Goals for today

- Learn that policies can be optimized directly, without learning value functions, by *policy-gradient methods*
  - Glimpse how one could learn real-valued (continuous) actions
- Glimpse how to handle hidden state

Policy-gradient methods

A new approach to control

Approaches to control

1. Previous approach: *Action-value methods*:
   - learn the value of each action;
   - pick the max (usually)
2. New approach: *Policy-gradient methods*:
   - learn the parameters of a stochastic policy
   - update by gradient ascent in performance
   - includes *actor-critic methods*, which learn *both* value and policy parameters

Why approximate policies rather than values?

- In many problems, the policy is simpler to approximate than the value function
- In many problems, the optimal policy is stochastic
  - e.g., bluffing, POMDPs
- To enable smoother change in policies
- To avoid a search on every step (the max)
- To better relate to biology

We first saw this in Chapter 2, with the

Gradient-bandit algorithm

- Store action preferences $H_t(a)$
  rather than action-value estimates $Q_t(a)$

- Instead of ε-greedy, pick actions by an exponential soft-max:

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

- Also store the sample average of rewards as $\bar{R}_t$
- Then update:

$$H_{t+1}(a) = H_t(a) + \alpha\big(R_t - \bar{R}_t\big)\big(\mathbf{1}_{a=A_t} - \pi_t(a)\big)$$

$\frac{\partial \pi_t(A_t)}{\partial H_t(a)} / \pi_t(A_t)$

1 or 0, depending on whether the predicate (subscript) is true

eg, linear-exponential policies (discrete actions)

- The "preference" h for action a in state s is linear in $\boldsymbol{\theta}$ and a state-action feature vector

$$h(s, a, \boldsymbol{\theta}) = \boldsymbol{\theta}^\top \mathbf{x}(s, a)$$

- The probability of each action is exponential in its preference

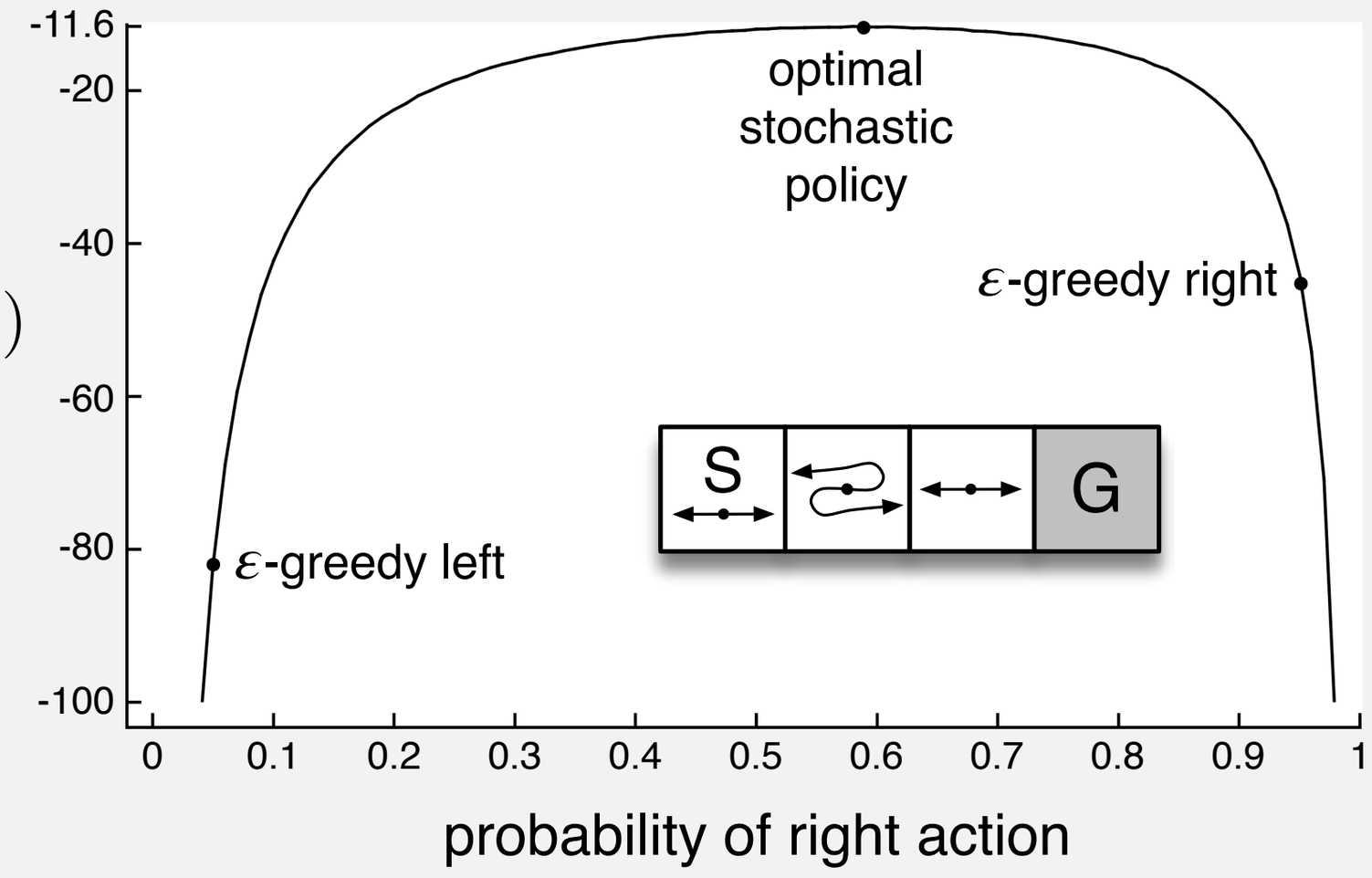
$$\pi(a|s, \boldsymbol{\theta}) = \frac{\exp(h(s, a, \boldsymbol{\theta}))}{\sum_b \exp(h(s, b, \boldsymbol{\theta}))}$$

- Corresponding *eligibility function*:

$$\nabla_{\boldsymbol{\theta}} \ln \pi(a|s, \boldsymbol{\theta}) = \mathbf{x}(s, a) - \sum_b \pi(b|s, \boldsymbol{\theta}) \mathbf{x}(s, b)$$

Example of the need for stochastic policies: The short corridor with switched actions

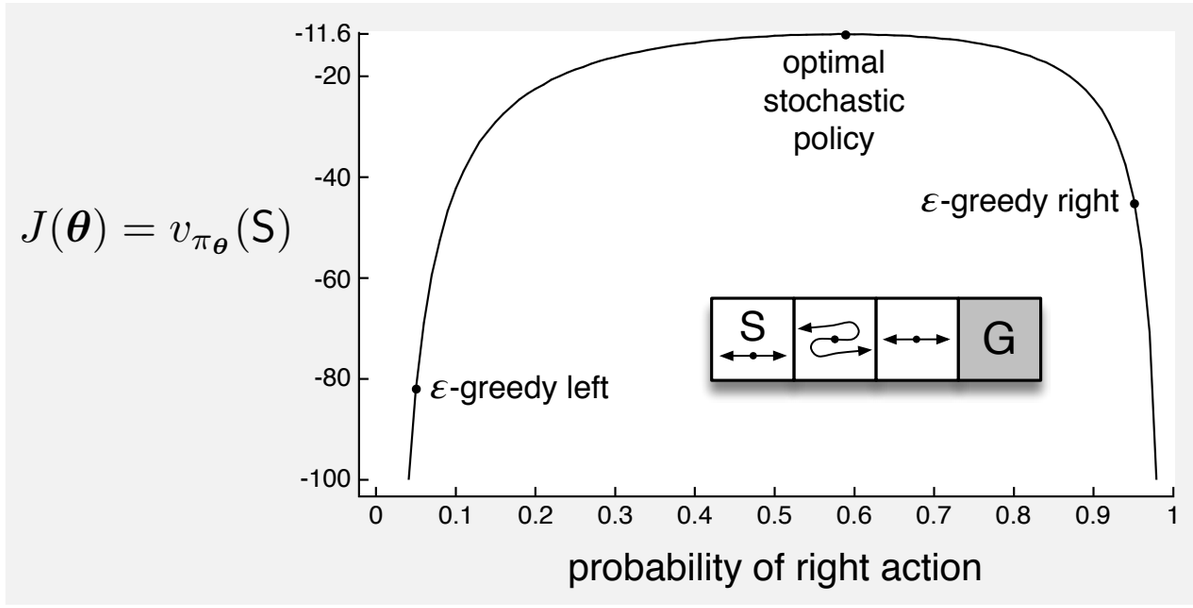
$J(\boldsymbol{\theta}) = v_{\pi_{\boldsymbol{\theta}}}(\mathrm{S})$



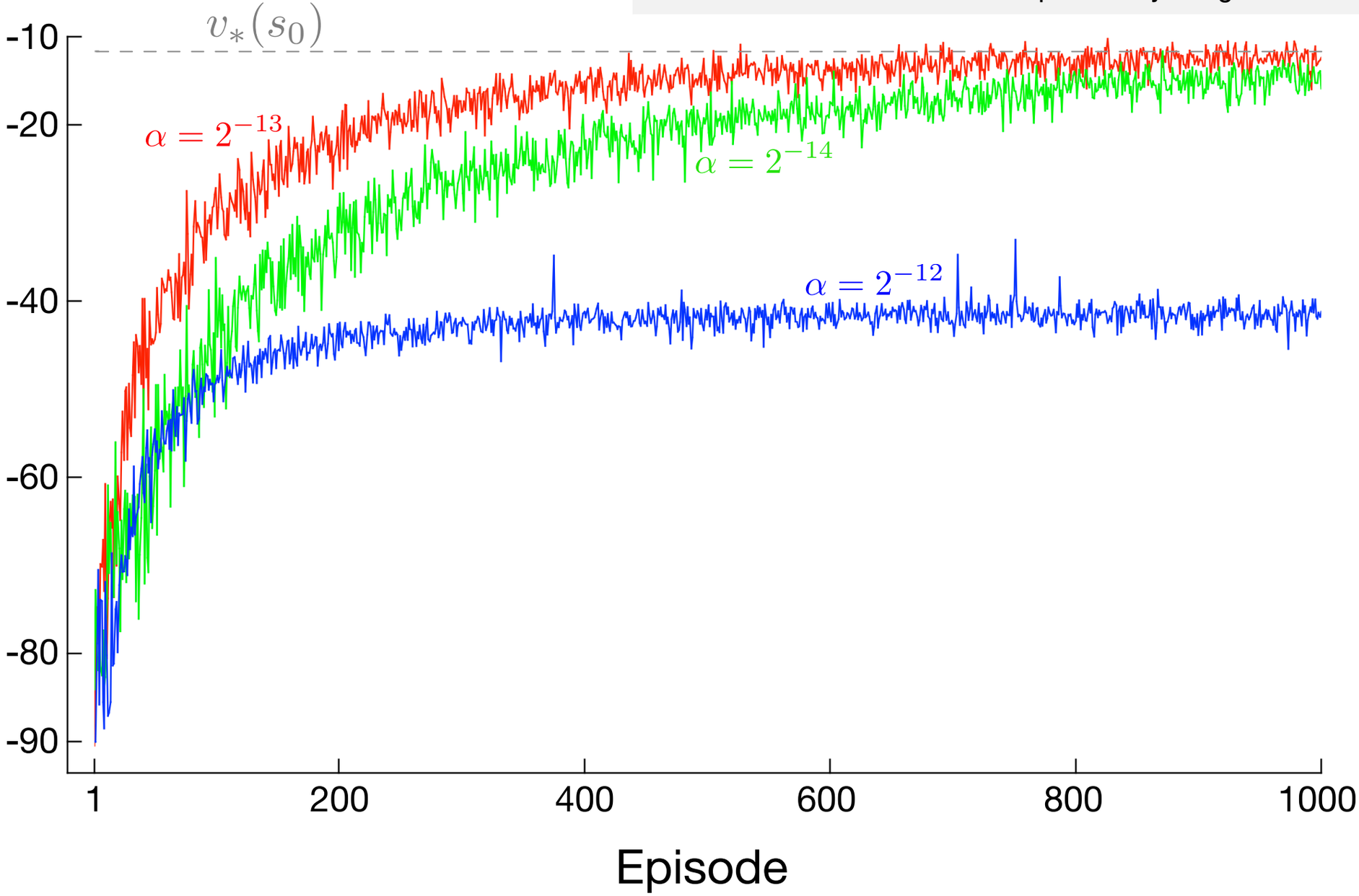
Neither e-greedy policy is good

Optimal performance requires acting with particular probabilities

The REINFORCE algorithm finds the right probability



Total reward per episode G_0



Policy-gradient setup

Given a policy parameterization:

$$\pi(a|s, \boldsymbol{\theta})$$

And objective:

$$J(\boldsymbol{\theta}) \doteq v_{\pi_{\boldsymbol{\theta}}}(s_0) \quad \text{(or average reward)}$$

Approximate stochastic gradient ascent:

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha \widehat{\nabla J(\boldsymbol{\theta}_t)}$$

Typically, based on the Policy-Gradient Theorem:

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_\pi(s, a) \nabla_{\boldsymbol{\theta}} \pi(a|s, \boldsymbol{\theta})$$

on-policy distribution

Proof of the Policy Gradient Theorem (episodic case)

With just elementary calculus and re-arranging terms we can prove the policy gradient theorem from first principles. To keep the notation simple, we leave it implicit in all cases that π is a function of $\boldsymbol{\theta}$, and all gradients are also implicitly with respect to $\boldsymbol{\theta}$. First note that the gradient of the state-value function can be written in terms of the action-value function as

$$
\begin{aligned}
\nabla v_\pi(s) &= \nabla \left[\sum_a \pi(a|s) q_\pi(s,a) \right], \quad \text{for all } s \in \mathcal{S} && \text{(Exercise 3.15)} \\
&= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla q_\pi(s,a) \Big] && \text{(product rule)} \\
&= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla \sum_{s',r} p(s',r|s,a) \big(r + v_\pi(s') \big) \Big] \\
& && \text{(Exercise 3.16 and Equation 3.2)} \\
&= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) \nabla v_\pi(s') \Big] && \text{(Eq. 3.4)} \\
&= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) && \text{(unrolling)} \\
&\qquad \sum_{a'} \big[\nabla \pi(a'|s') q_\pi(s',a') + \pi(a'|s') \sum_{s''} p(s''|s',a') \nabla v_\pi(s'') \big] \Big] \\
&= \sum_{x \in \mathcal{S}} \sum_{k=0}^{\infty} \Pr(s \to x, k, \pi) \sum_a \nabla \pi(a|x) q_\pi(x,a),
\end{aligned}
$$

after repeated unrolling, where $\Pr(s \to x, k, \pi)$ is the probability of transitioning from state s to state x in k steps under policy π. It is then immediate that

$$= \sum_a \Big[\nabla\pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla \sum_{s',r} p(s',r\,|\,s,a)\big(r + v_\pi(s')\big) \Big]$$

(Exercise 3.16 and Equation 3.2)

$$= \sum_a \Big[\nabla\pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'\,|\,s,a) \nabla v_\pi(s') \Big] \qquad \text{(Eq. 3.4)}$$

$$= \sum_a \Big[\nabla\pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'\,|\,s,a) \qquad \text{(unrolling)}$$

$$\sum_{a'} \big[\nabla\pi(a'|s') q_\pi(s',a') + \pi(a'|s') \sum_{s''} p(s''\,|\,s',a') \nabla v_\pi(s'') \big] \Big]$$

$$= \sum_{x\in\mathcal{S}} \sum_{k=0}^{\infty} \Pr(s \to x, k, \pi) \sum_a \nabla\pi(a|x) q_\pi(x,a),$$

after repeated unrolling, where $\Pr(s \to x, k, \pi)$ is the probability of transitioning from state s to state x in k steps under policy π. It is then immediate that

$$\begin{aligned}
\nabla J(\boldsymbol{\theta}) &= \nabla v_\pi(s_0) \\
&= \sum_s \left(\sum_{k=0}^{\infty} \Pr(s_0 \to s, k, \pi) \right) \sum_a \nabla\pi(a|s) q_\pi(s,a) \\
&= \sum_s \eta(s) \sum_a \nabla\pi(a|s) q_\pi(s,a) && \text{(box page 163)} \\
&= \left(\sum_s \eta(s) \right) \sum_s \frac{\eta(s)}{\sum_s \eta(s)} \sum_a \nabla\pi(a|s) q_\pi(s,a) \\
&\propto \sum_s \mu(s) \sum_a \nabla\pi(a|s) q_\pi(s,a). \qquad \text{Q.E.D.} && \text{(Eq. 9.3)}
\end{aligned}$$

Deriving REINFORCE from the PGT

$$
\begin{aligned}
\nabla J(\boldsymbol{\theta}) &\propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla_{\boldsymbol{\theta}} \pi(a|s,\boldsymbol{\theta}) \\
&= \mathbb{E}_\pi \left[\sum_a q_\pi(S_t,a) \nabla_{\boldsymbol{\theta}} \pi(a|S_t,\boldsymbol{\theta}) \right] \\
&= \mathbb{E}_\pi \left[\sum_a \pi(a|S_t,\boldsymbol{\theta}) q_\pi(S_t,a) \frac{\nabla_{\boldsymbol{\theta}} \pi(a|S_t,\boldsymbol{\theta})}{\pi(a|S_t,\boldsymbol{\theta})} \right] \\
&= \mathbb{E}_\pi \left[q_\pi(S_t,A_t) \frac{\nabla_{\boldsymbol{\theta}} \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})} \right] \\
&= \mathbb{E}_\pi \left[G_t \frac{\nabla_{\boldsymbol{\theta}} \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})} \right],
\end{aligned}
$$

Thus

$$
\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha G_t \frac{\nabla_{\boldsymbol{\theta}} \pi(A_t|S_t,\boldsymbol{\theta}_t)}{\pi(A_t|S_t,\boldsymbol{\theta}_t)}
$$

REINFORCE with baseline

Policy-gradient theorem with baseline:

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla_{\boldsymbol{\theta}} \pi(a|s,\boldsymbol{\theta})$$

$$= \sum_s \mu(s) \sum_a \Big(q_\pi(s,a) - b(s) \Big) \nabla_{\boldsymbol{\theta}} \pi(a|s,\boldsymbol{\theta})$$

any function of state, not action

Because

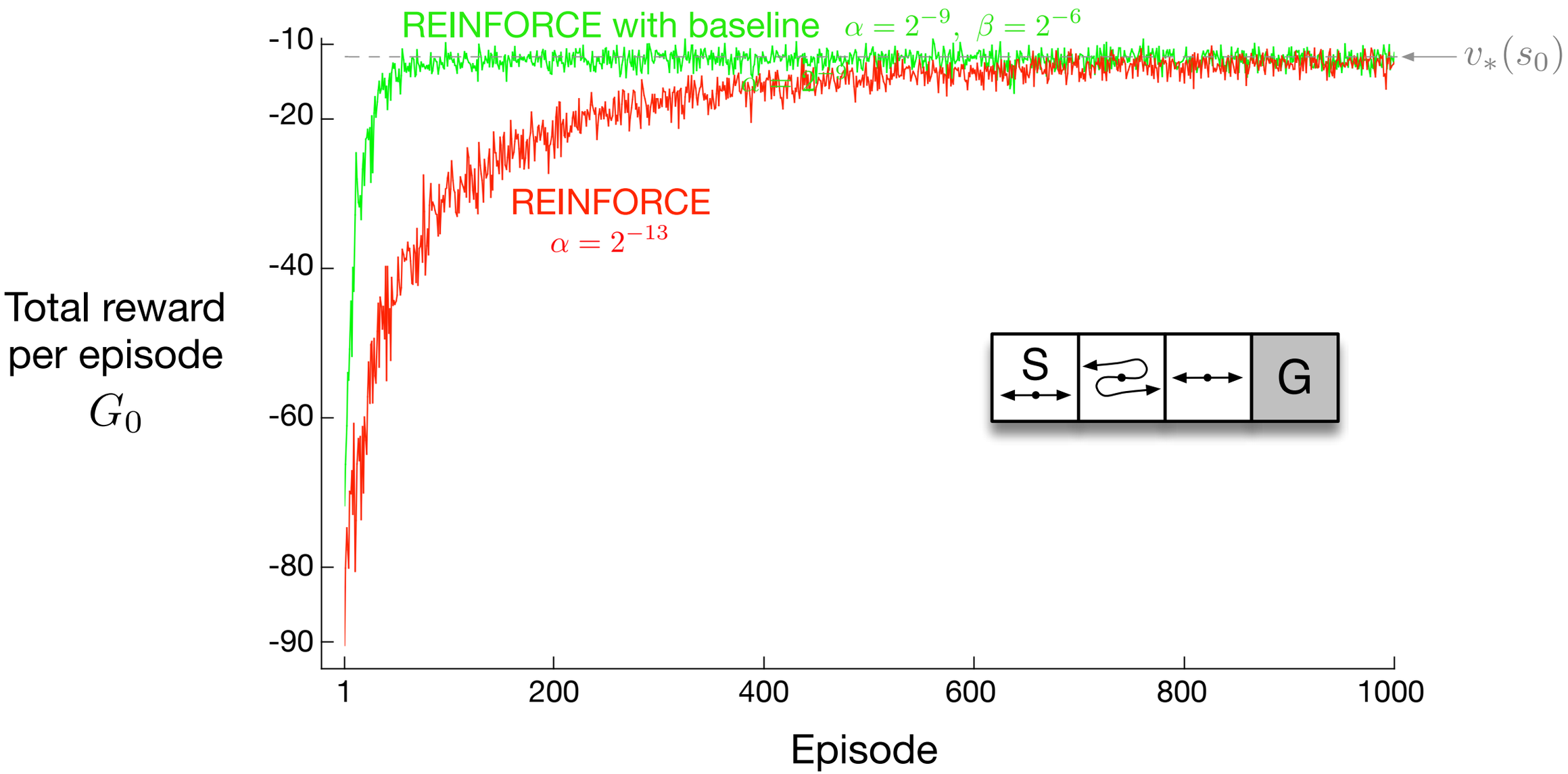
$$\sum_a b(s) \nabla_{\boldsymbol{\theta}} \pi(a|s,\boldsymbol{\theta}) = b(s) \nabla_{\boldsymbol{\theta}} \sum_a \pi(a|s,\boldsymbol{\theta}) = b(s) \nabla_{\boldsymbol{\theta}} 1 = 0 \qquad \forall s \in \mathcal{S}$$

Thus

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha \Big(G_t - b(S_t) \Big) \frac{\nabla_{\boldsymbol{\theta}} \pi(A_t|S_t,\boldsymbol{\theta}_t)}{\pi(A_t|S_t,\boldsymbol{\theta}_t)}$$

e.g., $b(s) = \hat{v}(s,\mathbf{w})$

REINFORCE with baseline is faster than plain REINFORCE



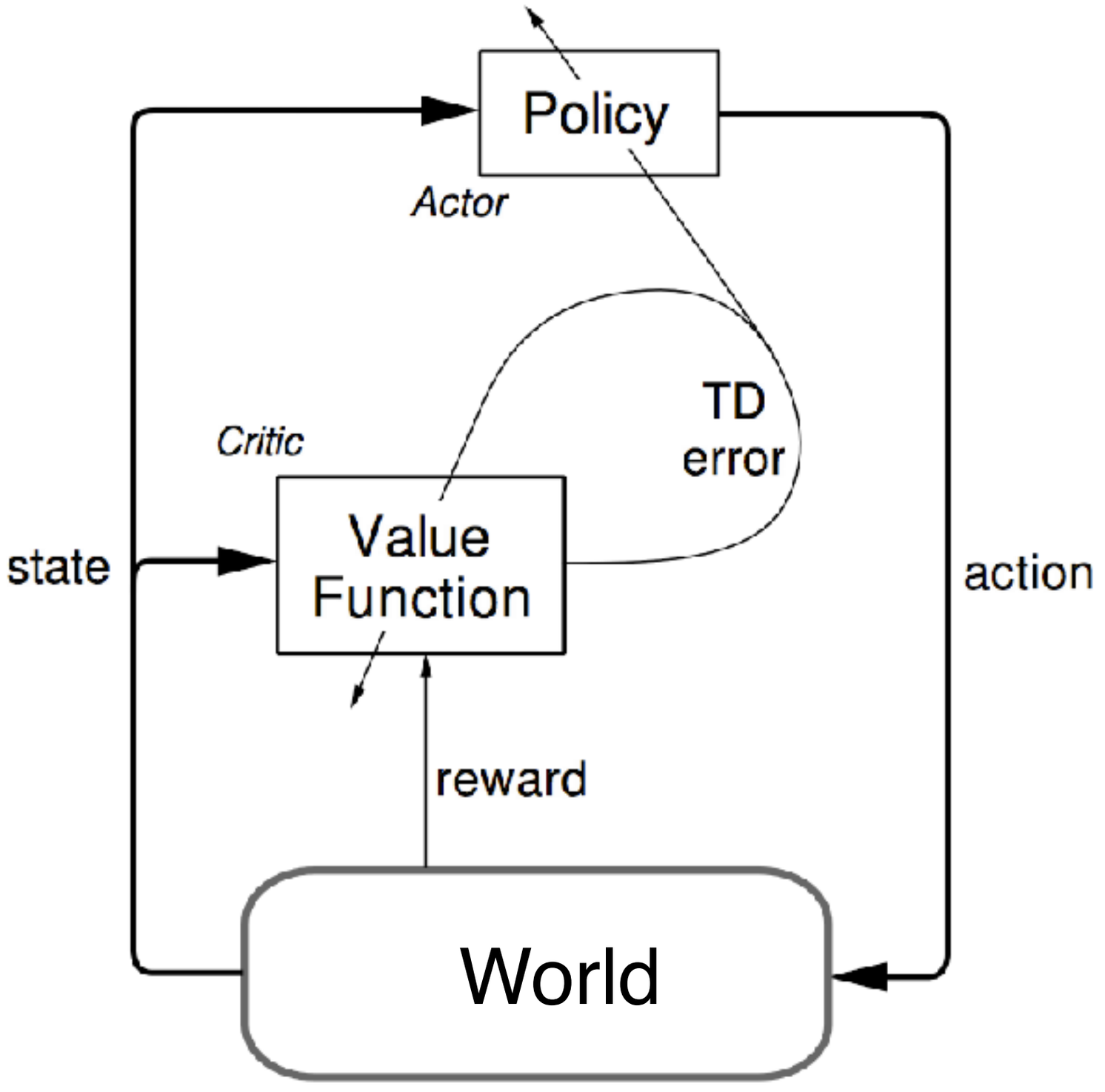
REINFORCE with Baseline (episodic)

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value parameterization $\hat{v}(s,\mathbf{w})$
Parameters: step sizes $\alpha > 0$, $\beta > 0$

Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$
Repeat forever:
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot,\boldsymbol{\theta})$
  For each step of the episode $t = 0, \ldots, T-1$:
    $G_t \leftarrow$ return from step t
    $\delta \leftarrow G_t - \hat{v}(S_t,\mathbf{w})$
    $\mathbf{w} \leftarrow \mathbf{w} + \beta\gamma^t\delta\nabla_{\mathbf{w}}\hat{v}(S_t,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\gamma^t\delta\nabla_{\boldsymbol{\theta}}\ln\pi(A_t|S_t,\boldsymbol{\theta})$

$$\nabla_{\boldsymbol{\theta}}\ln\pi(a|s,\boldsymbol{\theta}) = \frac{\nabla_{\boldsymbol{\theta}}\pi(a|s,\boldsymbol{\theta})}{\pi(a|s,\boldsymbol{\theta})}$$

Actor-critic architecture



Actor-Critic methods

REINFORCE with baseline:

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha\Big(G_t - b(S_t)\Big)\frac{\nabla_{\boldsymbol{\theta}}\pi(A_t|S_t,\boldsymbol{\theta}_t)}{\pi(A_t|S_t,\boldsymbol{\theta}_t)}$$

Actor-Critic method:

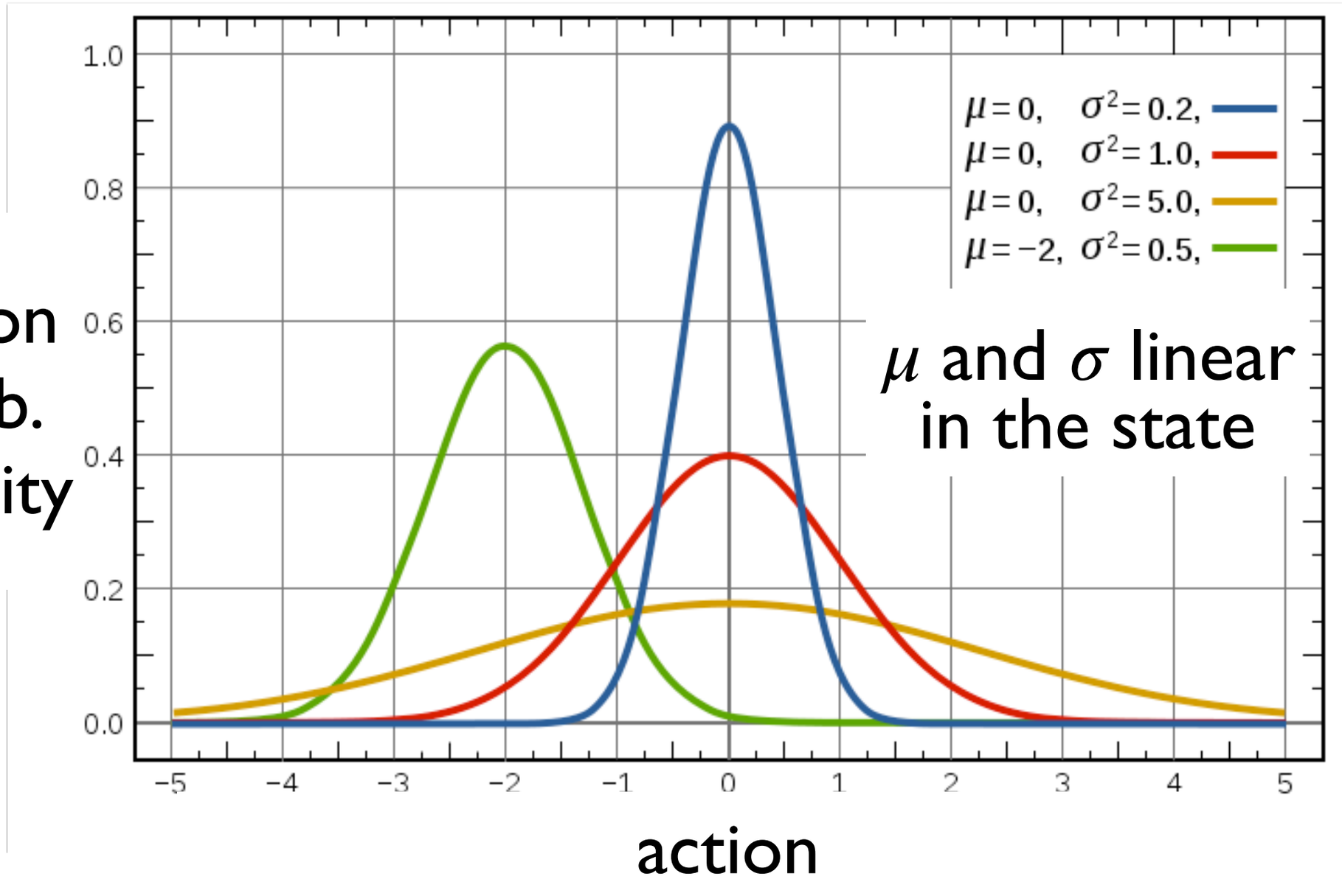
$$\begin{aligned}\boldsymbol{\theta}_{t+1} &\doteq \boldsymbol{\theta}_t + \alpha\Big(G_{t:t+1} - \hat{v}(S_t,\mathbf{w})\Big)\frac{\nabla_{\boldsymbol{\theta}}\pi(A_t|S_t,\boldsymbol{\theta}_t)}{\pi(A_t|S_t,\boldsymbol{\theta}_t)} \\ &= \boldsymbol{\theta}_t + \alpha\Big(R_{t+1} + \gamma\hat{v}(S_{t+1},\mathbf{w}) - \hat{v}(S_t,\mathbf{w})\Big)\frac{\nabla_{\boldsymbol{\theta}}\pi(A_t|S_t,\boldsymbol{\theta}_t)}{\pi(A_t|S_t,\boldsymbol{\theta}_t)} \\ &= \boldsymbol{\theta}_t + \alpha\delta_t\frac{\nabla_{\boldsymbol{\theta}}\pi(A_t|S_t,\boldsymbol{\theta}_t)}{\pi(A_t|S_t,\boldsymbol{\theta}_t)}.\end{aligned}$$

Actor–Critic with Eligibility Traces (continuing)

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value parameterization $\hat{v}(s,\mathbf{w})$
Parameters: step sizes $\alpha > 0$, $\beta > 0$, $\eta > 0$

$\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \mathbf{0}$ (d'-component eligibility trace vector)
$\mathbf{z}^{\mathbf{w}} \leftarrow \mathbf{0}$ (d-component eligibility trace vector)
Initialize $\bar{R} \in \mathbb{R}$ (e.g., to 0)
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$ (e.g., to $\mathbf{0}$)
Initialize $S \in \mathcal{S}$ (e.g., to s_0)
Repeat forever:
$A \sim \pi(\cdot|S,\boldsymbol{\theta})$
Take action A, observe S', R
$\delta \leftarrow R - \bar{R} + \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
$\bar{R} \leftarrow \bar{R} + \eta\delta$
$\mathbf{z}^{\mathbf{w}} \leftarrow \lambda^{\mathbf{w}}\mathbf{z}^{\mathbf{w}} + \nabla_{\mathbf{w}}\hat{v}(S,\mathbf{w})$
$\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \lambda^{\boldsymbol{\theta}}\mathbf{z}^{\boldsymbol{\theta}} + \nabla_{\boldsymbol{\theta}} \ln \pi(A|S,\boldsymbol{\theta})$
$\mathbf{w} \leftarrow \mathbf{w} + \beta\delta\mathbf{z}^{\mathbf{w}}$
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\delta\mathbf{z}^{\boldsymbol{\theta}}$
$S \leftarrow S'$

eg, linear-gaussian policies (continuous actions)



eg, linear-gaussian policies (continuous actions)

- The mean and std. dev. for the action taken in state s are linear and linear-exponential in

$$\boldsymbol{\theta} \doteq (\boldsymbol{\theta}_\mu^\top; \boldsymbol{\theta}_\sigma^\top)^\top \qquad \mu(s, \boldsymbol{\theta}) \doteq \boldsymbol{\theta}_\mu{}^\top \mathbf{x}_\mu(s) \qquad \sigma(s, \boldsymbol{\theta}) \doteq \exp\Big(\boldsymbol{\theta}_\sigma{}^\top \mathbf{x}_\sigma(s)\Big)$$

- The probability density function for the action taken in state s is gaussian

$$\pi(a|s, \boldsymbol{\theta}) \doteq \frac{1}{\sigma(s, \boldsymbol{\theta})\sqrt{2\pi}} \exp\left(-\frac{(a - \mu(s, \boldsymbol{\theta}))^2}{2\sigma(s, \boldsymbol{\theta})^2}\right)$$

Gaussian eligibility functions

$$\frac{\nabla_{\boldsymbol{\theta}_\mu} \pi(a|s,\boldsymbol{\theta})}{\pi(a|s,\boldsymbol{\theta})} = \frac{1}{\sigma(s,\boldsymbol{\theta})^2}(a - \mu(s,\boldsymbol{\theta}))\mathbf{x}_\mu(s)$$

$$\frac{\nabla_{\boldsymbol{\theta}_\sigma} \pi(a|s,\boldsymbol{\theta})}{\pi(a|s,\boldsymbol{\theta})} = \left(\frac{(a - \mu(s,\boldsymbol{\theta}))^2}{\sigma(s,\boldsymbol{\theta})^2} - 1\right)\mathbf{x}_\sigma(s)$$

Steps to understanding Policy-gradient methods

- Policy approximations $\pi(a|s,\boldsymbol{\theta})$
  - and their eligibility functions
- Approximate stochastic gradient ascent
- The policy-gradient theorem and its proof
- Approximating the gradient (REINFORCE)
- REINFORCE with a baseline
- Actor-critic methods

The generality of the policy-gradient strategy

- Can be applied whenever we can compute the effect of parameter changes on the action probabilities, $\nabla\pi(A_t|S_t,\boldsymbol{\theta})$
- E.g., has been applied to spiking neuron models
- There are many possibilities other than linear-exponential and linear-gaussian
  - e.g., mixture of random, argmax, and fixed-width gaussian; learn the mixing weights, drift/diffusion models

Goals for today

- Learn that policies can be optimized directly, without learning value functions, by *policy-gradient methods*
  - Glimpse how one could learn real-valued (continuous) actions
- Glimpse how to handle hidden state

Goals for today

- Learn that policies can be optimized directly, without learning value functions, by *policy-gradient methods*
  - Glimpse how one could learn real-valued (continuous) actions
- Glimpse how to handle hidden state

Hidden State

What it is
What to do about it

What is hidden state?

- Sometimes the environment includes state variables that are not visible to the agent
  - the agent sees only *observations*, not *state*
  - e.g., the object in the box, or in other rooms, velocities, even real positions as distinct from sensor readings
- This makes the environment Non-Markov

- All real problems involve extensive hidden state

- The agent's approximation to the hidden state of the environment will be imperfect and non-Markov

- But all of our methods rely on the Markov (state) property to some extent

- What to do?

Don't Panic

The usual over-reaction

- Introducing a whole new mathematical theory
  - like POMDPs (Partially Observable MDPs)
    - or HMMs (Hidden Markov Models)
- Relying on *complete models* of the hidden underlying environment and observation generators
  - even though these things are all hidden
- Thereby making both learning and planning *intractably complex*

There may be nothing you can do

- If the agent's approximate state is very poor, then any policy based on it will be poor

Use your tools!
1. Function approximation

- Features can be anything; they can be an arbitrary summary of past observations
- Nothing in our theory relies on the features being Markov

∴ FA will work ok with non-Markov features

Use your tools!
2. Eligibility traces

- Monte Carlo methods are much less reliant on having a good state approximation
  - because they don't bootstrap
- Eligibility traces allow our learning methods to be fully or partly Monte Carlo
  - and thus resistant to hidden state

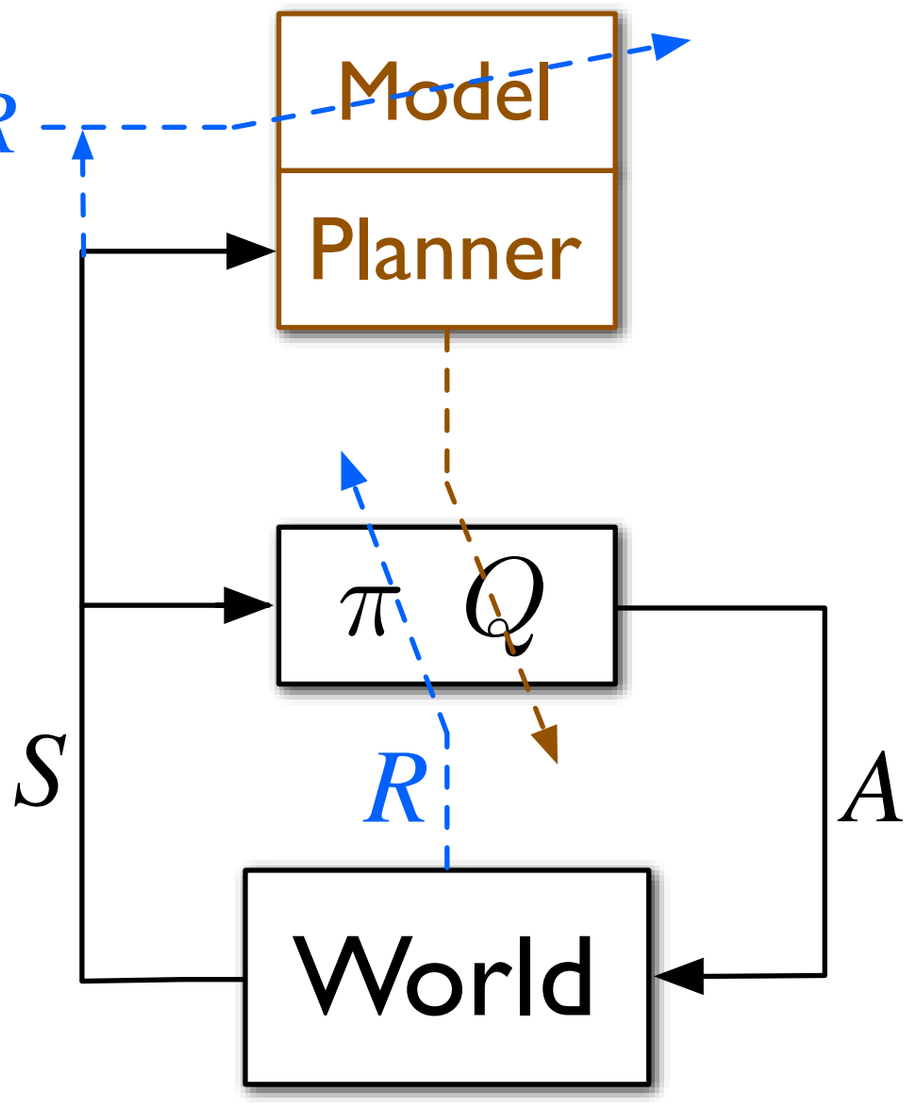
Remember: the bound of approximation accuracy depends on λ

Remember: why do we ever bootstrap?

The long-term solution

- Don’t panic
- Use your tools
- Embrace approximation
- Develop a recurrent process for updating the agent’s approximate state
  - Accept that it will be approximate, imperfect
  - And that it will have to monitored, debugged, improved…forever approximate

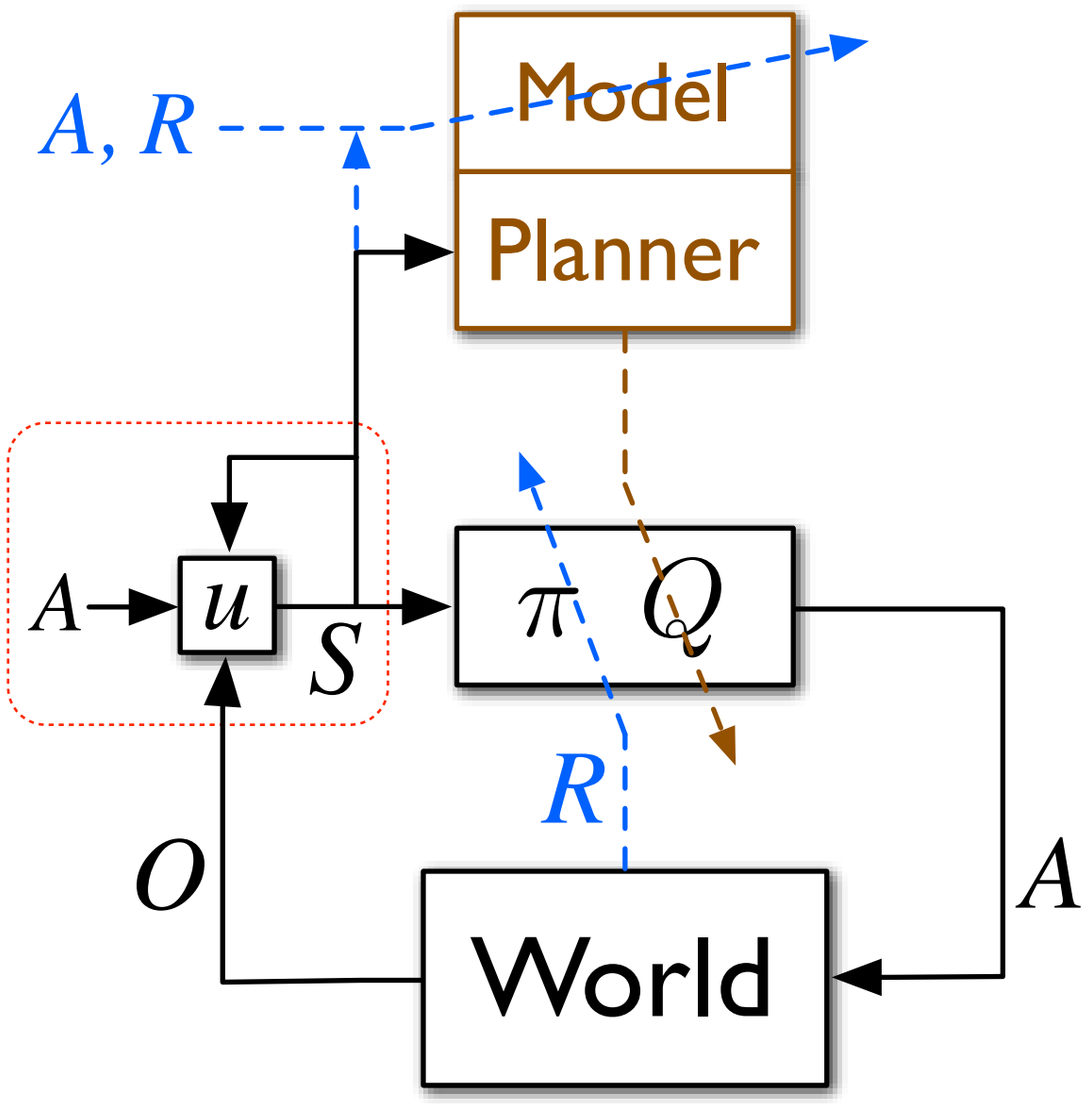
Foreground-background architecture



Planning is in the background

Interaction and learning are in the foreground

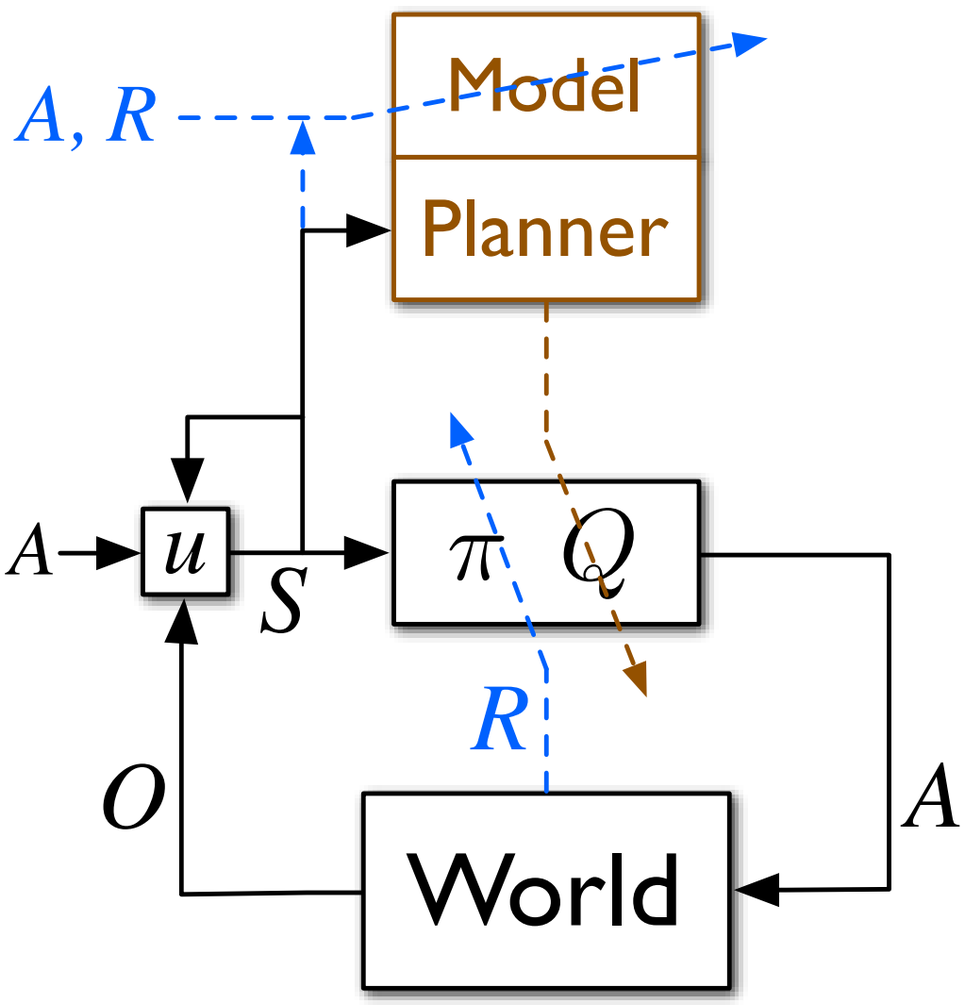
Foreground-background architecture
with partial observability



Planning is in the background

Interaction and learning are in the foreground

Agent state and its update

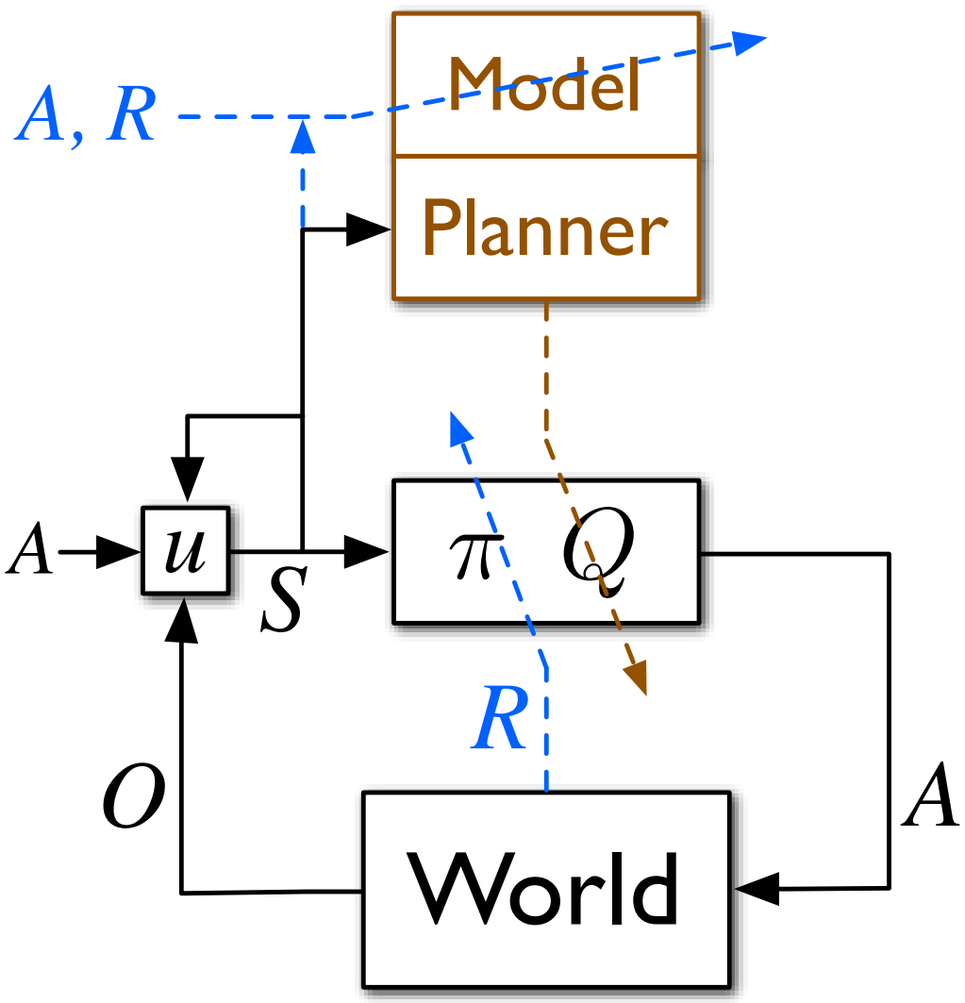


- Agent state is whatever the agent uses as state
  - in policy, value fn, model...
  - may differ from env state and information state
- State update:

$$S_{t+1} = u(S_t, A_t, O_{t+1})$$

  - e.g., Bayes rule, k-order Markov (history), PSRs, predictions

Planning should be state-to-state



$$S_{t+1} = u(S_t, A_t, O_{t+1})$$

- State update is in the foreground!
  - Planner and model see *only states*, never observations
- We lost this with POMDPs; Why?
  - Classical and MDP planning were always state-to-state
  - Planning can always be state-to-state in information state
- Function approximation makes planning in the info state a natural, flexible, and scalable approach

Goals for today

- Learn that policies can be optimized directly, without learning value functions, by *policy-gradient methods*
  - Glimpse how one could learn real-valued (continuous) actions
- Glimpse how to handle hidden state