Modular iPOMDPs for HRL

Momchil

December 9, 2017

1 Background

1.1 Reinforcement Learning and Markov Decision Processes

In reinforcement learning (RL), tasks are traditionally represented as Markov decision processes (MDPs) [1] and we will use those two terms interchangably. We define an MDP as a tuple $(\mathcal{S}, \mathcal{A}, T, R, I, \mathcal{F})$ where:

- $\mathcal{S}$ is a set of states that the agent can occupy,

- $\mathcal{A}$ is a set of actions that the agent can perform in those states,

- $T(\cdot|s, a)$ is a distribution of next states, such that $T(s'|s, a)$ is the probability that the agent ends up in state $s' \in \mathcal{S}$ after executing action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$,

- $R(s, a)$ is the amount of reward an agent obtains after executing action a in state s,

- $I(\cdot)$ is a distribution of initial states in which the agent may start the task,

- $\mathcal{F}$ is a set of terminal states in which the task is completed.

The agent begins the task in state $s_0 \sim I$. At each time step t, the agent performs action a_t that takes it from state s_t to state $s_{t+1} \sim T(\cdot|s_t, a_t)$ and receives reward $r_t \sim R(s_t, a_t)$. The agent is said to follow a policy π if it chooses its actions according to $a_t \sim \pi(\cdot|s_t)$.

For a fixed policy π, the state-value function $V_\pi(s)$ represents the total expected discounted reward for each state s. It is the sum of all future rewards that the agent will receive, on average, if it starts in state s and follows π:

$$V_\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \middle| s_0 = s \right],$$

where γ is the discount rate which determines the present value of future rewards. The goal of the agent is to select an optimal policy π^* that maximizes $V_{\pi^*}(s)$. The optimal value function $V_{\pi^*}(s)$ satisfies the Bellman optimality equation [2]:

$$V_{\pi^*}(s) = \max_a \left\{ R(s,a) + \gamma \sum_{s'} V_{\pi^*}(s') \; T(s'|s,a) \right\}$$

The Bellman equation serves as a basis of a number of algorithms for solving optimal control problems.

1.1.1 RL in the Brain

RL algorithms often fall into one of two broad categories: model-free algorithms, in which the agent learns V_π directly; and model-based algorithms, in which the agent learns R and T to compute V_π. Computational cognitive neuroscience has borrowed this dichotomy as a formal description of the distinction between habitual and goal-directed behaviors [3]. Model-free accounts of habitual behaviors posit that animals learn automatic responses to stimuli if they tend to lead to rewarding outcomes. Model-based accounts of goal-directed behaviors posit that animals learn the statistics of the environment independently of the rewards.

Model-free accounts are less computationally intensive, however they require many training examples and have to be re-trained whenever the reward distribution changes. This is consistent with empirical observations that overtrained animals exhibit fast reaction times and rigid behavioral patterns that tend to persist even reward contingencies change. Model-free accounts are further bolstered by the hallmark discovery that the phasic firing of midbrain dopaminergic neurons closely tracks the model-free reward prediction error (RPE) signal.

Model-based accounts, on the other hand, predict that agents will flexibly adapt to changes in reward distributions, at the cost of slower reaction times due to the greater computational complexity. Model-based accounts explain phenomena such as latent learning and accord with the intuition that people don't have to learn a new policy, say, every time they go shopping for different groceries.

1.2 Hierarchical Reinforcement Learning and Semi-Markov Decision Processes

A long-standing challenge for traditional RL is the combinatorial explosion that occurs when planning and learning take place over long time horizons. This challenge been addressed by hierarchical reinforcement learning (HRL), which breaks down the problem into sub-problems at multiple levels of abstraction.

One strand of HRL extends the agent's action repertoire to include *options* [4] – temporally-extended sequences of actions, sometimes referred to as subroutines, partial policies, or macro-actions. Each option consists of a sequence of actions that are executed

as a single behavioral unit. Actions in the original MDP are referred to as *primitive actions*, in order to distinguish them from options.

Formally, including options turns the underlying MDP into a discrete-time semi-Markov decision process (SMDP), which allows transitions between states to take variable amounts of time. Thus we augment the original MDP with the following definitions:

- $\mathcal{O}$ is a set of options,

- π_o is a policy for each option $o \in \mathcal{O}$, such that while o is being executed, actions are selected according to $a \sim \pi_o(\cdot|s)$,

- $\mathcal{F}_o$ is a set of subgoal states for each option o, such that π_o terminates when reaching any one of those states,

- $T(\cdot, \cdot|s, o)$ is a joint distribution of terminal states and step counts for each option o, such that $T(s', t|s, o)$ is the probability that the agent ends up in subgoal state $s' \in \mathcal{F}_o$ exactly t time steps after executing option $o \in \mathcal{O}$ in state $s \in \mathcal{S}$,

- $R(s, o)$ is the total expected discounted reward for starting in state s and executing option o.

The rewards can be computed as:

$$
R(s, o) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \middle| s_0 = s, a_t \sim \pi_o\right]
$$

It is also convenient to define a discounted distribution of subgoal states [4]:

$$
T_\gamma(s'|s, o) = \sum_{t=0}^{\infty} T(s', t|s, a)\gamma^t
$$

In the SMDP, the agent follows a policy μ defined over options, such that options are chosen according to $o_t \sim \mu(\cdot|s_t)$. The Bellman equation for the optimal options policy μ^* thus becomes:

$$
V_{\mu^*}(s) = \max_o \left\{ R(s, o) + \sum_{s'} V_{\mu^*}(s') \, T_\gamma(s'|s, o) \right\}
$$

An options policy μ in the SMDP uniquely determines a *flat policy* π consisting solely of primitive actions in the original MDP. Thus any solution of the SMDP can be mapped back to a solution in the original MDP.

Introducing options allows agent to perform "jumps" in the state space. When considering an option o, the agent only needs to compute the value of its subgoal states $s' \in \mathcal{F}_o$, without computing the values of intermediate states that will be traversed while executing the option. If the subgoals and the options policies are chosen appropriately, exploration and planning can be performed much more efficiently by allowing the agent reason over short sequences of options, rather than over long sequences of primitive actions.

1.2.1 HRL in the Brain

In addition to its computational benefits, HRL provides an appealing account of habitual behaviors [5] that competes with model-free RL. Model-free RL fails to capture habits which unfold as sequences of actions in response to a single cue, disregarding intervening cues [6]. It also fails to specify how habitual responses can be generalized across multiple task contexts.

Options as defined in HRL map nicely to habitual action sequences and their neural substrates. Research on rodents has implicated striatal and midbrain neurons in the formation and initiation of habitual action sequences, with task-bracketing activity emerging over the course of training in both regions [7, 8]. In these tasks, "bumps" of neural activity emerge at key decision points during the trial (e.g., at the trial start or at maze junctions), while activity is generally low in the remainder of the trial (e.g., while running straight). Similar task-bracketing activity has also been observed in birds and primates, suggesting a conserved neural implementation [7]. The acquisition, initiation and termination of options in HRL may account for these diverse findings.

Additionally, the HRL framework has been directly supported by both behavioral [9] and neural evidence [10]. It is also consistent with our intuition that people can plan over different time scales (e.g., go shopping, pick up the kids, etc.) without being bogged down by the details (e.g., get up, turn left, step forward, etc.).

1.3 Option Discovery and Transfer

The traditional HRL does not specify how options are transferred across tasks. Intuitively, under some circumstances, transferring an option from a previous task would speed up learning on a novel task (for example, learning to make a cup of coffee after learning to make a cup of tea). In other instances, learning a new task might be inhibited by previously acquired options, such as the tendency to get stuck in pre-existing ways of thinking that preclude creative solutions to new problems, a phenomenon known as the Einstellung effect [11]. HRL does not make it clear why an agent faced with a new task would choose to execute an option acquired in a previous task, particularly if the state space of the new task is different. The problem of transferring behaviors from a source task to a target task has been investigated in the field of transfer learning and is often solved via an explicit inter-task mapping between states in the two tasks [12]. This mapping is often supplied manually

or is discovered offline via batch heuristic methods which lack biological plausibility.

Furthermore, HRL fails to specify how options are learned in the first place, with most HRL accounts assuming a pre-specified set of options [5]. The problem of discovering useful options has traditionally been addressed in an ad-hoc fashion, often by breaking down a task into multiple sub-tasks and defining the optimal solution to each sub-task as a separate option. The sub-tasks are often defined either by manually designating certain states as subgoal states using internal rewards ("pseudo-rewards"), or by partitioning the original task space based on the topology of the state transition structure (for example, by identifying "bottleneck" states such as doorways, or clusters of interconnected states such as rooms) [5]. Solway et al. [13] proposed a more principled solution based on Bayesian model selection. In their study, they showed that useful options can be learned by considering an ensemble of tasks occurring in the same environment. Human behavior was consistent with a decomposition of the environment into sub-tasks, each solved by a corresponding option, such that the resulting task hierarchy yields the most parsimonious explanation of optimal behaviors in the full task ensemble.

Even though it is limited by strong assumptions about the state space and the transition dynamics, the approach of Solway et al. highlights the importance of explicitly considering a distribution of tasks. In fact, most heuristic approaches to option discovery implicitly assume a task distribution, often by assuming that each accessible state has equal probability of being the start state or the goal state. To illustrate why option discovery only makes sense in the context of ensembles of tasks, consider what would happen if the agent only ever faced a single task in the world: it could then just learn the optimal policy for that one task using regular RL. Options in the HRL sense would thus be unnecessary as the entire policy would be executed automatically, like a single option. Options thus imply a world with multiple tasks. However, if those tasks were completely unrelated, options would again not make sense as the agent would have to learn a separate policy for each task from scratch, much like in the single-task scenario. In order for the agent to be able to learn useful options that can be transferred across tasks, there must be a shared structure between those tasks. While this point may seem trivial, we believe it lies at the crux of understanding how people and animals acquire their rich repertoire of habitual behaviors.

In order to understand how options are learned and facilitate performance on novel tasks, it is therefore necessary to define a probability distribution over tasks in the environment. Crucially, this distribution of tasks would be highly structured [14], such that tasks are grouped into families of tasks that share certain parameters which are not directly observable. Option discovery would thus correspond to a particular kind of hidden state inference, where the hidden state refers to the parameters that govern the distribution of tasks.
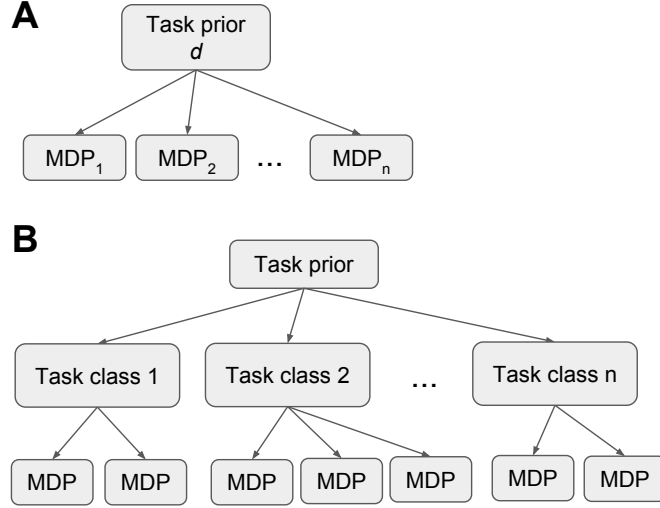
Figure 1: (A) The MTRL framework, (B) MTRL with classes of tasks. Adapted from [15].

1.4 Multi-task Reinforcement Learning

The problem of optimizing behaviors for a set of tasks (MDPs) drawn from a common distribution is the focus of multi-task reinforcement learning (MTRL) (Figure 1A). In MTRL, MDPs are assumed to be drawn from the same prior distribution. Experiencing a sequence of MDPs can allow the agent to infer the hidden parameters of the distribution and to quickly discover the optimal policy for a new MDP drawn from the same distribution. In its simplest instantiation, MTRL allows the agent to learn a "library" of MDPs and reuse policies from MDPs it has seen in the past.

Wilson et al. [15] extended this framework to include *classes* of tasks, where each class defines its own distribution of MDPs (Figure 1B). Their model allows the agent to rapidly discover the structure of tasks similar to previous tasks, while simultaneously supporting the discovery of entirely new classes of tasks. Policies acquired in old tasks can thus be adapted to new tasks from the same class, while avoiding incorrect generalization to tasks from different classes.

1.4.1 MTRL for HRL

While Wilson et al.'s approach beings to address the question of policy transfer across MDPs, it leaves open several questions:

- How does ambiguity between MDPs arise? Traditional tabular MDPs assume that the agent is fully aware of what state it is in, which implies it is also fully aware of which MDP it is in. In order to be able to perform inference over MDPs, it must

be the case that states are in some sense "shared" across MDPs, or at least "look similar".

- What are the distributions over MDPs? Intuitively, they should be generic enough to capture a wide variety of tasks that agents may encounter in the world, while being constrained enough to facilitate efficient learning and generalization.

- Why would options arise? In order to be able to define useful options and convert a given MDP into an SMDP, there must be some form of modular or subtask structure build into the MDP.

- Why would options be transferred? In order to transfer options, the modules or subtasks need to be shared across MDPs.

We address the first question by appealing to the formalism of partially observable Markov decision processes (POMDPs), in which states are not directly observable but instead define distributions over observations. The agent must thus infer which state it is in based on these observations. In MTRL with POMDPs [16], the agent additionally infers which POMDP it is in.

We address the second question using infinite POMDPs (iPOMDPs) [17], which allow the agent to flexibly adapt to environments with different state spaces.

We address the third question by constraining the iPOMDPs to a particular modular structure, such that states are grouped into *modules*, sometimes referred to as communities or blocks or states. This is a novel contribution of our work.

We address the last question by allowing the structure and properties of these modules to be shared across different iPOMDPs.

1.5   Partially Observable Markov Decision Processes

Partially observable Markov decision processes (POMDPs) are useful for modeling environments in which states are not directly observable but rather generate observations, which the agent must use to infer the underlying hidden states (Figure 2B). POMDPs extend MDPs by including the following:

- Ω is a set of observations,

- $O(\cdot|s,a)$ is a distribution of observations, such that $O(o|s,a)$ is the probability of seeing observation $o \in \Omega$ after executing action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$.

In POMDPs, the agent gets an observation after each action, so the agent's history is now a sequence of the form $\mathbf{h}_t = [o_0, a_0, o_1, a_1, o_2, a_2...a_{t-1}, o_t]$. The agent's history induces a probability distribution over the current state it is in:
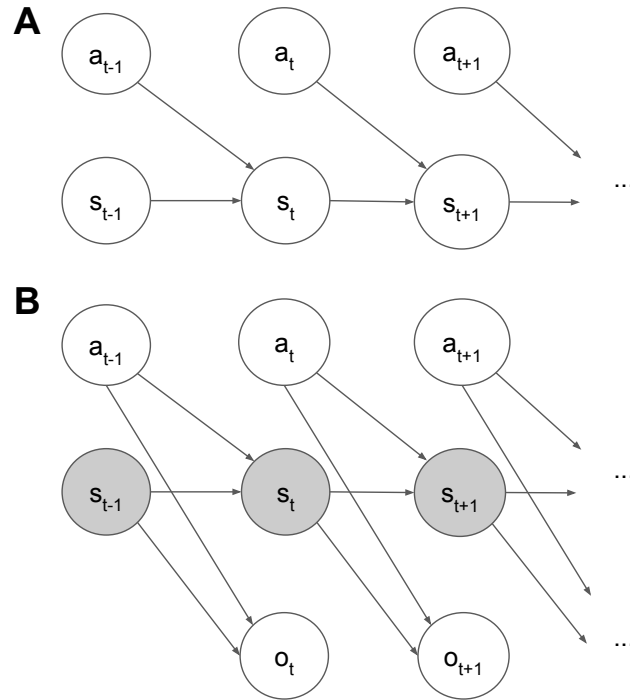
Figure 2: (A) A time slice of an MDP showing the dependence between successive states and actions. (B) A time slice through the POMDP. Grey circles indicate that states are not directly observable.

$$b_t(s_t) = P(s_t|o_0, a_0...a_{t-1}, o_t)$$

This summary statistic is known as a *belief state*. It obeys the Markov property and can be updated recursively at each time step using Bayes' rule:

$$
\begin{aligned}
b_{t+1}(s_{t+1}) &= & P(s_{t+1}|o_0, a_0...a_t, o_{t+1}) \\
&= & \sum_{s_t} T(s_{t+1}|s_t, a_{t+1}) \ P(s_t|o_0, a_0...a_t, o_{t+1}) \\
&\propto & \sum_{s_t} T(s_{t+1}|s_t, a_{t+1}) \ P(o_{t+1}|o_0, a_0...a_t, s_t) \ P(s_t|o_0, a_0...a_{t-1}, o_t) \\
&= & \sum_{s_t} T(s_{t+1}|s_t, a_{t+1}) \ O(o_{t+1}|s_t, a_t) \ b_t(s_t)
\end{aligned}
$$

Rewards are treated as additional observations (omitted here to keep the notation uncluttered).

Now we can define MTRL using POMDPs [16]. Different POMDPs generate similar observations, creating ambiguity between POMDPs and necessitating inference over multiple POMDPs.

1.5.1 POMDPs in the Brain

Recent work [18] using a task in which stimuli provide ambiguous information about the true state of the environment showed that the activity of midbrain dopaminergic neurons corresponds to the RPE computed on the inferred hidden state of the environment, consistent with a POMDP representation of the task.

1.6 Infinite POMDPs

In order to allow agents to learn a diverse set of tasks with different state spaces and transition functions, we turn to the formalism of infinite POMDPs (iPOMDPs) [17]. This extends POMDPs in two ways. First, the transition probabilities T are themselves treated as random variables with their own probability distributions. The unknown transitions T together with the hidden state s thus form an expanded hidden *hyperstate* [19]. This allows the agent to learn a POMDP with any arbitrary transition structure. Second, a nonparametric prior is defined over T, which allows for an unbounded number of states and transitions between them.

Formally, we generate a model from the iPOMDP prior in the following steps:

- A state "popularity" vector $\bar{T}$ is drawn from a stick-breaking process (GEM) [20] with concentration parameter α_0, such that $\bar{T}(s')$ corresponds to the average probability that a transition will lead to state s'. It can be thought of as the average transition distribution.

- A transition probability vector $T(\cdot|s,a)$ is drawn from a Dirichlet process (DP) [20] with concentration parameter α_1 and base distribution $\bar{T}$ for each state-action pair (s,a).

- An observation distribution $O(\cdot|s,a)$ is drawn from a base distribution H_O for each state-action pair (s,a).

- A reward distribution $R(\cdot|s,a)$ is drawn from a base distribution H_R for each state-action pair (s,a).

- States s_t, observations o_t, and rewards r_t are drawn as usual from their corresponding distributions.

The steps can be summarized as:

$$
\begin{aligned}
\bar{T} &\sim \text{GEM}(\alpha_0) && \text{average transition distribution} \\
T(\cdot|s,a) &\sim \text{DP}(\alpha_1, \bar{T}) && \text{transition distributions} \\
O(\cdot|s,a) &\sim H_O && \text{observation distributions} \\
R(\cdot|s,a) &\sim H_R && \text{reward distributions} \\
s_t &\sim T(\cdot|s_{t-1}, a_{t-1}) && \text{states} \\
o_t &\sim O(\cdot|s_{t-1}, a_{t-1}) && \text{observations} \\
r_t &\sim R(\cdot|s_{t-1}, a_{t-1}) && \text{rewards}
\end{aligned}
$$

The popularity vector $\bar{T}$ and the transition vectors $T(\cdot|s,a)$ are of infinite size. However, the generative process favors a small number of states, and hence most of the probability mass will be concentrated on low-numbered states. The extent to which the model prefers "small" states is controlled by the concentration parameter α_0. The concentration parameter α_1 determines the similarity between each transition distribution $T(\cdot|s,a)$ and the mean distribution $\bar{T}$.

Importantly, since the agent can only perceive a finite number of observations during its lifetime, it will only ever need to learn about a finite number of states.

An example iPOMDP is shown in Figure 4A.
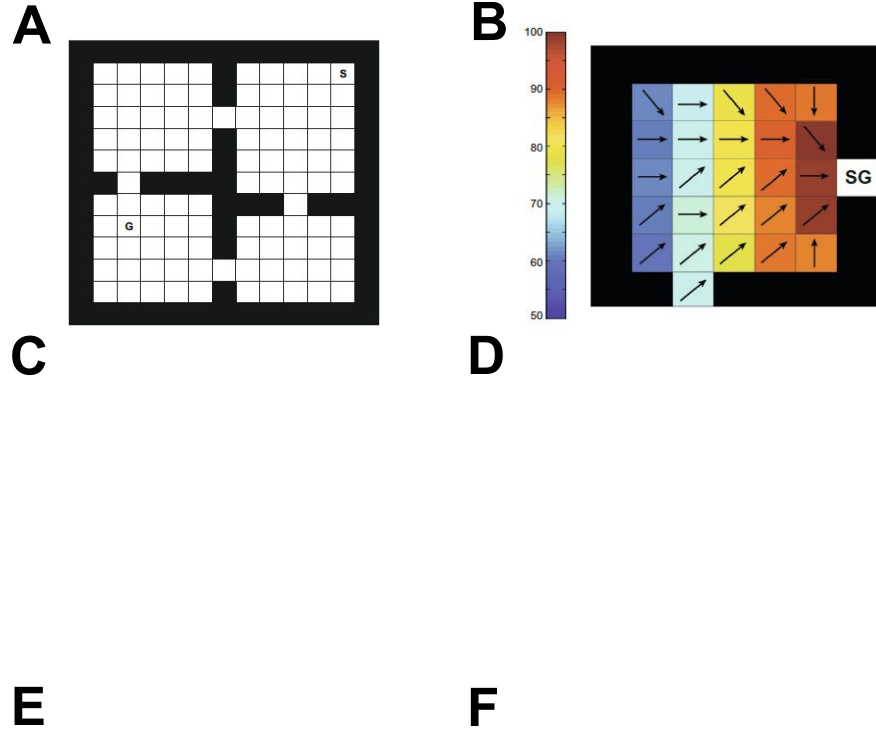
A

B

C

D

E

F

Figure 3: (A) Rooms domain from [4], adapted from [5]. G, goal state, S, starting state. (B) Upper left room with its corresponding option policy (arrows) leading to the subgoal state (SG). Colors indicate option-specific state values.

1.7 Motivating Examples

While the ultimate goal of this project is to design a generative model and inference machinery for a wide range of natural tasks, it is helpful to consider a few illustrative examples of constrained tasks people use to study HRL and actions sequences.

1.7.1 Machine Learning

Rooms domain A popular task used to illustrate HRL is the rooms domain (Figure 3A and B), which consists of a grid world in which some locations (states) are passable while others are not, dividing the space into rooms. Transitions are deterministic and there is often a single goal state in which the task terminates. Usually any transition to the goal state gives some fixed reward, while all other transitions yield zero reward. Alternatively, all states have a negative reward associated with them, with the exception of the goal state which has zero reward. In both cases, the optimal policy is the shortest path from the agent's starting state to the goal state.

In such tasks, options arise naturally, with subgoals corresponding to the doorways between the rooms. If the reward is not in the same room as the agent, then it would have to go explore an adjacent room as fast as possible. Thus it helps to have option policies that lead to the exit of each room from any location within that room. That way the agent can directly perform the option "go to doorway X" and just follow the corresponding option policy.

Taxi domain Another task used to study HRI is taxi domain [**?**]. TODO structure, figure

repeated modules, densely connected, similar observations
subgoals / parse cross-module
for each action; collapsed all actions

1.7.2 Humans

While the ultimate goal of this project can be seen as discovering a generative distribution

1.7.3 Monkeys

1.7.4 Rodents

2 The Model

2.1 Modular iPOMDPs

Combining MTRL with iPOMDPs allows the agent to learn and transfer policies across tasks with arbitrary state spaces and transition structures. However, it is still unclear
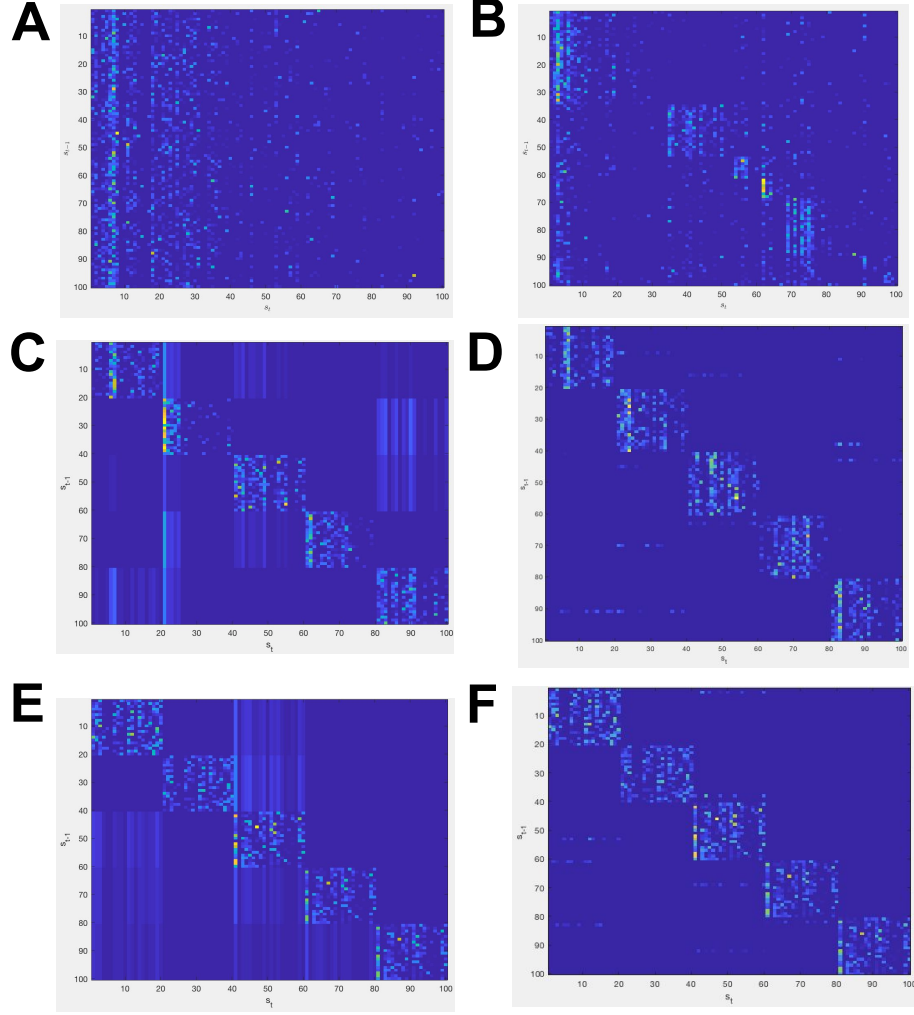
Figure 4: (A) The transition probabilities $T(s_t|s_{t-1}, a_{t-1})$ for a fixed action a_t in a iPOMDP sampled from the prior. (B) Modular iPOMDP with module-specific popularity vectors. (C) Two-layer iPOMDP. (D) Two-layer iPOMDP with subgoals. Notice the sparse off-diagonal transitions. (E) Two-layer iPOMDP with templates. Notice that the within-module transition structures of the first two modules (top left) are identical since they are taken from the same template. Similarly, the within-module transition structures of the last three modules (bottom right) are also identical but drawn from a different template. (F) Two-layer iPOMDP with subgoals and templates.

how options would arise in such a framework. Intuitively, options imply a certain modular structure, such that some parts of the state space have characteristic transition structures or observation distributions which are repeated across POMDPs or even within the same POMDP. Learning an appropriate sequence of actions in one of those modules will allow the agent to transfer it to other instances of the same module.

2.1.1 Module-specific popularity vectors

One way to incorporate modules in iPOMDPs is by extending block diagonal infinite hidden Markov models (BD-iHMMs) [21]. Intuitively, each state is assigned to a module, and states from the same module tend to transition to states within the module more often than to states outside of the module. As an example, modules can be thought of as rooms, while states in each module can be thought of as locations within the room.

Formally, we can introduce the following:

- $\mathcal{M}$ is a set of modules,

- $\bar{T}_m$ is a state popularity vector for each module $m \in \mathcal{M}$,

- $z_s \in \mathcal{M}$ is the module that state $s \in \mathcal{S}$ belongs to.

To generate a modular iPOMDP:

- A module popularity vector β is drawn from a stick-breaking process with concentration parameter γ.

- A module assignment z_s is drawn for each state s from a categorial distribution defined by β.

- An overall state popularity vector $\bar{T}$ is drawn from a stick-breaking process with concentration parameter α_0.

- A module-specific state popularity vector $\bar{T}_m$ is created for each module m. It is equal to $\bar{T}$, except with $\bar{T}_m(s)$ scaled by a factor of $(1 + \xi)$ for all states s such that $z_s = m$. The whole vector $\bar{T}_m$ is then renormalized to sum to 1.

- A transition probability vector $T(\cdot|s, a)$ is drawn from a DP with concentration parameter α_1 and base distribution $\bar{T}_m$ for each state-action pair (s, a), where $m = z_s$.

- Reward distributions R, observation distributions O, states s_t, observations o_t, and rewards r_t are drawn as for regular iPOMDPs.

The steps can be summarized as:

$$\beta \sim \text{GEM}(\gamma) \qquad\qquad\qquad \text{module popularities}$$

$$z_s \sim \beta \qquad\qquad\qquad \text{module assignments}$$

$$\bar{T} \sim \text{GEM}(\alpha_0) \qquad\qquad\qquad \text{average transition distribution}$$

$$C_m = 1 + \xi \sum_s \bar{T}(s) \; \delta(z_s = m) \qquad\qquad\qquad \text{normalization constant}$$

$$\bar{T}_m(s) = \begin{cases} \frac{1+\xi}{C_m}\bar{T}(s) & \text{if } z_s = m \\ \frac{1}{C_m}\bar{T}(s) & \text{if } z_s \neq m \end{cases} \qquad \text{average transition distribution for each module}$$

$$T(\cdot|s,a) \sim \text{DP}(\alpha_1, \bar{T}_m), \text{ where } m = z_s \qquad\qquad\qquad \text{transition distributions}$$

$$O(\cdot|s,a) \sim H_O$$

$$R(\cdot|s,a) \sim H_R$$

$$s_t \sim T(\cdot|s_{t-1}, a_{t-1})$$

$$o_t \sim O(\cdot|s_{t-1}, a_{t-1})$$

$$r_t \sim R(\cdot|s_{t-1}, a_{t-1})$$

Unlike regular iPOMDPs, here there is a separate average transition distribution $\bar{T}_m$ for each module m. Importantly, that distribution is artificially skewed to prefer states within m. The parameter ξ controls the bias for within-module transitions, with $\xi = 0$ leading to no bias. The concentration parameter γ determines whether the prior is biased towards a small or a large number of modules.

An example modular iPOMDP is shown in Figure 4B.

The module-specific average transition distributions would allow the model to discover the modular or "community" structure of tasks in the environment, something which is characteristic of human behavior [22, 13]. However, since each transition distribution $T(\cdot|s,a)$ is drawn independently (conditional on $\bar{T}_m$), each module will have its own unique transition structure. An option learned in one module would not apply to a different module.

2.1.2 Two-layer iPOMDPs

In order to facilitate the acquisition of options, modules need to be repeated, which is infeasible with the previous approach. The easiest way to achieve this is by treating modules as self-contained units. One way to accomplish this is by inducing a separate iPOMDP for each module, which would be "nested" in a higher-level iPOMDP defined over modules.

To illustrate this, consider the original iPOMDP formulation in which states induce distributions over observations. Now imagine that states are modules, and that the observations themselves are the states, with their own module-specific transition distributions.

Formally, we can generate a two-layer iPOMDP as follows:

- A **module** popularity vector $\bar{T}$ is drawn from a stick-breaking process with concentration parameter α_0, such that $\bar{T}(m')$ corresponds to the average probability that a transition will lead to module m'.

- A **module** transition probability vector $T(\cdot|m)$ is drawn from a DP with concentration parameter α_1 and base distribution $\bar{T}$ for each module m, such that $T(m'|m)$ is the average probability that an action in a state in module m will lead to a state in module m'.

- A within-module **state** popularity vector $\bar{T}_m$ is drawn from a stick-breaking process with concentration parameter α_2 for each module m, such that $\bar{T}_m(s')$ corresponds to the average probability that a transition within module m will lead to state s'. This pertains to transitions to states within m only.

- A within-module **state** transition probability vector $T_m(\cdot|s,a)$ is drawn from a DP with concentration parameter α_3 and base distribution $\bar{T}_m$ for each state-action pair (s,a) within each module m. Thus $T_m(s'|s,a)$ is the probability of transitioning to state s' after taking action a in state s within module m. Note that both s and s' belong to module m.

- Global transitions $T(s'|s,a)$ across states (both within and across modules) are defined as follows. If both s and s' are in the same module m, then $T(s'|s,a)$ equals the within-module transition probability $T_m(s'|s,a)$, scaled by the probability that a module will transition to itself, $T(m|m)$. If s and s' belong to different modules m and m', respectively, then $T(s'|s,a)$ is taken to be the popularity of s' in module m', $\bar{T}_{m'}(s')$, scaled by the probability of a transition from m to m', $T(m'|m)$.

- Reward distributions R, observation distributions O, states s_t, observations o_t, and rewards r_t are drawn as for regular iPOMDPs.

The steps can be summarized as:

$$\bar{T} \sim \mathrm{GEM}(\alpha_0) \qquad \text{module popularities}$$
$$T(.|m) \sim \mathrm{DP}(\alpha_1, \bar{T}) \qquad \text{module transitions}$$
$$\bar{T}_m \sim \mathrm{GEM}(\alpha_2) \qquad \text{within-module state popularities}$$
$$T_m(\cdot|s, a) \sim \mathrm{DP}(\alpha_3, \bar{T}_m), \ \forall s \in m \qquad \text{within-module state transitions}$$
$$T(s'|s, a) = \begin{cases} T_m(s'|s, a) \ T(m|m) & \text{if } s, s' \in m \\ \bar{T}_{m'}(s') \ T(m'|m) & \text{if } s \in m, s' \in m' \end{cases} \qquad \text{transitions between all states}$$
$$O(\cdot|s, a) \sim H_O$$
$$R(\cdot|s, a) \sim H_R$$
$$s_t \sim T(\cdot|s_{t-1}, a_{t-1})$$
$$o_t \sim O(\cdot|s_{t-1}, a_{t-1})$$
$$r_t \sim R(\cdot|s_{t-1}, a_{t-1})$$

As before, it makes sense to introduce a bias that favors transitions within modules by scaling $T(m|m)$ by a factor of $(1 + \xi)$:

$$T(m'|m) \leftarrow \begin{cases} \frac{1+\xi}{1+\xi T(m|m)} \ T(m|m) & \text{if } m' = m \\ \frac{1}{1+\xi T(m|m)} \ T(m'|m) & \text{if } m' \neq m \end{cases}$$

Now each module has its own separate transition structure defined by T_m. Even though each module is still unique, this version of the model allows us to bias the prior towards repeated modules, which would naturally give rise to options.

An example of a two-layer iPOMDP is shown in Figure 4C.

2.1.3 Two-layer iPOMDPs with subgoals

For options to arise, intuitively there need to be certain subgoals. One common definition of subgoals is "bottleneck" states, such as doorways or passages, which in our framework would correspond to states which transition from one module to another. Intuitively, we want the cross-module transitions to be sparse, with a few states from one module (the subgoal or *exit* states) transitioning to a few states to another module (the *entrance* states).

Two-layer iPOMDPs as defined above posit that each state from a given module m has the same transition distribution for all states in another module m', which is implausible (notice the off-diagonal structure in Figure 4C).

We would like to define **cross-module** state transition distributions $T_{m,m'}(s'|s, a)$, specifying the probability of transitioning from state $s \in m$ to state $s' \in m'$ after taking action a (for $m \neq m'$). We also want this distribution to be sparse, i.e. all probabilities

would be zero for most states s, except for a few states s (the exit states of m) which would transition to a few states s' (the entrance states of m'). One way to achieve this is by selecting a single exit state $s_{m,m'}$ that acts as a "bridge" from module m to module m':

- An **exit state** probability vector $\bar{T}_{m,\cdot}$ is drawn from a GEM with concentration parameter α_4, specifying which states in module m are likely to be exit states.

- An **entrance state** probability vector $\bar{T}_{\cdot,m'}$ is drawn from a GEM with concentration parameter α_5, specifying which states in module m' are likely to be entrance states.

- A single **exit state** $s_{m,m'} \in m$ is chosen according to $\bar{T}_{m,\cdot}$ for each pair of distinct modules m, m'. Its transition distribution $T_{m,m'}(\cdot|s,a)$ to states $s' \in m'$ is then chosen according to a DP with base distribution $\bar{T}_{\cdot,m'}$ and concentration parameter α_6.

- Global transitions $T(s'|s,a)$ are modified accordingly.

Thus the overall two-layer iPOMDP generation process can be summarized as:

$$
\begin{aligned}
\bar{T} &\sim \text{GEM}(\alpha_0) && \text{module popularities}\\
T(.|m) &\sim \text{DP}(\alpha_1, \bar{T}) && \text{module transitions}\\
\bar{T}_m &\sim \text{GEM}(\alpha_2) && \text{within-module state popularities}\\
T_m(\cdot|s,a) &\sim \text{DP}(\alpha_3, \bar{T}_m), \ \forall s \in m && \text{within-module state transitions}\\
\bar{T}_{m,\cdot} &\sim \text{GEM}(\alpha_4) && \text{exit state popularities}\\
\bar{T}_{\cdot,m'} &\sim \text{GEM}(\alpha_5) && \text{entrance state popularities}\\
s_{m,m'} &\sim \bar{T}_{m,\cdot} && \text{exit states}\\
T_{m,m'}(\cdot|s,a) &\sim \text{DP}(\alpha_6, \bar{T}_{\cdot,m'}), \ \text{for } s = s_{m,m'} && \text{cross-module state transitions}\\
T(s'|s,a) &= \begin{cases} T_m(s'|s,a)\, T(m|m) & \text{if } s, s' \in m\\ \bar{T}_{m,m'}(s'|s,a)\, T(m'|m) & \text{if } s \in m, s' \in m' \end{cases} && \text{transitions between all states}\\
O(\cdot|s,a) &\sim H_O\\
R(\cdot|s,a) &\sim H_R\\
s_t &\sim T(\cdot|s_{t-1}, a_{t-1})\\
o_t &\sim O(\cdot|s_{t-1}, a_{t-1})\\
r_t &\sim R(\cdot|s_{t-1}, a_{t-1})
\end{aligned}
$$

With the same additional bias for higher $T(m|m)$, favoring within-module transitions. An example of a two-layer iPOMDP with subgoals is shown in Figure 4D.

2.1.4   Two-layer iPOMDPs with templates

One way to bias the prior towards repeated structures to allow transition structures to be shared between different modules. Thus two different modules m and m' may have identical state transition structures $T_m \equiv T_{m'}$.

To accomplish this, it is convenient to introduce *templates* τ that define unique transition structures T_τ. Each module m is assigned to a template τ, resulting in $T_m \equiv T_\tau$.

We introduce the following components to the two-layer iPOMDP generation process:

- $\bar{T}_\tau$ and T_τ are drawn in the same way as $\bar{T}_m$ and T_m, respectively, except defined for templates τ instead of modules m.

- A template popularity vector β is drawn from a stick-breaking process with concentration parameter γ.

- A template assignment z_m is drawn for each module m from a categorial distribution defined by β.

- $\bar{T}_m$ and T_m are assigned equal to $\bar{T}_\tau$ and T_τ, respectively, for the corresponding template $\tau = z_m$.

Hence instead of drawing $\bar{T}_m$ and T_m for modules m, we now draw them for templates τ in the exact same manner:

$$\bar{T}_\tau \sim \text{GEM}(\alpha_2)$$
$$T_\tau(\cdot|s,a) \sim \text{DP}(\alpha_3, \bar{T}_\tau), \ \forall s \in \tau$$

Modules are assigned to templates according to:

$$\beta \sim \text{GEM}(\gamma)$$
$$z_m \sim \beta$$
$$\bar{T}_m \equiv \bar{T}_\tau, \ \text{where } \tau = z_m$$
$$T_m(\cdot|s,a) \equiv T_\tau(\cdot|s,a), \ \text{where } \tau = z_m$$

An example of a two-layer iPOMDP with templates is shown in Figure 4E.

2.2   MTRL with two-layer iPOMDPs

There are different ways to adapt two-layer iPOMDPs to the MTRL framework that will facilitate the acquisition and transfer of different kinds of options.

2.2.1 Two-layer iPOMDPs with templates and subgoals

The iPOMDP with subgoals can easily be extended with templates. An example is shown in Figure 4F.

2.2.2 Prior over tasks

One convenient choice is a Chinese restaurant process (CRP) [20] prior over tasks. Intuitively, we can think of time as being divided into discrete episodes (or trials). At the start of each episode, the agent is presented with a task, which is either identical to a task seen in a previous episode, or is an entirely new task. The CRP prior favors a rich-gets-richer dynamic whereby tasks which are selected frequently tend to be selected even more frequently.

Formally, if the distinct tasks are indexed by $k = 1..K$, then on episode e, the task is drawn according to:

$$P(\text{task}_e = k | \text{task}_0, \text{task}_1...\text{task}_{e-1}) \propto \begin{cases} n_k & \text{if } k \leq K \\ \alpha & \text{if } k = K + 1 \end{cases}$$

where K is the number distinct tasks presented so far, n_k is the number of times task k has been chosen before, and α is a concentration parameter.

In our case, the tasks are represented by two-layer iPOMDPs. During each episode, the agent can use the CRP prior to decide whether it is learning a new iPOMDP, or whether it is performing in a previously encountered iPOMDP.

2.2.3 Shared templates

In the framework presented so far, templates are defined separately for each iPOMDP. In order to allow the acquisition of options that apply across multiple iPOMDPs, we can allow templates to be shared across iPOMDPs. This can be done in a straightforward manner by drawing $\bar{T}_\tau$ and T_τ once and using them in all iPOMDPs.

2.2.4 Shared observations

2.2.5 Shared rewards

2.2.6 Classes of tasks

Bibliography & References Cited

[1] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning.* MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[2] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1957.

[3] Ray J Dolan and Peter Dayan. Goals and habits in the brain. *Neuron*, 80(2):312–325, 2013.

[4] Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1):181 – 211, 1999.

[5] Matthew Botvinick, Yael Niv, and Andrew C Barto. Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective. *Cognition*, 113:262–80, 11 2008.

[6] Amir Dezfouli and Bernard W. Balleine. Actions, action sequences and habits: Evidence that goal-directed and habitual action control are hierarchically organized. *PLOS Computational Biology*, 9(12):1–14, 12 2013.

[7] Kyle Smith and Ann Graybiel. Using optogenetics to study habits. *Brain Research*, 1511, 01 2013.

[8] Xin Jin, Fatuel Tecuapetla, and Rui M. Costa. Basal ganglia subcircuits distinctively encode the parsing and concatenation of action sequences. *Nature Neuroscience*, 17:423 EP –, Jan 2014. Article.

[9] Matthew Botvinick and Ari Weinstein. Model-based hierarchical reinforcement learning and human action control. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 369(1655), 2014.

[10] José F. Ribas-Fernandes, Alec Solway, Carlos Diuk, Joseph T McGuire, Andrew G Barto, Yael Niv, and Matthew M Botvinick. A neural signature of hierarchical reinforcement learning. *Neuron*, 71(2):370–379, 2017/10/26 2011.

[11] Merim Bilalić, Peter McLeod, and Fernand Gobet. Inflexibility of experts – reality or myth? quantifying the einstellung effect in chess masters. *Cognitive Psychology*, 56(2):73 – 102, 2008.

[12] Matthew E. Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *J. Mach. Learn. Res.*, 10:1633–1685, December 2009.

[13] Alec Solway, Carlos Diuk, Natalia Córdova, Debbie Yee, Andrew G. Barto, Yael Niv, and Matthew M. Botvinick. Optimal behavioral hierarchy. *PLOS Computational Biology*, 10(8):1–10, 08 2014.

[14] Matthew Botvinick, Ari Weinstein, Alec Solway, and Andrew Barto. Reinforcement learning, efficient coding, and the statistics of natural tasks. *Current Opinion in Behavioral Sciences*, 5:71 – 77, 2015. Neuroeconomics.

[15] Aaron Wilson, Alan Fern, Soumya Ray, and Prasad Tadepalli. Multi-task reinforcement learning: A hierarchical bayesian approach. *ACM International Conference Proceeding Series*, 227:1015–1022, 2007.

[16] Hui Li, Xuejun Liao, and Lawrence Carin. Multi-task reinforcement learning in partially observable stochastic environments. *Journal of Machine Learning Research*, 10:1131–1186, June 2009.

[17] Finale Doshi-Velez. The infinite partially observable markov decision process. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 477–485. Curran Associates, Inc., 2009.

[18] Clara Kwon Starkweather, Benedicte M. Babayan, Naoshige Uchida, and Samuel J. Gershman. Dopamine reward prediction errors reflect hidden-state inference across time. *Nat Neurosci*, 20(4):581–589, Apr 2017. Article.

[19] Michael O'Gordon Duff. *Optimal Learning: Computational Procedures for Bayesadaptive Markov Decision Processes*. PhD thesis, 2002. AAI3039353.

[20] Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

[21] Thomas Stepleton, Zoubin Ghahramani, Geoffrey Gordon, and Tai-Sing Lee. The block diagonal infinite hidden markov model. In David van Dyk and Max Welling, editors, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 552–559, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009. PMLR.

[22] Anna C. Schapiro, Timothy T. Rogers, Natalia I. Cordova, Nicholas B. Turk-Browne, and Matthew M. Botvinick. Neural representations of events arise from temporal community structure. *Nature Neuroscience*, 16(4):486–492, Apr 2013.