6.7900 Machine Learning (Fall 2023)

Lecture 17: Reinforcement Learning Cont'd (supporting slides)

Shen Shen

Outline

- Value-based RL
  - (Tabular) Q-learning

- Policy-based RL
  - What does the policy gradient do?
  - Policy gradient derivation
  - Policy gradient estimates
  - Variance reduction
    - Constant Baselines
    - Temporal structure
    - Actor-critic intro

References

- More RL-flavored presentation:
  - Reinforcement Learning: An Introduction, Sutton and Barton; The MIT Press, 2018. Chapter 6 and 13

- Seminal papers referenced on slides.
- Some slides adapted from: Philip Isola, Pieter Abbeel, and Andrej Karpathy

Reinforcement Learning

	Actions Don't Impact State	Actions Change State
Unknown Model	Multi-Armed Bandits	Reinforcement Learning
Known Model	Stochastic Optimization	Markov Decision Process

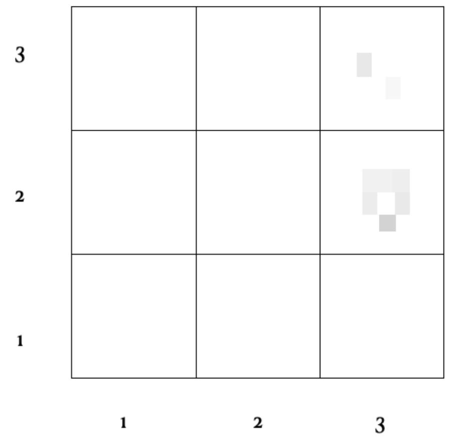
~~Markov Decision Process~~ RL

- $\mathscr{S}$: a state space which contains all possible states s of the system.
- $\mathscr{A}$: an action space which contains all possible actions a an agent can take.
- ~~$P(s' \mid s, a)$: the probability of transition from state s to s' if action a~~ is taken.
- ~~$R(s, a)$: a function that takes in the (state and action) and returns a real-valued~~ reward.

Sometimes, also:

- s_0: initial state.
- Objective version (may involve a $\gamma \in [0,1]$: discount factor (details later), and/or T: horizon. Details later).

Example: Grid World (in RL)



(Almost deterministic) Transitions:

- Normally, actions take us deterministically to the "intended" state. E.g., in state (1,1), action "North" gets us to state (1,2)
- If an action would take us out of this world, stay put
- In state (3,2), action "North" leads to two possible next state:
  - chance ends in (3,3)
  - chance ends in (2,3)

State space: 9 cells

Actions space: {North, South, East, West}

Discount $\gamma = 0.9$

Deterministic Rewards:

- State (3,3), any action gets reward
- State (3,2), any action gets reward
- Any other (state, action) pairs get reward 0

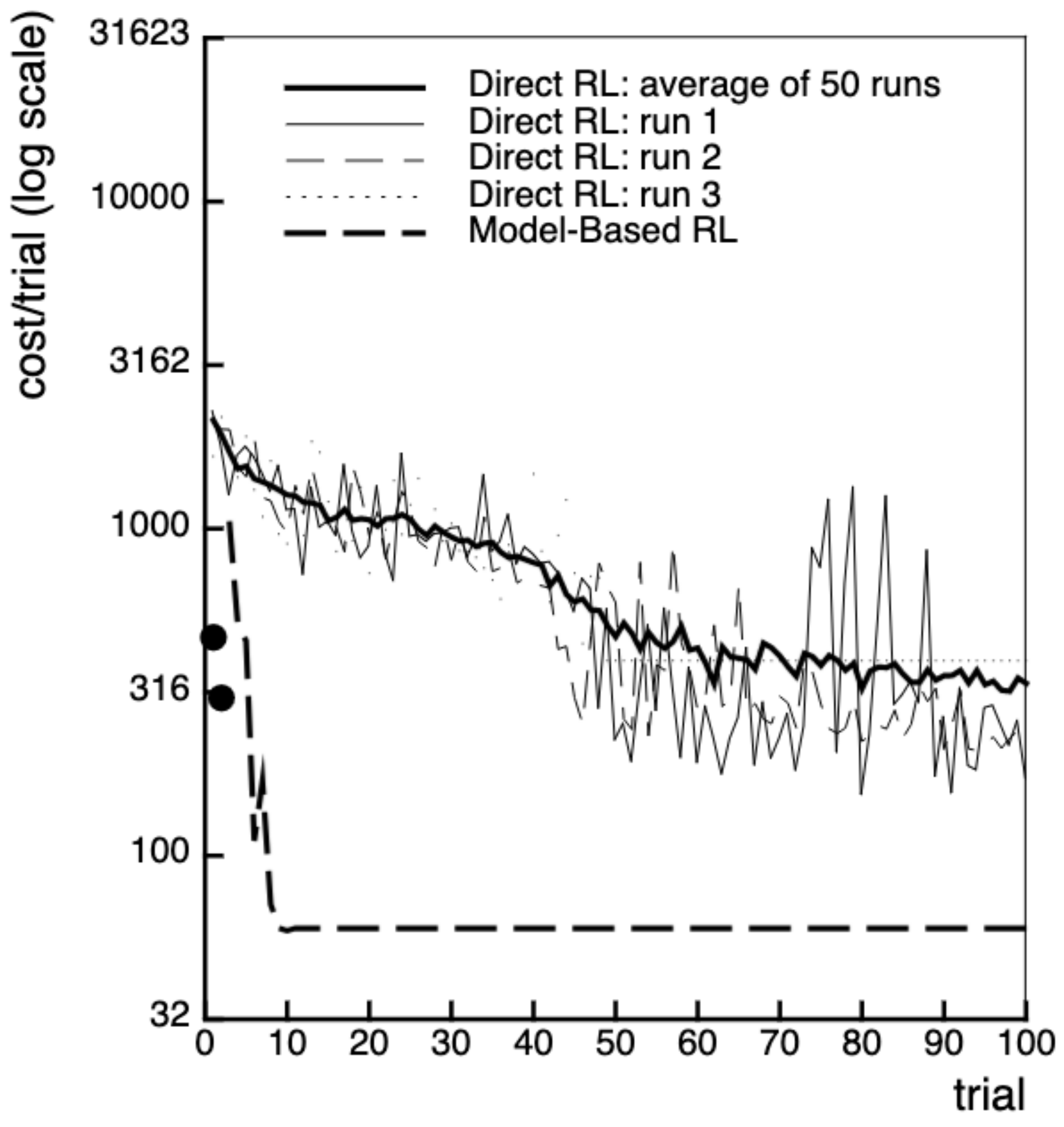
RL ~~MDP~~ Goal

- Find a policy $\pi : S \to A$, such that:

  $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \mid s_0 = s]$ is maximized for all s_0

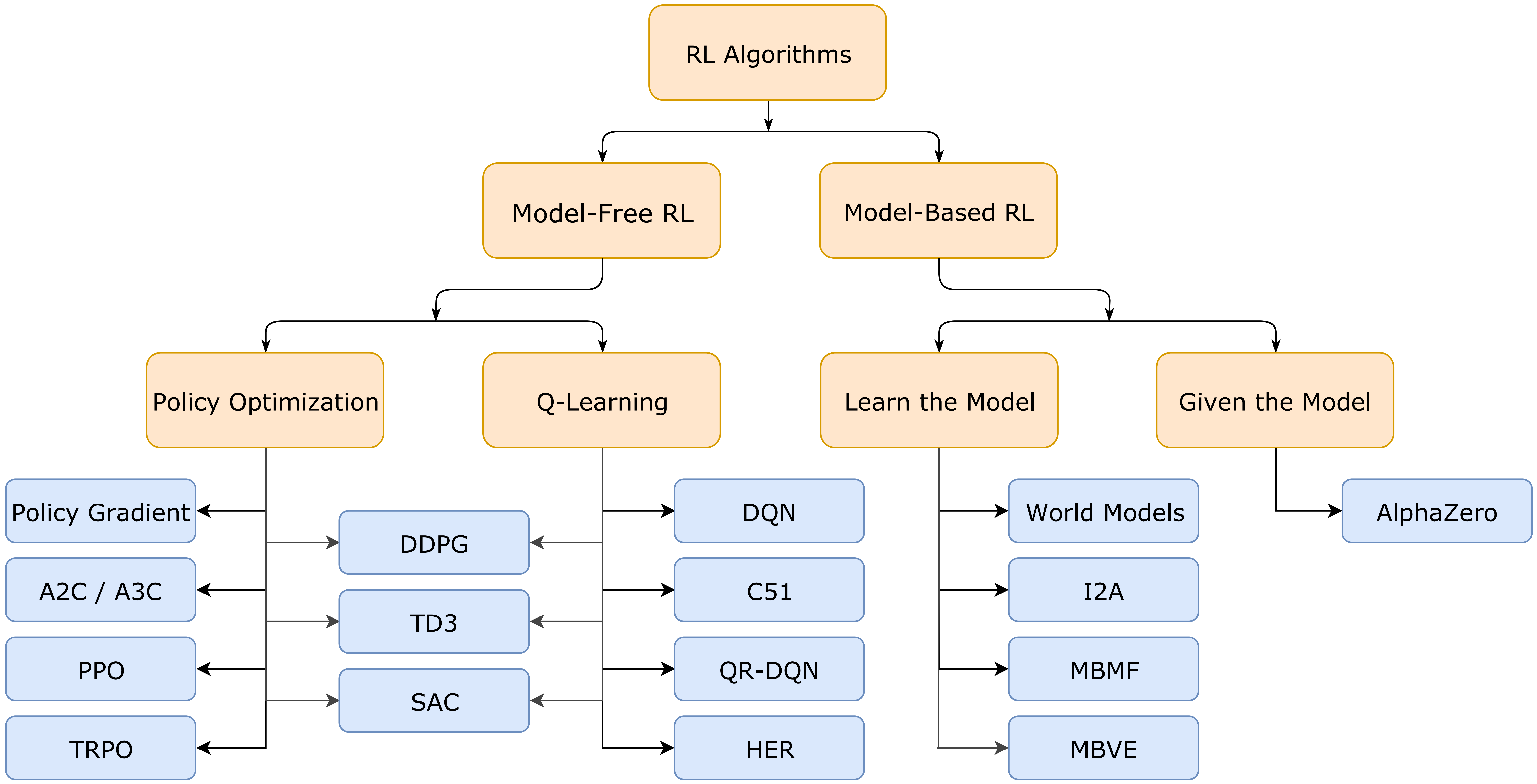
Model-Based RL

- Collect trajectories to estimate the transition and rewards model (system identification in control)
  - $\hat{P}\left(s' \mid s, a\right) = \frac{1}{N(s,a)} \sum_{k=1}^{K} \sum_{t=1}^{L_k-1} 1\left(s_{k,t} = s, a_{k,t} = a, s_{k,t+1} = s'\right)$
  - $\hat{R}(s,a) = \frac{1}{N(s,a)} \sum_{k=1}^{K} \sum_{t=1}^{L_k-1} 1\left(s_{k,t} = s, a_{k,t} = a\right) r_{t,k}$
  - Where $1(\cdot)$ is indicator function and $N(s,a)$ is the count of trajectories starting from (s,a)
- Then solve the estimated MDP
- Typically more sample efficient than the so-called model-free RL
- Inherit limitations of MDP exact methods, e.g., can be very computationally expensive



[Atkeson and Santamaría, 96]

A Glance of RL Algorithms



[Credit: Open AI Spinningup]

Q Learning

- Recall that using Q-value iteration, we update our estimate of Q via $Q_{\text{new}}(s,a) \leftarrow \mathbb{E}[R(s,a)] + \gamma \sum_{s'} p\left(s' \mid s,a\right) \max_{a'} Q\left(s',a'\right)$
- Without access to P and R, how would we be able to use this?
- One idea is to sample a state and action pair (s,a), simulate, observe s', r, and then $Q_{\text{new}}(s,a) \leftarrow r + \gamma \max_{a'} Q\left(s',a'\right)$
- But this is too "current sample dependent" — assumes the observed r is the only possible reward, assumes the observed s' is the only possible next state.
- So, instead, "smooth" the update with a step-size α $Q_{\text{new}}(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a'))$ target

(Tabular) Q Learning

$\text{Q-Learning}(\mathcal{S}, \mathcal{A}, s_0, \gamma, \alpha, \epsilon)$

$Q(s,a) = 0$ for $s \in \mathcal{S}, a \in \mathcal{A}$
$s = s_0$ // (e.g., s_0 can be drawn randomly from $\mathcal{S}$)
while True:
    $a = \text{select\_action}(s, Q)$
    $r, s' = \text{execute}(a)$
    $Q_{new}(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s', a'))$
    $s \leftarrow s'$
    if $|Q - Q_{new}| < \epsilon$: // (or, if reached some max iteration)
        return Q_{new}
    $Q \leftarrow Q_{new}$

target

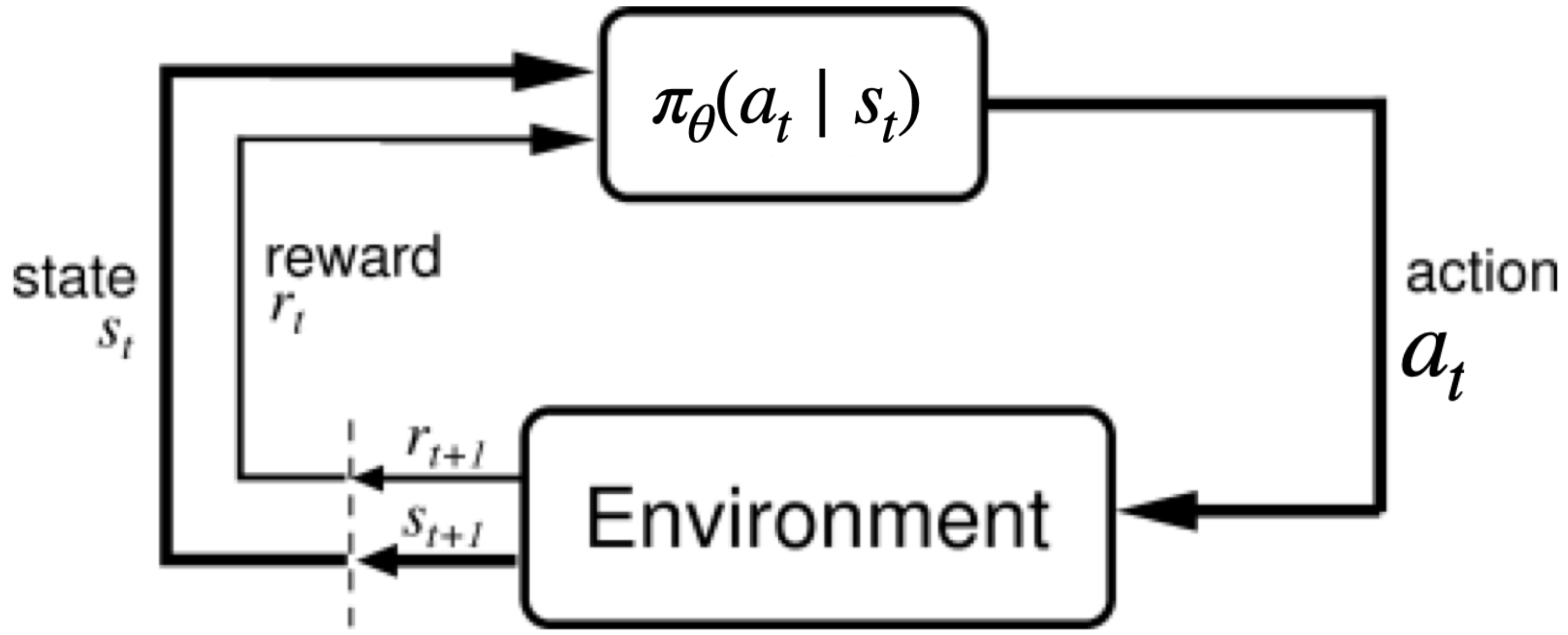
Q-learning Comments

- Face the same exploration versus exploitation dilemma as in bandits (due to unknown model)
- selection_action in line4 often uses epsilon-greedy; many other options available
- Rearranging terms in line 6, the update can also be interpreted via temporal-difference (TD) error: $Q_{new}(s, a) \leftarrow Q(s, a) + \alpha(\text{target} - Q(s, a))])$ TD error
- In TD-error form, the update looks quite like SGD.
- Closely connects to Fitted Q-learning (coming up in future lecture).

```
Q-LEARNING(S, A, s0, γ, α, ε)
1   Q(s, a) = 0 for s ∈ S, a ∈ A
2   s = s0 // (e.g., s0 can be drawn randomly from S)
3   while True:
4       a = select_action(s, Q)                                   target
5       r, s' = execute(a)
6       Qnew(s, a) ← (1 − α)Q(s, a) + α(r + γ max_a' Q(s', a'))
7       s ← s'
8       if |Q − Qnew| < ε: // (or, if reached some max iteration)
9           return Qnew
10      Q ← Qnew
```

- Can converge to true Q^* if:
  - All states and actions visited infinity often
  - Step-size α are annealed (i.e. if α_k, k being the iteration number of line 6, satisfy: $\sum_{k=1}^{\infty} \alpha_k = \infty$ and $\sum_{k=1}^{\infty} \alpha_k^2 < \infty$)

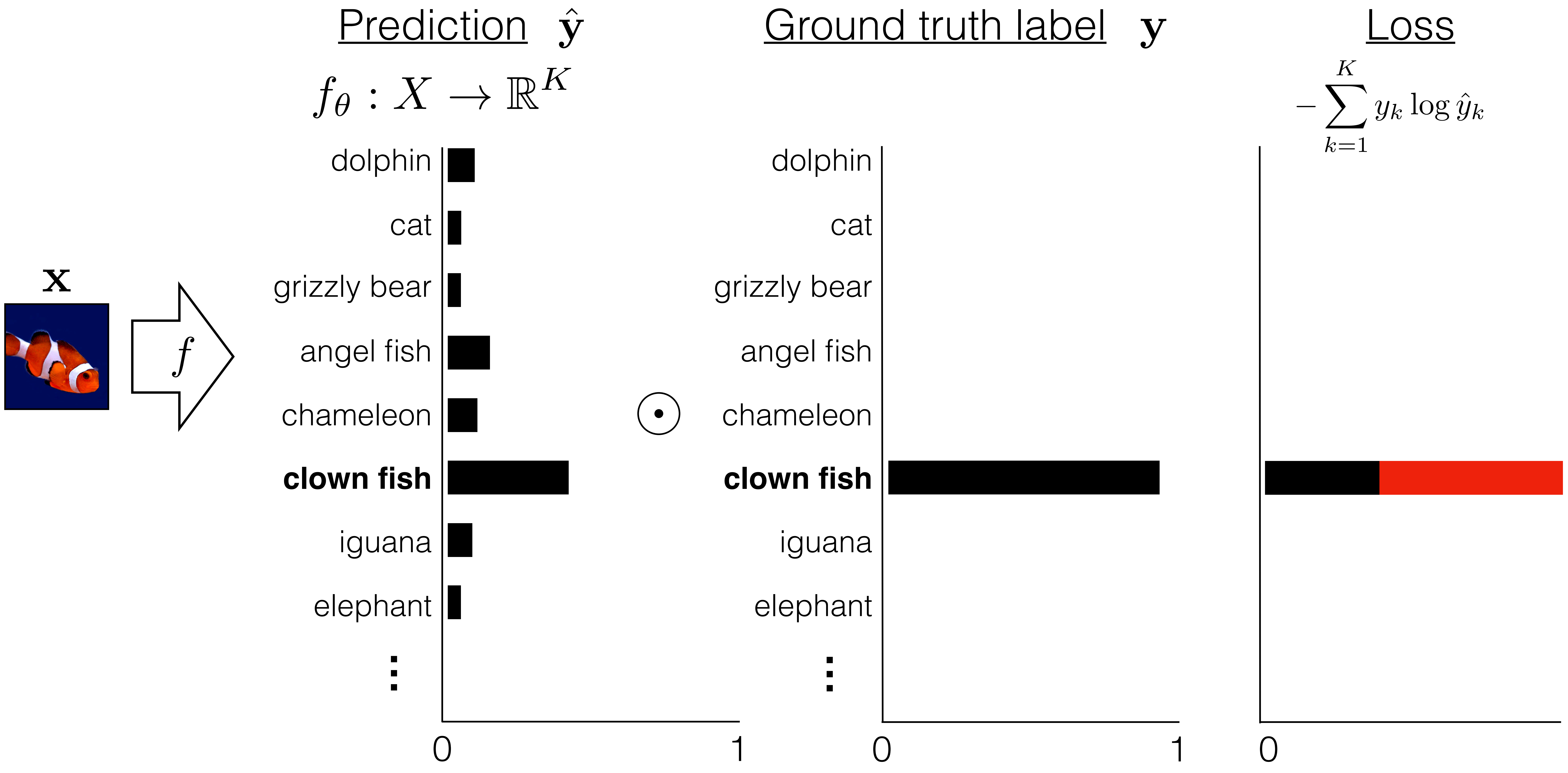
Policy Optimization



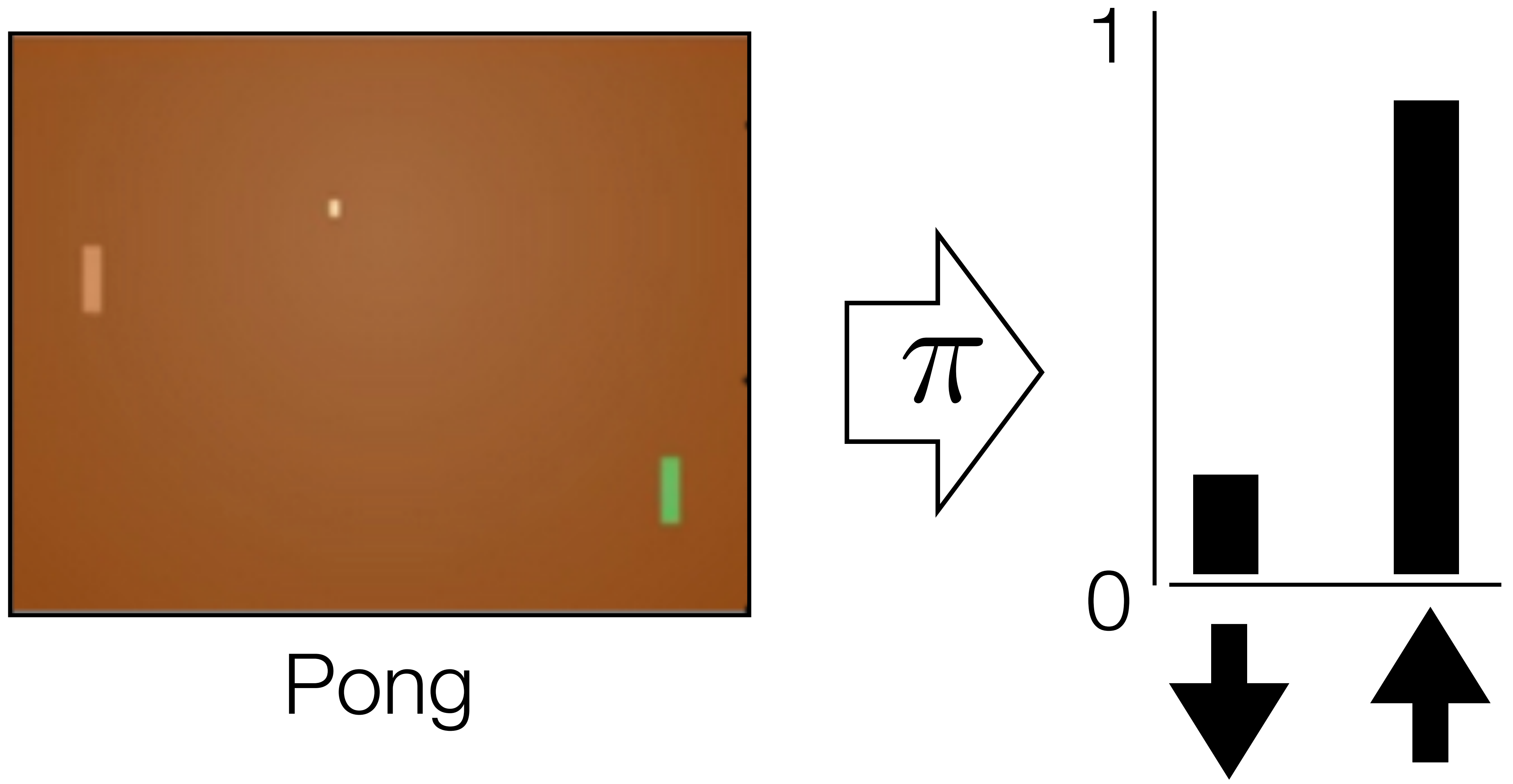
- Parameterize policy by θ and directly try $\max_\theta \mathbb{E}\left[\sum \gamma^t R\left(s_t, a_t\right) \mid \pi_\theta\right]$
- Stochastic policy class $\pi_\theta(a \mid s)$: probability of action a in state s
  - Discrete $\mathscr{A}$: e.g. $\pi_\theta(a \mid s)$ softmax
  - Continuous $\mathscr{A}$: e.g. $\pi_\theta(a \mid s)$ Gaussian with mean/variance parameterized by θ
  - Smoothes out the optimization problem
  - Also encourages exploration

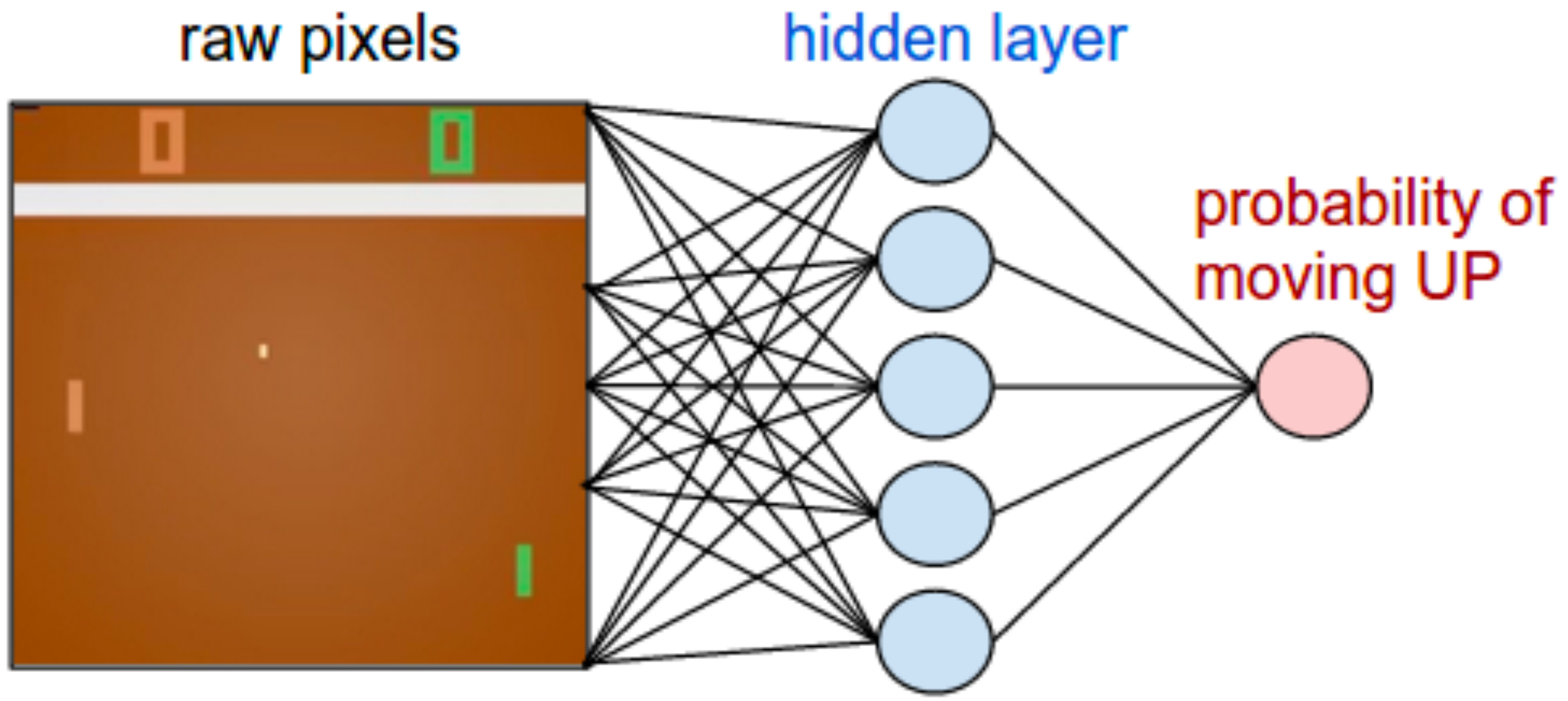
Why Policy Optimization

- Often π can be simpler than Q or V
  - e.g. lots of π are roughly good
- $V(s)$: doesn't prescribe actions
  - $\pi^*(s) = \arg\max_a \left[\mathbb{E}[R(s,a)] + \gamma \sum_{s'} p(s' \mid s,a) V^*(s')\right]$
  - Would still need world model (and compute one-step Bellman update)
- Q: need to be able to efficiently solve $\arg\max_a Q(s,a)$
  - $\pi^*(s) = \arg\max_a Q^*(s,a)$
  - Can be challenging for continuous / high-dimensional action spaces
- Maybe makes sense to direct optimize policy end-to-end
- So how do we do this?


Prediction $\hat{\mathbf{y}}$
$f_\theta : X \rightarrow \mathbb{R}^K$
Ground truth label $\mathbf{y}$
Loss
$-\sum_{k=1}^{K} y_k \log \hat{y}_k$
$\mathbf{x}$
f
dolphin
cat
grizzly bear
angel fish
chameleon
clown fish
iguana
elephant
⋮
$\odot$
dolphin
cat
grizzly bear
angel fish
chameleon
clown fish
iguana
elephant
⋮
0
1
0
1
0

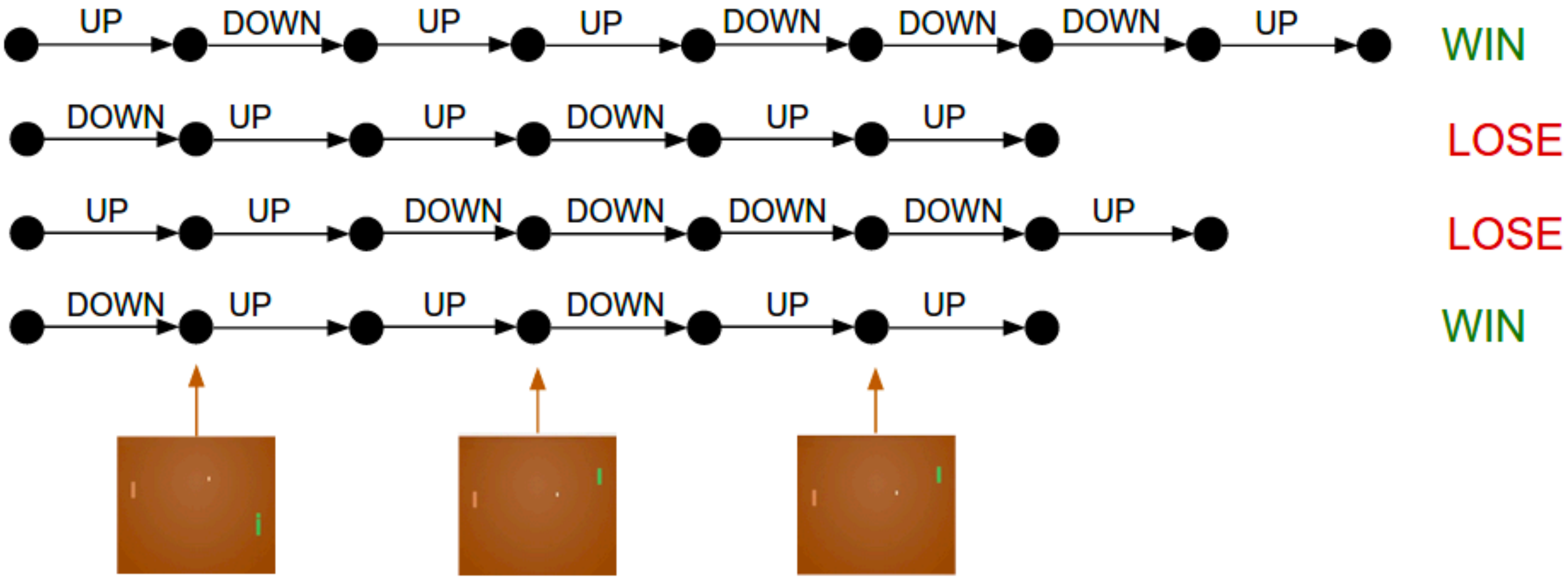
- If explicit “good” state-action pair is given, also supervised learning.
- Behavior cloning or imitation learning.
- But what if no explicit guide?



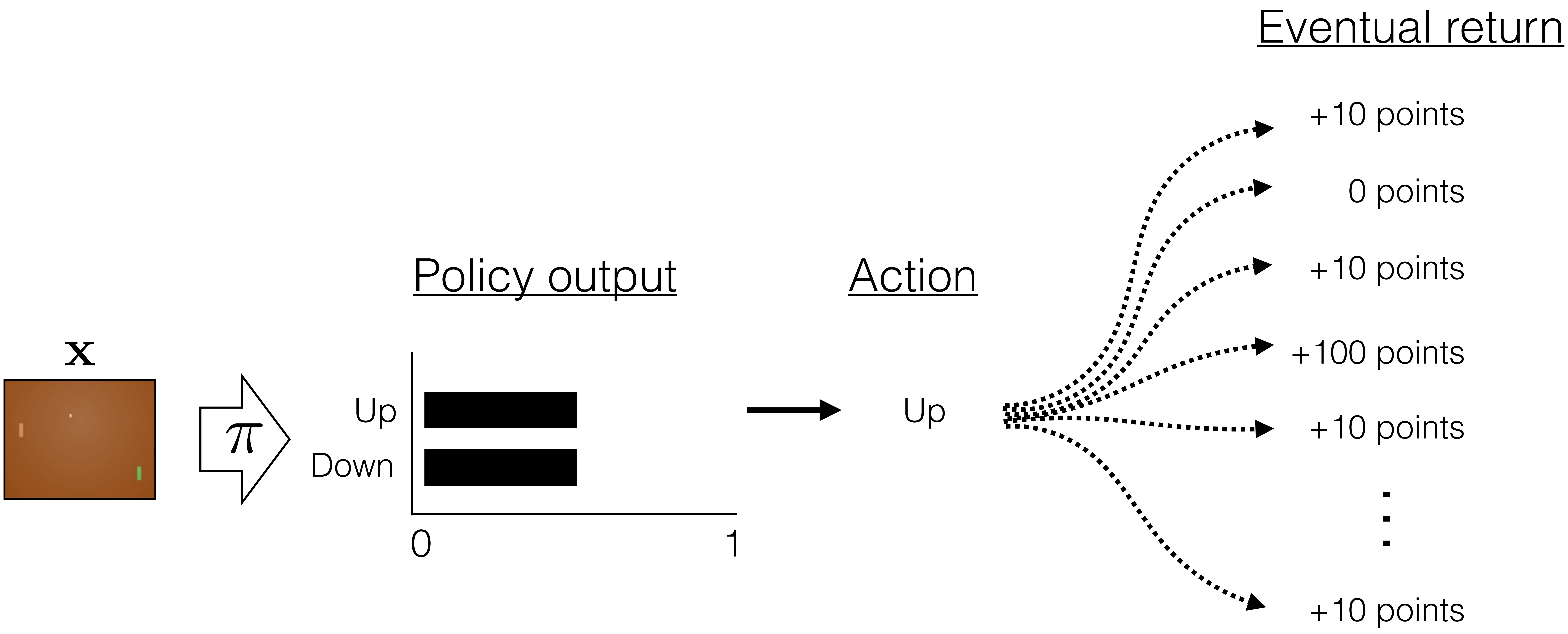


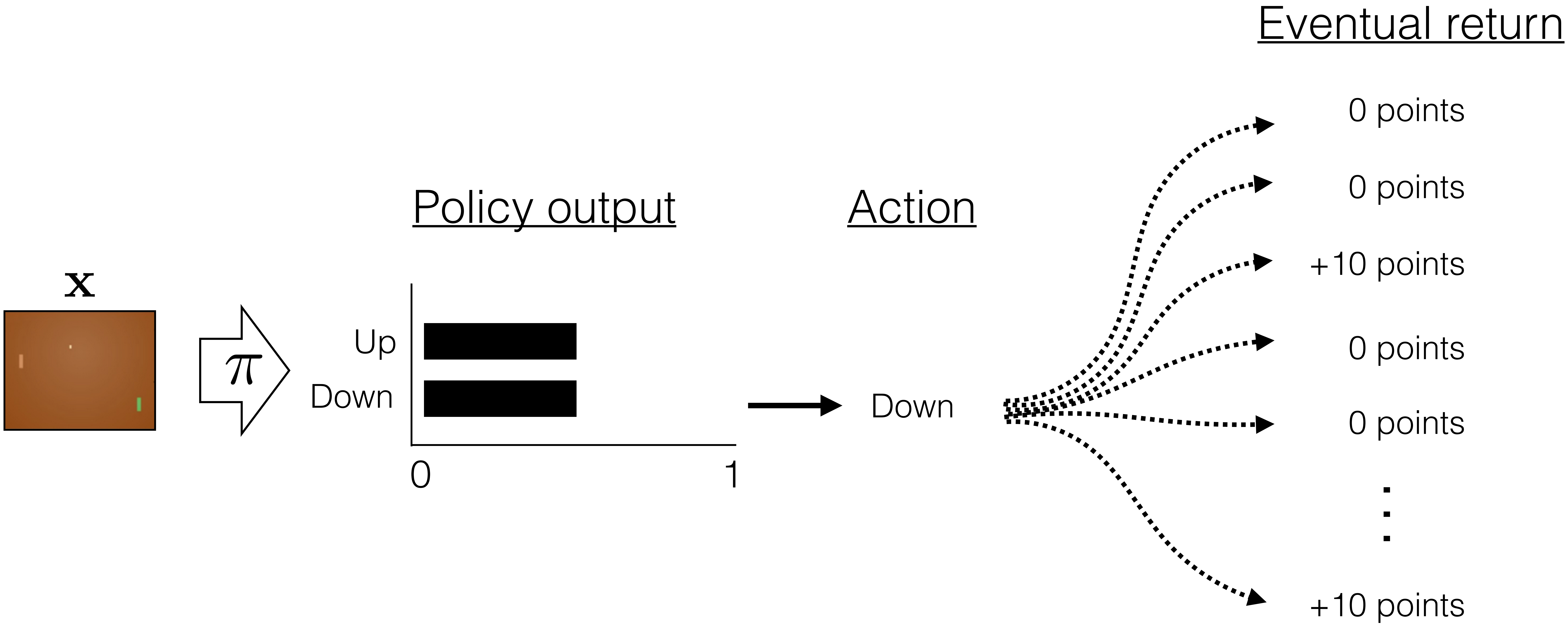
[Adapted from Andrej Karpathy: http://karpathy.github.io/2016/05/31/rl/]

Policy gradients: Run a policy for a while. See what actions led to good return. Increase their likelihood.

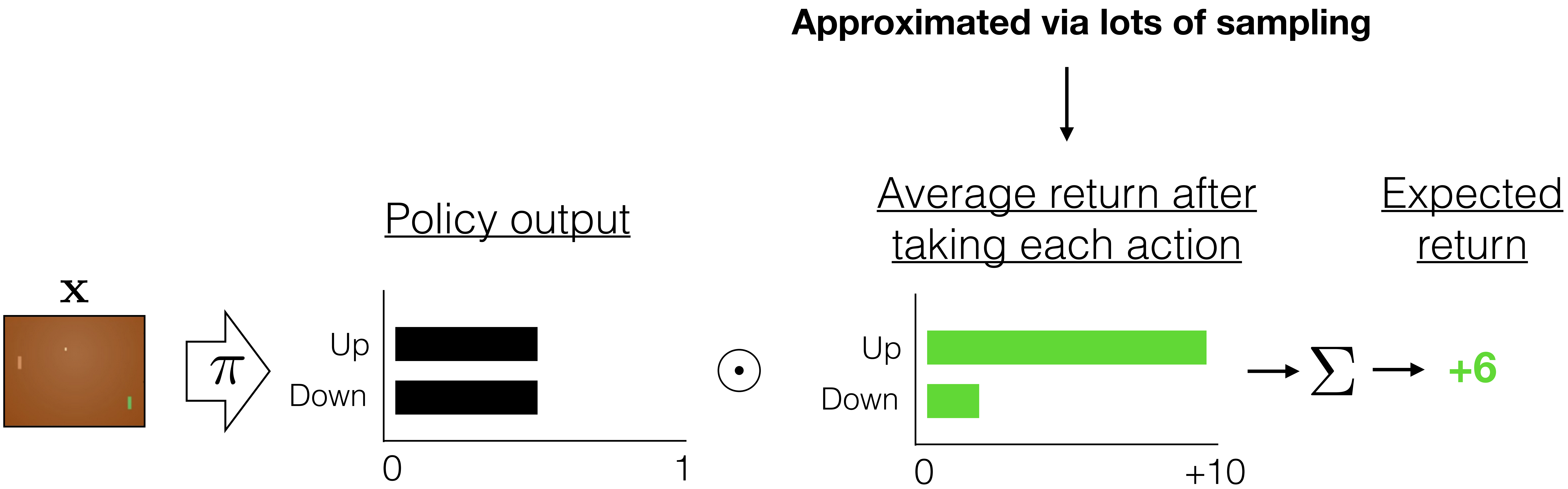


[Adapted from Andrej Karpathy: http://karpathy.github.io/2016/05/31/rl/]

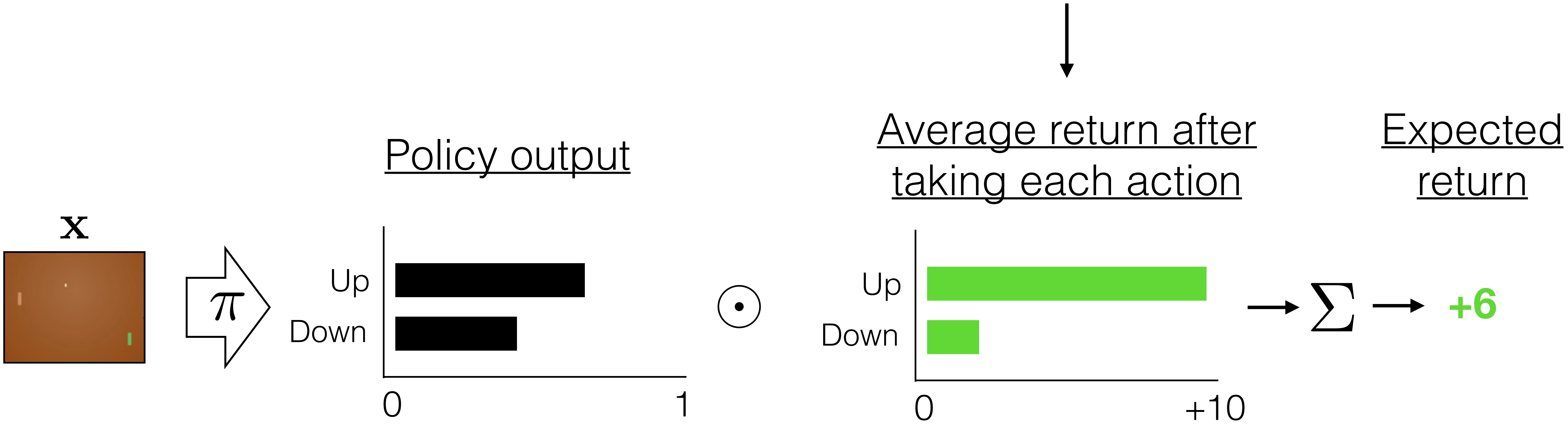

Eventual return
+10 points
0 points
+10 points
+100 points
+10 points
+10 points
Policy output
Action
x
π
Up
Down
0
1
Up



Eventual return
0 points
0 points
Policy output
Action
+10 points
x
Up
π
0 points
Down
Down
0 points
0
1
+10 points


Approximated via lots of sampling
x
π
Policy output
Up
Down
0
1
⊙
Average return after taking each action
Up
Down
0
+10
Σ
Expected return
+6

Approximated via lots of sampling



How is this gradient update done though (as we don't have the world model Pong)?

Likelihood Ratio Policy Gradient

- We overload notation:
  - Let τ denote a state-action sequence: $\tau = s_0, a_0, s_1, a_1, \ldots$
  - Let $R(\tau)$ denote the sum of discounted rewards on τ: $R(\tau) = \sum_t \gamma^t R\left(s_t, a_t\right)$
  - W.l.o.g. assume $R(\tau)$ is deterministic in τ
  - Let $P(\tau; \theta)$ denote the probability of trajectory τ induced by π_θ
  - Let $U(\theta)$ denote the objective: $U(\theta) = \mathbb{E}[\sum_t \gamma^t R\left(s_t, a_t\right) \mid \pi_\theta]$

- Our goal is to find θ: $\max_\theta U(\theta) = \max_\theta \sum_\tau P(\tau; \theta) R(\tau)$

Likelihood Ratio Policy Gradient

Taking the gradient w.r.t. θ gives

$$\nabla_\theta U(\theta) = \nabla_\theta \sum_\tau P(\tau;\theta)R(\tau)$$
$$= \sum_\tau \nabla_\theta P(\tau;\theta)R(\tau)$$
$$= \sum_\tau \frac{P(\tau;\theta)}{P(\tau;\theta)} \nabla_\theta P(\tau;\theta)R(\tau)$$
$$= \sum_\tau P(\tau;\theta)\frac{\nabla_\theta P(\tau;\theta)}{P(\tau;\theta)}R(\tau)$$
$$= \sum_\tau P(\tau;\theta)\,\nabla_\theta \log P(\tau;\theta)R(\tau)$$

$$U(\theta) = \sum_\tau P(\tau;\theta)R(\tau)$$

But $P(\tau;\theta) = \prod_{t=0} \underbrace{P\left(s_{t+1} \mid s_t, a_t\right)}_{\text{transition}} \cdot \underbrace{\pi_\theta\left(a_t \mid s_t\right)}_{\text{policy}}]$

Use identity

$$\nabla_\theta p_\theta(\tau) = p_\theta(\tau)\frac{\nabla_\theta p_\theta(\tau)}{p_\theta(\tau)}$$
$$= p_\theta(\tau)\,\nabla_\theta \log p_\theta(\tau)$$

Likelihood Ratio Policy Gradient

$$\nabla_\theta U(\theta) = \sum_\tau P(\tau;\theta)\,\nabla_\theta \log P(\tau;\theta) R(\tau)$$

Approximate with the empirical estimate for m sample traj. under policy π_θ

$$\nabla_\theta U(\theta) \approx \hat{g} = \frac{1}{m}\sum_{i=1}^{m} \nabla_\theta \log P\left(\tau^{(i)};\theta\right) R\left(\tau^{(i)}\right)$$

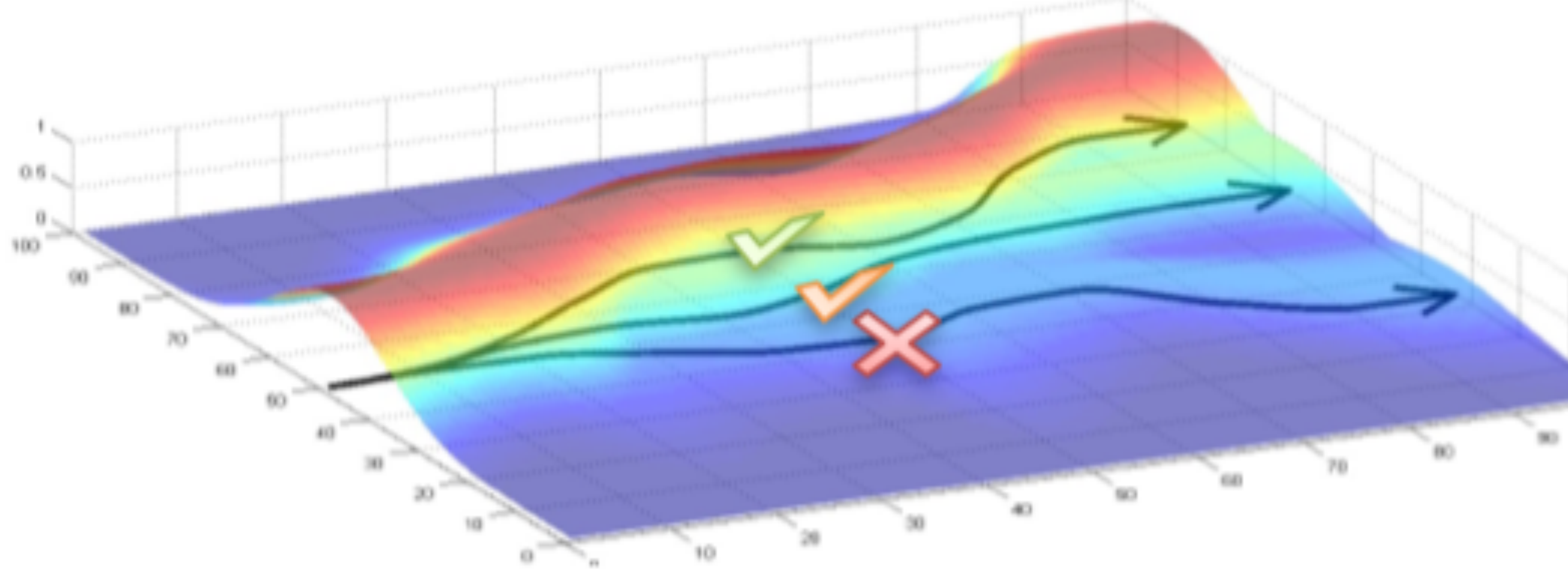
Valid even when:

- Reward function discontinuous and/or unknown
- Discrete state and/or action spaces

Likelihood Ratio Gradient

$$\nabla_\theta U(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \log P\left(\tau^{(i)}; \theta\right) R\left(\tau^{(i)}\right)$$

- Checks out with our intuition that:
  - Increase likelihood of trajectory with big reward
  - Decrease prob of trajectory with negative reward



- How do we evaluate $\nabla_\theta \log P\left(\tau^{(i)}; \theta\right)$ though?

  Didn't we say we don't know the transition?

$$P(\tau; \theta) = \prod_{t=0} \underbrace{P\left(s_{t+1} \mid s_t, a_t\right)}_{\text{transition}} \cdot \underbrace{\pi_\theta\left(a_t \mid s_t\right)}_{\text{policy}}]$$

Decompose a trajectory

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

$$= \nabla_\theta \left[\sum_{t=0} \log P\left(s_{t+1} \mid s_t, a_t\right) + \sum_{t=0} \log \pi_\theta\left(a_t \mid s_t\right) \right]$$

$$= \nabla_\theta \sum_{t=0} \log \pi_\theta\left(a_t \mid s_t\right)$$

$$= \sum_{t=0} \underbrace{\nabla_\theta \log \pi_\theta\left(a_t \mid s_t\right)}_{\text{no transition model required,}}$$

Likelihood Ratio Gradient - Summary

- The following expression provides us with an unbiased estimate of the gradient, and we can compute it without access to the world model:

$$\hat{g} = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \log P\left(\tau^{(i)}; \theta\right) R\left(\tau^{(i)}\right)$$

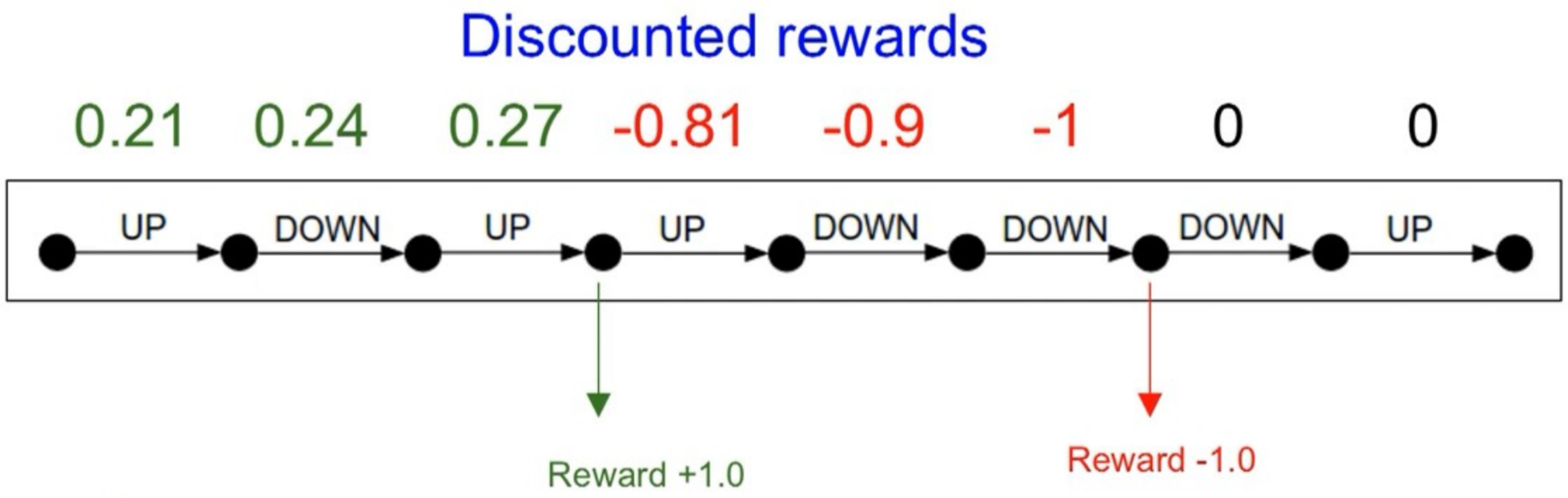
- Here

$$\nabla_\theta \log P(\tau; \theta) = \sum_{t=0} \underbrace{\nabla_\theta \log \pi_\theta\left(a_t \mid s_t\right)}_{\text{no need of dynamics}}$$

- Unbiased estimator $\mathrm{E}[\hat{g}] = \nabla_\theta U(\theta)$, but very noisy.

Variance Reduction - Discount

Blame each action assuming that its effects have exponentially decaying impact into the future.



- In the extreme, if discount of 0, almost no variance at all.
- So discount can be both a problem definition, or a hyper-parameter.

Variance Reduction - Baseline

- Sample estimate, unbiased but can be very noisy

$$\nabla U(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \log P\left(\tau^{(i)}; \theta\right) R\left(\tau^{(i)}\right)$$

- Can we keep unbiasedness but reduce variance? Yes!
- Subtract an appropriate baseline can keep the unbiasedness

$$\nabla U(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \log P\left(\tau^{(i)}; \theta\right) \left(R\left(\tau^{(i)}\right) - b\right)$$

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

[[Williams, REINFORCE paper,1992]

Variance-reduction Baselines

- Constant $b = \frac{1}{m} \sum_{i=1}^{m} R\left(\tau^{(i)}\right)$
- Optimal constant baseline: $b = \dfrac{\sum_i \left(\nabla_\theta \log P\left(\tau^{(i)}; \theta\right) \right)^2 R\left(\tau^{(i)}\right)}{\sum_i \left(\nabla_\theta \log P\left(\tau^{(i)}; \theta\right) \right)^2}$

[Greensmith, Bartlett, Baxter, JMLR 2004 for variance reduction techniques.]

- Estimated state-dependent value functions: $b\left(s_t\right) = \hat{V}^\pi\left(s_t\right)$
  - I.e., $\nabla U(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \log P\left(\tau^{(i)}; \theta\right) \left(R\left(\tau^{(i)}\right) - \hat{V}^\pi(s) \right)$ Advantage
  - We'll discuss methods on how to estimate $\hat{V}^\pi\left(s_t\right)$ later.
  - This kind of "value" baseline very roughly gets us to actor-critic methods.

Variance Reduction - Temporal Structure

- Current gradient estimate:

$$\hat{g} = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \log P\left(\tau^{(i)}; \theta\right) \left(R\left(\tau^{(i)}\right) - b\right)$$

$$= \frac{1}{m} \sum_{i=1}^{m} \left(\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta \left(a_t^{(i)} \mid s_t^{(i)}\right) \right) \left(\sum_{t=0}^{H-1} R\left(s_t^{(i)}, a_t^{(i)}\right) - b \right)$$

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

- Removing terms that don't depend on current action can lower variance:

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

[Policy Gradient Theorem: Sutton et al 1999; GPOMDP: Bartlett & Baxter, 2001; Survey: Peters & Schaal, 2006]

Estimation of V^π (coming up later)

- State-dependent expected return: $b\left(s_t\right) = \mathbb{E}\left[r_t + r_{t+1} + r_{t+2} + \ldots + r_{H-1}\right] = V^\pi\left(s_t\right)$
  - Increase the prob of action proportionally to how much its returns are better than the expected return under the current policy
- Can't exactly solve for V^π; again need to estimate. How?
  - Either collect $\tau_1, \ldots, \tau_m$, and regress against empirical return:

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

  - Or similar to fitted Q-learning, do fitted V-learning:

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

[See: Greensmith, Bartlett, Baxter, JMLR 2004 for variance reduction techniques.]

Algorithm 1 "Vanilla" policy gradient algorithm

Initialize policy parameter θ, baseline b
for iteration$=1, 2, \dots$ **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

Thanks!

Questions?