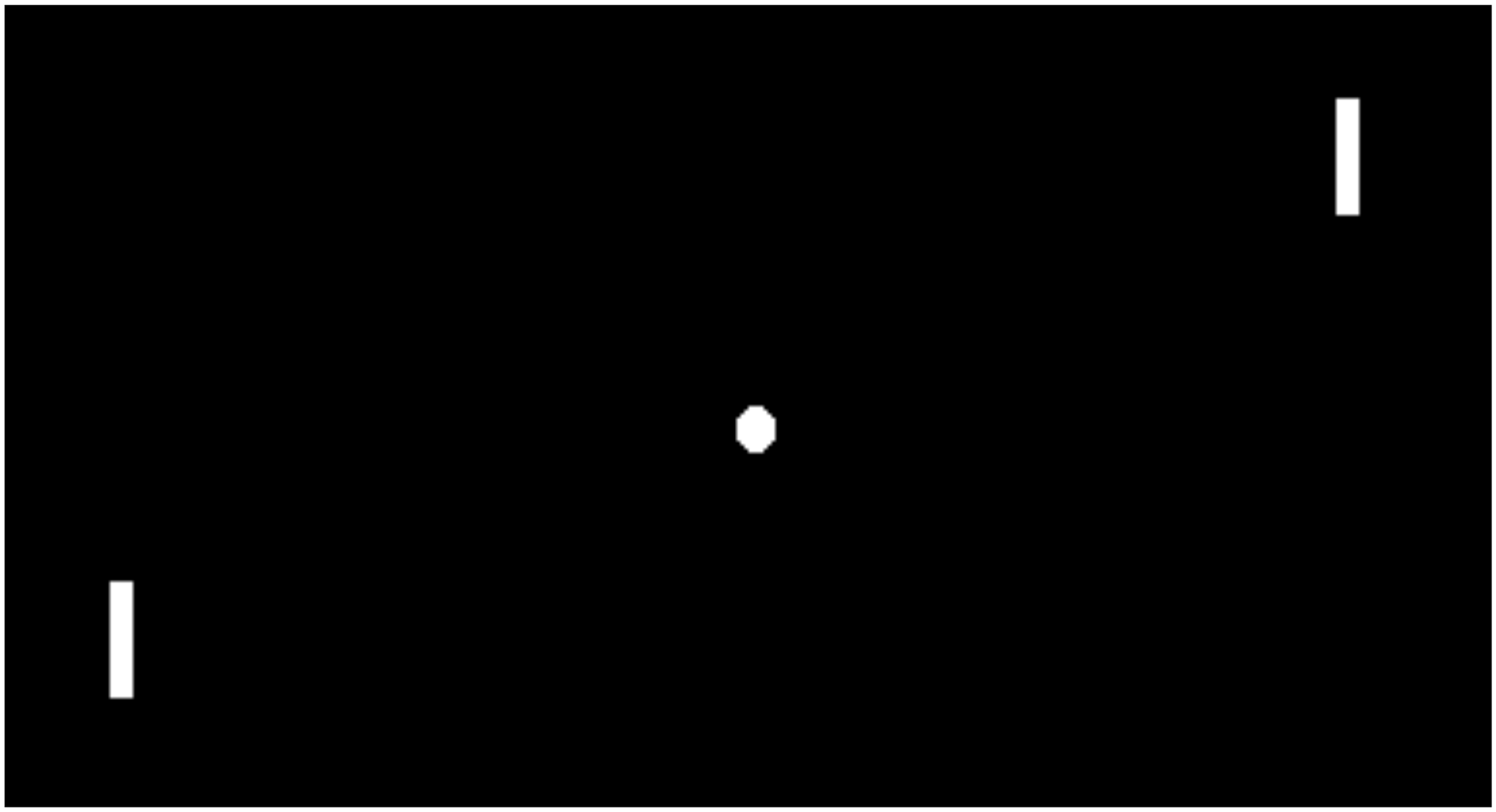
[**80** x **80**]
array




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

[(80*80)*200 + 200] + [200*1 + 1] = **~1.3M parameters**

Layer 1 Layer 2



Network does not see this. Network sees 80*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

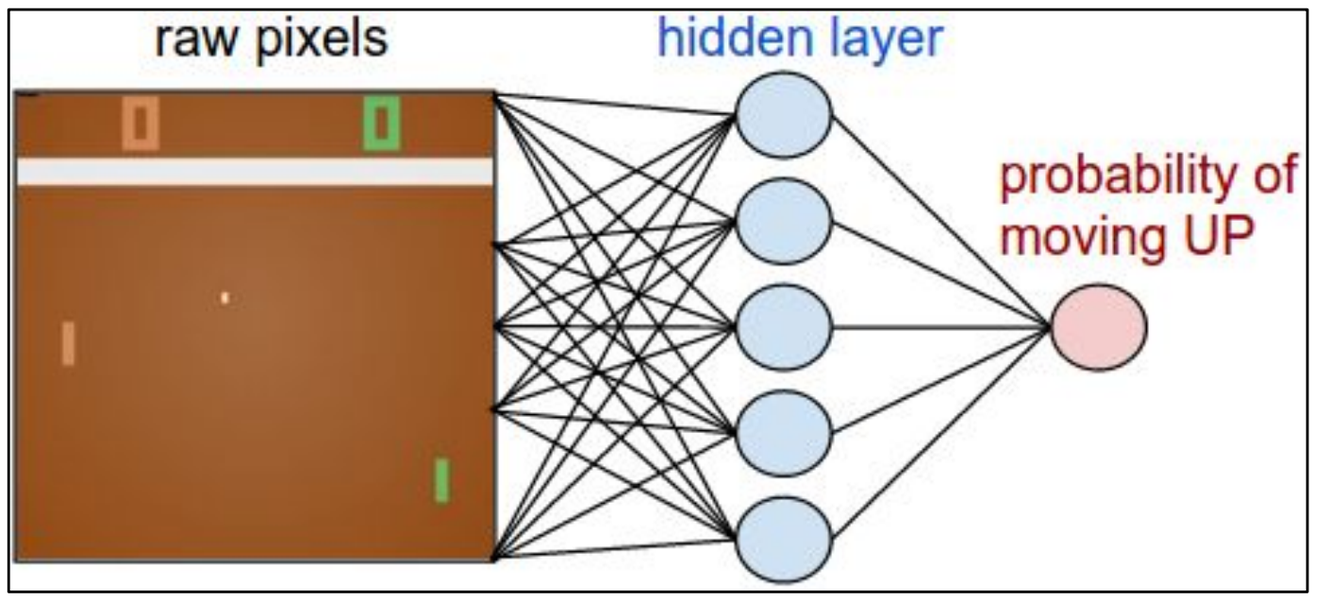
Policy gradients

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)

(x1,UP)
(x2,DOWN)
(x3,UP)
…

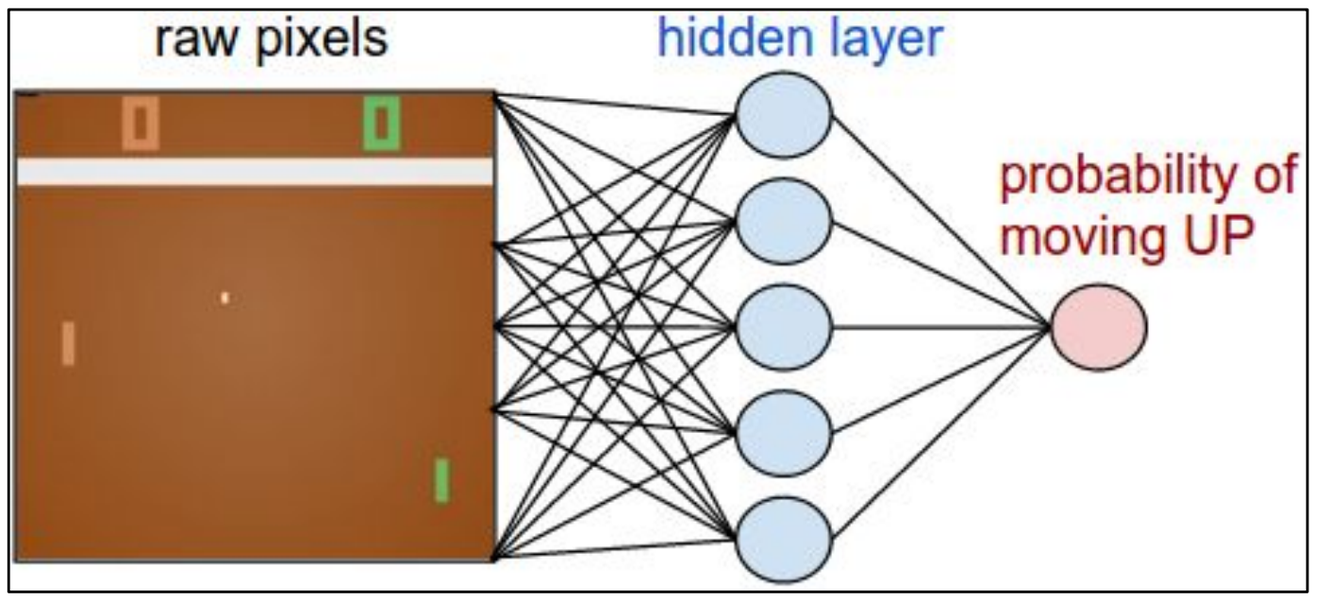


Suppose we had the training labels…
(we know what to do in any state)

(x1,UP)
(x2,DOWN)
(x3,UP)
…

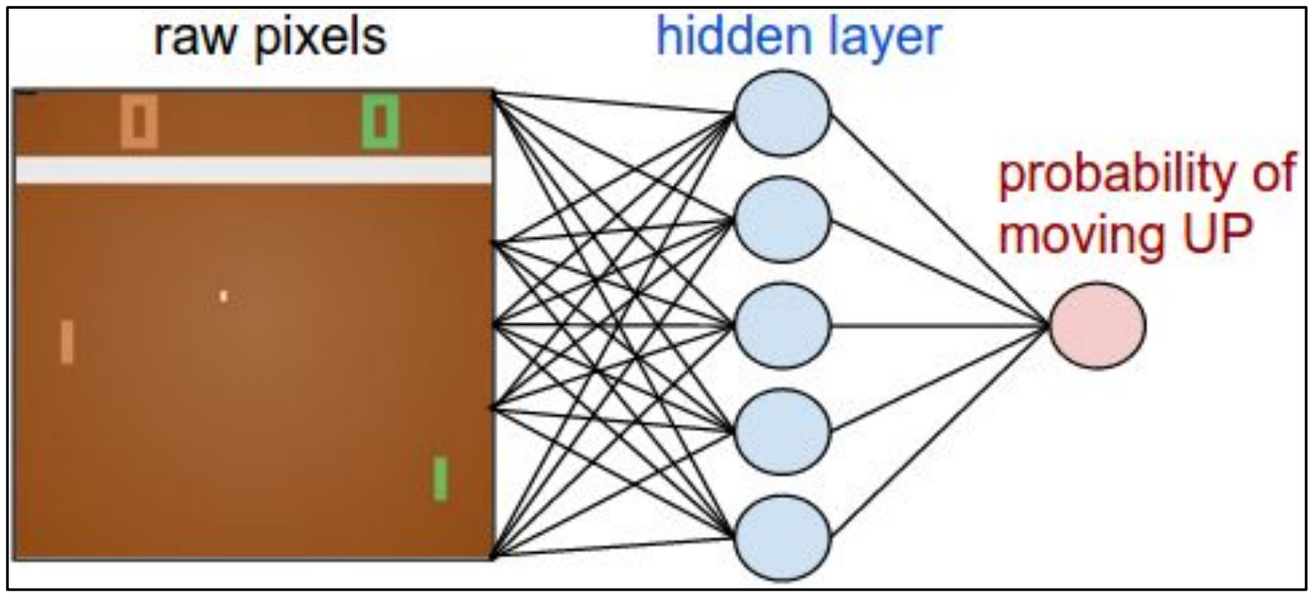
maximize:

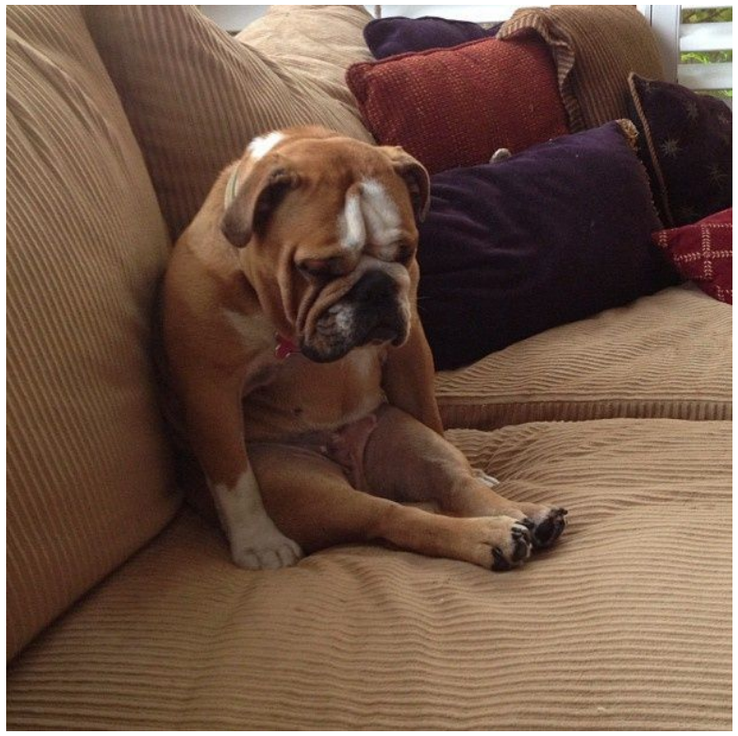
$$\sum_i \log p(y_i \mid x_i)$$




supervised learning

# Except, we don’t have labels...



Should we go UP or DOWN?

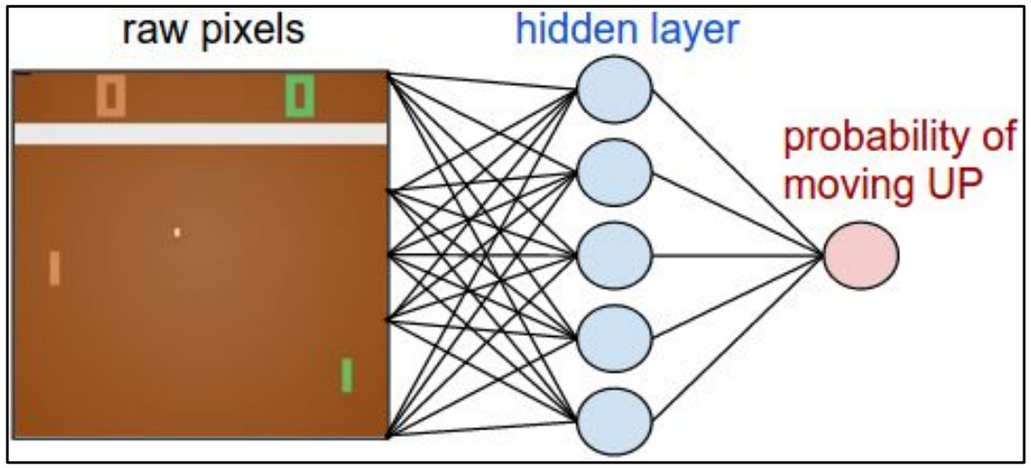
Except, we don’t have labels...

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

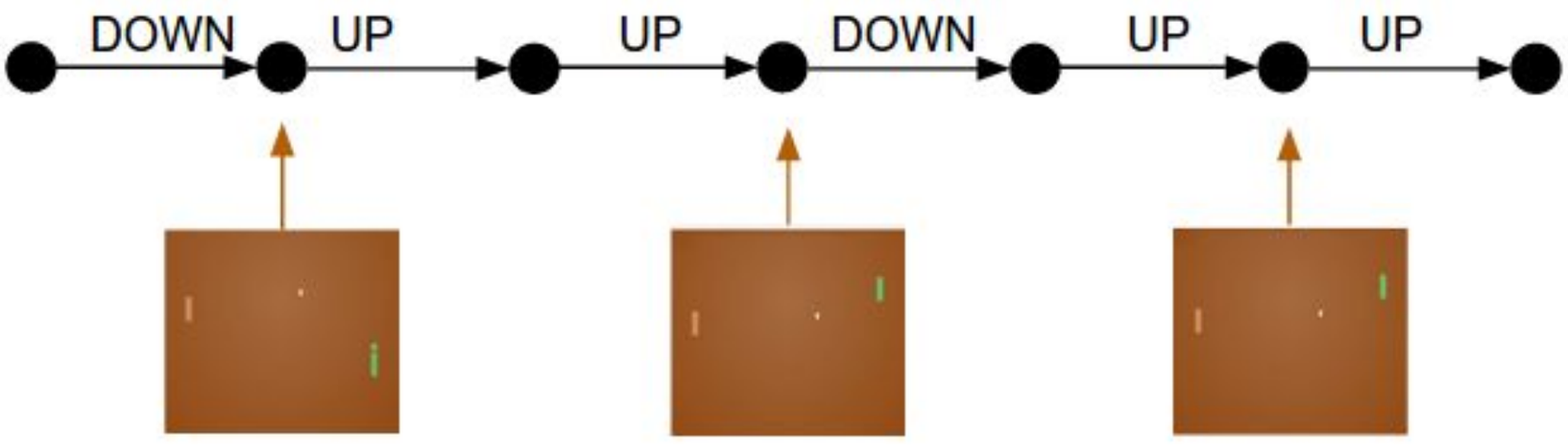
-RL

trial-and-error learning

# Let’s just act according to our current policy...



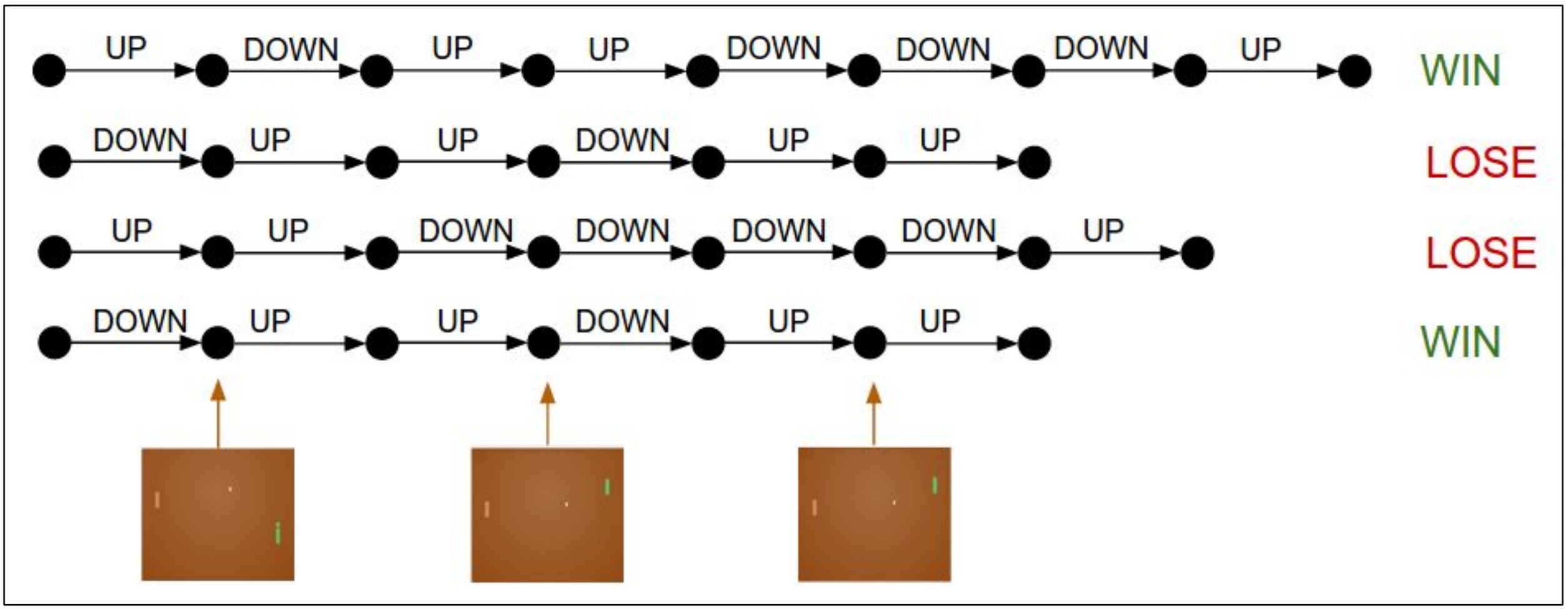

Rollout the policy and collect an episode



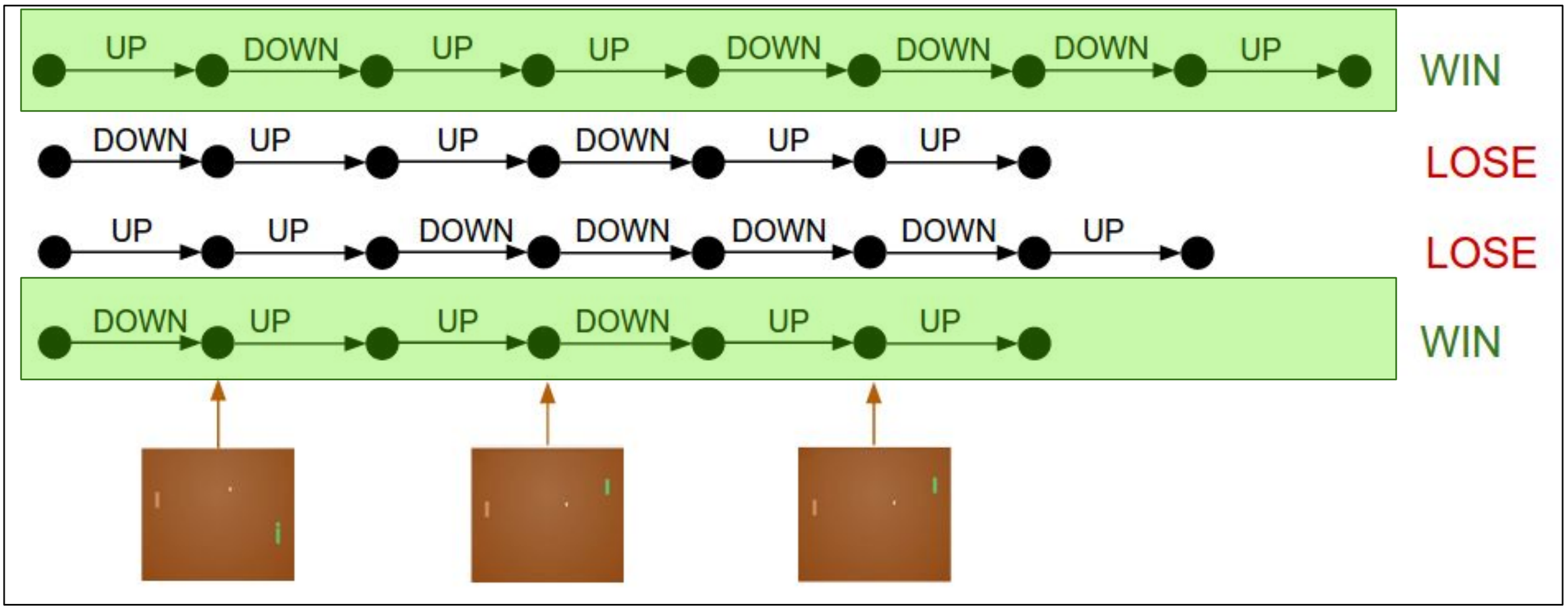

WIN

# Collect many rollouts...

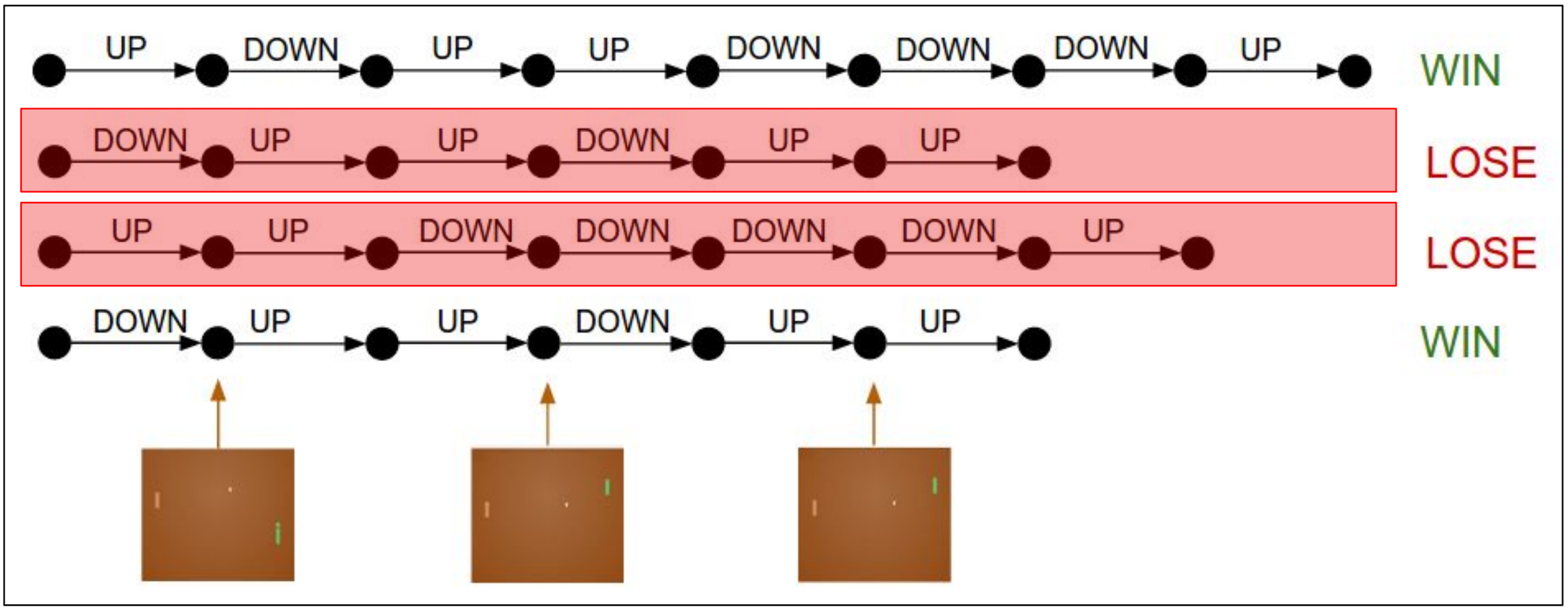
**4 rollouts:**



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



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

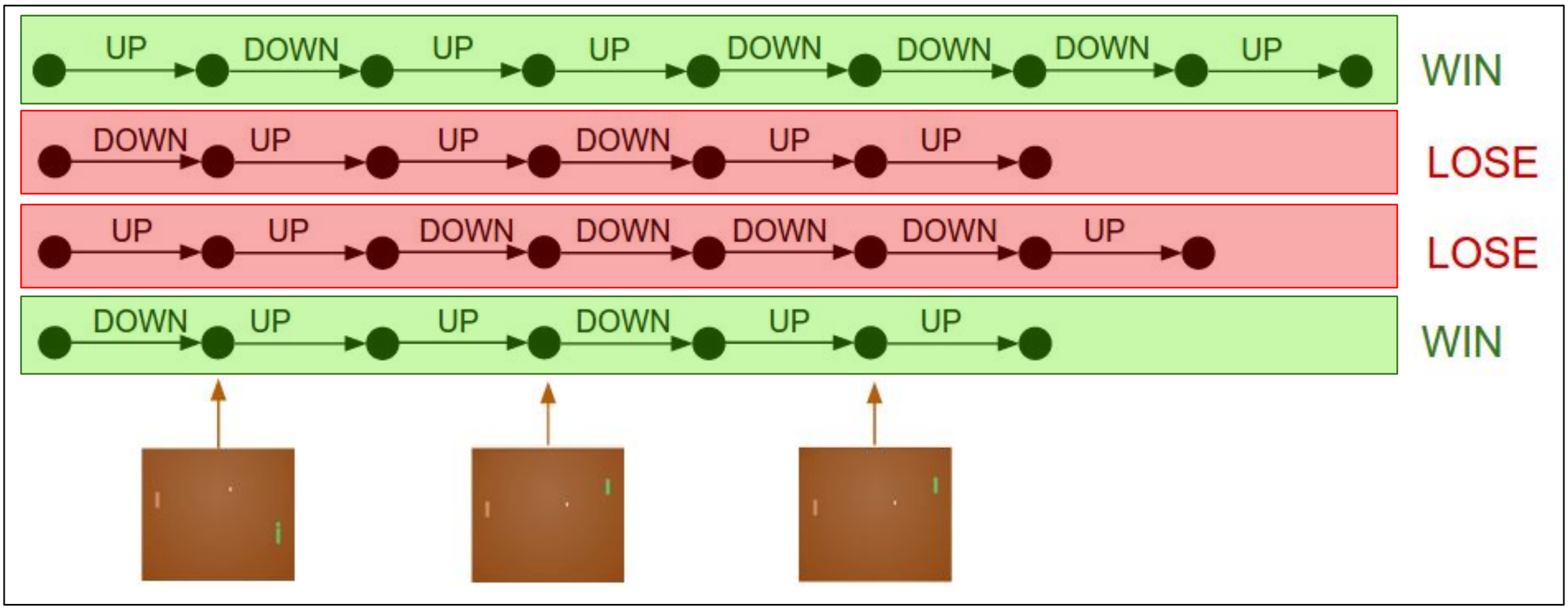


Pretend every action we took here was the correct label.

maximize: $\log p(y_i \mid x_i)$

Pretend every action we took here was the wrong label.

maximize: $(-1) * \log p(y_i \mid x_i)$




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

# 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.

Advantage is the same for all actions taken during a trajectory, and depends on the trajectory return (episode return)

# 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?

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

# Temporal structure

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

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

Each action takes the blame for the full trajectory!

# Temporal structure

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

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

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

Each action takes the blame for the full trajectory!

These rewards are not caused by actions that come after t

# Temporal structure

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

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

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

Each action takes the blame for the full trajectory!

Consider instead:

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

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:

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

- Let's us rewrite is as follows:

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha \gamma^t G_t \frac{\nabla_{\boldsymbol{\theta}} \pi \left(A_t \mid S_t, \boldsymbol{\theta}\right)}{\pi \left(A_t \mid S_t, \boldsymbol{\theta}\right)}$$

- 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:

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

- Let's us rewrite is as follows: move most in the directions that favor actions that yield the highest return

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha \gamma^t G_t \frac{\nabla_{\boldsymbol{\theta}} \pi \left(A_t \mid S_t, \boldsymbol{\theta}\right)}{\pi \left(A_t \mid S_t, \boldsymbol{\theta}\right)}$$

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:

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

The trace of the covariance matrix:

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

Our goal is to minimize the variance of our estimator

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

# Reducing variance with baseline

What if we subtract a constant b from the rewards:

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

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

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

# Reducing variance with baseline

What if we subtract a constant b from the rewards:

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

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

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

# Reducing variance with baseline

What if we subtract a constant b from the rewards:

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

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

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

# 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:

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

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

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

We still have an unbiased estimator of the gradient!

# Baseline choices

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

Constant Baseline: $b = \mathbb{E}[R(\tau)]$

Time-dependant 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 $\theta$, baseline $b$

**for** iteration=$1, 2, \ldots$ **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 $\nabla_\theta \log \pi(a_t \mid s_t, \theta) \hat{A}_t$

**end for**

~ [Williams, 1992]

# Baseline choices

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

Constant Baseline: $b = \mathbb{E}[R(\tau)]$

Time-dependant Baseline: $b_t = \sum_{i=1}^{N}\sum_{k=t}^{H} R(s_k^{(i)}, a_k^{(i)})$

State dependant expected return:

$$b\left(s_t\right) = \mathbb{E}\left[r_t + r_{t+1} + r_{t+2} + \ldots + r_{H-1}\right] = V_\pi\left(s_t\right)$$

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

# Variance

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

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

- Imagine in some state $S_1$ the rewards of all actions are ~3000 and in $S_2$ ~ -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 thatn 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 $V_\pi(s_t)$

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

- MC estimation
- Initialize $\phi$

Collect trajectories: $\tau_i, \ldots, \tau_N$

Regress against empirical return:

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

# Estimate $V_\pi(s_t)$

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

- TD estimation
- Initialize $\phi$

Collect data for: $s, a, s', r$

Fitted V iteration

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

Bootstrapping!

# Better estimates for cumulative future reward

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

We are essentially attempting to estimate Q from a single rollout:

$$Q^\pi(s,a) = \mathbb{E}[R_0 + R_1 + \cdots \mid S_0 = s, A_0 = a]$$

Minimize variance by:

- discounting
- introducing a learnt approximation for the extected return (critic), as opposed to use MC samples

# Actor-Critic

Reducing variance using a critic

$$Q^{\pi,\gamma}(s,a) = \mathbb{E}[R_0 + \gamma R_1 + \gamma^2 R_2 \cdots \mid S_0 = s, A_0 = a]$$

# Actor-Critic

Reducing variance using a critic

$$
\begin{aligned}
Q^{\pi,\gamma}(s,a) &= \mathbb{E}[R_0 + \gamma R_1 + \gamma^2 R_2 \cdots \,|\, S_0 = s, A_0 = a] \\
&= \mathbb{E}[R_0 + \gamma V^{\pi}(S_1) \,|\, S_0 = s, A_0 = a]
\end{aligned}
$$

# Actor-Critic

Reducing variance using a critic

$$
\begin{aligned}
Q^{\pi,\gamma}(s,a) &= \mathbb{E}[R_0 + \gamma R_1 + \gamma^2 R_2 \cdots \,|\, S_0 = s, A_0 = a] \\
&= \mathbb{E}[R_0 + \gamma V^{\pi}(S_1) \,|\, S_0 = s, A_0 = a] \\
&= \mathbb{E}[R_0 + \gamma R_1 + \gamma^2 V^{\pi}(S_2) \,|\, S_0 = s, A_0 = a]
\end{aligned}
$$

# Actor-Critic

Reducing variance using a critic

$$
\begin{aligned}
Q^{\pi,\gamma}(s,a) &= \mathbb{E}[R_0 + \gamma R_1 + \gamma^2 R_2 \cdots \mid S_0 = s, A_0 = a] \\
&= \mathbb{E}[R_0 + \gamma V^{\pi}(S_1) \mid S_0 = s, A_0 = a] \\
&= \mathbb{E}[R_0 + \gamma R_1 + \gamma^2 V^{\pi}(S_2) \mid S_0 = s, A_0 = a] \\
&= \mathbb{E}[R_0 + \gamma R_1 + \gamma^2 R_2 + \gamma^3 V^{\pi}(S_3) \mid S_0 = s, A_0 = a] \\
&= \ldots
\end{aligned}
$$

# 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

# REINFORCE/Actor-critic training

- Stability of training neural networks requires the **gradient updates to be de-correlated**
- This is not the case if data arrives **sequentially**
- Gradient updates computed from some part of the space can cause the value (Q) function approximator to **oscillate**
- Our solution so far has been: **Experience buffers** where experience tuples are mixed and sampled from. Resulting sampled batches are more stationary that the ones encountered online (without buffer)
- This limits deep RL to **off-policy** methods, since data from an older policy are used to update the weights of the value approximator (critic) (except if we take care and weight such data under our current stochastic policy->importance sampling)

# Asynchronous Deep RL for on policy learning

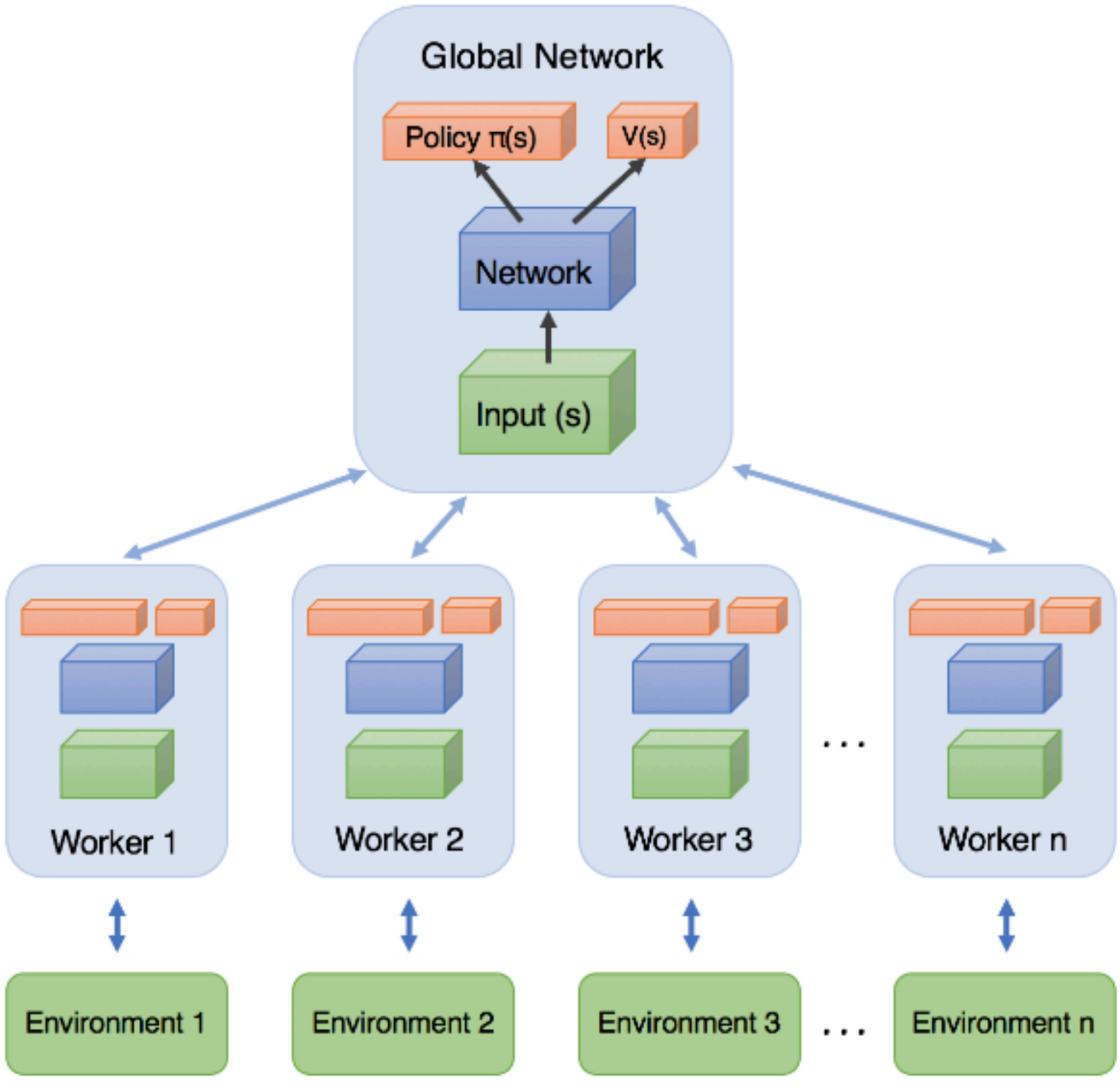
**Asynchronous Methods for Deep Reinforcement Learning**


**Volodymyr Mnih**[1] VMNIH@GOOGLE.COM
**Adrià Puigdomènech Badia**[1] ADRIAP@GOOGLE.COM
**Mehdi Mirza**[1,2] MIRZAMOM@IRO.UMONTREAL.CA
**Alex Graves**[1] GRAVESA@GOOGLE.COM
**Tim Harley**[1] THARLEY@GOOGLE.COM
**Timothy P. Lillicrap**[1] COUNTZERO@GOOGLE.COM
**David Silver**[1] DAVIDSILVER@GOOGLE.COM
**Koray Kavukcuoglu** [1] KORAYK@GOOGLE.COM

[1] Google DeepMind
[2] Montreal Institute for Learning Algorithms (MILA), University of Montreal


- Alternative: parallelize the collection of experience and stabilize training **without experience buffers**
- **Multiple threads of experience**, one per agent, each exploring in different part of the environment contributing experience tuples
- **Different exploration strategies** (e.g., various $\epsilon$ values) in different threads increase diversity
- Now you can train on-policy using any of our policy gradient methods

# Distributed RL



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

*// Assume global shared parameter vectors* $\theta$ *and* $\theta_v$ *and global shared counter* $T = 0$
*// Assume thread-specific parameter vectors* $\theta'$ *and* $\theta'_v$
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 $\theta'_v = \theta_v$ — Copying the weights
  $t_{start} = t$
  Get state $s_t$
  **repeat**
    Perform $a_t$ according to policy $\pi(a_t|s_t;\theta')$ — Rollout
    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** $i \in \{t-1, \ldots, t_{start}\}$ **do**
    $R \leftarrow r_i + \gamma R$
    Accumulate gradients wrt $\theta'$: $d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi(a_i|s_i;\theta')(R - V(s_i;\theta'_v))$ — Advantage
    Accumulate gradients wrt $\theta'_v$: $d\theta_v \leftarrow d\theta_v + \partial (R - V(s_i;\theta'_v))^2 / \partial \theta'_v$ — Learning the critic
  **end for**
  Perform asynchronous update of $\theta$ using $d\theta$ and of $\theta_v$ using $d\theta_v$.
**until** $T > T_{max}$

$s_1, s_2, s_3, s_4$

$r_1, r_2, r_3$

What is the approximation used for the advantage?

$R_3 = r_3 + \gamma V(s_4, \theta'_v)$ $\quad A_3 = R_3 - V(s_3;\theta'_v)$

$R_2 = r_2 + \gamma r_3 + \gamma^2 V(s_4, \theta'_v)$ $\quad A_2 = R_2 - V(s_2;\theta'_v)$

$R_3 - V(s_3)$

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

*// Assume global shared parameter vectors* $\theta$ *and* $\theta_v$ *and global shared counter* $T = 0$
*// Assume thread-specific parameter vectors* $\theta'$ *and* $\theta'_v$
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 $\theta'_v = \theta_v$

*``We also found that adding the entropy of the policy π to the objective function improved exploration by discouraging premature convergence to suboptimal deterministic policies." So you need to add to the policy gradient:* $+\beta \nabla_\theta \mathrm{H}(\pi_\theta(a_t \mid s_t; \theta))$

*We will look into the entropy as part of the reward in later lecture*

Perform asynchronous update of $\theta$ using $d\theta$ and of $\theta_v$ using $d\theta_v$.
**until** $T > T_{max}$

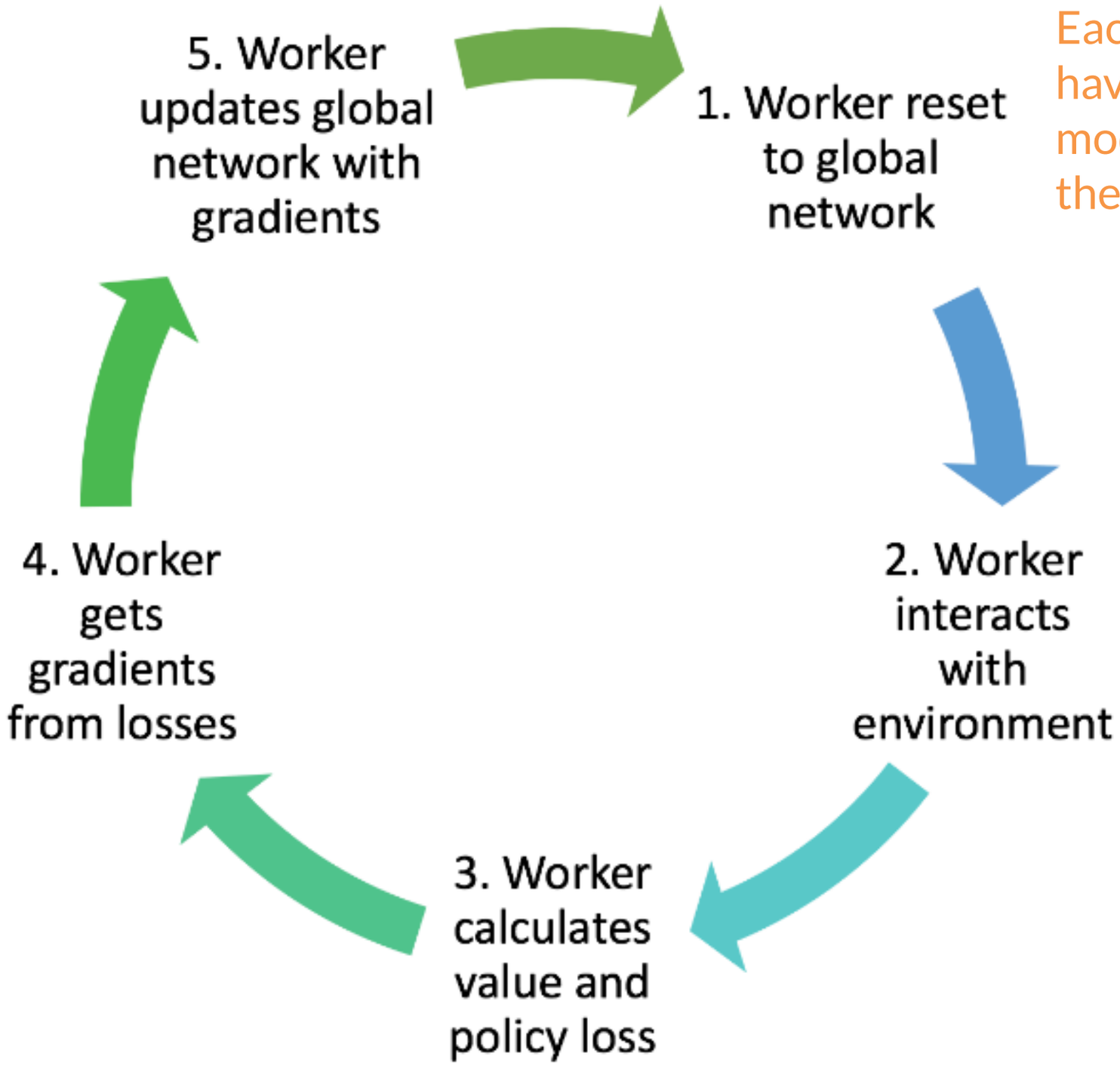
What is the approximation used for the advantage?

$R_3 = r_3 + \gamma V(s_4, \theta'_v)$ $\quad A_3 = R_3 - V(s_3; \theta'_v)$

$R_2 = r_2 + \gamma r_3 + \gamma^2 V(s_4, \theta'_v)$ $\quad A_2 = R_2 - V(s_2; \theta'_v)$

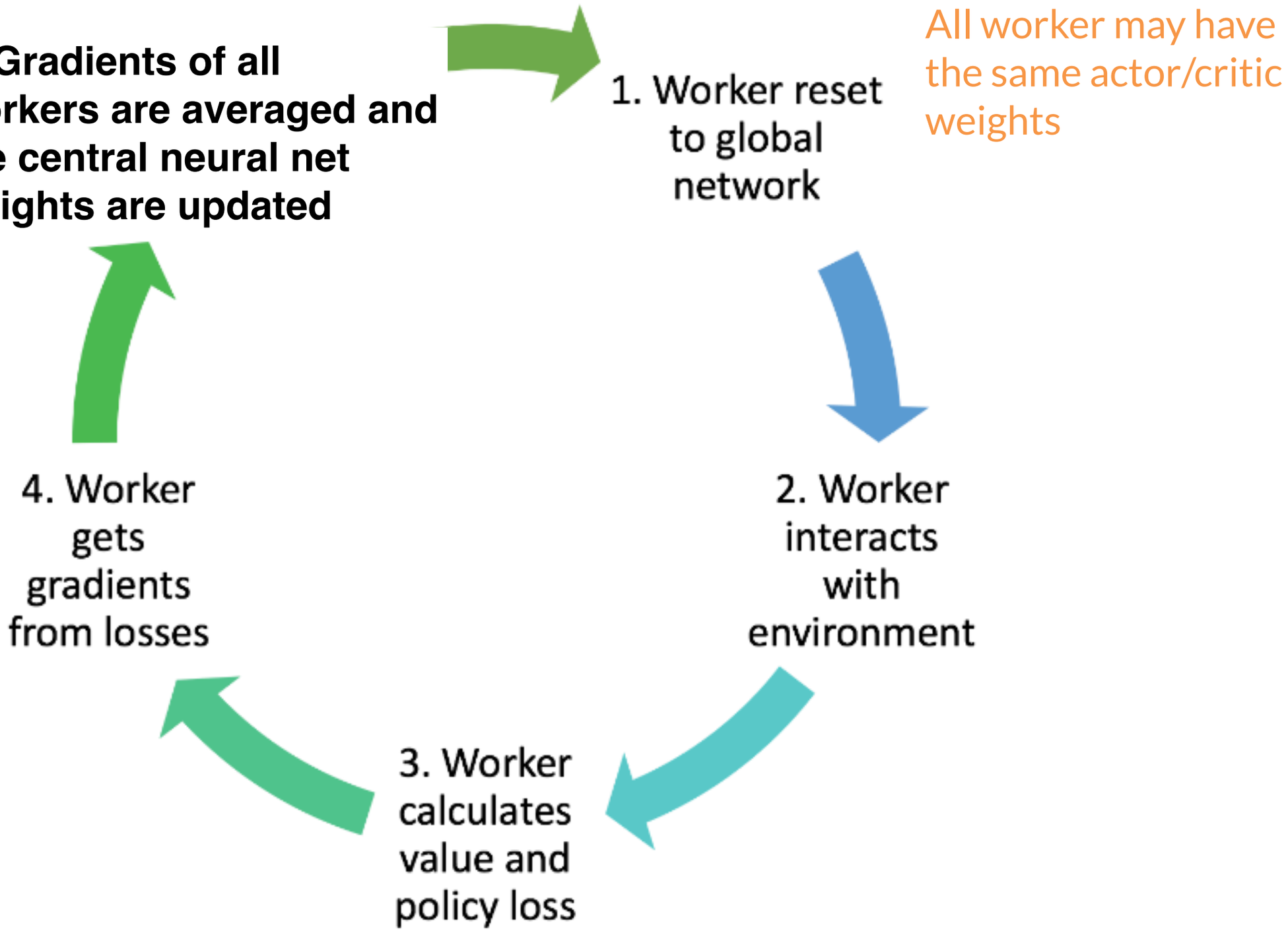
# Distributed Asynchronous RL-A3C

No locking



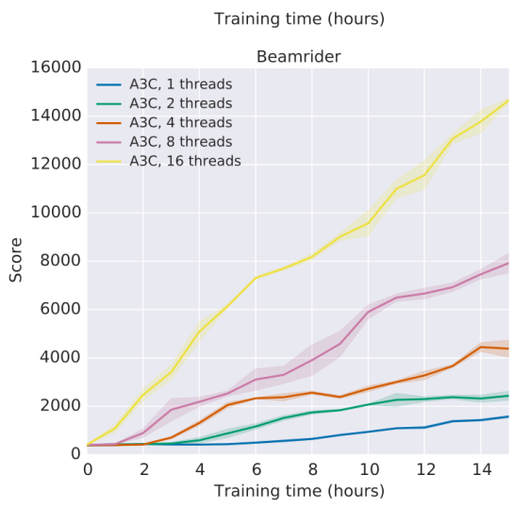

Each worker may have slightly modified version of the policy/critic

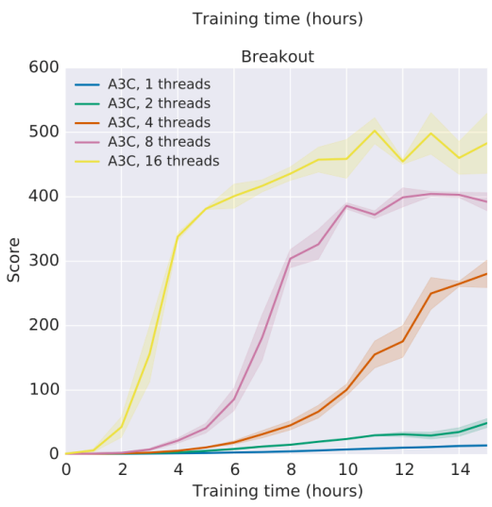
# Distributed Synchronous RL-A2C

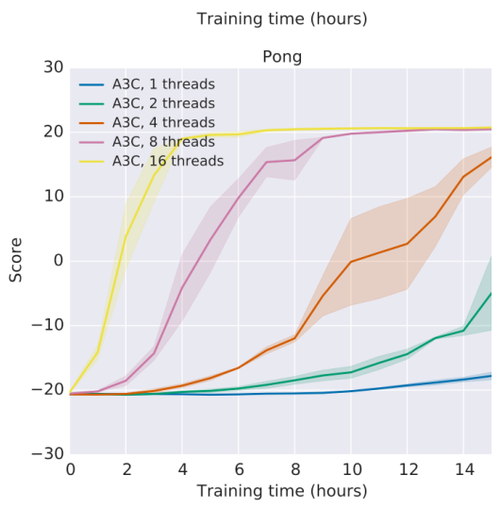


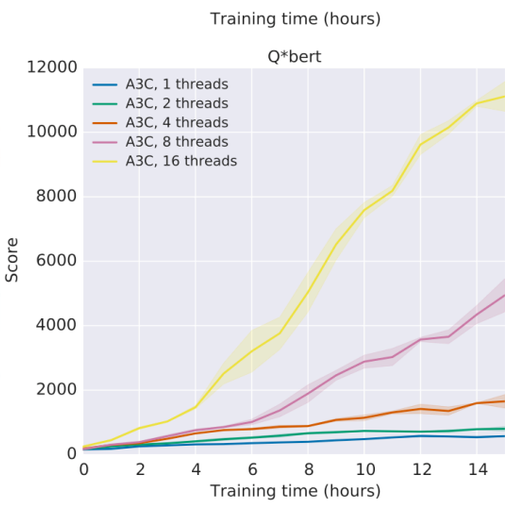

All worker may have the same actor/critic weights

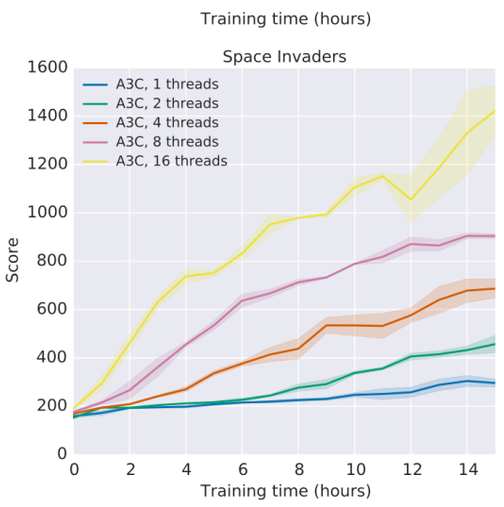
# Advantages of Asynchronous (multi-threaded) RL



# Summary of policy gradients so far

- The policy gradients has many equivalent forms:

$$\nabla_\theta U(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a) v_t\right] \quad \text{REINFORCE}$$

$$= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a) Q^w(s,a)\right] \quad Q \text{ Actor-Critic}$$

$$= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a) A^w(s,a)\right] \quad \text{Advantage Actor-Critic}$$

- Each leads to a stochastic gradient descent algorithm
- Critic uses policy evaluation (for e.g. MC or TD learning) to estimate

$Q^\pi(s,a), A^\pi(s,a)$ or $V^\pi(s)$