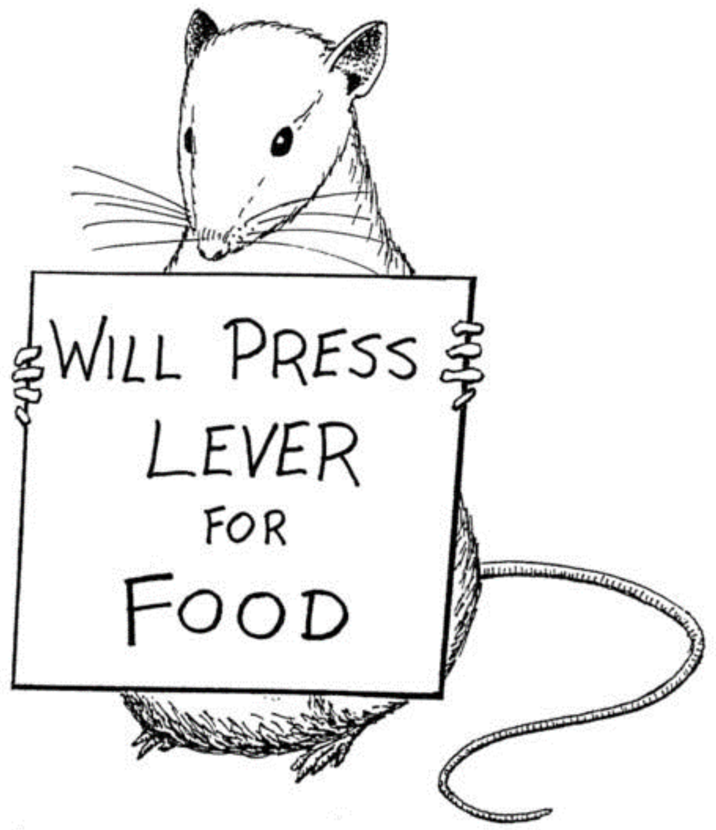
UCI University of California, Irvine

CS 277: Control and Reinforcement Learning

Winter 2024

Lecture 16: Structured Control

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



Logistics

participation

- If you participated in class
  - I appreciate it!
  - Get a ducky
  - Make sure I have your name
- Still time to participate in class / forum
- Don't forget course evaluations

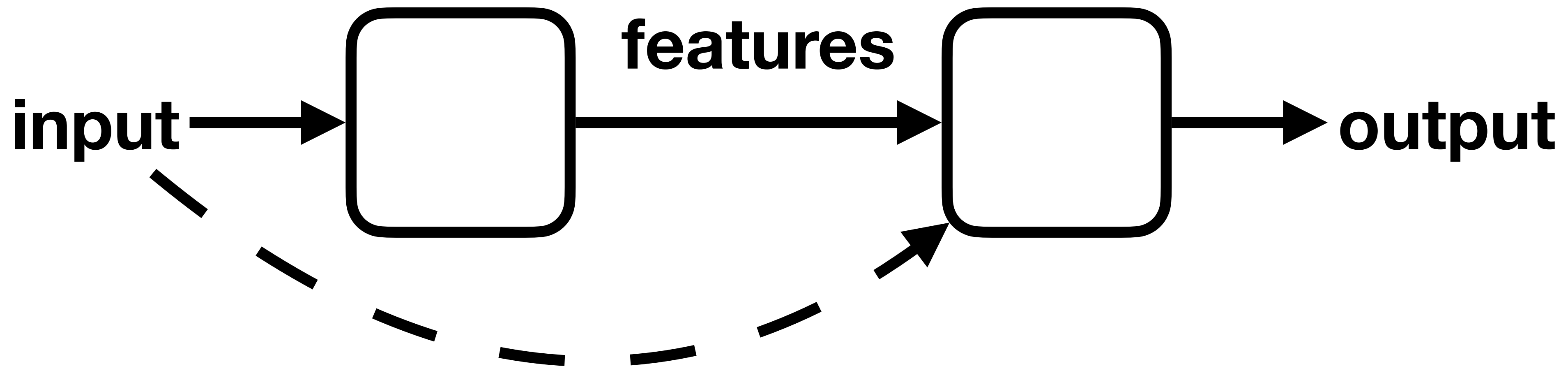
Today's lecture

Abstractions

Hierarchical planning

HRL methods

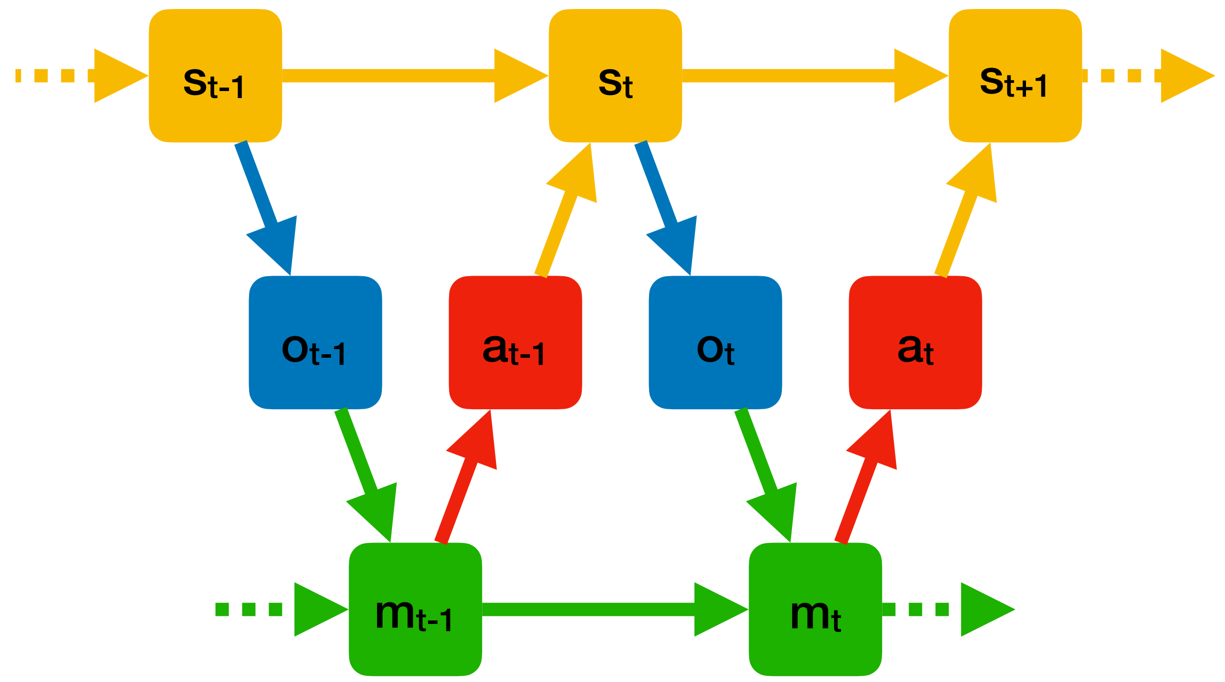
Abstractions in learning



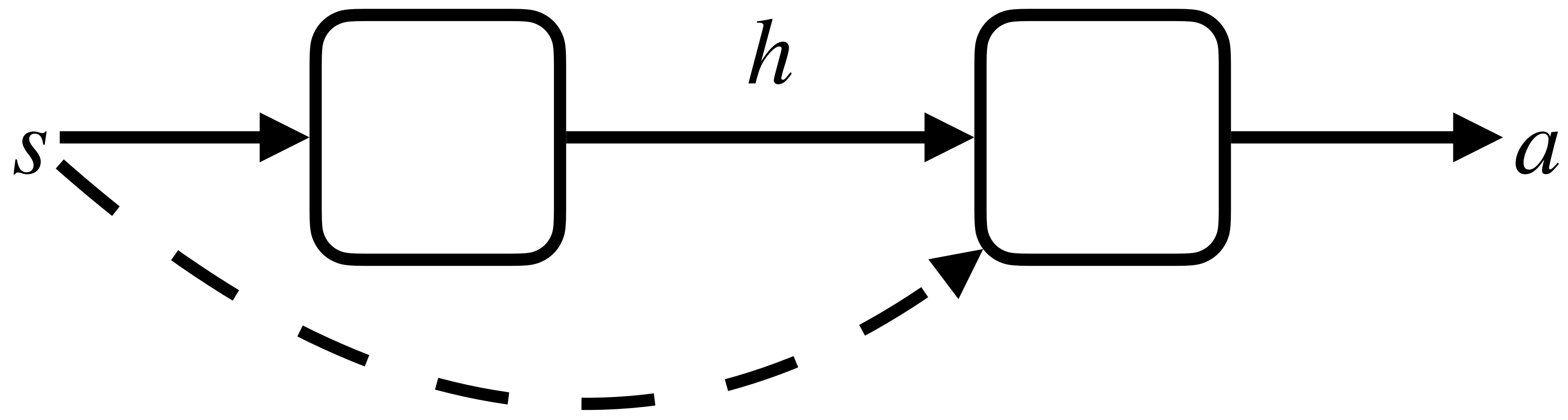
- Abstraction = succinct representation
  - Captures high-level features, ignores low-level
  - Can be programmed or learned
  - Can improve sample efficiency, generalization, transfer
- Input abstraction (in RL: state abstraction)
  - Allow downstream processing to ignore irrelevant input variation
- Output abstraction (in RL: action abstraction)
  - Allow upstream processing to ignore extraneous output details

Abstractions in sequential decision making

- Spatial abstraction: each decision has state / action abstraction
  - Easier to decide based on high-level state features (e.g. objects, not pixels)
  - Easier to make big decisions first, fill in the details later
- Temporal abstraction: abstractions can be remembered
  - No need to identify objects from scratch in every frame
    - High-level features can ignore fast-changing, short-term aspects
  - No need to make the big decisions again in every step
    - Focus on long-term planning, shorten the effective horizon

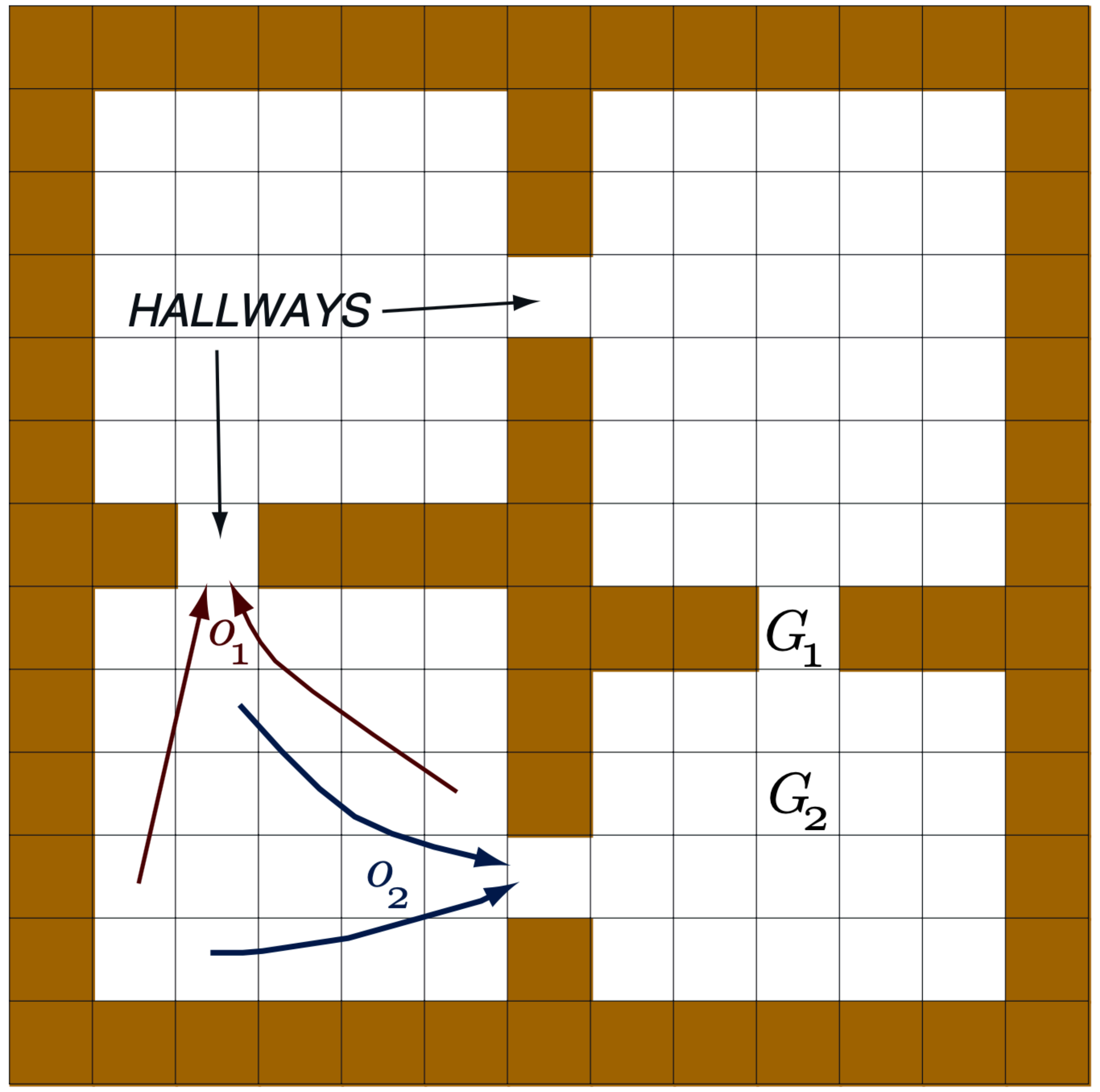


Options framework



- Option = "skill" = persistant action abstraction
  - High-level policy = select the active option $h \in \mathcal{H}$
  - Low-level option = "fills in the details", select action $\pi_h(a \mid s)$ every step
- When to switch the active option h?
  - Idea: option has some subgoal = postcondition it tries to satisfy
  - Option can detect when the subgoal is reached (or failed to be reached)
    - As part of deciding what action to take otherwise
  - $\Rightarrow$ the option terminates $\Rightarrow$ the high-level policy selects new option

Four-room example



4 stochastic primitive actions

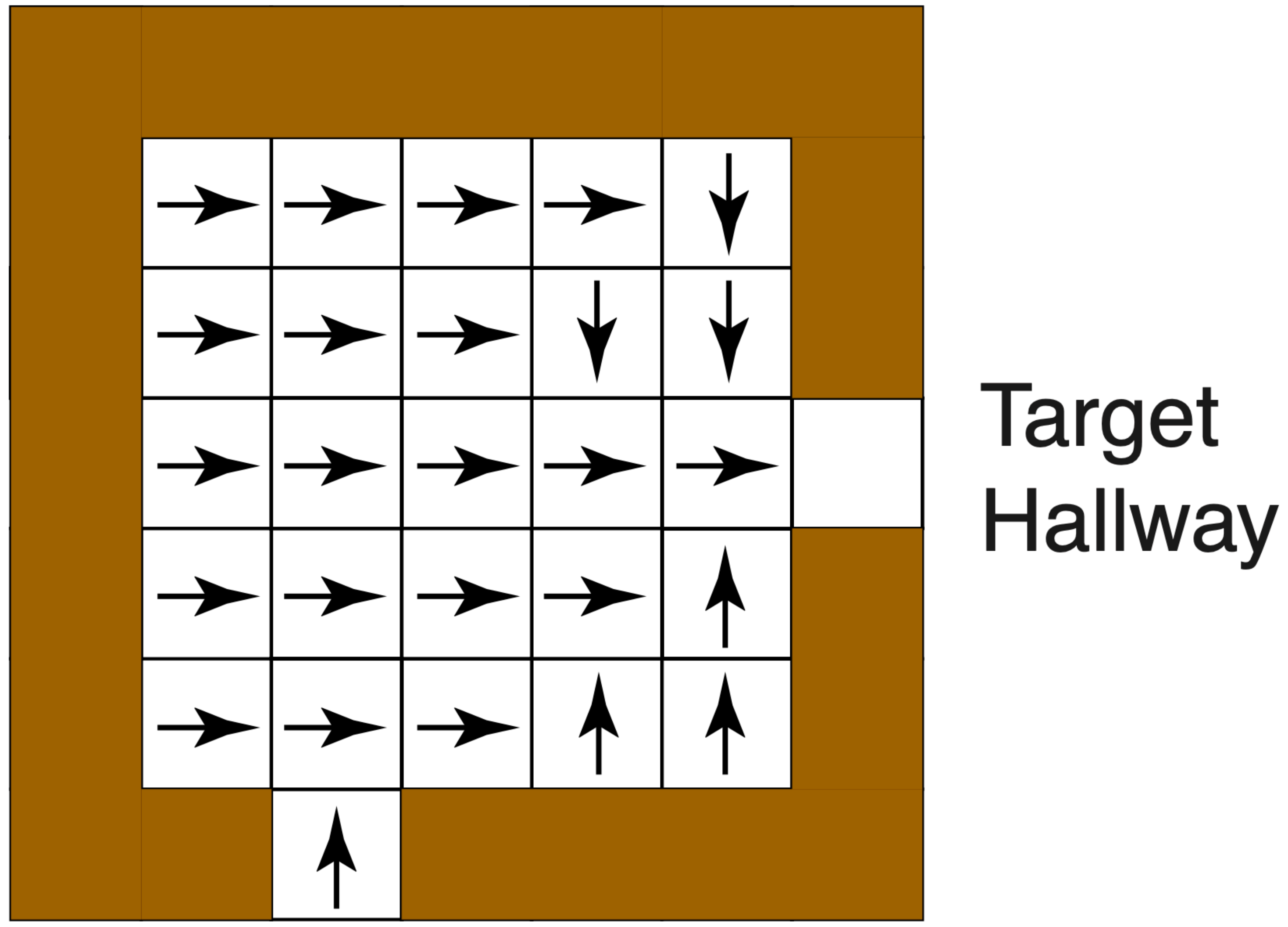
up
left
right
down

Fail 33% of the time

8 multi-step options

(to each room's 2 hallways)

one of the 8 options:



Options framework: definition

- Option = tuple $\langle I_h, \pi_h, \beta_h \rangle$
  - The option can only be called in its initiation set $s \in I_h$
  - It then takes actions according to policy $\pi_h(a \mid s)$
  - After each step, the policy terminates with probability $\beta_h(s)$

termination action

- Equivalently, define policy over extended action set $\pi_h : S \to \Delta(A \cup \{ \perp \})$
- Initiation set can be folded into option-selection meta-policy $\pi_H : S \to \Delta(\mathscr{H})$
- Together, π_H and $\{\pi_h\}_{h \in \mathscr{H}}$ form the agent policy

Today's lecture

Abstractions

Hierarchical planning

HRL methods

Planning with options

- Given a set of options, Bellman equation for the meta-policy:

$$V_H(s) = \max_{h \in \mathcal{H}} r_h(s) + \mathbb{E}_{(s'|s)\sim p_h}[V_H(s')]$$

until it terminates

  - Option meta-reward: $r_h(s) = \mathbb{E}_{\xi \sim p_h}\left[\sum_{\Delta t=0}^{T-1} \gamma^{\Delta t} r(s_{t+\Delta t}, a_{t+\Delta t}) \,\middle|\, s_t = s, a_T = \perp \right]$

rewards during option's run

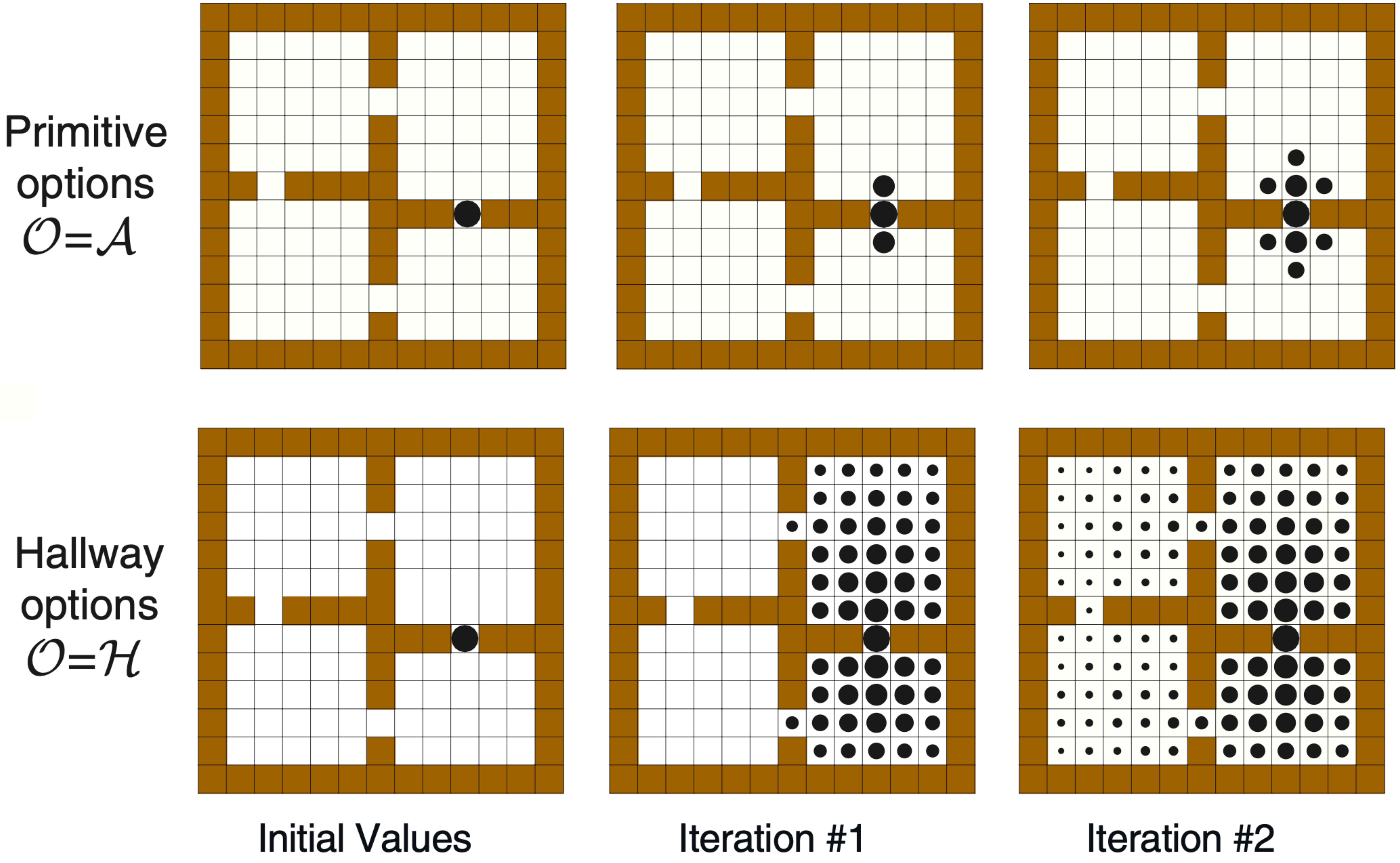
  - Option transition distribution: $p_h(s'|s) = \mathbb{E}_{\xi \sim p_h}[1_{[s_T = s']}\gamma^{T-t} \,|\, s_t = s, a_T = \perp\,]$

variable amount of discounting

- Special case of base actions = option says: take one action and terminate

$$r_a(s) = r(s, a) \qquad p_a(s'|s) = \gamma p(s'|s, a)$$

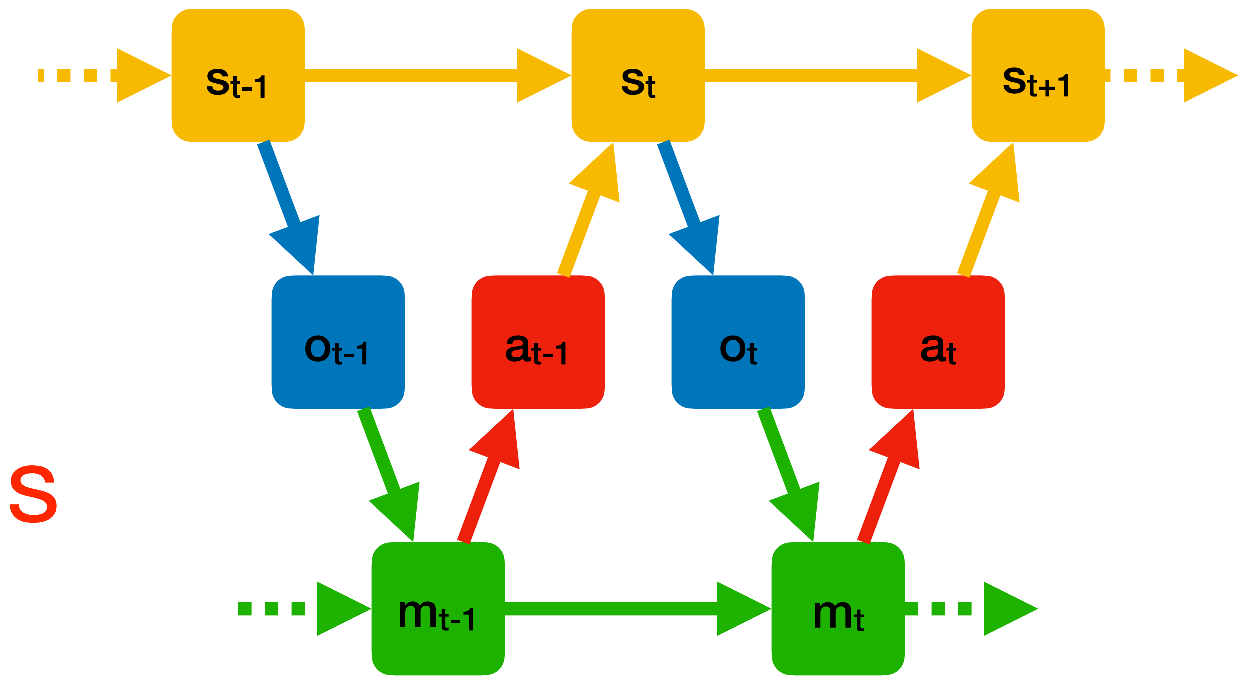
Planning: four-room example



- Options allow fast value backup
- Transfer to other tasks in same domain

Memory structure of options agent

- Options are a pre-commitment, thus an uncontrolled part of the state
- Option terminate after variable time: Semi-Markov Decision Process (SMDP)
- Can be viewed as structured memory
  - The option index is committed to memory
    - although it's not about past observations, it's about future actions
  - Memory remains unchanged until option termination
  - → memory is interval-wise constant



Planning within options

non-terminating action $a \neq \perp$

can terminate

$$Q_h(s,a) = r(s,a) + \gamma \mathbb{E}_{(s'|s,a)\sim p}[V_h^{\text{term?}}(s')] \qquad V_h^{\text{term?}}(s) = \max_a Q_h(s,a)$$

$$Q_h(s,\perp) = V_H(s) = \max_h V_h^{\text{nonterm}}(s) \qquad V_h^{\text{nonterm}}(s) = \max_{a\neq\perp} Q_h(s,a)$$

new option:
take at least 1 action

- Problem: jointly finding V_H and $\{V_h\}_{h\in\mathscr{H}}$ is under-determined
- High-fitting: some π_h tries to solve entire task, never terminates
  - If π_h is expressive enough, this is guaranteed to happen in many algorithms
- Low-fitting: options terminate immediately, emulating base actions
  - Now meta-policy carries the entire burden

Today's lecture

Abstractions

Hierarchical planning

HRL methods

Option–critic method

- For the critic, define $V_h(s) = \mathbb{E}_{(a|s)\sim\pi_{\theta_h}}[Q_h(s,a)]$, $V_H(s) = \mathbb{E}_{(h|s)\sim\pi_\psi}[V_h(s)]$

- Then for on-policy experience (s, h, a, r, s') define the losses:
  - Critic loss: $L_Q = (r + \gamma((1 - \beta_h(s'))V_h(s') + \beta_h(s') \max_{h'} V_{h'}(s')) - Q_h(s,a))^2$
  - For actor behavior π_{θ_h}: $\nabla_{\theta_h} L_\pi = - Q_h(s,a) \nabla_{\theta_h} \log \pi_{\theta_h}(a \,|\, s)$
  - For actor termination β_{ϕ_h}: $\nabla_{\phi_h} L_\beta = (V_h(s) - V_H(s)) \nabla_{\phi_h} \beta_{\phi_h}(s)$
  - For actor high level π_ψ: $\nabla_\psi L_H = - V_h(s) \nabla_\psi \pi_\psi(h \,|\, s)$

- Suffers badly from high- and low-fitting

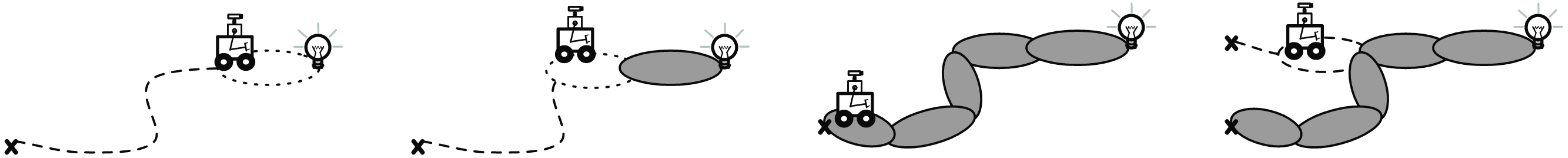
Subgoals

- Can we discover natural points to separate the high and low levels?
- Insight: the high level defines the termination value for the low level

$$Q_h(s, \perp) = V_H(s)$$

  - Brings value back from a far future horizon to the low level's horizon
- We can think of the terminal-state value function as a subgoal
  - Defines in which states the option should try to terminate
  - E.g. doorways in the four-room domain
- Can we discover good subgoals?

Learning skill trees



(a) (b) (c) (d)

Algorithm Skill Tree

$S \leftarrow \{\text{goal}\}$

repeat

$(\pi, \beta) \leftarrow$ option for subgoal $V_H(s) = r \cdot \mathbb{1}_{[s \in S]}$

$\mathcal{I} \leftarrow$ initiation set from which (π, β) reaches subgoal

$S \leftarrow S \cup \mathcal{I}$

until $s_0 \in S$

Spectral methods

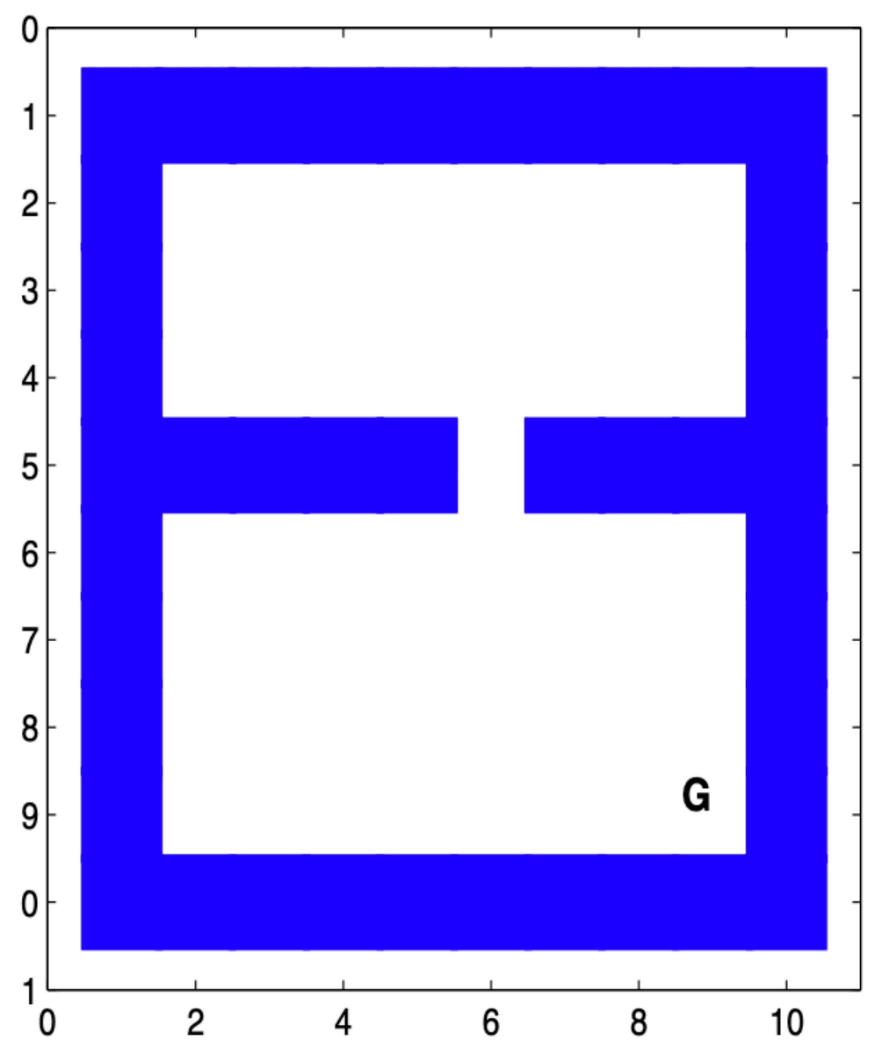
- Consider a state clustering into “good” and “bad” states
- The clustering indicator is a subgoal
- Let's use spectral clustering on the visitation graph

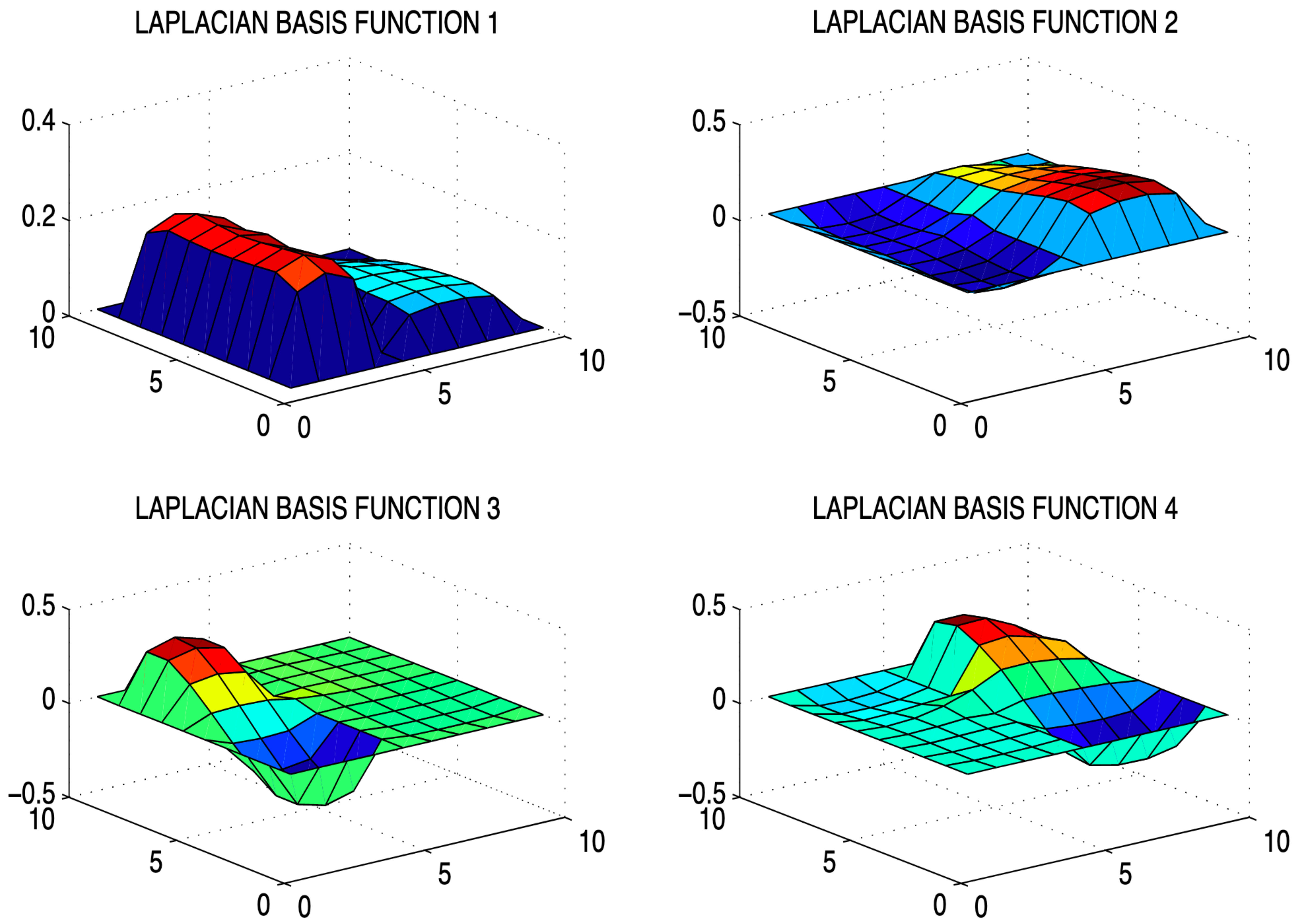
$$W_{s,s'} = 1_{[s' \text{ is reachable from } s]}$$

$$D(s) = \sum_{s'} W_{s,s'} = \text{out-degree of } s$$

- Normalized graph Laplacian $L = D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}$ finds connectivity
  - Related to random walk $D^{-\frac{1}{2}}(I - L)D^{\frac{1}{2}} = D^{-1}W = \{p_0(s' | s)\}_{s,s'}$
  - Eigenvectors of least positive eigenvectors find nearly stationary state clusters

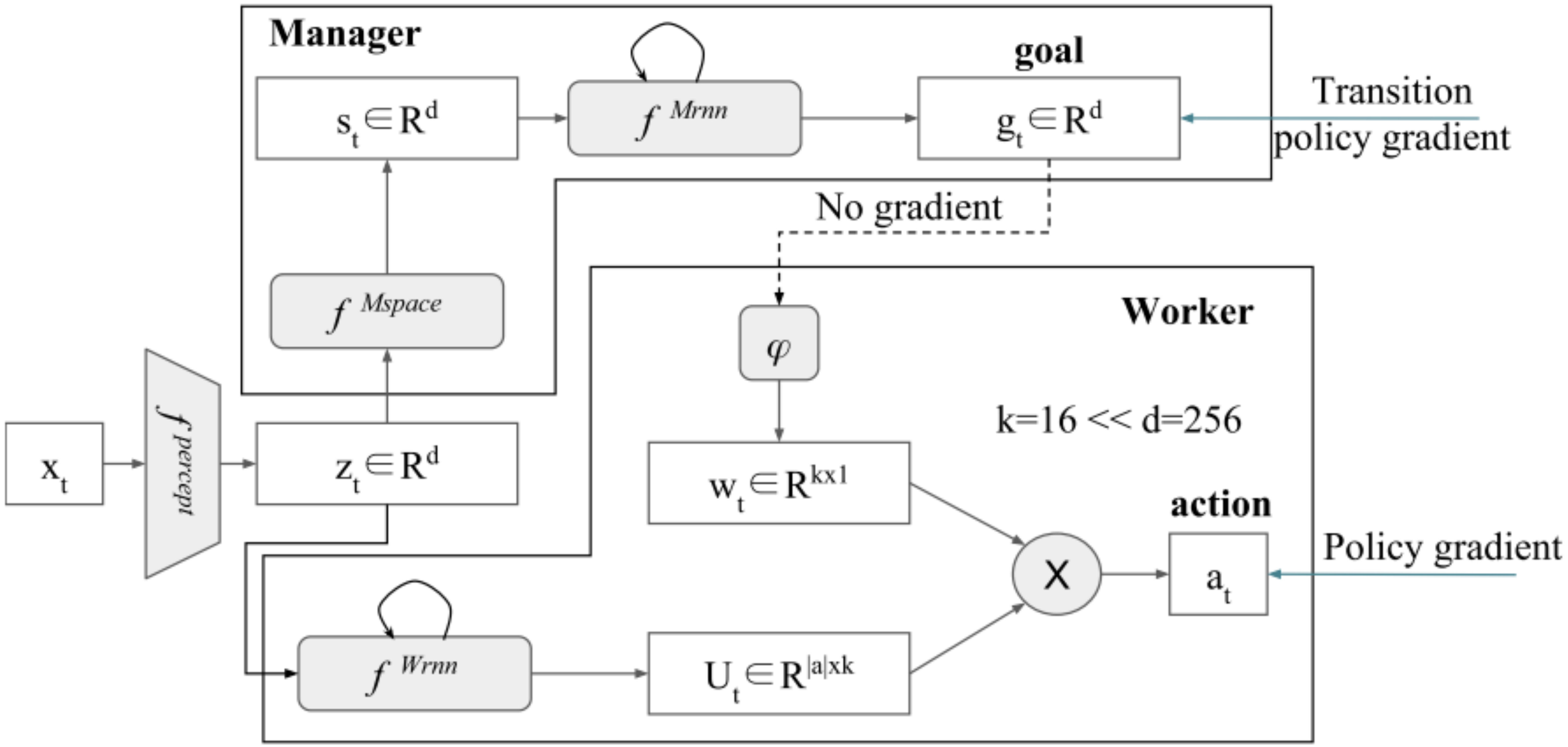
Spectral subgoal discovery





- Roll out random walk
- Find eigenvectors of graph Laplacian with small eigenvalues
- Learn options for these subgoals

Feudal networks



- Manager sets goals in learned latent space, every H steps
- Worker uses the goals as hints for learning long-term valuable behavior

Recap

- Abstractions: succinct representations; better data efficiency, generalization
- Hierarchical policy is foremost a memory structure
- Structure can be programmed, demonstrated, or discovered
- Subgoals can be represented by terminal-state value functions
- Many more hierarchical frameworks:
  - HAMQ, MAXQ, HEXQ, HDQN, QRM, HVIL, ...
- Many more opportunities for structure in control
  - Multi-task learning
  - Structured exploration