LEARNING TRANSFERABLE TASK SCHEMAS BY REPRESENTING CAUSAL INVARIANCES

Anonymous authors
Paper under double-blind review

ABSTRACT

Real-world tasks are often compositional and possess causal structure over component subtasks that is independent of instantiation in a particular environment or the corresponding low-level observations. While rewards provide a natural way of controlling the behaviour of an agent solving such tasks, under the standard assumption for reinforcement learning problems rewards depend only on the present state and action. In contrast, in cases where rewards have complex, causal dependencies on the agent's history (such as completion of specific previous subtasks) this assumption either forces the state representation to encompass the whole history of the agent, or requires finding a clever way of compressing this history into a belief state on which to act. It is not clear how to achieve this while also allowing better generalization to novel environments, where one intuition is that faithfully capturing the underlying generative process of a task, using a modularized representation of disentangled causal factors is the beneficial approach. To address this problem, we propose a memory based meta-learning architecture that allows transfer of task structure through a dual approach: by basing its predictions about rewards (including subtask completions) on a memory indexed by a learned task decomposition, thus conditioning away low-level observations, while also learning to infer and represent order-dependence, the strength of temporal causal dependencies between different rewards . This allows the transfer of the task structure to novel environments , while also enabling the algorithm to trade-off visiting currently rewarding states against long-term consequences.

1 INTRODUCTION

What could be more satisfying than inviting over one's friends for a well-organized dinner party to a new home? While certainly desirable, such feats of human organization and knowledge transfer are still hard to capture algorithmically, as they rely on complex structural knowledge about the task at hand, allowing one to accomplish it more-or less successfully even after relocating to a new flat, country, or continent. In our particular example, organising a dinner party relies, among other things, on the knowledge that food should (mostly) be prepared before guests arrive, and that shopping needs to be done before cooking, but also that shopping for ingredients itself is a (mostly) order-independent process, allowing for the possibility to plan the most efficient route within and between stores and their isles, whereas cooking a recipe (mostly) requires following steps in strict order.

In this paper we explore a memory-based meta-learning approach to learn to represent different phases or 'task states' of a compositional tasks, and the relationships between these task states, in order to successfully complete tasks with such shared structure across different settings. In particular, we use a memory-augmented neural network (Santoro et al., 2016) as a generative model to represent task states as well as order-dependency, predict eligible subtasks and rewards, learn policies, and plan ahead to complete subtasks in the order promising of the highest total return.

2 BACKGROUND:

Markov Decision Processes (MDPs) have been successful as a model for solving many simulated and real-world problems while using reward functions obeying the Markov property, where the reward distribution is a function of the current state, action and arrival state only. It has long been observed, however, that natural rewards guiding human and animal behaviour, or determining desirable performance criteria in robotics can often be non-Markovian (Bacchus et al., 1996; Littman et al., 2017) when specified using intuitive, tractable state spaces. In the case of Non-Markovian Reward Decision Processes (NM-RDPS) one approach to this problem is to use the entire history of the agent as an extend state space on which to define the reward distribution, however this is clearly problematic because of the resulting combinatorial explosion. Earlier work has thus focused on learning a minimal extension of the state space that makes rewards Markovian, using the formulations of linear temporal logic, or finite state automata (Li et al., 2016; Camacho et al., 2017; Brafman et al., 2018; Icarte et al., 2018). While the motivation behind most of such work is to reduce the size of the extended state space to mitigate its effect on the sample complexity of learning a particular task, here we focus expressly on *generalization*: namely learning the *structure* of the reward function for a family of tasks. In the next sections we outline our approach, its relationship to learning state space models, and the implementation in a factorized recurrent neural network.

3 Motivation

Our approach is to augment the state space of a task specified by a non-Markov reward function (NMRF) using a task state variable, in order to segment the task into sub-parts in a generalizable way, such that, conditional on the current task state, the rewards, and thus the optimal policy is independent of the history during previous task states. This idea is related to so-called semi-Markov options (Sutton et al., 1999), where the termination condition is the function of the history during the execution of the option, hence we refer to this type of segmentation as semi-Markov task states. Unlike in the case of options, however, the task states are inherent in the structure of the task itself,and not simply a way to define temporal abstraction over actions. We make the assumption that the true reward function at a specific time is always expressible as a function of the complete state and reward history, where we leave out the history of actions for convenience, though this assumption is not central to our approach. Given this assumption, we desire the decomposition of the task into task states to be generalizable and minimal, in the sense that we make more precise shortly. Intuitively, to learn a general task *schema*, we want the transition between task states to depend as much as possible on general features (such as a certain number of rewards signalling the completion of a subtask) and as little as possible on more specific features, such as sensory features attached to the reward, or the state visited to acquire it (both of which might change in a different instantiation of a task within the same family of tasks). In our original example, if we learned that shopping is an order-independent task state of collecting n items on the shopping list, we know to transition to the next task state after n reward signals, with the local history during the task state guiding the collection of rewards (remembering the items already bought and where the others might be). This way we are not forced into an unnecessary ordering of this process present in previous work (e.g.(Andreas et al., 2017)). In contrast, if the details of the items bought (represented as reward features or state visitations) factor into task state transitions, or buying each individual item is represented as a separate task state, this creates an order-dependent evolution of task states, resulting in having to evaluate the effect of an exponentially growing number of different orderings on the evolution of the reward function.

We use these intuitions of the desirable inductive biases to construct a model that learns task states that predict rewards in an NM-RDP while encouraging generalizability by regularizing the flow of state information into the task state transition process.

4 Algorithm and Implementation

We implement the learning of task states in a neural generative model using the variational autoencoder (VAE) framework (Kingma & Welling, 2013; Rezende et al., 2014), learning a factorized state space model that handles an external memory, similarly to previous approaches implementing a generative temporal model with external memory to generate sensory predictions (Gemici et al., 2017; Fraccaro et al., 2018; Whittington et al., 2018)

4.1 Generative model

The agent's generative model (Fig. 1a) predicts rewards in the arrival state and can be used to construct reward maps for the environment to select actions. The generative model factorizes as

$$p_\theta(\mathbf{r}_{\leq T}, \mathbf{c}_{\leq T}, \mathbf{g}_{\leq T} | \mathbf{s}_{\leq T}) = \prod_{t=1}^{t=T} p_\theta(g_t | g_{t-1}, r_{t-1}) p_\theta(c_t | c_{t-1}, g_t, K_t, r_{t-1}) p_\theta(r_t | c_t, M_t, s_t),$$

where c is the task state (implemented as a vector), M a memory of transitions (including previous episodes) indexed by task state, g a scalar gating variable that gates the effect of state information on the task state c, with this state information represented by a compressed state-reward history K (implemented as the hidden unit of an LSTM with parameters θ_K). Thus the task state moves forward given the reward in the current step, and optionally the identity of the reward as signalled by the state information, as would be desirable in order-dependent parts of a task where differences in histories can lead to very different future predictions. All of the transition densities were modelled as Gaussians with diagonal covariances. The predictive densities for rewards come from using the task state variable c_t and arrival state s_{t+1} as keys to the memory M_t to retrieve sufficient statistics, which are then passed into a small MLP (the *reward network* with parameters θ_r) to give the parameters for a scalar Gaussian density. In line with our motivation above, the current task state c_t at time t in episode e_t retrieves memories from the current episode (working or short-term memory) and previous episodes(long-term memory) using a dot product attention, to condition predictions using the part of history under the same (or similar) task state, using soft attention. We used a set of three sufficient statistics as input to the reward network: from previous episodes, the weighted sum of rewards retrieved using the keys c_t and s_t per episode, averaged over episodes, $\frac{1}{e_t-1}\sum_{e=1}^{e=e_t-1}\sum_k (c_t \cdot c_{e,k}) \cdot (s_t \cdot s_{e,k}) \cdot r_{e,k}$. The maximum reward retrieved with the keys of c_t and s_t across all previous episodes in the memory $\max_{e<e_t,k}(c_t \cdot c_{e,k}) \cdot (s_t \cdot s_{e,k}) \cdot r_{e,k}$, and the sum of rewards already collected during the current episodes $\sum_k (c_t \cdot c_{e_t,k}) \cdot (s_t \cdot s_{e_t,k}) \cdot r_{e,k}$. This gives us the flexibility to predict locally non-Markovian rewards, e.g. rewards that can be 'picked up' once, or that deplete according to some well-defined rule, or indeed are Markovian.

4.2 Inference model and two-filter smoothing

We approximate the intractable filtering and smoothing posteriors, $q_\phi(\mathbf{g}_{\leq t}, \mathbf{c}_{\leq t} | \mathbf{r}_{\leq \mathbf{t}}, \mathbf{s}_{\leq \mathbf{t}})$ and $q_\phi(c_t | \mathbf{r}_{\leq \mathbf{T}}, \mathbf{s}_{\leq \mathbf{T}})$ respectively using variational inference with auto-regressive posteriors. Smoothing is necessary in our model since later rewards are causally dependent on current ones, and inference over the current task state and order-dependence is only possible once we observe the consequence on future rewards. While at the action selection step we can only use the prior based on transitioning from the previous, filtered posterior, we can still overwrite the filtered posterior with the smoothing posterior in the memory, such that will be able to make better inference over the task states and plan more optimally later. In our implementation with recurrent neural networks it will be important to be able to compute a per-step variational lower bound (ELBO), so we use a variational implementation of a two-filter smoother (Briers et al., 2004) where the smoothing posterior at time t is computed using the product of two independent (a forward and backward) filters. The form of the variational filtering and smoothing posteriors is given in the Supplementary Information.

The VAE framework than allows us to represent the component conditional densities by neural networks, sample the stochastic variables, and jointly optimize the generative, posterior,smoothing, LSTM (for computing K), and reward network parameters. We maximize two separate ELBOs, the filtering ELBO at every step of the agent wrt. parameters $\theta, \theta_r, \theta_K$ and ϕ, and a smoothing ELBO at the end of the episode when we run the backward information filter to the beginning of the episode, and optimize wrt. parameters θ, θ_K and ϕ_2

4.3 Reward maps, control, and planning

To test the effectiveness of this framework, we use a heuristic strategy for control and planning. Given a current task state c_t, our generative model can predict a reward for any state s, so we can use it to learn, through replay of previously visited states, a conditional linear reward map $\mathbf{w}_t$ such that the reward received expected for visiting state s during the current task state c_t and with memory M_t is $\Phi(s) \cdot \mathbf{w}$, where $\Phi(s)$ is the state embedding. Rather than applying Q learning

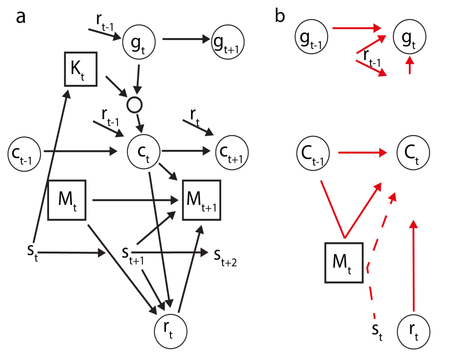

Figure 1: a, Generative model of rewards r using task states c and memory M. Circled letters denote stochastic variables. While we show arrows from c and s to r, these influences actually go through the Memory, playing the part of attention keys. b, Inference network. Dashed arrow indicates optional computations.

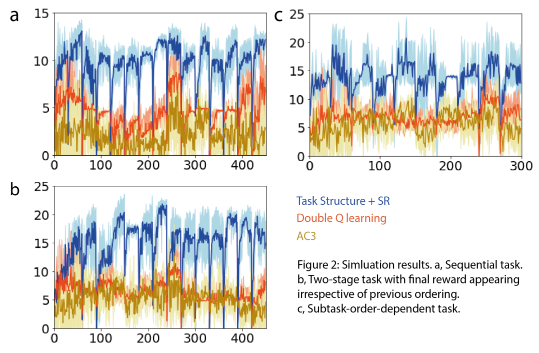

Figure 2: Simluation results. a, Sequential task. b, Two-stage task with final reward appearing irrespective of previous ordering. c, Subtask-order-dependent task.

anew for each task, we will use this reward map with the successor representation (Dayan, 1993) to compute approximations to the Q value function,even though the non-Markovian nature of the reward function violates the assumptions under which this factorization normally holds. Further, we can define a value function for the task state c $V(c, s, h^c$ where h^c is the history during the task state c. In practice we replace the history h^c with the collected reward in c retrieved from the memory $\sum_k (c_t \cdot c_{e_t,k}) \cdot r_{e,k}$. Our forward dynamics model over the task states also allows for planning in this abstract space, by imagining the consequences (ignoring all intermediate steps) of visiting a state s', receiving reward r and transitioning to task state c'. We use the approximate relationship $V(c, s, h^c) \approx r + V(c', s', \emptyset)$, and add the imagined value $V(c', s', \emptyset)$ to the predicted reward r when learning $\mathbf{w}$ through preplay.

5 Experiments

We ran simulations on 8 by 8 (5 by 5 for task 3) tabular grid-world environments under three conditions. In the *sequential task*, three rewards (each worth 5 points) appeared sequentially in predetermined order, each appearing once the previous one was collected and removed by the agent stepping onto the corresponding square. The location of the three rewards changed every 30 episodes (after an initial phase of 60). In the order-independent tasks, three rewards (each worth 5) appeared initially together, and a fourth (worth 10 points) appeared once all of these were collected and removed, in any order. Finally, in the order-dependent task the last reward only appeared if the first three were collected in a particular order. Figure 2 shows the results compared to controls using double Q-learning and the A3C algorithm. Further, we also found that as expected, the absolute value of g was indeed on average much larger during task 3 and smallest in task 2.

6 Related Work and Conlcusion

We introduced a framework to decompose a task depending on its temporal subtask dependencies in a generalisable way using a neural state space model. Unlike previous works we don't assume predefined policies for subtasks (Sohn et al., 2020), a set propositional symbols corresponding to important events (Icarte et al., 2018), or entrenched orderings (Andreas et al., 2017) and in ongoing work we are focusing on the implementation of more sophisticated control approaches.

REFERENCES

Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, International Convention Centre, Sydney, Australia, 2017. PMLR. URL `http://proceedings.mlr.press/v70/andreas17a.html`.

Fahiem Bacchus, Craig Boutilier, and Adam Grove. Rewarding behaviors. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*. AAAI Press, 1996. ISBN 026251091X.

Ronen Brafman, Giuseppe De Giacomo, and Fabio Patrizi. Ltlf/ldlf non-markovian rewards. In *AAAI Conference on Artificial Intelligence*, 2018.

Mark Briers, Arnaud Doucet, and Simon Maskell. Smoothing algorithms for state-space models. In *in Submission IEEE Transactions on Signal Processing*, 2004.

Alberto Camacho, Oscar Chen, Scott Sanner, and Sheila A. McIlraith. Non-markovian rewards expressed in ltl: Guiding search via reward shaping. In *Proceedings of the Tenth International Symposium on Combinatorial Search, SOCS 2017, 16-17 June 2017, Pittsburgh, Pennsylvania, USA*, 2017.

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Comput.*, 1993. ISSN 0899-7667.

Marco Fraccaro, Danilo Jimenez Rezende, Zwols Yori, Alexander Pritzel, S. M.Ali Eslami, and Viola Fabio. Generative temporal models with spatial memory for partially observed environments. In *Proceedings of 35th International Conference on Machine Learning, ICML 2018*, 35th International Conference on Machine Learning, ICML 2018, 2018.

Mevlana Gemici, Chia-Chun Hung, Adam Santoro, Greg Wayne, Shakir Mohamed, Danilo J. Rezende, David Amos, and Timothy Lillicrap. Generative temporal models with memory. *arXiv preprint 1702.04649*, 2017.

Rodrigo Toro Icarte, Toryn Klassen, Richard Valenzano, and Sheila McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, Proceedings of Machine Learning Research, 2018.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint 1312.6114*, 2013.

Xiao Li, Cristian-Ioan Vasile, and Calin Belta. Reinforcement learning with temporal logic rewards, 2016.

Michael L. Littman, Ufuk Topcu, Jie Fu, Charles Isbell, Min Wen, and James MacGlashan. Environment-independent task specifications via gltl. *arXiv preprint 1704.04341*, 2017.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, pp. II1278II1286. JMLR.org, 2014.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In Maria Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, New York, New York, USA, 2016.

Sungryull Sohn, Hyunjae Woo, Jongwook Choi, and Honglak Lee. Meta reinforcement learning with autonomous inference of subtask dependencies. 2020.

R.S. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112:181–211, 1999.

James C.R. Whittington, Timothy H. Muller, Shirley Mark, Caswell Barry, and Timothy E.J. Behrens. Generalisation of structural knowledge in the hippocampal-entorhinal system. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018.

A Supplementary Information

The variational smoothing distribution is given by

$$q_\phi(\mathbf{g}_{\leq t}, \mathbf{c}_{\leq t}|\mathbf{r}_{\leq \mathbf{t}}, \mathbf{s}_{\leq \mathbf{t}}) = \prod_{\tau=1}^{\tau=t} q_\phi(g_\tau|g_{\tau-1}, c_\tau, M_t, r_{t-1}, r_t) \cdot q_\phi(c_\tau|c_{\tau-1}, g_{\tau-1}, M_t, k_t, r_{t-1}, r_t),$$

whereas the smoothing posterior for c is given by

$$\begin{aligned} q_{\phi,\phi_2}(c_t|\mathbf{r}_{\leq \mathbf{T}}, \mathbf{s}_{\leq \mathbf{T}}) = q_\phi(c_t|\mathbf{r}_{\leq \mathbf{t}}, \mathbf{s}_{\leq \mathbf{t}}) q_{\phi_2}(c_t|\tilde{c}_{t+1}, M_t, r_t, r_{t+1}) \cdot \\ \prod_{\tau=T}^{\tau=t+2} q_{\phi_2}(\tilde{c}_{\tau-1}|\tilde{c}_\tau, M_{\tau-1}, r_{\tau-1}, r_\tau) \cdot q_{\phi_2}(\tilde{c}_T|r_T), \end{aligned}$$

where $q_{\phi_2}(\tilde{c}_\tau)$ are the (independently) computed *information filter* posteriors using backward filtering.