University of California, Irvine

CS 295: Optimal Control and Reinforcement Learning

Winter 2020

Lecture 16: Structured Control

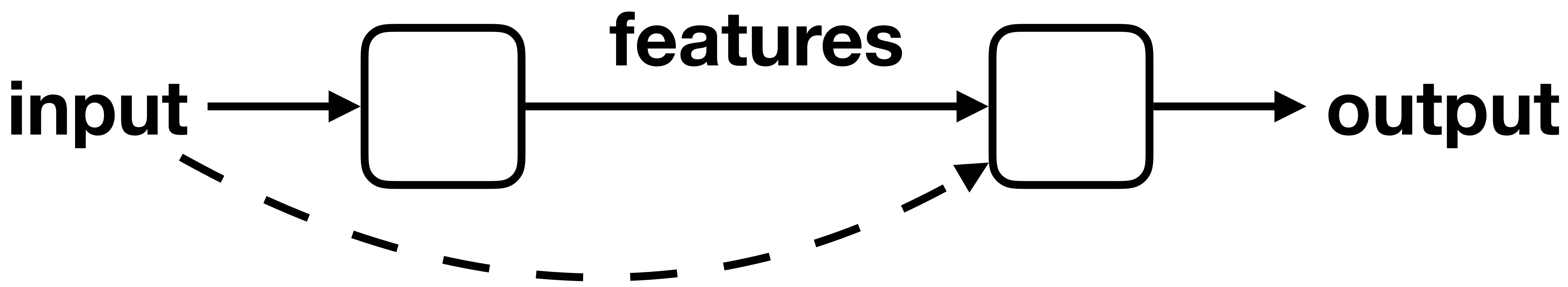
Roy Fox
Department of Computer Science
Bren School of Information and Computer Sciences
University of California, Irvine

Today's lecture

- Abstractions in ML and RL
- Options framework
- Planning with options, within options
- Option discovery
- Multi-level hierarchies
- Feudal Networks

Abstractions in learning

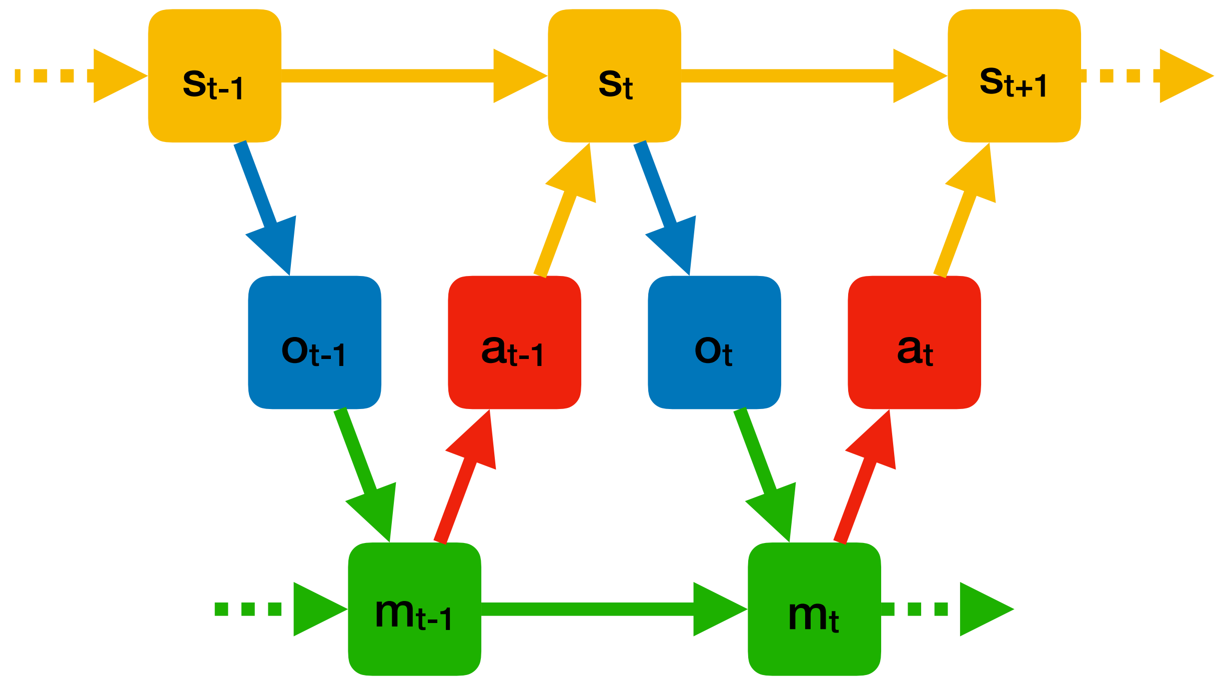
- Input abstraction
  - Allow downstream processing to ignore irrelevant input variation
  - In RL: state abstraction



- Output abstraction
  - Allow upstream processing to ignore extraneous output details
  - In RL: action abstraction
- Can be programmed or learned
- Can improve sample efficiency, generalization, transfer

Abstractions in sequential decision making

- Each decision can have state / action abstraction
  - Spatial abstraction
- Better: abstractions can be remembered
- Even better: abstraction dynamics can have a longer time-scale
  - Temporal abstraction
  - The abstract features can ignore fast-changing, short-term aspects
  - Focus on long-term planning, shorten the effective horizon



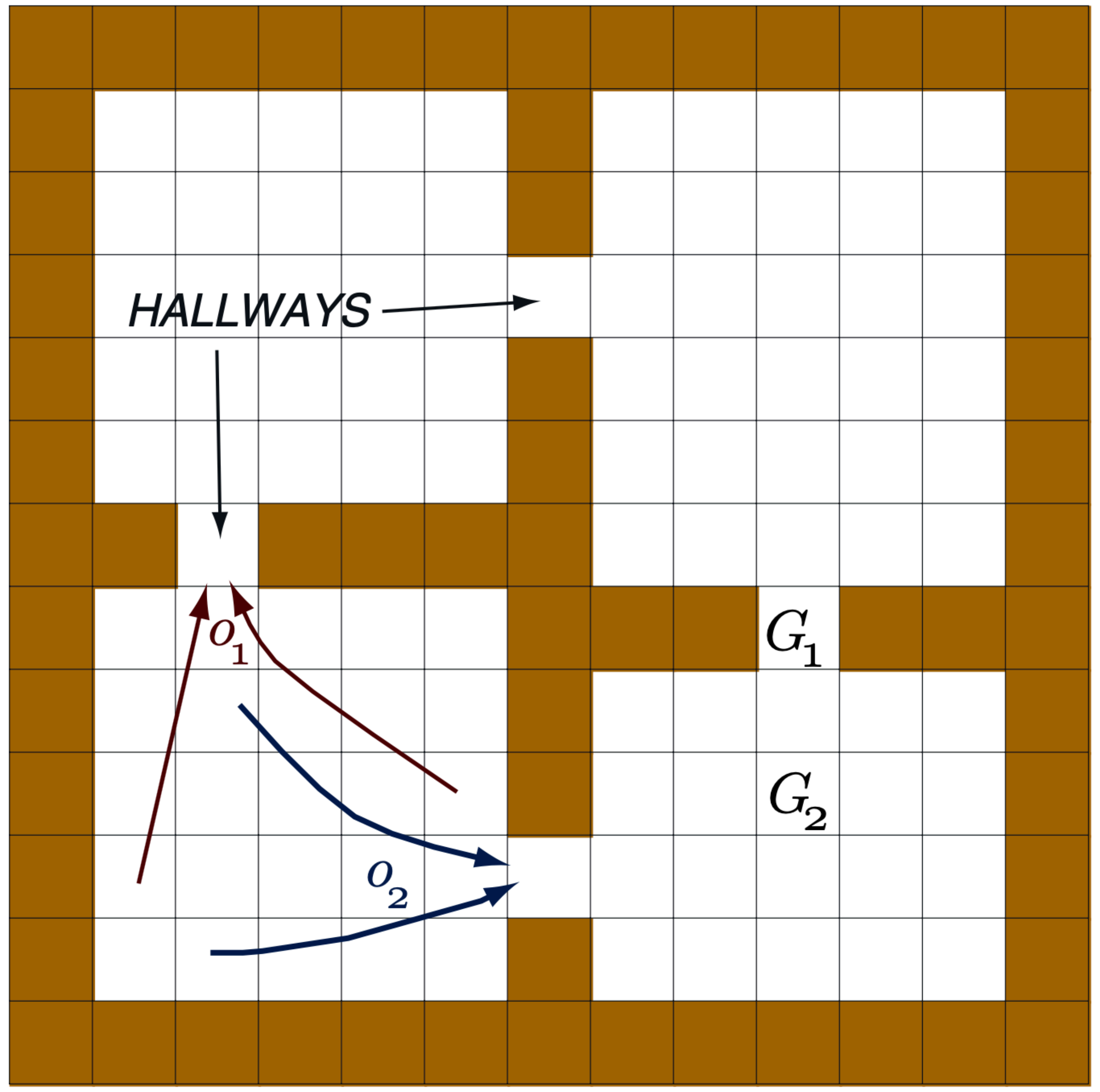
Options framework

- Option = persistant action abstraction
- High-level policy selects the active option $h \in \mathcal{H}$
- The active option "fills in the details" by selecting concrete actions every step

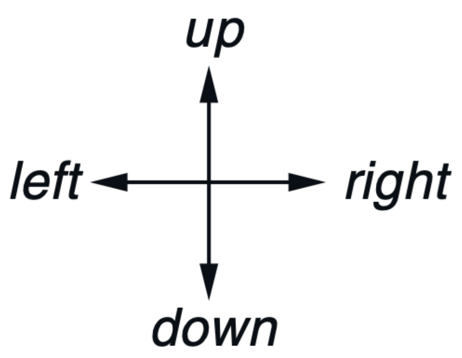
$$\pi_h(a|s)$$

- When to switch the active option?
  - The option already attends to state details, and "knows" its subgoal
  - So let the option detect when it achieved the subgoal (or failed to do so)
  - Then the option will terminate; the high-level policy will select new option

Four-room example



4 stochastic primitive actions

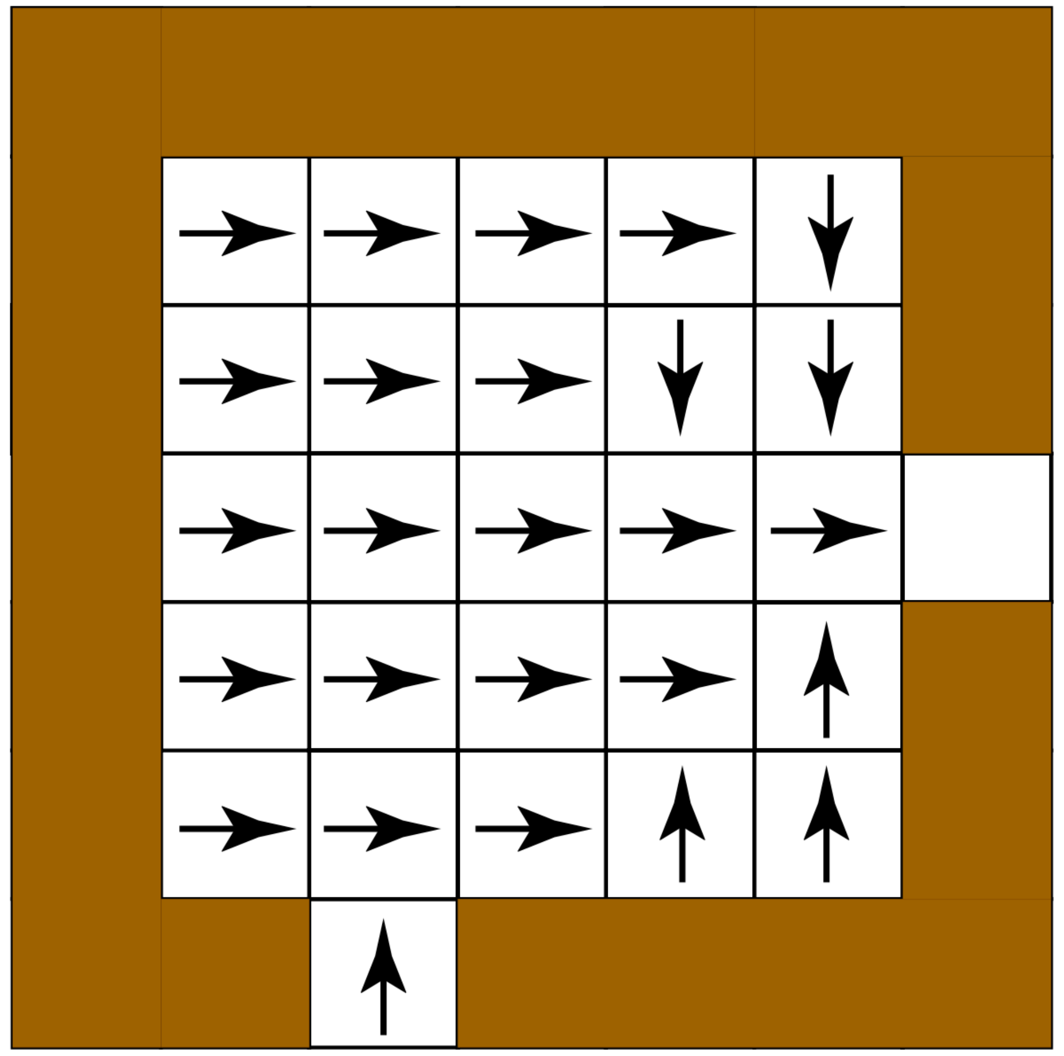


Fail 33% of the time

8 multi-step options

(to each room's 2 hallways)

one of the 8 options:



Target Hallway

Options framework: definition

- Option: tuple $\langle \mathcal{I}_h, \pi_h, \beta_h \rangle$
  - The option can only be called in its initiation set $s \in \mathcal{I}_h$
  - It then takes actions according to policy $\pi_h(a|s)$
  - After each step, the policy terminates with probability $\beta_h(s)$
- Equivalently, define policy over extended action set $\pi_h : \mathcal{S} \to \Delta(\mathcal{A} \cup \{\bot\})$
- Initiation set can be folded into option-selection meta-policy $\pi_\bot : \mathcal{S} \to \Delta(\mathcal{H})$
- Together, $\pi_\bot$ and $\{\pi_h\}_{h \in \mathcal{H}}$ form the agent policy

Planning with options

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

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

  - such that with $a_T = \perp$ at the time of option termination time

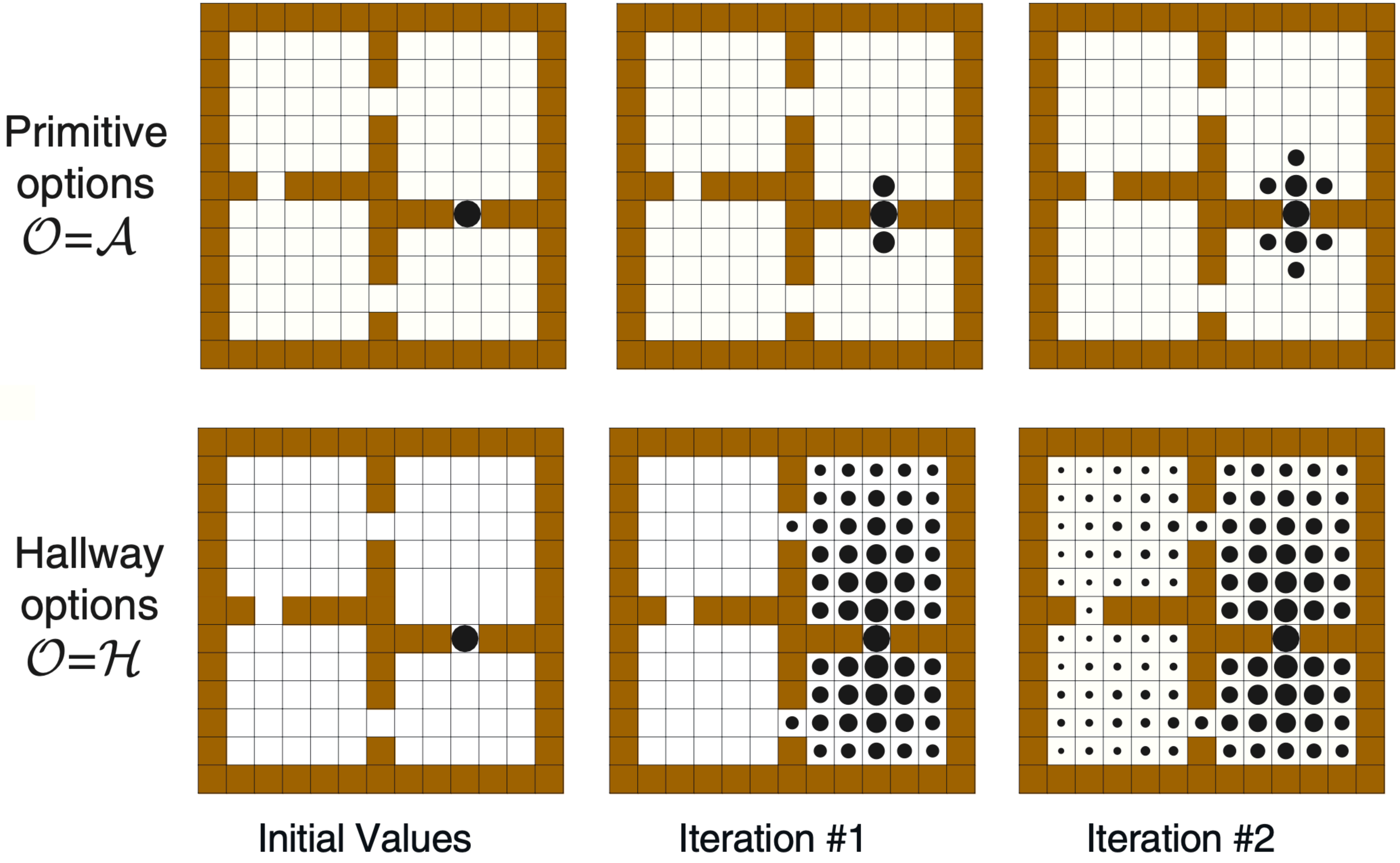
$$r_h(s_t) = \mathbb{E}\left[\sum_{t'=t}^{T-1} \gamma^{t'-t} r(s_{t'}, a_{t'}) | s_t\right]$$

$$p_h(s'|s_t) = \mathbb{E}[\mathbb{1}_{[s_T = s']} \gamma^{T-t} | s_t]$$

- Special case of primitive actions:

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

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 <u>uncontrolled</u> 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

$$V_h(s) = \max_a Q_h(s,a)$$ **including or excluding termination?**

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

$$Q_h(s,\perp) = V_\perp(s) = \max_h V_h^{\not\perp}(s)$$

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

Option–critic method

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

- Then

$$\mathcal{L}_Q(s,h,a,r,s') = (r + \gamma((1-\beta_h(s'))V_h(s') + \beta_h(s')\max_{h'} V_{h'}(s') - Q_h(s,a))^2$$

$$\mathcal{L}_\pi(s,h,a) = -\nabla_{\theta_h} \log \pi_{\theta_h}(a|s) Q_h(s,a)$$

$$\mathcal{L}_\beta(s,h) = \nabla_{\phi_h} \beta_{\phi_h}(s)(V_h(s) - \max_{h'} V_{h'}(s))$$

- Suffers badly from high- and low-fitting

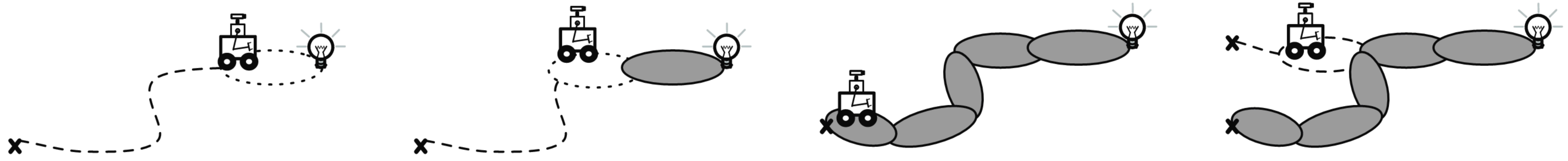
Subgoals

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

$$Q_h(s, \bot) = V_\bot(s)$$

  - 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)

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

repeat

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

$\mathcal{I} \leftarrow$ initiation set, on which (π, β) succeeds reaching 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

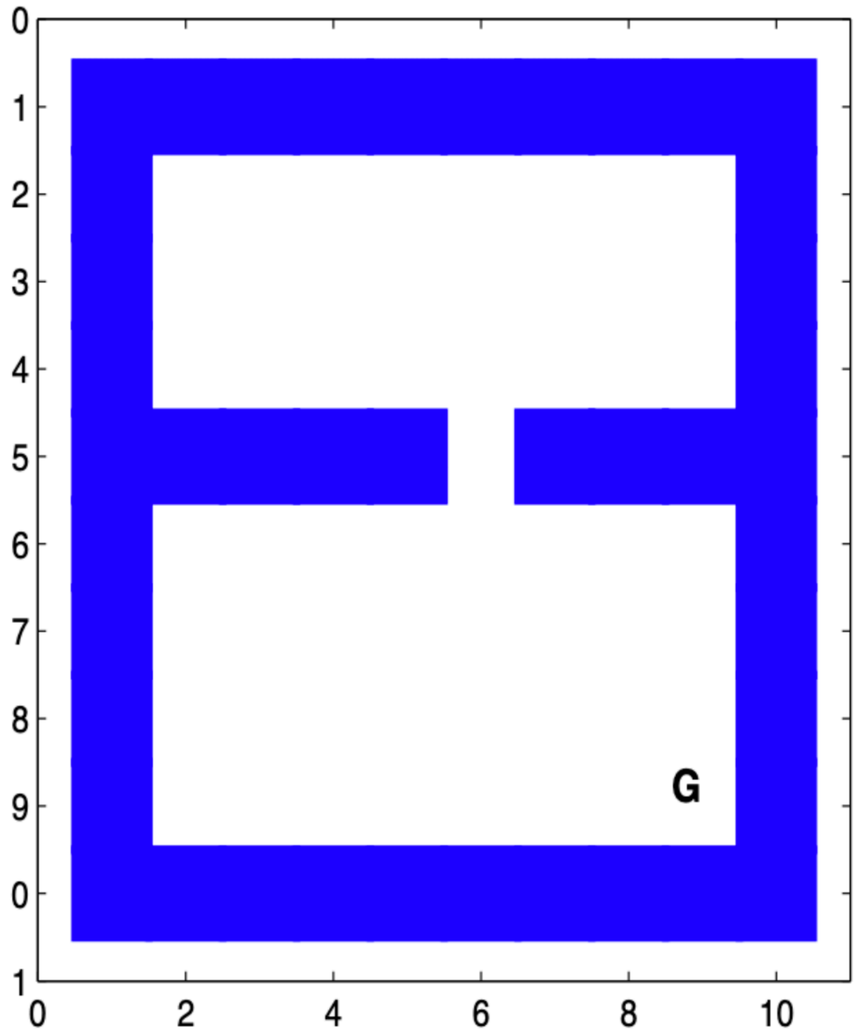
- Let's use spectral clustering on the visitation graph

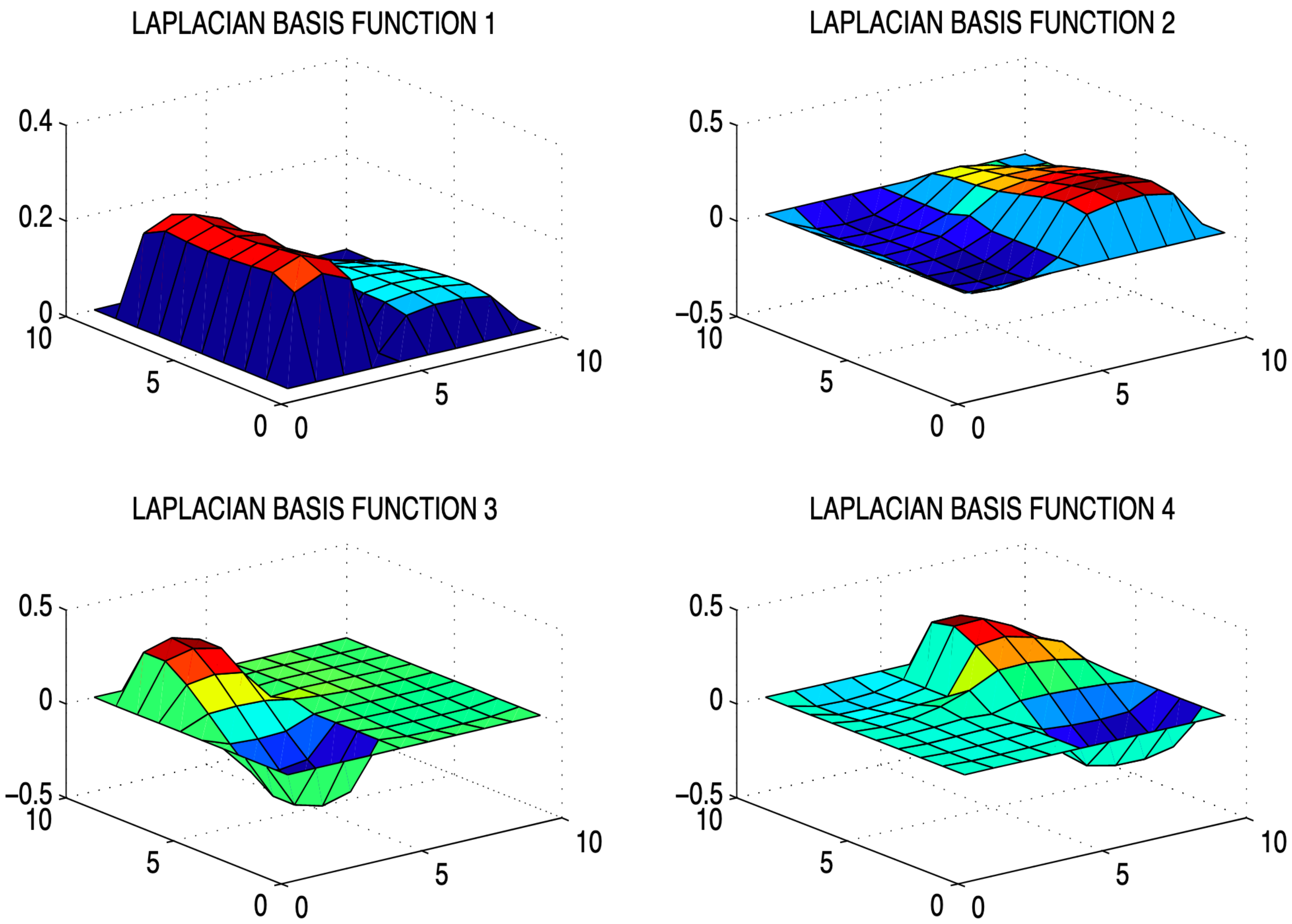
$$W_{s,s'} = \mathbb{1}_{[s' \text{ is reachable from } s]}$$

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

- 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





- Random walk
- Find eigenvectors of graph Laplacian with small eigenvalues
- Learn options for these subgoals

Option inference

- A (hierarchical) policy is a generator

$$p_\theta(h_t, a_t | h_{t-1}, s_t) = ((1 - \beta_{h_{t-1}}(s_t))\,\mathbb{1}_{[h_t = h_{t-1}]} + \beta_{h_{t-1}}(s_t)\pi_\perp(h_t|s_t))\pi_{h_t}(a_t|s_t)$$

- Easy to compute when $\zeta = h_0, h_1, \ldots$ is known; otherwise we can infer

$$\nabla_\theta \log p_\theta(\xi) = \frac{\nabla_\theta p_\theta(\xi)}{p_\theta(\xi)} = \sum_\zeta \frac{p_\theta(\zeta, \xi)}{p_\theta(\xi)} \nabla_\theta \log p_\theta(\zeta, \xi) = \mathbb{E}_{\zeta|\xi \sim p_\theta}[\nabla_\theta \log p_\theta(\zeta, \xi)]$$

$$= \sum_t \mathbb{E}_{h_{t-1}, h_t | \xi \sim p_\theta}[\nabla_\theta \log p_\theta(h_t, a_t | h_{t-1}, s_t)]$$

- In one-level hierarchy, $p_\theta(h_{t-1}, h_t | \xi)$ can be computed exactly
  - Forward–backward algorithm, similar to Baum–Welch in HMMs

Expectation–Gradient

- E-step: compute posterior over latent options
- G-step: compute policy gradient

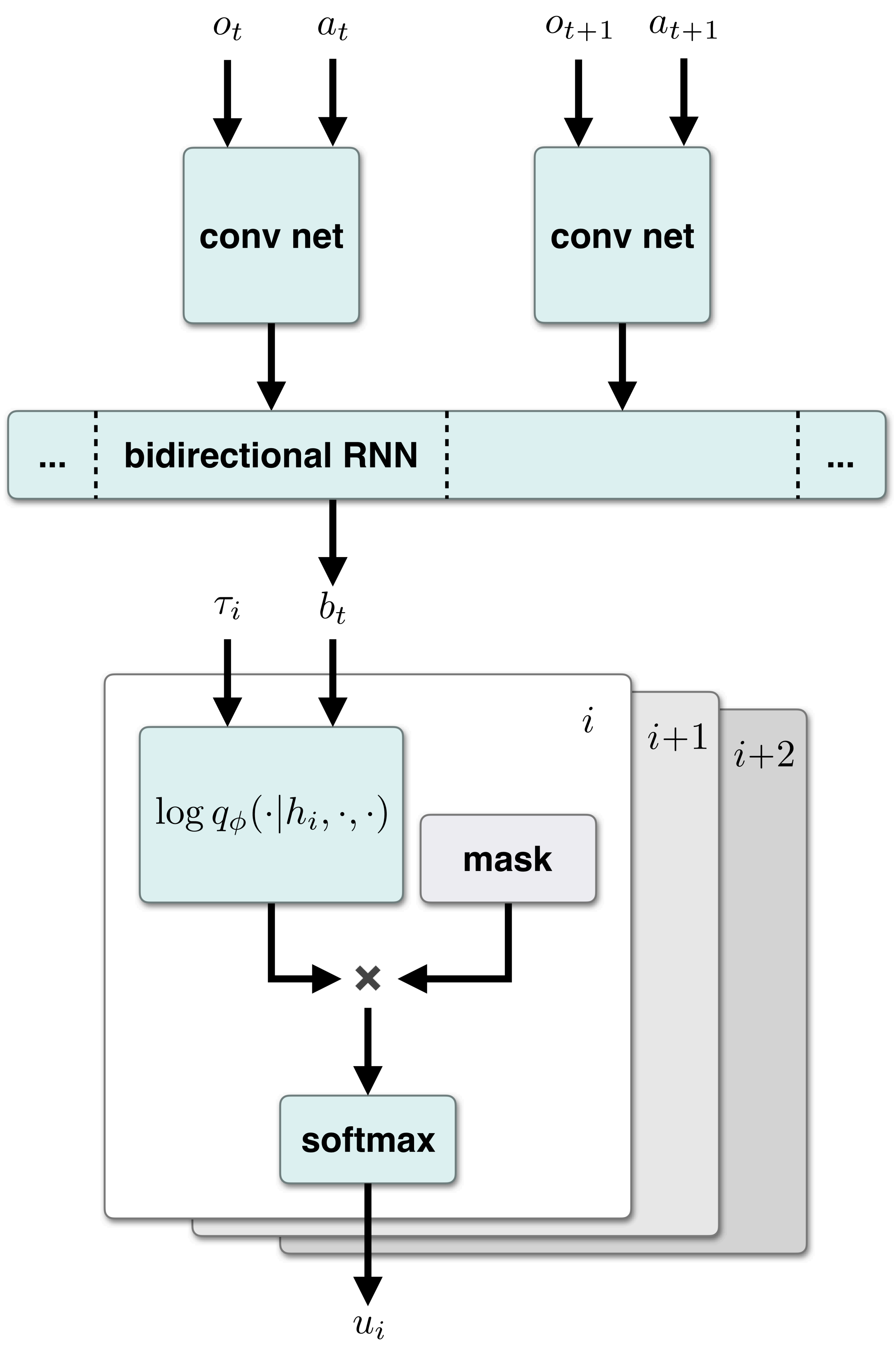
- Effectively, we jointly
  - segment (successful) trajectories into homogenous control intervals
  - cluster segments with similar behavior = options
  - take a policy gradient step for the policy of each cluster

Multi-level hierarchies

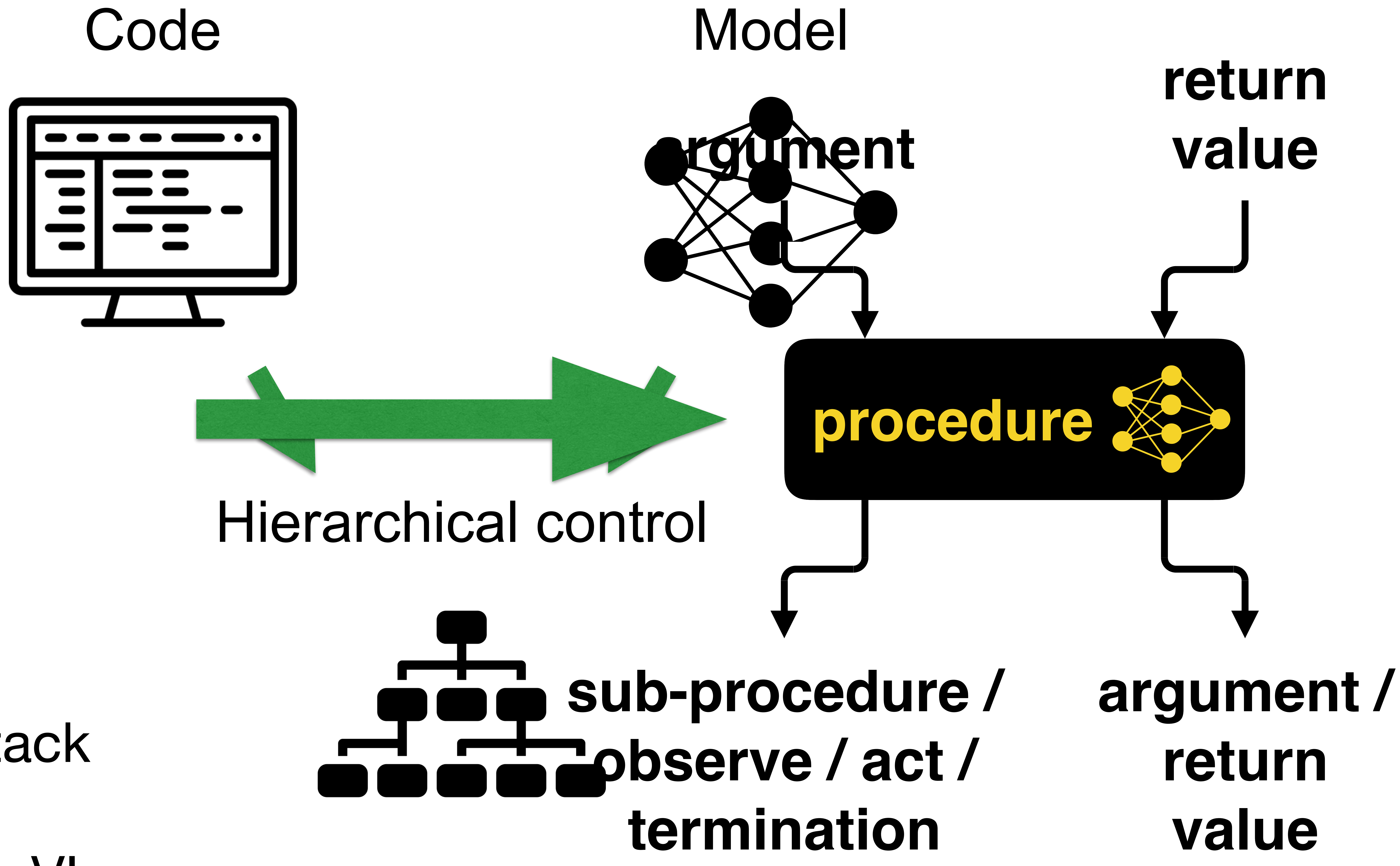
- Multi-level hierarchies useful for same reasons as one-level
  - Many algorithms don't easily extend
- Exact inference no longer possible
  - use variational inference

$$\log p_\theta(\xi) \geqslant \mathbb{E}_{\zeta|\xi\sim q_\phi}\left[\log \frac{p_\theta(\zeta,\xi)}{q_\phi(\zeta|\xi)}\right]$$

- Proposal distribution in training time can depend on past and future
  - Better data efficiency

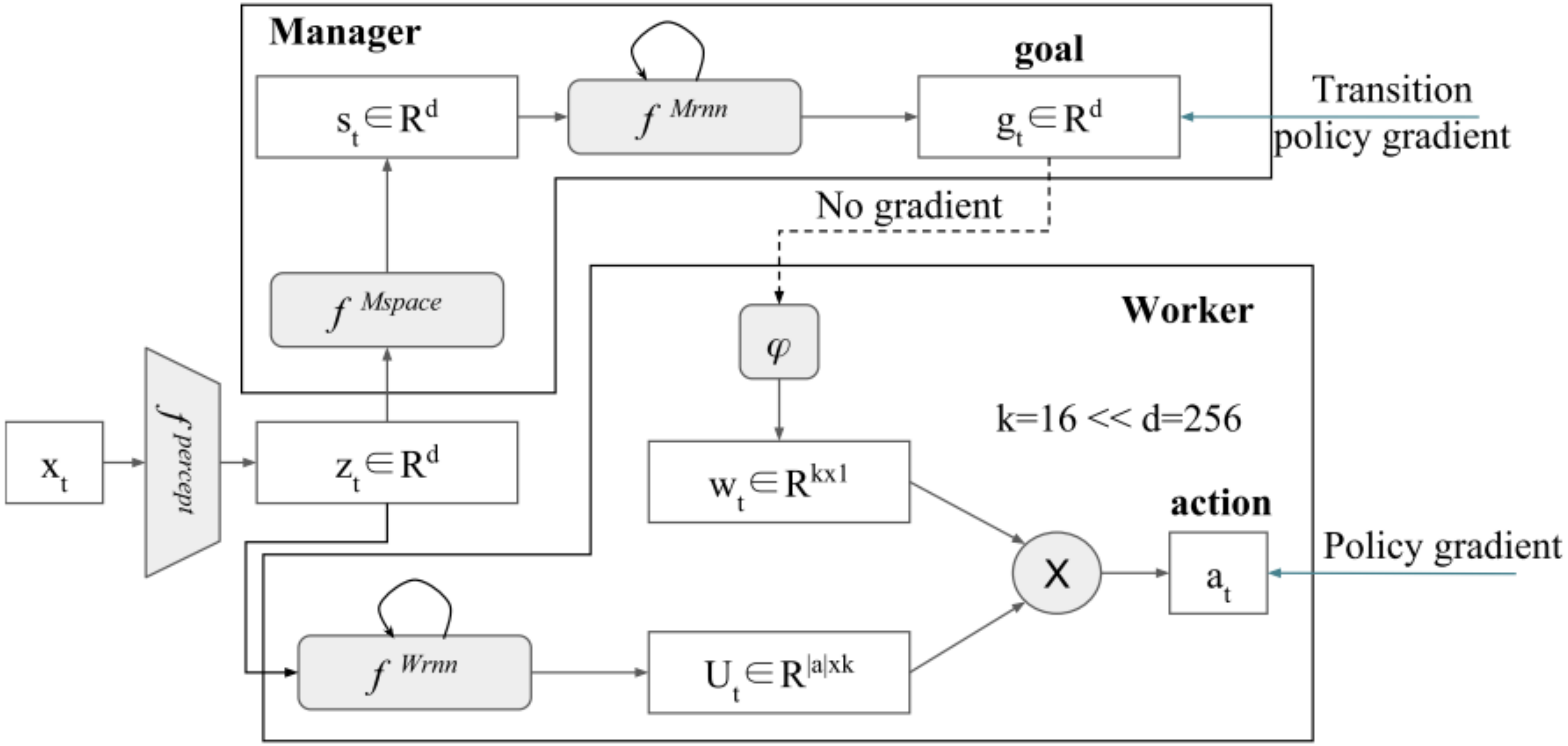


Parametrized Hierarchical Procedures (PHPs)



- Memory is a call-stack
- Can be trained with VI

Feudal networks



- Manager sets goals in learned latent space, every H steps
- Worker uses the goals as hints for 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, ...
- Many more opportunities for structure in control
  - Multi-task learning
  - Structured exploration