Summary for off-policy

Off-policy RL with FA and TD remains challenging; there are multiple solution ideas, plus combinations

- Higher λ (less TD)
- Better state rep'ns (less FA)
- Recognizers (less off-policy)
- LSTD ($O(n^2)$ methods)
- Gradient TD, proximal gradient TD, and hybrids
- Emphatic TD

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

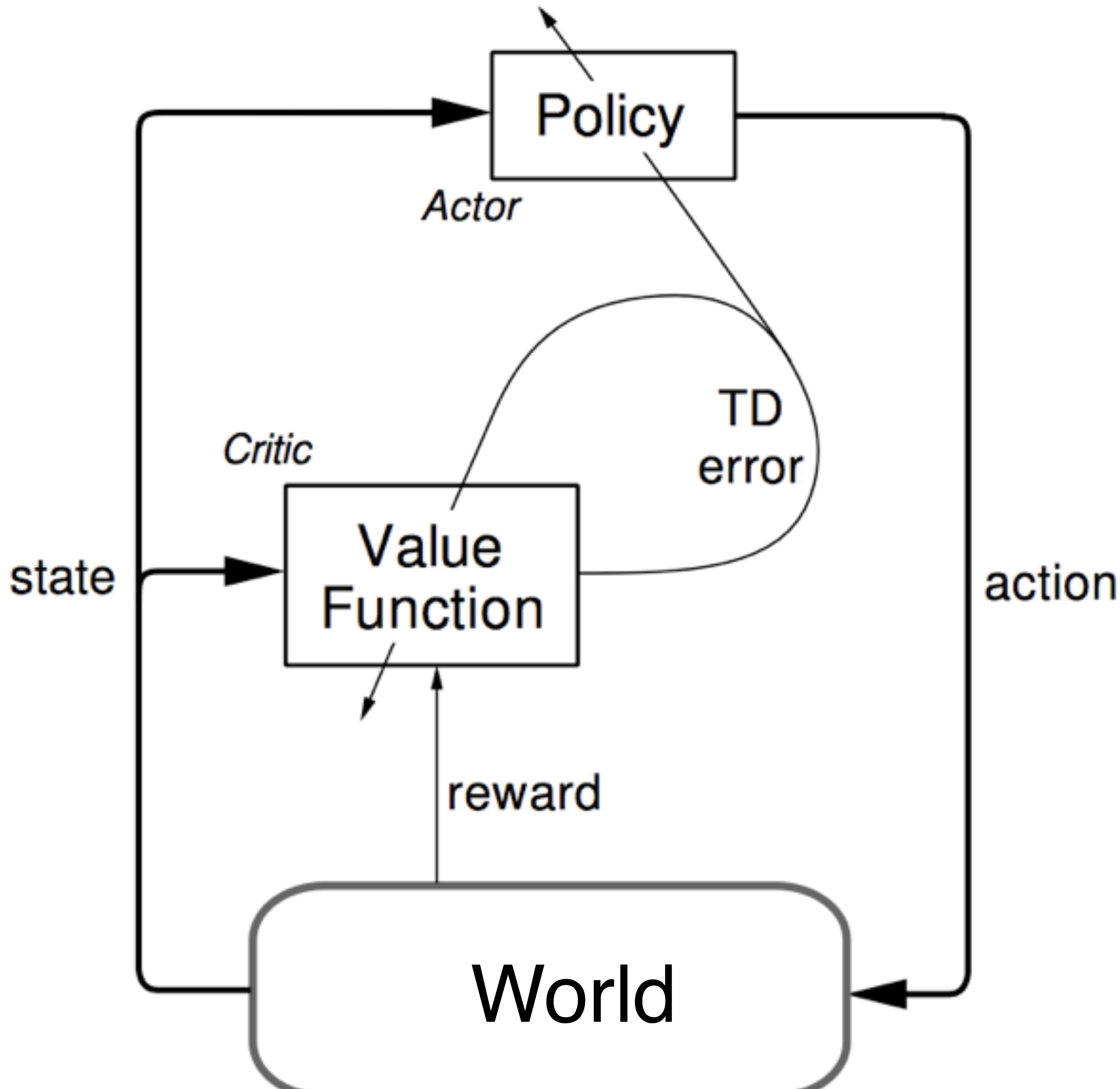
Policy-gradient methods

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

Actor-critic architecture



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

Policy Approximation

- Policy = a function from state to action
  - How does the agent select actions?
  - In such a way that it can be affected by learning?
  - In such a way as to assure exploration?
- Approximation: there are too many states and/or actions to represent all policies
  - To handle large/continuous action spaces

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 $\bar{R}_t$

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

- Then update:

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

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

eg, linear-exponential policies (discrete actions)

- The "preference" for action a in state s is linear in θ and a state-action feature vector $\phi(s,a)$

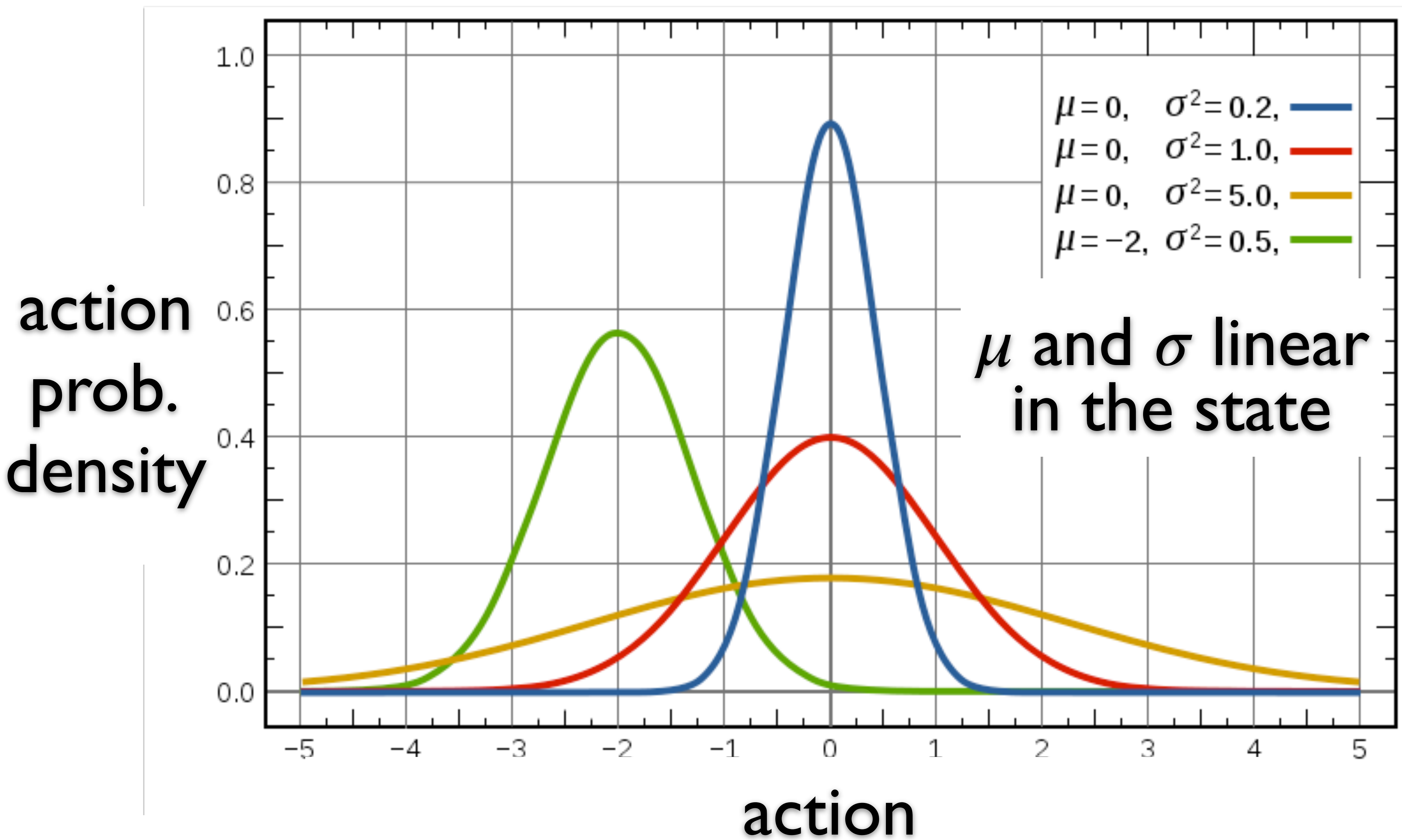
- The probability of action a in state s is exponential in its preference

$$\pi(a|s,\boldsymbol{\theta}) \doteq \frac{\exp(\boldsymbol{\theta}^\top \boldsymbol{\phi}(s,a))}{\sum_b \exp(\boldsymbol{\theta}^\top \boldsymbol{\phi}(s,b))}$$

- Corresponding *eligibility function:*

$$\frac{\nabla \pi(a|s,\boldsymbol{\theta})}{\pi(a|s,\boldsymbol{\theta})} = \boldsymbol{\phi}(s,a) - \sum_b \pi(b|s,\boldsymbol{\theta})\boldsymbol{\phi}(s,b)$$

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

$$\boldsymbol{\theta} \doteq (\boldsymbol{\theta}_\mu^\top; \boldsymbol{\theta}_\sigma^\top)^\top \qquad \mu(s) \doteq \boldsymbol{\theta}_\mu^\top \boldsymbol{\phi}(s) \qquad \sigma(s) \doteq \exp(\boldsymbol{\theta}_\sigma^\top \boldsymbol{\phi}(s))$$

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

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

Gaussian eligibility functions

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

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

Policy-gradient setup

Given a policy parameterization:

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

And objective:

$$\eta(\boldsymbol{\theta}) \doteq v_{\pi_{\boldsymbol{\theta}}}(S_0)$$ (or average reward)

Approximate stochastic gradient ascent:

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

Typically, based on the Policy-Gradient Theorem:

$$\nabla \eta(\boldsymbol{\theta}) = \sum_s d_\pi(s) \sum_a q_\pi(s,a) \nabla_{\boldsymbol{\theta}} \pi(a|s,\boldsymbol{\theta})$$

Proof of the Policy-Gradient Theorem

(from the 2nd Edition)

$$\nabla v_\pi(s) = \nabla \left[\sum_a \pi(a|s) q_\pi(s,a) \right], \quad \forall s \in \mathcal{S} \qquad \text{(Exercise 3.11)}$$

$$= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla q_\pi(s,a) \Big] \qquad \text{(product rule)}$$

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

$$\text{(Exercise 3.12 and Equation 3.8)}$$

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

$$\text{(Eq. 3.10)}$$

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

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

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

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

$$\nabla \eta(\boldsymbol{\theta}) = \nabla v_\pi(s_0)$$

$$= \sum_s \sum_{k=0}^{\infty} \gamma^k \Pr(s_0 \to s, k, \pi) \sum_a \nabla \pi(a|s) q_\pi(s,a)$$

$$= \sum_s d_\pi(s) \sum_a \nabla \pi(a|s) q_\pi(s,a). \qquad \text{Q.E.D.}$$

Deriving REINFORCE from the PGT

$$
\begin{aligned}
\nabla \eta(\boldsymbol{\theta}) &= \sum_s d_\pi(s) \sum_a q_\pi(s,a) \nabla_{\boldsymbol{\theta}} \pi(a|s,\boldsymbol{\theta}), \\
&= \mathbb{E}_\pi\left[\gamma^t \sum_a q_\pi(S_t,a) \nabla_{\boldsymbol{\theta}} \pi(a|S_t,\boldsymbol{\theta})\right] \\
&= \mathbb{E}_\pi\left[\gamma^t \sum_a \pi(a|S_t,\boldsymbol{\theta}) q_\pi(S_t,a) \frac{\nabla_{\boldsymbol{\theta}} \pi(a|S_t,\boldsymbol{\theta})}{\pi(a|S_t,\boldsymbol{\theta})}\right] \\
&= \mathbb{E}_\pi\left[\gamma^t q_\pi(S_t,A_t) \frac{\nabla_{\boldsymbol{\theta}} \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})}\right] \qquad \text{(replacing } a \text{ by the sample } A_t \sim \pi) \\
&= \mathbb{E}_\pi\left[\gamma^t G_t \frac{\nabla_{\boldsymbol{\theta}} \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})}\right] \qquad \text{(because } \mathbb{E}_\pi[G_t|S_t,A_t] = q_\pi(S_t,A_t))
\end{aligned}
$$

Thus

$$
\boldsymbol{\theta}_{t+1} \triangleq \boldsymbol{\theta}_t + \alpha \widehat{\nabla \eta(\boldsymbol{\theta}_t)} \triangleq \boldsymbol{\theta}_t + \alpha \gamma^t G_t \frac{\nabla_{\boldsymbol{\theta}} \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})}
$$

REINFORCE with baseline

Policy-gradient theorem with baseline:

any function of state, not action

$$\begin{aligned}\nabla\eta(\boldsymbol{\theta}) &= \sum_s d_\pi(s)\sum_a q_\pi(s,a)\nabla_{\boldsymbol{\theta}}\pi(a|s,\boldsymbol{\theta}) \\ &= \sum_s d_\pi(s)\sum_a \Big(q_\pi(s,a) - b(s)\Big)\nabla_{\boldsymbol{\theta}}\pi(a|s,\boldsymbol{\theta})\end{aligned}$$

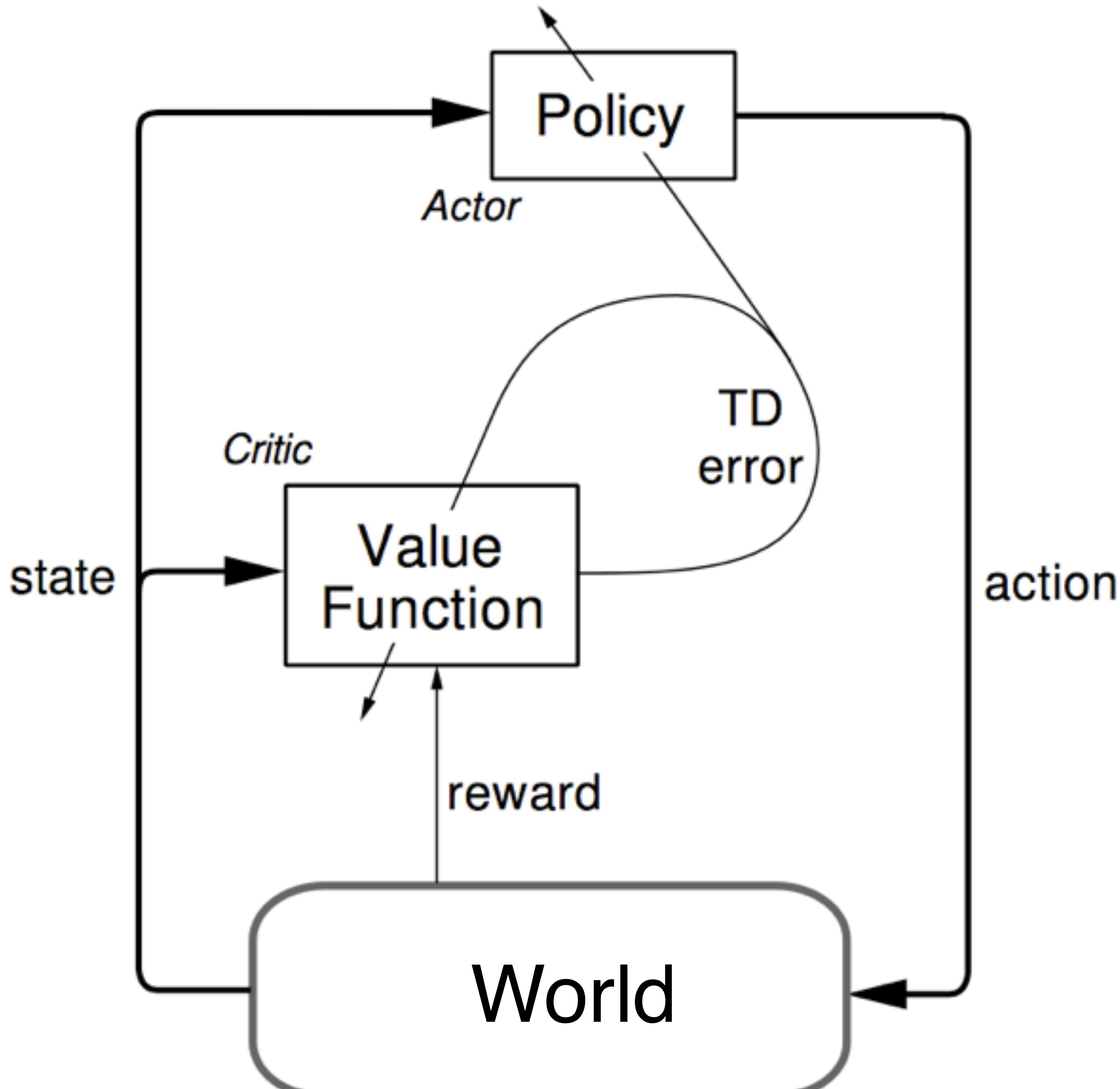
Because

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

Thus

$$\boldsymbol{\theta}_{t+1} \triangleq \boldsymbol{\theta}_t + \alpha\gamma^t\Big(G_t - b(S_t)\Big)\frac{\nabla_{\boldsymbol{\theta}}\pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})} \qquad \text{e.g., } b(s) = \hat{v}(s,\mathbf{w})$$

Actor-critic architecture



Actor-Critic methods

REINFORCE with baseline:

$$\boldsymbol{\theta}_{t+1} \triangleq \boldsymbol{\theta}_t + \overset{\gamma^t}{\underset{\wedge}{\alpha}}\Big(G_t - b(S_t)\Big)\frac{\nabla_{\boldsymbol{\theta}}\pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})}$$

Actor-Critic method:

$$\begin{aligned}\boldsymbol{\theta}_{t+1} &\triangleq \boldsymbol{\theta}_t + \overset{\gamma^t}{\underset{\wedge}{\alpha}}\Big(G_t^{(1)} - \hat{v}(S_t,\mathbf{w})\Big)\frac{\nabla_{\boldsymbol{\theta}}\pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})} \\ &= \boldsymbol{\theta}_t + \overset{\gamma^t}{\underset{\wedge}{\alpha}}\Big(R_{t+1} \overset{-\bar{R}_t}{\underset{\wedge}{}} + \gamma\hat{v}(S_{t+1},\mathbf{w}) - \hat{v}(S_t,\mathbf{w})\Big)\frac{\nabla_{\boldsymbol{\theta}}\pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})}\end{aligned}$$

Complete PG algorithm

Initialize parameters of policy $\boldsymbol{\theta} \in \mathbb{R}^n$, and state-value function $\mathbf{w} \in \mathbb{R}^m$

Initialize eligibility traces $\mathbf{e}^{\boldsymbol{\theta}} \in \mathbb{R}^n$ and $\mathbf{e}^{\mathbf{w}} \in \mathbb{R}^m$ to $\mathbf{0}$

Initialize $\bar{R} = 0$

On each step, in state S:

- Choose A according to $\pi(\cdot|S, \boldsymbol{\theta})$
- Take action A, observe S', R
- $\delta \leftarrow R - \bar{R} + \hat{v}(S', \mathbf{w}) - \hat{v}(S, \mathbf{w})$ form TD error from critic
- $\bar{R} \leftarrow \bar{R} + \alpha^{\boldsymbol{\theta}} \delta$ update average reward estimate
- $\mathbf{e}^{\mathbf{w}} \leftarrow \lambda \mathbf{e}^{\mathbf{w}} + \nabla_{\mathbf{w}} \hat{v}(S, \mathbf{w})$ update eligibility trace for critic
- $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \mathbf{e}^{\mathbf{w}}$ update critic parameters
- $\mathbf{e}^{\boldsymbol{\theta}} \leftarrow \lambda \mathbf{e}^{\boldsymbol{\theta}} + \frac{\nabla \pi(A|S,\boldsymbol{\theta})}{\pi(A|S,\boldsymbol{\theta})}$ update eligibility trace for actor
- $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \delta \mathbf{e}^{\boldsymbol{\theta}}$ update actor parameters

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

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

$\therefore$ 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

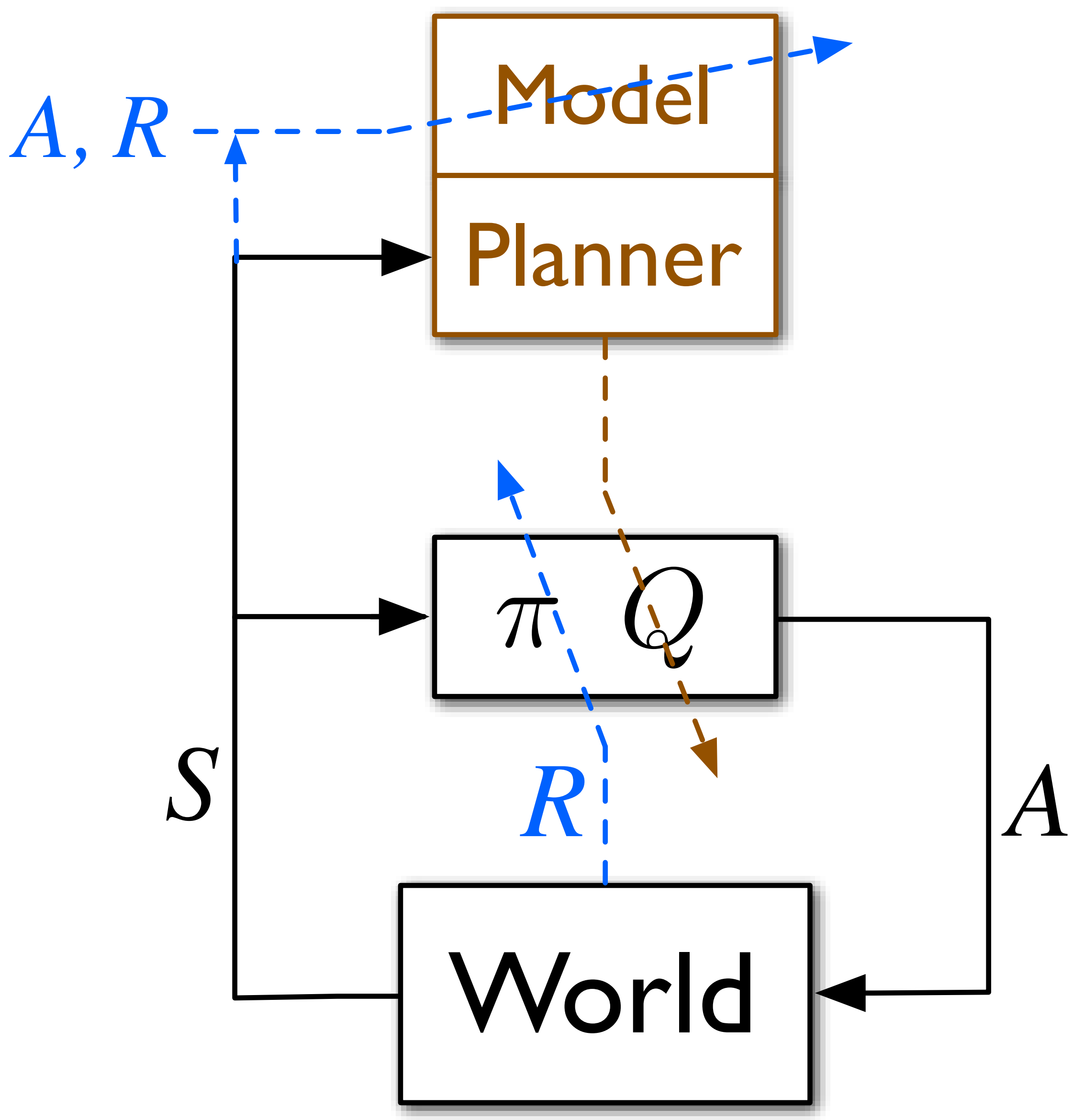
Remember: the bound of approximation accuracy depends on λ

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

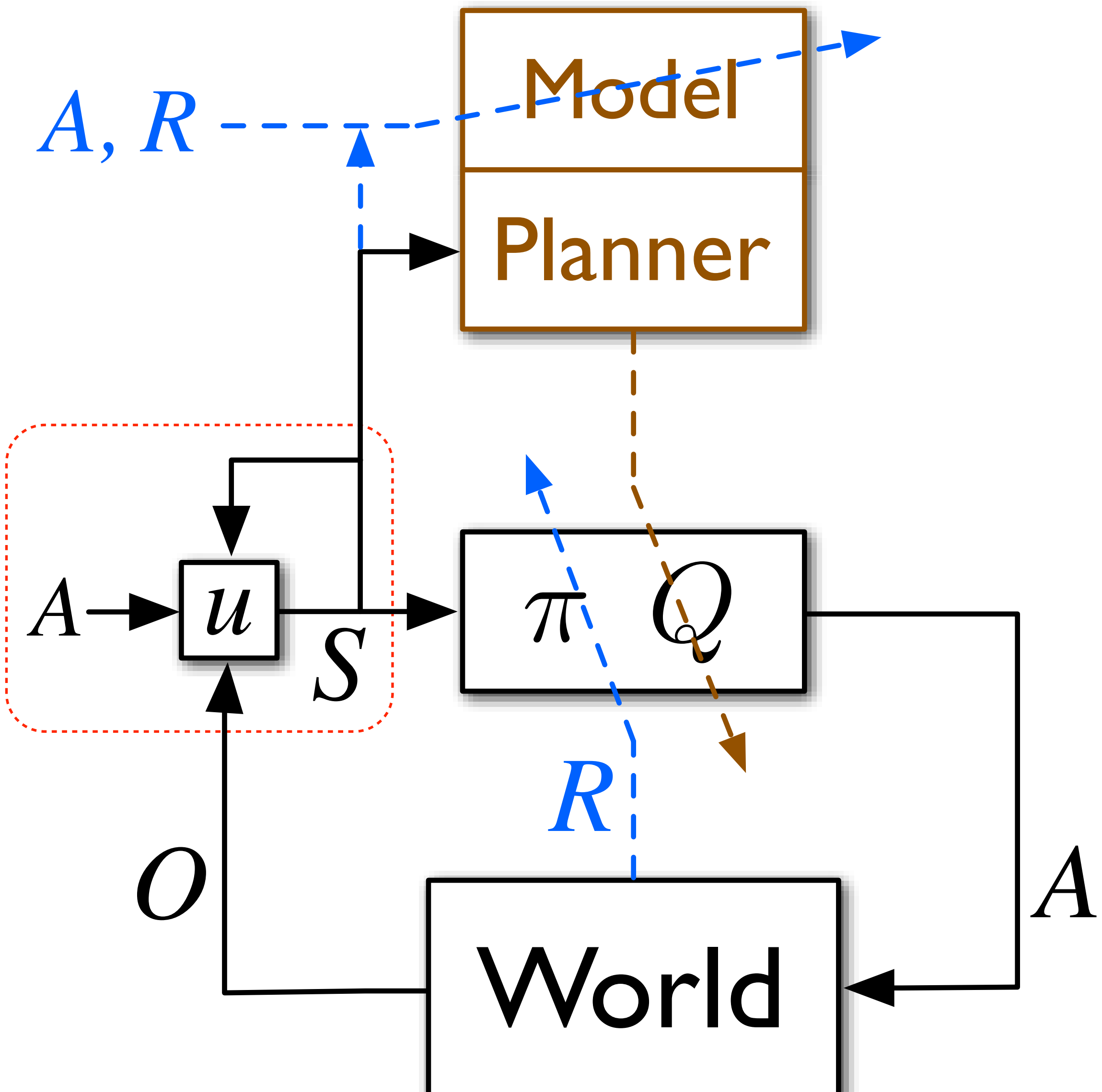


Planning is in the background

Interaction and learning are in the foreground

Foreground-background architecture *with partial observability*

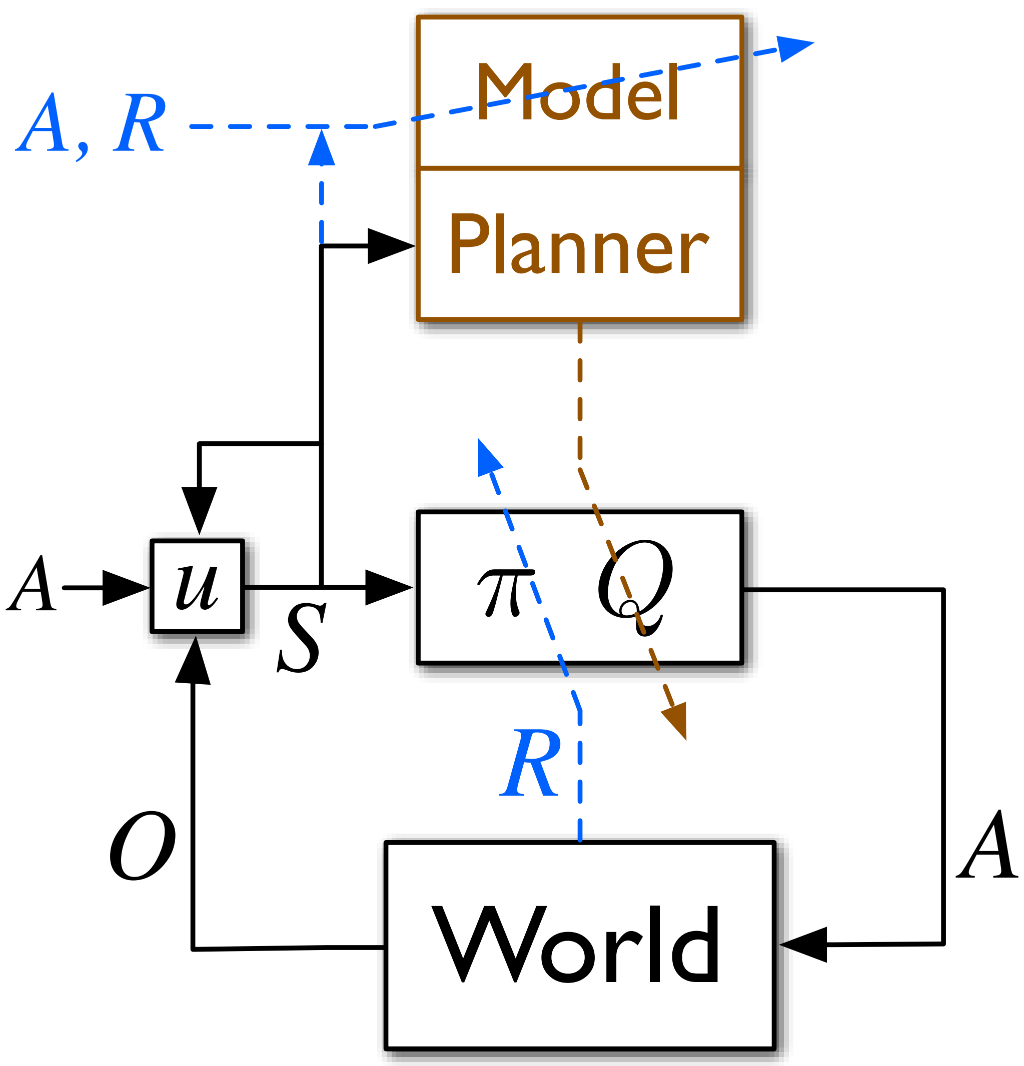
State update



Planning is in the background

Interaction and learning are in the foreground

Agent state and its update

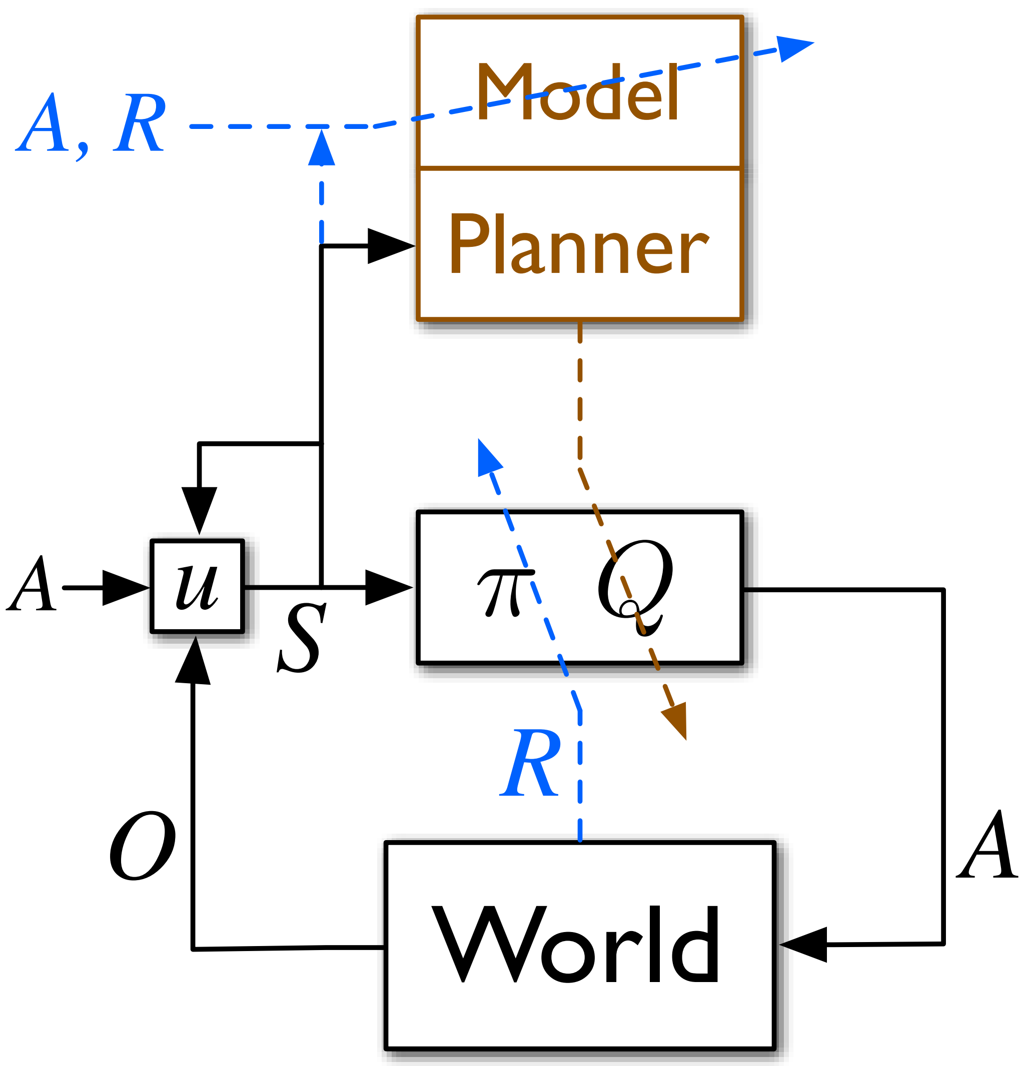


- Agent state is whatever the agent uses as state
  - in policy, value fn, model...
  - may differ from env state and information state
- State update:

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

  - 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