UCI University of California, Irvine

CS 277: Control and Reinforcement Learning

Winter 2021

Lecture 11: Partial Observability

Roy Fox
Department of Computer Science
Bren School of Information and Computer Sciences
University of California, Irvine

Today's lecture

MPC, Local Models

Partially Observable MDPs (POMDPs)

Belief-state MDPs

RNNs

Issues with approximate models (1)

- In large state / action spaces, we can only approximate the dynamics
- No guarantees outside of training distribution
    - As in model-free RL, we can't be too far off-policy
- Solution: keep interacting using learner policy and updating the model

Issues with approximate models (2)

- Model inaccuracy accumulates
  - If $|p_\phi(s'|s,a) - p(s'|s,a)|_1 \leq \epsilon$ then $|p_\phi(s_t) - p(s_t)|_1 \leq \epsilon t$
  - We have to plan far enough ahead to realize the consequences of actions
  - But we don't have to execute those plans far ahead!
- Model Predictive Control (MPC):

$\mathcal{D} \leftarrow$ collect data
repeat
  $\hat{\mathcal{M}} \leftarrow$ train model $\hat{p}, \hat{r}$ from $\mathcal{D}$
  repeat
    $\pi \leftarrow$ plan in $\hat{\mathcal{M}}$ from current state s to horizon H
    take *one action* a according to π
    add empirical (s, a, r, s') to $\mathcal{D}$

How to use a learned model

- Recall how planning benefitted from access to a model:
  - As a fast simulator
  - As an arbitrary-reset simulator
  - As a differentiable model

Local models

- Can we use a learned model for iLQR?
  - Option 1: learn global model, linearize locally $\implies$ wasteful
  - Option 2: directly learn local linearizations:

initialize a policy $\pi(u_t|x_t)$
repeat
  roll out π to horizon T for N trajectories
  fit $p(x_{t+1}|x_t, u_t)$
  plan new policy π

How to fit local dynamics

- Option 1: linear regression
  - Find $(A_t, B_t)_{t=0}^{T-1}$ such that $x_{t+1} \approx A_t x_t + B_t u_t$
  - Do we care about error / noise?
    - If we assume it's Gaussian, doesn't affect policy; but could help evaluate the method
- Option 2: Bayesian linear regression
  - Use global model as prior
  - More data efficient across time steps and across iterations

How to plan with local models

- Option 1: as in iLQR, find optimal control sequence $\hat{u}$
  - Problem: model errors will cause actual trajectory to diverge
- Option 2: execute the optimal policy $\hat{L}_t \delta x_t + \hat{\ell}_t + \hat{u}_t$ directly in the world
  - Problem: need spread for linear regression, dynamics may be too deterministic
- Option 3: make control stochastic $\hat{L}_t \delta x_t + \hat{\ell}_t + \hat{u}_t + \epsilon_t$
  - Idea: have $\epsilon_t \sim \mathcal{N}(0, R^{-1})$
    - Optimal for the incurred costs, not for the spread needed for regression

Recap

- Roughly two schemes:
  - Plan in a learned model
  - Improve model-free RL using a learned model
- Good theory for how to explore optimally for learning a model

Today's lecture

MPC, Local Models

Partially Observable MDPs (POMDPs)

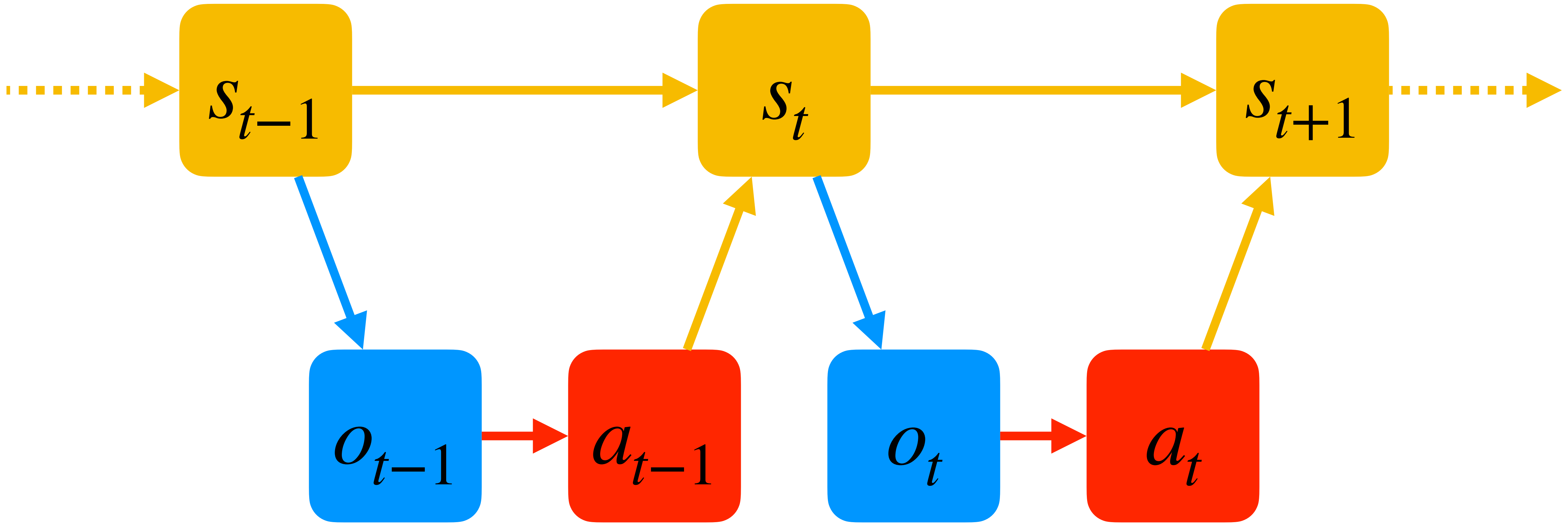
Belief-state MDPs

RNNs

What does the policy depend on?

- Minimally: nothing
  - Just an open-loop sequence of actions $a_0, a_1, \ldots$
    - Except, even this depends on a clock $a_t = \pi(t)$
- Typically: the current state $\pi(a_t \mid s_t)$
- What if the state is not fully observable to the agent's sensors?
  - Completely unobservable → forced open loop
  - Partially observable → $\pi(a_t \mid o_t)$?

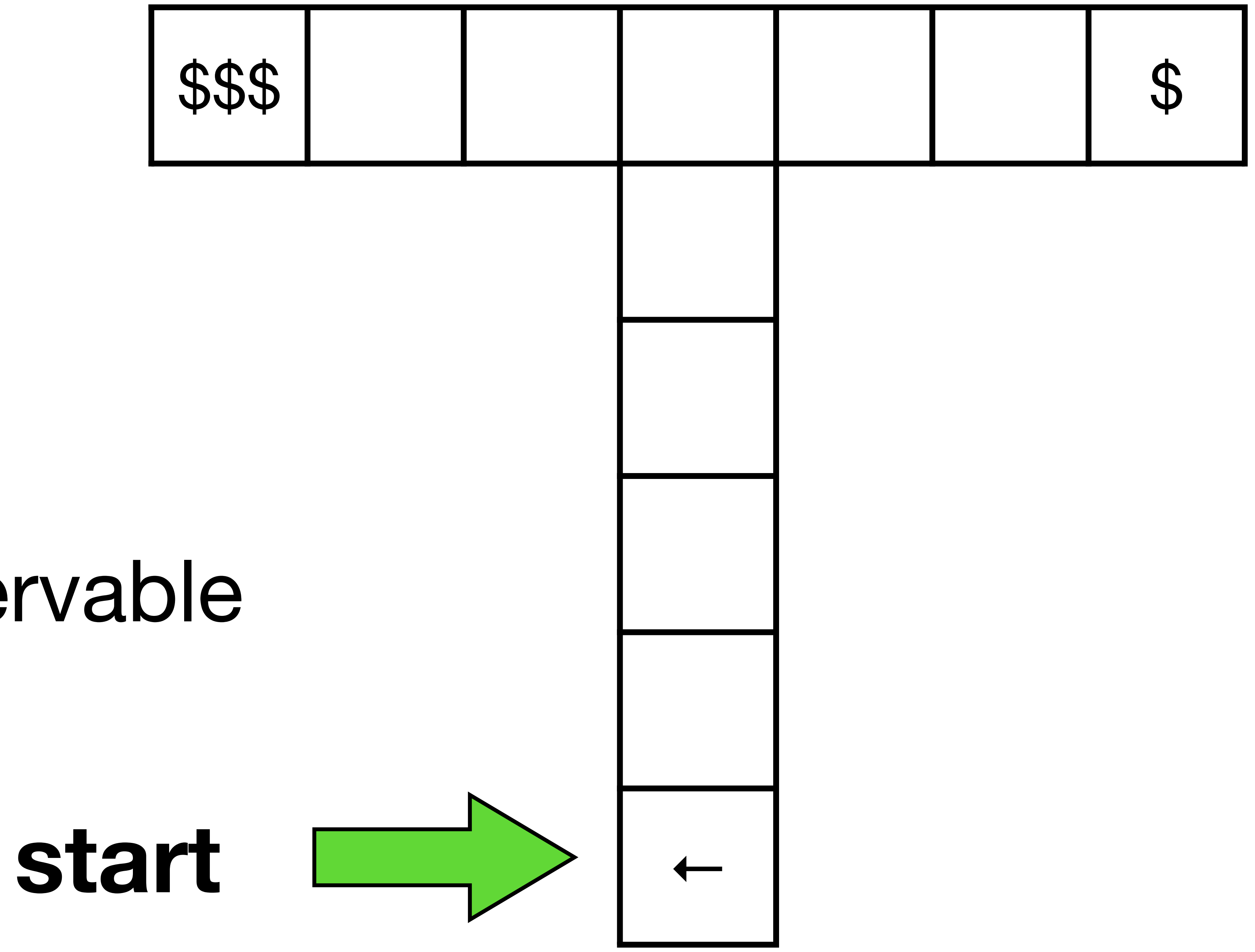
Partially Observable Markov Decision Process (POMDP)



- States $\mathcal{S}$
- Actions $\mathcal{A}$
- Observations $\mathcal{O}$
- Transitions $p(s_{t+1} | s_t, a_t)$
- Emissions $p(o_t | s_t)$
- Rewards $r(s_t, a_t)$

T-maze domain

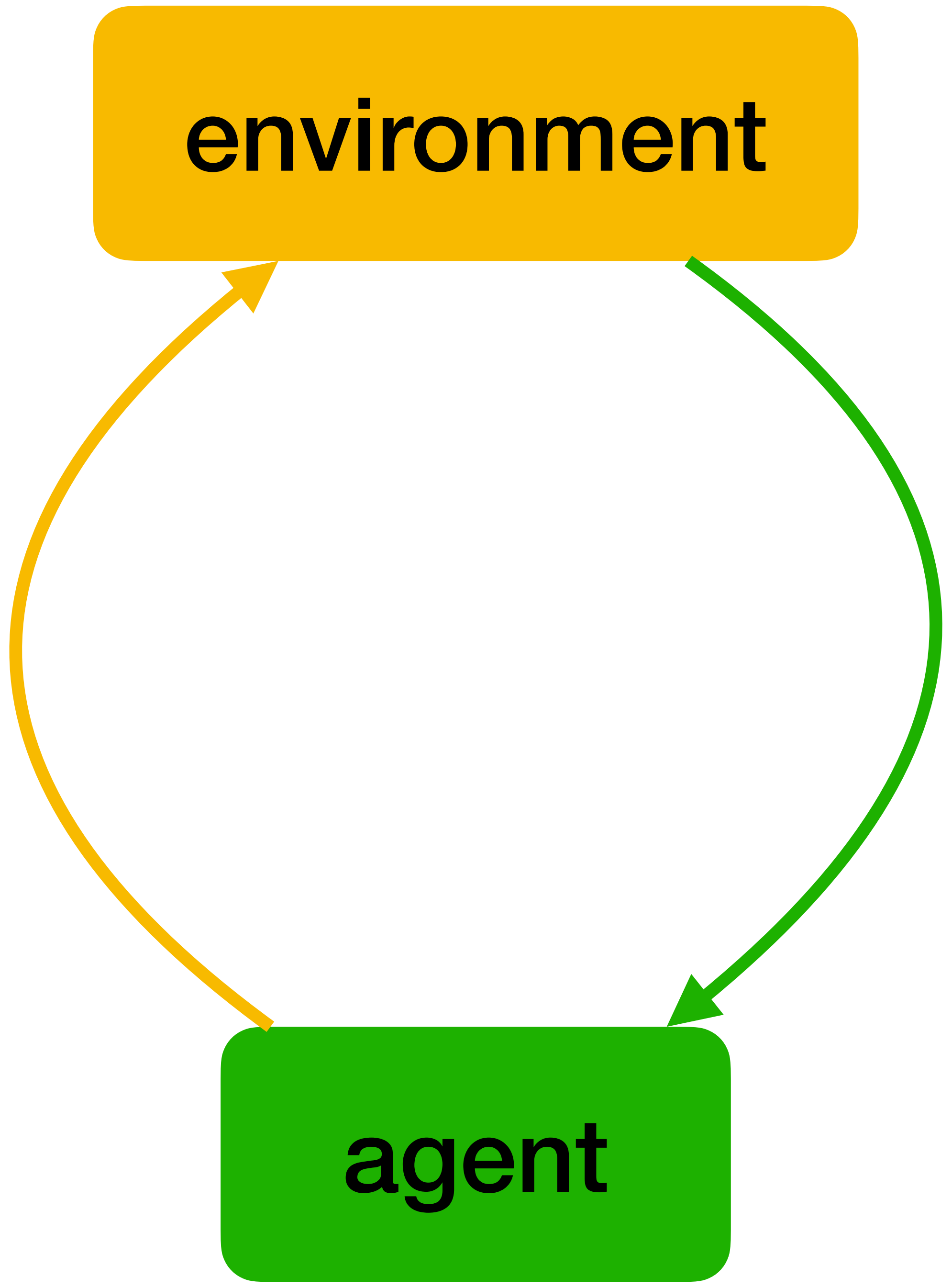
- Observation: current cell
- Observe cue at start
  - Decision at T-junction — cue no longer observable
- Memory is needed

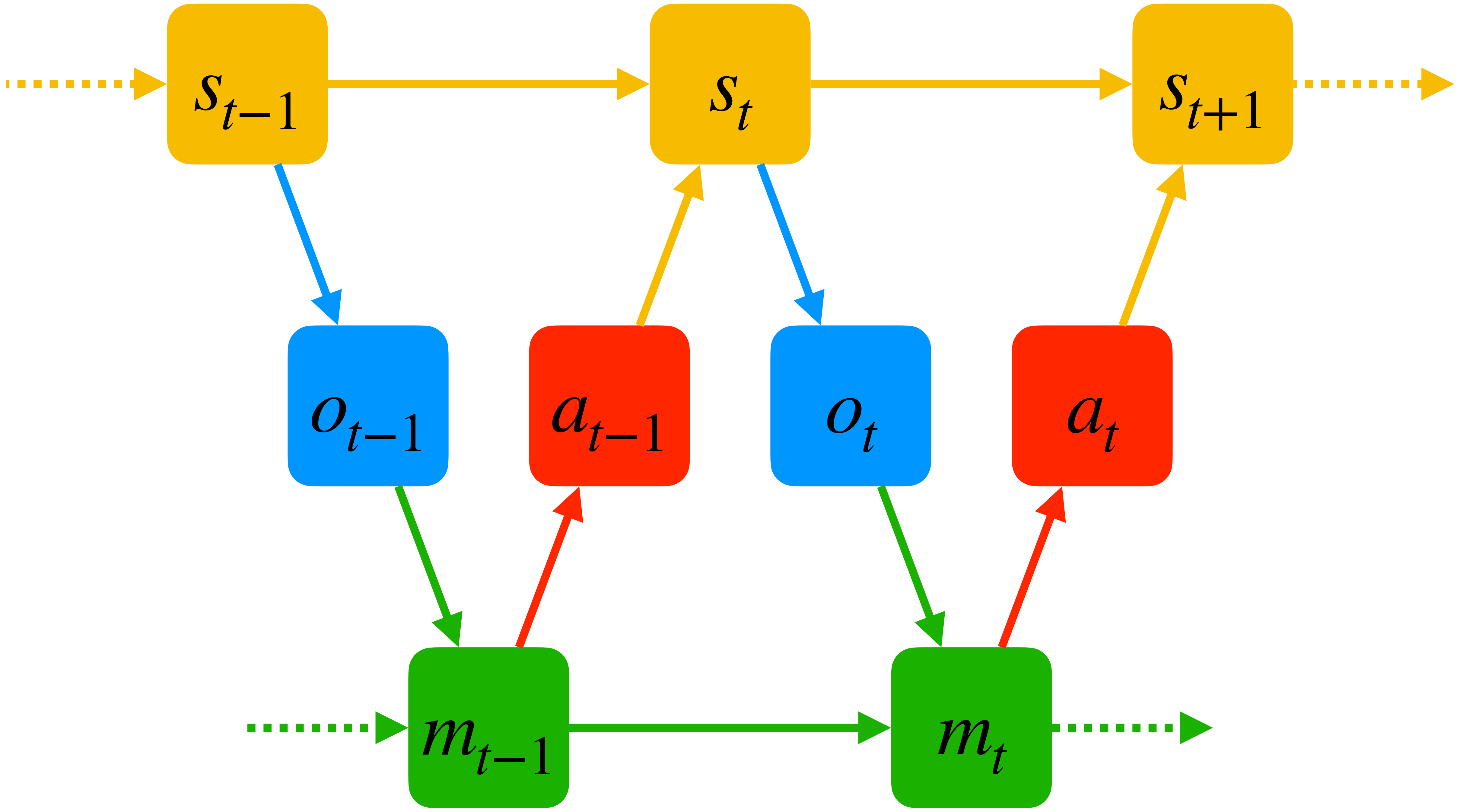


What does the policy depend on? (revisited)

- Maximally: the entire observable history $\pi(a_t | h_t = (o_0, o_1, \ldots, o_t))$
  - Should we remember past actions?
    - In a stochastic policy, yes $h_t = (o_0, a_0, o_1, a_1, \ldots, o_t)$
    - In a deterministic policy, we could regenerate them (with compute cost)
- Problem: we can't have unbounded memory that grows with t
- Solution 1: keep a window of k last observations $\pi(a_t | o_{t-k+1}, \ldots, o_{t-1})$
- Solution 2: keep a statistic of the observable history $\pi(a_t | m_t)$, with some $\pi(m_t | h_t)$
  - Memory must allow sequential updates: $\pi(m_t | m_{t-1}, o_t)$

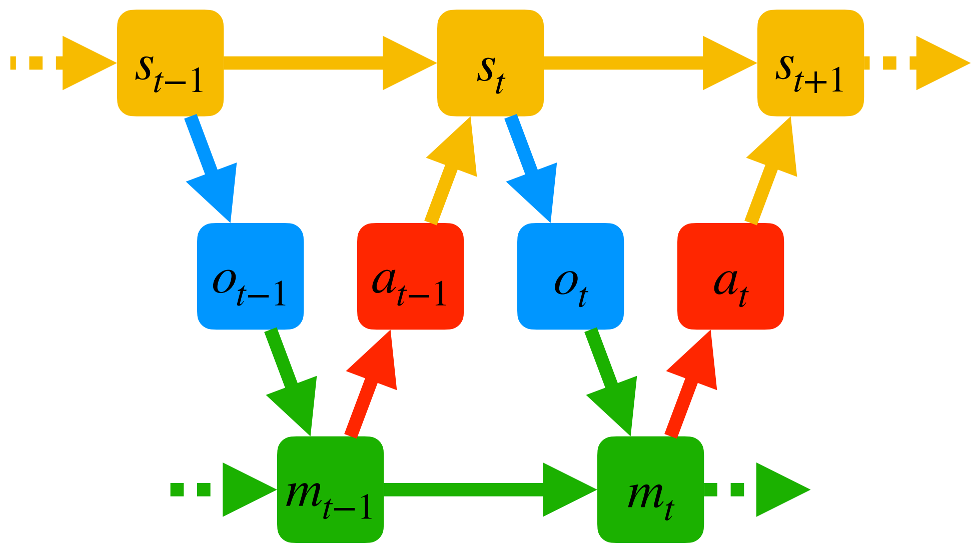
Agent–environment interaction





- Agent policy: $\pi(m_t, a_t | m_{t-1}, o_t) = \pi(m_t | m_{t-1}, o_t)\pi(a_t | m_t)$
- For simplicity, no edge from a_{t-1} to m_t
  - Can make a_{t-1} explicitly observable in o_t, or explicitly remembered in m_{t-1}

So what is memory?



- There's no Markov property in the observable process alone
  - All past observations may be informative of future actions
- Filter the observable past to provide more information about the hidden state
- No less important: plan for the future
  - Previously, we needed to trade off short-term with long-term rewards
  - Now we also need to trade off with information-gathering = active perception
- In multi-agent: state of the world is incomplete without other agent's memory
  - Theory of mind

Tiger domain

- 2 states: which door leads to a tiger (-100 reward) and which to $$$ (+10)

- You can stop and listen: $p(o_t = s_t | s_t) = 0.8$

$p(s_0 = \text{left}) = 0.5; \quad \mathbb{E}[r(s_0, \text{left})] = -45$ → listen → $o_1 = \text{right}$

$p(s_1 = \text{left}) = 0.2; \quad \mathbb{E}[r(s_1, \text{left})] = -12$ → listen → $o_2 = \text{left}$

$p(s_2 = \text{left}) = 0.5; \quad \mathbb{E}[r(s_2, \text{left})] = -45$ → listen → $o_3 = \text{right}$

$p(s_3 = \text{left}) = 0.2; \quad \mathbb{E}[r(s_3, \text{left})] = -12$ → listen → $o_4 = \text{right}$

$$p(s_4 = \text{left}) = \frac{0.04}{0.04 + 0.64} \approx 0.06; \quad \mathbb{E}[r(s_4, \text{left})] \approx 3.5$$

$$p(s_5 = \text{left}) = \approx 0.015; \quad \mathbb{E}[r(s_5, \text{left})] \approx 8.3$$

Today's lecture

MPC, Local Models

Partially Observable MDPs (POMDPs)

Belief-state MDPs

RNNs

Sufficient statistics

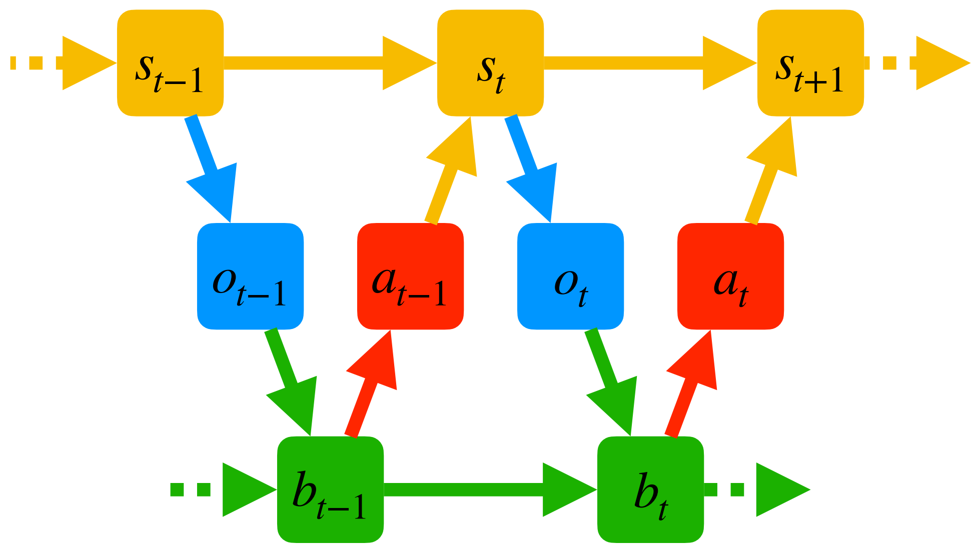
- Statistic of h = independent of all else given h
  - Satisfying the Markov chain $s - h - m$
  - Data processing inequality (DPI): $\mathbb{I}[s; m] \leq \mathbb{I}[s; h]$
- Sufficient statistic of h for s = statistic that has $s - m - h$
  - $\implies \mathbb{I}[s; m] = \mathbb{I}[s; h] \implies p(s \mid m) = p(s \mid h)$
- Belief = distribution over the state $b(s)$
- Bayesian belief $b(s) = p(s \mid h)$: a sufficient statistic of h for s

what is $p(s \mid b)$ for a Bayesian belief?

$p(s \mid b) = b(s) = p(s \mid h)$

not true for all beliefs!

Computing the Bayesian belief



- In the linear–Gaussian case: the Kalman filter
  - Bayesian belief is Gaussian $p(x_t | \hat{x}_t) = \mathcal{N}(\hat{x}_t, \Sigma_t)$
  - Precomputed covariance $\mathrm{var}(\mathrm{x_t} | \hat{\mathrm{x}}_\mathrm{t}) = \Sigma_t$; mean updated linearly:

$$\hat{x}'_t = A\hat{x}_{t-1} + Bu_{t-1} \qquad e_t = y_t - C\hat{x}'_t \qquad \hat{x}_t = \hat{x}'_t + K_t e_t$$

- More generally — use Bayes' rule:

total probability over s_t / previous belief / known dynamics

$$b'_t(s_{t+1} | h_t, a_t) = \sum_{s_t} p(s_t | h_t) p(s_{t+1} | s_t, a_t) = \sum_{s_t} b_t(s_t) p(s_{t+1} | s_t, a_t)$$

previous prediction / observation model

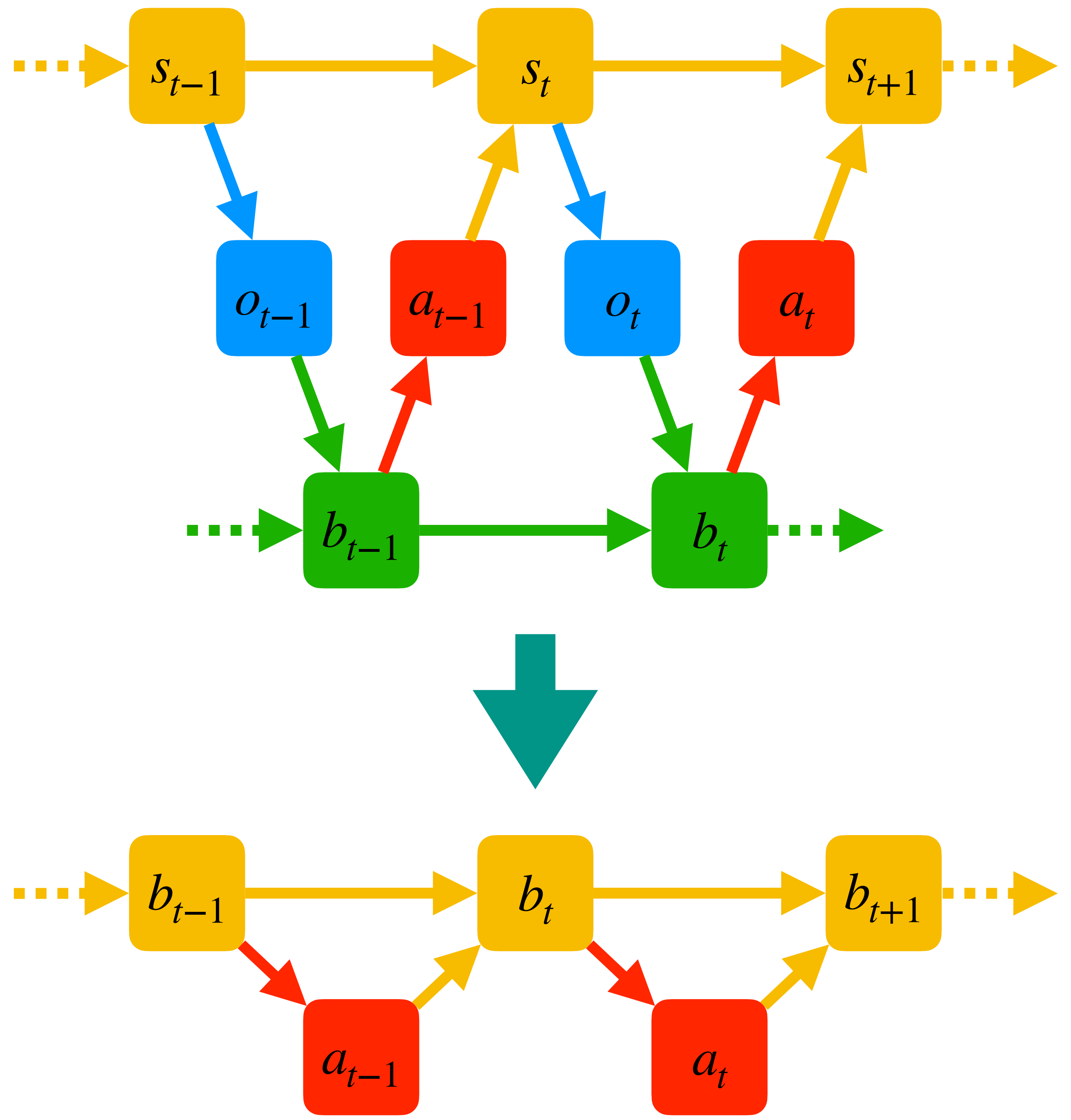
$$b_{t+1}(s_{t+1} | h_{t+1} = (h_t, a_t, o_{t+1})) = \frac{p(s_{t+1} | h_t, a_t) p(o_{t+1} | s_{t+1})}{p(o_{t+1} | h_t, a_t)} = \frac{b'_t(s_{t+1}) p(o_{t+1} | s_{t+1})}{\sum_{\bar{s}_{t+1}} b'_t(\bar{s}_{t+1}) p(o_{t+1} | \bar{s}_{t+1})}$$

Bayes' rule / $o_{t+1} - s_{t+1} - (h_t, a_t)$ / normalizer

  - This is a deterministic update of belief-state b_t, given an action a_t and next observation o_{t+1}

Belief-state MDP

- In the linear–quadratic–Gaussian case: certainty equivalence
  - Plan using $\hat{x}_t$ as if it was x_t
- More generally (though vastly less useful): belief-state MDP
  - States: $\Delta(\mathcal{S})$ Actions: $\mathcal{A}$ Rewards: $r(b_t, a_t) = \sum_{s_t} b_t(s_t) r(s_t, a_t)$



- Transitions: each possible observation o_{t+1} contributes its probability

$$p(o_{t+1} \mid b_t, a_t) = \sum_{s_t, s_{t+1}} b_t(s_t) p(s_{t+1} \mid s_t, a_t) p(o_{t+1} \mid s_{t+1})$$

to the total probability that the belief that follows (b_t, a_t, o_{t+1}) is the Bayesian belief

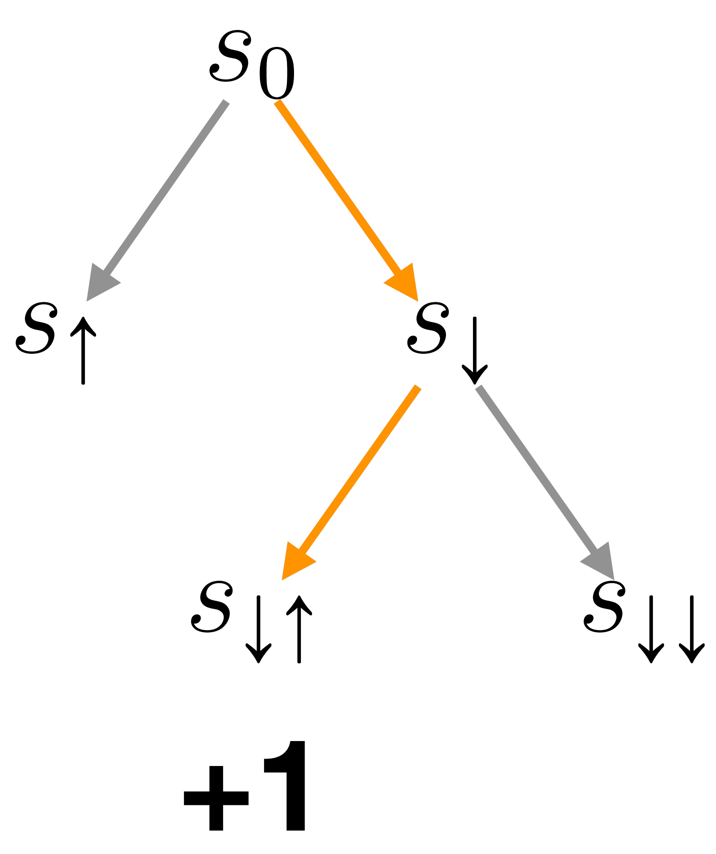
$$b_{t+1}(s_{t+1}) = p(s_{t+1} \mid b_t, a_t, o_{t+1}) = \frac{\sum_{s_t} b_t(s_t) p(s_{t+1} \mid s_t, a_t) p(o_{t+1} \mid s_{t+1})}{p(o_{t+1} \mid b_t, a_t)}$$

Memory is hard...

- Belief space $b(s_t)$ is continuous, as high-dimensional as the state space
  - Curse of dimensionality
  - Beliefs are naturally multi-modal — how do we even represent them?
- The number of reachable beliefs may grow exponentially with time
  - Curse of history
- As we'll see, belief-value function very complex, hard to approximate
- There may not be optimal stationary deterministic policy $\implies$ instability

Stationary deterministic policy counterexample

- Assume no observability
- Stationary deterministic policies gets no reward
- Non-stationary policy: ↓, ↑; expected return: +1
  - But non-stationary = observability of a clock t
- Stationary stochastic policy: ↓ / ↑ with equal prob.; expected return: +0.25



- Open problem: Bellman backup is inherently stationary and deterministic

no dependence on t → $V(s)$; **maximum achieved for some action** → $\max_a$

$$V(s) = \max_a r(s,a) + \gamma \mathbb{E}_{s'|s,a\sim p}[V(s')]$$

Today's lecture

MPC, Local Models

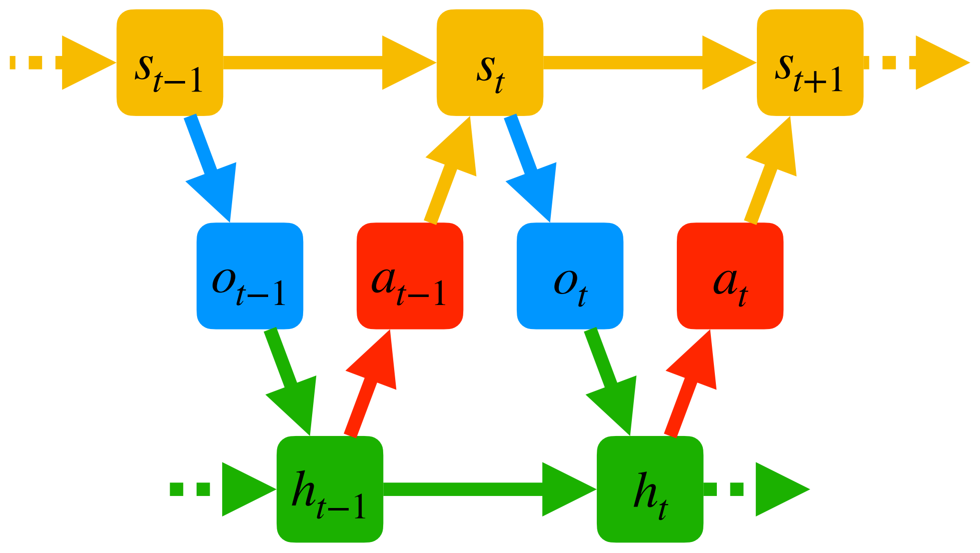
Partially Observable MDPs (POMDPs)

Belief-state MDPs

RNNs

Filtering with function approximation

- Instead of Bayesian belief, compute memory update $h_t = f_\theta(h_{t-1}, o_t)$



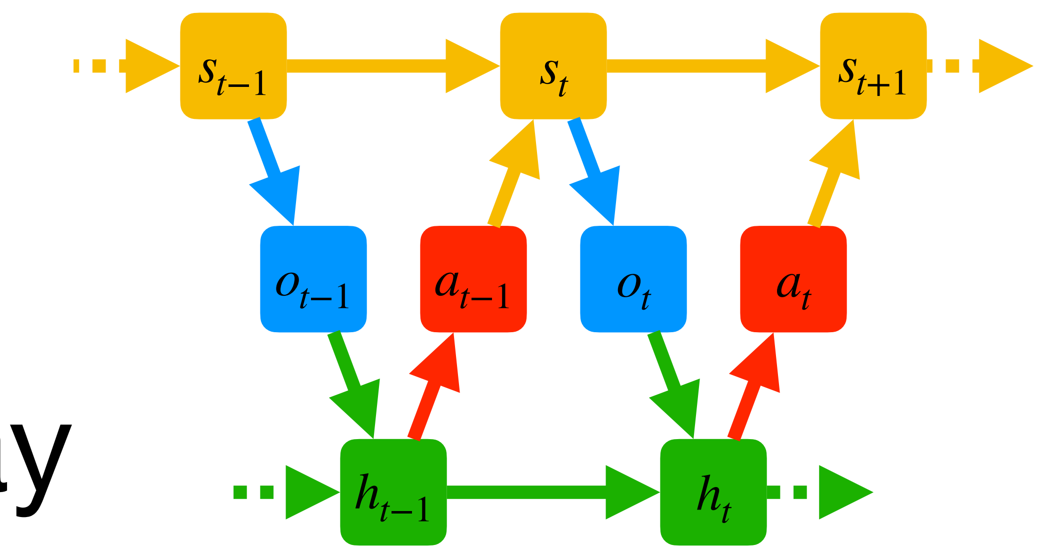
  - Action policy: $\pi_\theta(a_t | h_t)$
  - Sequential structure = Recurrent Neural Network (RNN)
- Training = back-propagate gradients through the whole sequence
  - Back-propagation through time (BPTT)
- Unfortunately, gradients tend to vanish → 0 / explode → ∞
  - Long term coordination of memory updates + actions is challenging
  - RNN can't use information not remembered, but no memory gradient unless used

RNNs in on-policy methods

- Training RNNs with on-policy methods is straightforward (and backward)
  - Roll out policy: parameters of a_t distribution are determined by $\pi_\theta(h_t)$ with

$$h_t = f_\theta(\cdots f_\theta(f_\theta(o_0), o_1), \cdots o_t)$$

  - Compute $\nabla_\theta \log \pi_\theta(a_t \mid h_t)$ with BPTT all the way to initial observation o_0
- Problems: computation graph > RAM, vanishing / exploding grads
- Solution: stop gradients every k steps
- Problem: cannot learn longer memory — but that's hard anyway



RNNs in off-policy methods

- Problem: RNN states in replay buffer disagree with current RNN params
- Solution 1: use n-step rollouts

$$Q_\theta(s_t, h_t, a_t) \to r_t + \gamma r_{t+1} + \cdots + \gamma^{n-1} r_{t+n-1} + \gamma^n \max_{a'} Q_\theta(s_{t+n}, h_{t+n}, a')$$

- Solution 2: "burn in" h_t from even earlier stored steps
- In practice: RNNs rarely used
  - Stacking k frames every step $(o_{t-k+1}, \ldots, o_t)$ may help with short-term memory

Deep RL as partial observability

- Memory-based policies fail us in Deep RL, where we need them most:
  - Deep RL is inherently partially observable
- Consider what deeper layers get as input:
  - High-level / action-driven state features are not Markov!
- Memory management is a huge open problem in Deep RL
  - Actually, in other areas of ML too: NLP, time-series analysis, video processing, ...

Recap and further considerations

- Let policies depend on observable history through memory
- Memory update: Bayesian, approximate, or learned
  - Learning to update memory is one of the biggest open problems in all of ML
- Let policy be stochastic
  - Should memory be stochastic? interesting research question...
- Let policies be non-stationary if possible, otherwise learning may be unstable
  - Time-dependent policies for finite-horizon tasks
  - Periodic policies for periodic tasks