GOAL-CONDITIONED LEARNING USING LINKED EPISODIC MEMORY

Effie Li
CS330 Project Report

ABSTRACT

Extended abstract. Memory modules have played a key role in deep reinforcement learning. The most common practice is to keep a replay buffer of seen transitions and subsequently use it to sample transition batches for training the value or policy networks. Some recent work have introduced memory mechanisms such as constructing memory graphs and using graph search methods to help with planning, exploiting contributions from memory beyond its role in optimization (Eysenbach et al., 2019; Liu et al., 2020; Savinov et al., 2018; Zhu et al., 2020). In line with these work, I explore how treating the replay buffer as a content-addressable instance-based memory can help deconstruct the task problem. I draw inspiration from instance-based memory models proposed by psychologists, in particular a model of human hippocampal memory that is capable of transitive inference (REMERGE, Kumaran & McClelland (2012)). I test a goal-conditioned DQN agent equipped with a REMERGE-like, linked episodic memory module in the grid-world navigation domain. In particular, I train the agent in a goal-conditioned setting with a capped maximum task difficulty (within X steps), and test how it transfers its learning on increased task difficulty ($>$ X steps). Although this particular task can be solved with shaped reward (e.g., euclidean distance), my goal is to explore an observation- and reward-independent memory mechanism that can generalize to a broad range of task formulations. To create the linked episodic memory module, I build a bidirectional network linking a conjunctive layer of unique transitions to their corresponding states and next states in two visible layers. There are three key features of this memory module: 1) it attempts to find linked intermediate subgoal states through recurrent similarity computation, 2) it directly makes use of the stored state representations in the regular replay buffer, and 3) it dynamically adds higher-order (multi-step) transitions in the conjunctive layer to aid planning. To perform plan generation, the visible layers in the memory network are probed with the current state and the goal state in a given task. Through a series of recurrent steps, the system retrieves relevant successors and predecessors and attempts to settle on a pair of linked intermediate subgoals, which the base agent can choose to replace the true task goal with at each environment step. Preliminary results indicate that the memory module is capable of generating sensible plans, but linking plan generation with the base agent at scale has not shown fruitful improvement and needs future work. I discuss some ideas on improving the system to better exploit this memory mechanism for generalization.

1 INTRODUCTION

Although the dichotomy of model-free and model-based methods dominates reinforcement learning (RL), memory, and specifically episodic memory, has supported learning in RL systems in important ways. For example, the use of a replay buffer that stores encountered transitions in past episodes is a ubiquitous practice nowadays, with the replay buffer supporting batch optimization by breaking correlated environmental steps as well as preventing the forgetting of learning relevant to previous episodes. Many seminal methods have addressed specifically what content to store in the replay buffer as well as how to sample in a way that better supports optimization (Andrychowicz et al., 2017; Schaul et al., 2016).

However, beyond its role in optimization, episodic memory can support RL by directly helping with control and task solving. Control based on episodic memory, termed episodic RL, was suggested as a third learning system in addition to model-free and model-based RL, and has the potential of boosting early learning performances (Lengyel & Dayan, 2008). Several recent RL work have pushed the front of how memory modules may enable flexible generalization (Eysenbach et al., 2019; Liu et al., 2020; Savinov et al., 2018; Zhu et al., 2020). Similar integration of episodic memory and reinforcement learning has been suggested to underlie human cognition, especially in support of efficient learning with little data and on complex state spaces, as well as enabling long-term reasoning (Gershman & Daw, 2017).

In this work, I investigate how the replay buffer, treated as a content-addressable instance-based memory, can potentially link distant transitions and deconstruct a goal-conditioned task into subproblems. The key insight is that the replay buffer is a perfect instantiation of an instance-based memory system, and therefore we may be able to better exploit information present in the replay buffer by building a transition linking mechanism on top of the stored instances. I draw insights from instance-based models of human memory proposed by psychologists, some of which serve as theories of fast generalization capabilities in humans. I focus on a particular model of human hippocampal memory called REMERGE (Kumaran & McClelland, 2012). The key feature of REMERGE is the retrieval-time recurrent similarity computation, which allowed rapid generalization in tasks such as transitive inference tasks. Note that transitive inference has an intuitive synergy with goal-conditioned RL in tasks where the goal representation is in the same space with the state representation. Ideally, when an agent has learned how to solve the tasks [$s_0 = A$, $s_g = B$] and [$s_0 = B$, $s_g = C$] (s_0 denotes the starting state, s_g denotes the goal state), we would want the agent to be able to solve the new task [$s_0 = A$, $s_g = C$] at the same rate of optimality. This is exactly generalization in the form of transitive inference. Therefore, the goal of this work is to explore whether REMERGE-like, linked episodic memory mechanisms may help a goal-conditioned RL agent to achieve transitive inference generalization.

2 RELATED WORK

Recent work has pushed the front of how memory modules may function as an integrative part in RL systems and provide key generalization capabilities (Eysenbach et al., 2019; Liu et al., 2020; Savinov et al., 2018; Zhu et al., 2020), especially in the context of goal-conditioned RL (Schaul et al., 2015). A shared feature of these memory architectures is building a graph representation of the observations (in raw observation space or in a latent space) and subsequently use the memory graph to aid planning. In particular, Savinov et al. (2018) uses the memory graph to represent observation connectivity and uses a parametric retrieval network to provide subgoals to the given goal-conditioned task. In Eysenbach et al. (2019), the edges of the memory graph are learned distance estimates based on shaped reward, and temporally extended planning is achieved by using graph search methods on the memory graph to generate a sequence of subgoals. Similarly, both Liu et al. (2020) and Zhu et al. (2020) construct memory graphs with states in the replay buffer, and use graph search methods to perform planning. Liu et al. (2020) further generates hallucinated observation samples based on a context image, allowing zero-shot generalization in novel environments, while Zhu et al. (2020) uses the memory graph for fast value propagation between previously unexperienced state pairs in a reverse-trajectory manner.

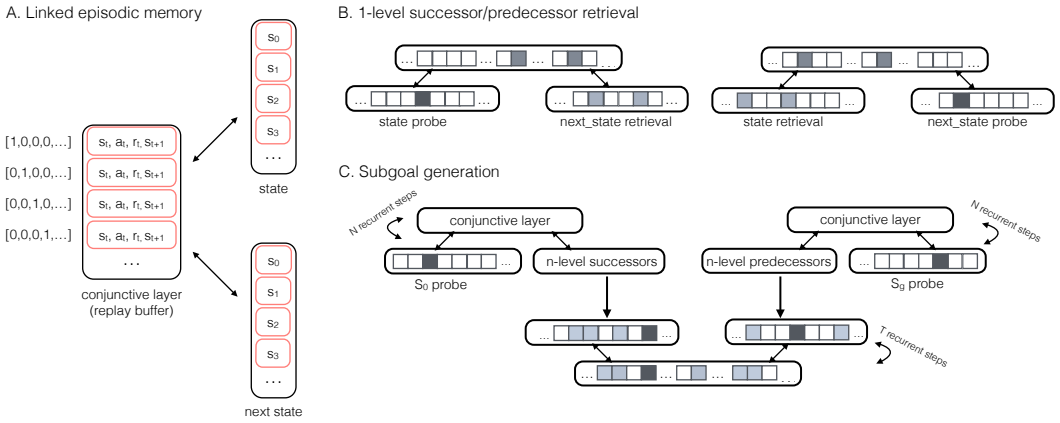

Figure 1: Bidirectionally linked episodic memory. A. The conjunctive layer corresponds to unique transitions in the regular replay buffer or higher-order transitions. Each transition is linked to two visible layers, one marking the state and the other marking the next state. B. At baseline, the memory system performs 1-level successor and predecessor retrieval, when probed with a state or a next state, respectively. C. Plan generation contains two steps. First the given state and goal are used to probe for all the 1-level to n-level successors and predecessors. The n-level successor and predecessor activation are then used as probe states and next states to find a linked transition in the conjunctive layer.

3 METHOD

Linked episodic memory. The (non-parametric) linked episodic memory module adapts the RE-MERGE memory model into an RL setting (Kumaran & McClelland, 2012). The conjunctive layer contains the unique transitions stored in the regular replay buffer and higher-level (multi-step) transitions, with each node of the conjunctive layer corresponding to one such transition (Figure 1A). Because multiple transitions can be stored in the regular replay buffer paired with different goals, the size of the linked episodic memory module can be far smaller than the regular replay buffer. Each node in the conjunctive layer maps to two visible layers, one marking the state in the transition, the other marking the next state in the transition. All three layers use an instance-based onehot coding of the memory content, i.e., for the conjunctive layer the onehot transition code can be used to retrieve the stored transition, for the visible layers the shared onehot state code can be used to retrieve the relevant state. The bidirectional weights connecting the conjunctive layer and the visible layers are set as 1.0 if the corresponding nodes are linked, or 0.0 if otherwise. The weights are adjusted dynamically as new memory traces are added and old ones discarded, to control for capacity. Following Kumaran & McClelland (2012), at each recurrent step t, the net input signal of a given node is a weighted combination of the current input from sending layers and the net input signal at the previous time step:

$$net(t) = \lambda net_{in}(t) + (1 - \lambda) net(t - 1)$$

where $\lambda = 0.2$ for all reported experiments, and $net_{in}(t)$ for the hidden layer given by:

$$net_{in}(t) = \sum_{v} \boldsymbol{w}_{vh} \boldsymbol{a}_{v}(t)$$

where v denotes a sending (visible) layer, $\boldsymbol{w}_{vh}$ denotes the corresponding weight matrix, and $\boldsymbol{a}_{v}(t)$ denotes the activation of that visible layer at time t. $net_{in}(t)$ for the visible layers is given by:

$$net_{in}(t) = \boldsymbol{w}_{hv} \boldsymbol{a}_{h}(t) + estr * ext(t)$$

where $ext(t)$ is any external probe to the visible layers, and $estr$ is a scaling parameter set to 0.2 for all reported results.

The activation at both the conjunctive layer and the visible layers are gated by a temperature-modulated softmax function:

$$a_i = \frac{e^{net_i/\tau}}{\sum_i e^{net_i/\tau}}$$

for each node i in a given layer. τ for all three layers are fixed at 0.04.

At baseline, this linked episodic memory module functions as a 1-level successor or predecessor retrieval system, depending on which one of the two visible layers is probed with an external state (Figure 1B). Building on this baseline successor and predecessor retrieval operation, there are various ways that this linked episodic memory system can be used to perform plan generation. I have experimented a few. In this report I show results from the two-step plan generation method shown in Figure 1C. First, the given state and the goal state are used to probe for all the 1-level to n-level successors and predecessors through N recurrent steps. During this first step, the activation of the layers are binarized so that any positive activation is encoded as 1.0 and the inactivated nodes set to 0.0. The n-level successor activation and predecessor activation are then used as the probes to the state and next state visible layers to check if a potential linked transition exist in the conjunctive layer. During this second step, the softmax activation function is used. If such a link exists in the conjunctive layer, then the activation for the state and next state corresponding to that linked transition will gradually (through T recurrenct steps) have higher activation compared to the other n-level successor/predecessor states. For all reported experiments, $N = 1$ and $T = 4$. To further aid planning, I also dynamically add higher-level transitions in the conjunctive layer. Specifically, if an incoming new transition already exists in the conjunctive layer, then the memory module attempts to find a linked second-order next state for the next state in the incoming transition. The order of the higher-order transitions is capped at 4 steps, so the conjunctive layer can store 1- to 4-step transitions, with 1-step transitions dominating the storage (in the reported experiments, the conjunctive size was set to 2k, while the size of the regular replay buffer set to 10k).

Goal-conditioned DQN. The base agent is a goal-conditioned deep Q network, using a three-layer MLP as its core neural network to perform Q value estimation. For each epoch during training, the agent steps online through an episode until reaching the goal state or until a max horizon of H. The agent has some exploration capacity during training according to a decaying ϵ-schedule, but there is no exploration at test time. At each environment step, with some probability (set to a decaying probability schedule from 0.8 to 0 in the reported experiments) replace the goal with a subgoal sampled from the linked memory module, or re-plan with some probability (set to 0.5) by consulting the memory module with the updated current state. The agent also samples a batch of transitions (batch size=128) from the regular replay buffer to optimize its Q value prediction per each environment step. A target network is used to compute the estimated value of the next state for bootstrapped temporal difference learning and updated periodically to stabilize the optimization.

4 EXPERIMENTS

Task setup. As a proof-of-concept test, I choose to explore the integration of goal-conditioned DQN and the linked episodic memory module in a gridworld navigation domain. I initially set out to use a grid matrix representation for the observation space, including the agent location and the wall obstacles. However, I have not been able to get the vanilla DQN to solve the grid matrix observation representation. Therefore, for the reported experiments, I simplified the task to use the x-y coordinate information in an empty gridworld (with surrounding walls on four sides) as the observation representation space instead. For each new episode, the environment generates a starting location and a goal location and feeds into the agent to collect an action among 4 possible actions (up, down, left, right). An episode ends at the max horizon H or when the agent arrives at the goal state, at which point the agent is rewarded with +1.0.

The key training and test task difference is in the number of steps required if performing optimally. As mentioned above, the generalization behavior of interest is one in the form of transitive inference.

Thus, I constructed training tasks by capping the task to be within X steps. Subsequently at test time, the agent is tested on how its learning can transfer to tasks requiring $>$ X optimal steps. It is worth noting that for this particular task, perfect test-time generalization is achievable using shaped reward such as the euclidean distance between the agent's current location and the goal location. However, the linked episodic memory module is reward- and observation-independent. Even though I focus on this particular navigation task, I hope to explore this idea with the understanding that it may be applicable to a broad range of task specifications.

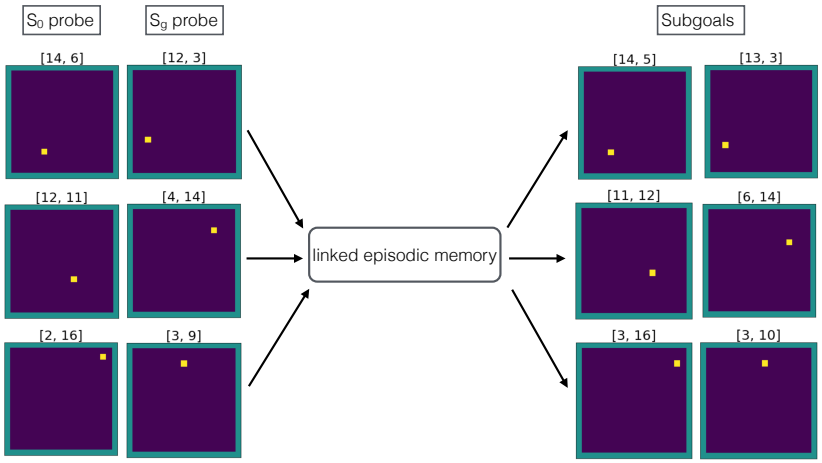

Figure 2: Subgoal generation using the linked episodic memory module.

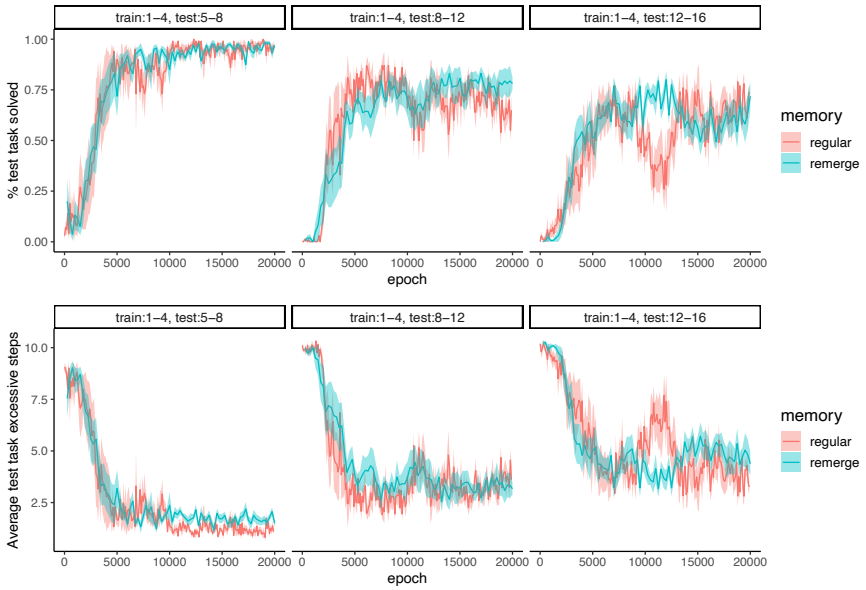

Figure 3: Generalization performance on the test tasks throughout training. The memory module has not shown fruitful improvement yet, and needs more work to scale up. Training tasks are 1-4 steps difficulty, while test tasks are of increasing difficulty, requiring generalization to 5-8 steps (horizon $H = 16$) to 8-12 steps ($H = 20$) to 12-16 steps ($H = 24$). Upper panel, proportion of test tasks solved within horizon. Lower panel, average excessive steps used across test tasks (evaluated against the minimum number of steps required to solve the task optimally).

Results. Figure 2 shows that the linked episodic memory module is capable of generating sensible subgoals given a distant state and goal pair. In practice, it works near perfect especially at a small scale. The agent learns to solve the training tasks perfectly around 5k training epoch, achieving near

100% tasks solved within horizon as well as near 0 excessive steps. Figure 3 shows the tracking of test task performance across training epochs, measured as the proportion of test tasks solved and the number of excessive steps used by the agent (evaluated against the minimum number of steps required to solve the task optimally). Long story short, the current implementation of the integration of plan generation using the linked episodic memory module and the base DQN agent has not shown fruitful improvement over a vanilla DQN with a regular replay buffer. A significant challenge has been to generalize this memory mechanism in larger state spaces while preserving its linking accuracy. The multiplex of hyper-parameters that the memory module adds to the entire system may also have prevented it from working perfectly when scaled up. In the next section, I will discuss some relevant factors that may contribute to efficient planning using the proposed memory mechanism in more detail.

5 DISCUSSION

In this work, I explore how casting the replay buffer as a linked instance-based memory system can bring fruitful generalization in an RL setting. Inspired by human memory models, the goal of the linked episodic memory module is to perform plan generation given a pair of distant probe states, and attempt to link the probe states through a set of recurrent steps to activate potential intermediate subgoal states stored in the memory. The hope is that this observation- and reward-independent memory module will facilitate generalization in the form of transitive inference on goal-conditioned tasks where the goal representation is in the same space as the state representation. As shown in the results above, so far I have not successfully scaled the memory module to bring fruitful improvement in generalization. Below, I discuss some of the factors that require future investigation.

A key challenge is that the memory module and the current method that integrates the memory module with the agent add a whole new set of hyper-parameters, including the temperature parameters τ for the softmax activation functions, the scaling of change to the net activation in memory network layers λ, the scaling of the external probe $estr$, N recurrence to activate all within n-level successors or predecessors (held constant in the reported results, but likely need to vary to ensure planning accuracy for different levels of test task difficulty), T recurrent steps to find linked pair among all activated n-level successors and predecessors, the re-plan probability, and the switch-goal probability at each environment step. Therefore, it is entirely possible that the system is sensitive to these hyper-parameters and would require further tuning. It may also be the case that some of these hyper-parameters need to vary based on task difficulty, and I may consider for them to be learned in future work.

Another important difference between the gridworld transition structure and the conventional transitive inference tasks is that the gridworld transition structure is perfectly inversible (i.e., we have both pairs of [state=s_1, next state=s_2] and [state=s_2, next state=s_1]). In the current implementation, a given state shares the same onehot coding whether at the state visible layer or at the next state visible layer. This means that any state is its own second-order transition next state. At plan generation time, this particular dynamic was especially luring for the bidirectional network to settle in, which hinders further meaningful linking of other transitions. Therefore, I may need to either implement separate coding for states when they appear in the two visible layers, or add some form of inhibition to prevent the repeated inversible transition pattern from intruding the plan generation procedure.

Lastly, the present result only exploit the linked episodic memory module in one specific way. As mentioned, this memory module functions as a 1-level successor/predecessor retrieval system at baseline. Thus, plan generation can take multiple forms by utilizing this baseline operation in different ways. It is possible that there are other forms of recurrent architecture on top of this base recurrent unit that can solve the plan generation problem more efficiently. For example, in my presentation I discussed another form of plan generation. In practice, I found that version to run prohibitively slow and it would require significant optimization, which is why I am showing results from a different way of using the memory model to perform plan generation. To this end, I am in the process of actively experimenting a few alternative implementations.

REFERENCES

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in Neural Information Processing Systems*, 2017-Decem(Nips):5049–5059, 2017. ISSN 10495258.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Search on the Replay Buffer: Bridging Planning and Reinforcement Learning. pp. 1–16, 2019. URL http://arxiv.org/abs/1906.05253.

Samuel J. Gershman and Nathaniel D. Daw. Reinforcement learning and episodic memory in humans and animals: an integrative framework. *Annual Review of Psychology*, 68:101–128, 2017. ISSN 15452085. doi: 10.1146/annurev-psych-122414-033625.

Dharshan Kumaran and James L. McClelland. Generalization through the recurrent interaction of episodic memories: A model of the hippocampal system. *Psychological Review*, 119(3):573–616, 2012. ISSN 0033295X. doi: 10.1037/a0028681.

Máté Lengyel and Peter Dayan. Hippocampal Contributions to Control: The Third Way. *Advances in neural information processing systems, NeuRIPS*, 20:889–896, 2008.

Kara Liu, Thanard Kurutach, Christine Tung, Pieter Abbeel, and Aviv Tamar. Hallucinative Topological Memory for Zero-Shot Visual Planning. 2020. URL http://arxiv.org/abs/2002.12336.

Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. *arXiv*, pp. 1–16, 2018.

Tom Schaul, Dan Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, 2015.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, pp. 1–21, 2016.

Guangxiang Zhu, Zichuan Lin, Guangwen Yang, and Chongjie Zhang. Episodic Reinforcement Learning With Associative Memory. pp. 1–15, 2020.