Modeling Consecutive Task Learning with Task Graph Agendas

Extended Abstract

David Isele, Eric Eaton
University of Pennsylvania
{isele,eeaton}@seas.upenn.edu

Mark Roberts, David W. Aha
Navy Center for Applied Research in AI
Naval Research Laboratory, Washington, DC
{mark.roberts,david.aha}@nrl.navy.mil

ACM Reference Format:
David Isele, Eric Eaton and Mark Roberts, David W. Aha. 2018. Modeling Consecutive Task Learning with Task Graph Agendas. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018,* IFAAMAS, 3 pages.

Recent advances in transfer, multi-task, and lifelong learning have demonstrated that agents can efficiently learn a challenging target task through a curriculum of simpler-to-harder tasks [15]. Yet relatively little work examines how learning can be self-directed, especially when there can be multiple underspecified targets, or when the environment combines their rewards, creating ambiguity.

Transitioning between tasks works best when they are similar [1] and can be done efficiently if easier (but related) tasks are learned first [2, 12]. The *task graph*, introduced for classification and regression problems by Eaton et al. [6] and applied to reinforcement learning (RL) by Svetlik et al. [19], is a general model of inter-task transferability. Task graphs make the relations between different tasks explicit, and provide methods for reasoning over the entire domain. But computing a task graph can be costly.

Instead of measuring transferability directly we use task descriptors, which provide an efficient and clearly defined mechanism to reason about transfer [3, 5, 9, 16, 19]. While task descriptors have been used to model transfer, they have primarily been used to transfer to a *single* target task, although a single target (and often a single source) may not always be appropriate. For example, goal reasoning agents may be tasked with many simultaneous goals [20]. We show that task descriptors can also be used to efficiently construct and navigate a task graph.

Agents may not always be provided a well-defined target task or may be asked to switch between multiple tasks in dynamic or otherwise demanding environments. For example, an agent may be guided to select related tasks to promote generalization [4] or to search through task space to identify a niche where it is most useful. We extend a transfer learning agent with an *agenda*, which is a general mechanism that can handle multiple guiding principles for how the agent should move through the space of tasks.

1 TASK GRAPHS AND AGENDAS

We assume a domain $\mathscr{D}$ of a set of Markov Decision Processes (MDPs) where each MDP can be described by a collection of features. Each task is an MDP: $\langle \mathcal{S}, \mathcal{A}, P, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ is the set of states, and $\mathcal{A}$ is the set of actions that the agent may execute $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a transition function describing the system's dynamics, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function, and $\gamma \in (0, 1]$ is a discount factor. Informally, $\mathscr{D}$ captures the universe of possible tasks an agent can encounter.

We adapt notation from Narvekar et al. [13] to define a task graph, except we assume some attributes (*e.g.*, degrees of freedom, restrictions) can be expressed as *features*. Let $\mathcal{T}^{(t)} \in \mathscr{D}$ be the t^{th} task from domain $\mathscr{D}$. Further, let $\Phi^{(t)} \in \mathbb{Z}^m$ be the task descriptor describing task $\mathcal{T}^{(t)}$, where m is the number of task features. Each feature $\phi_i^{(t)} \in \Phi^{(t)}$ is an element in a discrete set representing a property in $\mathscr{D}$. Using features assumes that task differences can be described directly (*e.g.*, in Minecraft, the room for the task contains lava or not), or indirectly (*e.g.*, the room has more orange pixels indicating a higher temperature).

We assume tasks are embedded in a task space that models the transfer relationships between tasks. Tasks with a high amount of transfer are close to each other in task space, and tasks that have low or negative transfer are far apart. We define the transferability from $\mathcal{T}^{(i)}$ to $\mathcal{T}^{(j)}$ as the change in performance on $\mathcal{T}^{(j)}$ that results from learning with and without transfer from $\mathcal{T}^{(i)}$. We assume that the tasks vary smoothly along a latent manifold that underlies the task space; the task graph represents a discrete approximation of this task manifold.

Let $G = \langle V, E \rangle$ be a graph where the tasks $V \in \mathscr{D}$ are the set of vertices and $E \in V \times V$ is the set of edges. Each edge can be weighted by a (dis)similarity $w(\mathcal{T}^{(i)}, \mathcal{T}^{(j)})$ between tasks. If we assume the distance between tasks is correlated with the distance between descriptors, we can estimate the distance between tasks as $w(\Phi^{(i)}, \Phi^{(j)})$.

Task Graph Agendas generalize the notion of *curricula* in RL where the curriculum is an agenda of source tasks terminating in a target task. Given a set of tasks $\mathscr{T} \subseteq \mathscr{D}$, an *agenda* is a sequence of tasks $\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_k \in \mathscr{T}$ on which an agent trains. A *walk* is the sequence $W = v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k$, whose terms are alternately vertices and edges, such that for $1 \leq i \leq k$ the edge e_i connects the previous vertex to the current vertex, $e_i = v_{i-1} \sim v_i$. An agenda is any valid walk on the task graph, so a curriculum is a special case of an agenda. In the following experiments, we examine mechanisms for an agent to manage its own agenda.

2 EXPERIMENTS

We train agents with different capabilities to optimize reward over both policies *and* tasks in a **search** problem. We then demonstrate how task graphs can be used to more efficiently learn a difficult task in our **agenda** problem. Although we have carried out a wider set of experiments, we report on results from the game of Minecraft

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. André, S. Koenig (eds.), July 10–15, 2018, Stockholm, Sweden. © 2018 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

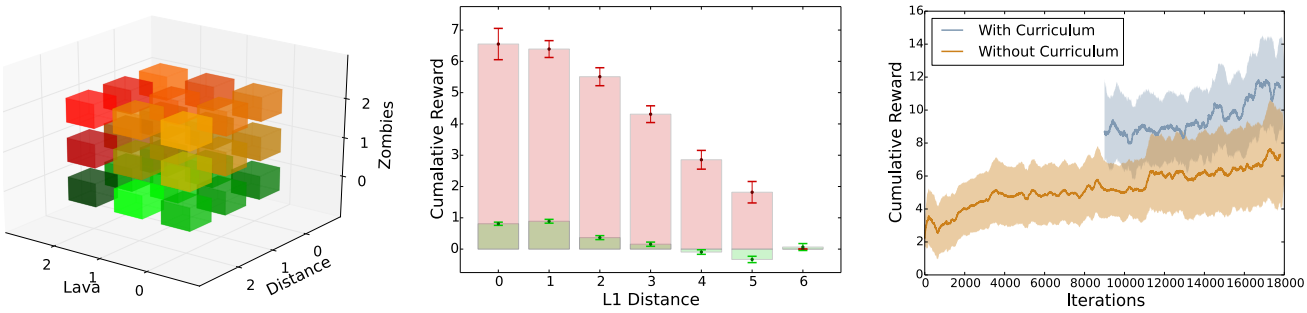

Figure 1: Left: Performance on *search* by task space for the fighter (red) and runner (green); brighter color indicates better performance. Center: Performance on *search* problem by distance from the task where the agent spent the most time. Right: Reward for *agenda*; the top curve offset by training on the earlier task.

[10], which allows for the flexible creation of a variety of scenarios and tasks. In both problems, we consider agents that obtain reward by collecting apples or defeating zombies. The agent is rewarded for collecting apples and injuring zombies, and its goal is to optimize the cumulative reward.

Search: Consider an agent that is trying to maximize its reward when it has the freedom to learn policies *and* select its tasks for applying those policies. We can frame this as a search problem on the task graph where an agent seeks to find the task/policy combination that maximizes reward.

We formalize this as a Semi-Markov Decision Process (SMDP) following the options framework [14, 18]. The MDP as defined earlier forms the basis of the SMDP, while the options framework formalizes temporally extended actions as a Markovian option $\omega \in \Omega$, where ω is triple $\langle \mathcal{I}_\omega, \pi_\omega, \beta_\omega \rangle$, $\mathcal{I}_\omega \subseteq \mathcal{S}$ is the initiation set, π_ω is the intra-option policy, and $\beta_\omega : \mathcal{S} \rightarrow [0, 1]$ is the termination function. For the task graph, options consist of steps on the task graph or training on a task. The state of the SMDP includes the current knowledge of the agent and the position on the task graph. The reward is a transition step cost for switching tasks, or the average cumulative reward from training on the current task.

We present results for two agents: a runner and a fighter. The runner has only motion commands and uses dynamic frame skipping to increase its exploration range [17]. The fighter has motion commands and the ability to attack. Tasks are encoded using the task descriptors $[lava, distance, zombies] \in \{0, 1, 2\}^3$, where higher values in each feature indicate greater challenge. Higher lava and distance increase the number of apples present, and fighting zombies is more rewarding than collecting apples.

Each agent uses cumulative reward to select among neighboring tasks as they walk through the task graph. A greedy epsilon strategy is used to balance the exploration versus exploitation trade-off in the task graph [21]. Agents are allowed to revisit prior tasks. We use a deep Q-network augmented to preserve experiences across tasks [8] as our learning agent.

Figure 1 (left) shows the performance of the runner and fighter overlaid. The runner's performance is in green and the fighter's in red. The runner spent more time on task $[0, 2, 0]$ (no lava, high distance, no zombies) as seen in the bottom row, while the fighter spent the most time on task $[0, 0, 2]$ (two zombies) shown on the top row. Brighter color indicates higher reward, and darker is lower reward. This demonstrates that zombies and lava together complicate the runner's task, while the fighter gets more hits in when there are two zombies. Figure 1 (center) shows how performance varies with distance measured from the task on which the agent spent the most time; the colors match the runner (green) and fighter (red) as above. We found that the agent performs well on the task where it spent the most time, confirming that the search was effective. We also found a clear decrease in performance as a function of distance from the preferred task. Eventually, the runner agent received the most reward on task $[1, 2, 0]$, but learned this task later as a result of its agenda, confirming the hypothesis that the agent cannot immediately jump to the hardest task.

Agenda Generation: Now suppose the runner must now learn to defend itself or consider an agent that has many simultaneous goals and must master tasks to achieve those goals (*e.g.*, a self-motivated agent [7, 20]). Given a desired target task, an agent could create its own agenda through the task graph.

We present results from Minecraft where the agent is rewarded for collecting an apple positioned behind a zombie. The feature dimensions control the amount of lava and the x and z positions of the apple. We use an A3C network [11] as our learning agent. Since performance on only the final task matters, there is no need for the system to preserve ability on prior tasks as it moves through the curriculum.

Figure 1 (right) shows the improvement that results from following an agenda. The agent shown in blue first trains on easier tasks where the apple is closer to the agent. Because the apple is easier to collect, the agent more quickly learns the importance of collecting the apple. The agent that trained directly on the target task (shown in orange) takes longer to discover the apple, and instead focuses exclusively on killing the zombie. At around 11,000 iterations, the agent starts to collect the apple on occasion, but by this time the agent following the curriculum is more accomplished at killing the zombie and proceeding to collect the reward.

ACKNOWLEDGMENTS

Supported by NRL and NREIP internship program.

REFERENCES

[1] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. 2007. Analysis of representations for domain adaptation. *Advances in Neural Information Processing Systems (NIPS)* 19 (2007), 137–144.

[2] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. *Proc. ICML* (2009), 41–48. https://doi.org/10.1145/1553374.1553380

[3] Edwin V Bonilla, Felix V Agakov, and Christopher Williams. 2007. Kernel multi-task learning using task-specific features. *International Conference on Artificial Intelligence and Statistics* (2007), 43–50.

[4] Rich Caruana. 1997. Multitask Learning. *Machine Learning* 28 (1997), 41–75.

[5] Bruno Da Silva, George Konidaris, and Andrew Barto. 2012. Learning parameterized skills. *arXiv:1206.6398* (2012).

[6] Eric Eaton, Marie desJardins, and Terran Lane. 2008. Modeling transfer relationships between learning tasks for improved inductive transfer. In *Joint European Conf. on Mach. Learn. and Knowledge Discovery in Databases*. Springer, 317–332.

[7] Nick Hawes. 2011. A survey of motivation frameworks for intelligent systems. *Art. Intel. J.* 175, 5-6 (April 2011), 1020–1036.

[8] David Isele and Akansel Cosgun. 2018. Selective Experience Replay for Lifelong Learning. *Proc. AAAI* (2018).

[9] David Isele, Mohammad Rostami, and Eric Eaton. 2016. Using task features for zero-shot knowledge transfer in lifelong learning. *Proc. IJCAI* (2016).

[10] Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell. 2016. The Malmo Platform for Artificial Intelligence Experimentation.. In *Proc. IJCAI*. 4246–4247.

[11] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy P Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *Proc. ICML*.

[12] Sanmit Narvekar, Jivko Sinapov, Matteo Leonetti, and Peter Stone. 2016. Source task creation for curriculum learning. In *Proc. AAMAS*. 566–574.

[13] Sanmit Narvekar, Jivko Sinapov, and Peter Stone. 2017. Autonomous Task Sequencing for Customized Curriculum Design in Reinforcement Learning. (2017).

[14] Doina Precup. 2000. Temporal abstraction in reinforcement learning. (2000).

[15] Mark B Ring. 1997. CHILD: A first step towards continual learning. *Machine Learning* 28, 1 (1997), 77–104.

[16] Jivko Sinapov, Sanmit Narvekar, Matteo Leonetti, and Peter Stone. 2015. Learning inter-task transferability in the absence of target task samples. In *Proc. AAMAS*. 725–733.

[17] Aravind Srinivas, Sahil Sharma, and Balaraman Ravindran. 2017. Dynamic Action Repetition For Deep Reinforcement Learning. *Proc. AAAI* (2017).

[18] Richard S Sutton, Doina Precup, and Satinder Singh. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence* 112, 1-2 (1999), 181–211.

[19] Maxwell Svetlik, Matteo Leonetti, Jivko Sinapov, Rishi Shah, Nick Walker, and Peter Stone. 2017. Automatic Curriculum Graph Generation for Reinforcement Learning Agents.. In *Proc. AAAI*. 2590–2596.

[20] S. Vattam, M. Klenk, M. Molineaux, and D.W. Aha. 2013. Breadth of approaches to goal reasoning: A research survey. In *Goal Reasoning: Papers from the ACS Workshop (Technical Report CS-TR-5029)*. College Park, MD: University of Maryland, Department of Computer Science., 222–231.

[21] Joannes Vermorel and Mehryar Mohri. 2005. Multi-armed bandit algorithms and empirical evaluation. In *ECML*, Vol. 3720. Springer, 437–448.