Online Multi-Task Gradient Temporal-Difference Learning

Vishnu Purushothaman Sreenivasan and **Haitham Bou Ammar** and **Eric Eaton**
University of Pennsylvania
Computer and Information Science Department
{visp, haithamb, eeaton}@seas.upenn.edu

Introduction

Reinforcement learning (RL) is an essential tool in designing autonomous systems, yet RL agents often require extensive experience to achieve optimal behavior. This problem is compounded when an RL agent is required to learn policies for different tasks within the same environment or across multiple environments. In such situations, learning task models jointly rather than independently can significantly improve model performance (Wilson et al. 2007; Lazaric and Ghavamzadeh 2010). While this approach, known as *multi-task learning* (MTL), has achieved impressive performance, the performance comes at a high computational cost when learning new tasks or when updating previously learned models. Recent work (Ruvolo and Eaton 2013) in the supervised setting has shown that online MTL can achieve nearly identical performance to batch MTL with large computational speedups. Building upon this approach, which is known as the Efficient Lifelong Learning Algorithm (ELLA), we develop an online MTL formulation of model-based gradient temporal-difference (GTD) reinforcement learning (Sutton, Szepesvári, and Maei 2008). We call the proposed algorithm GTD-ELLA. Our approach enables an autonomous RL agent to accumulate knowledge over its lifetime and efficiently share this knowledge between tasks to accelerate learning. Rather than learning a policy for an RL task *tabula rasa*, as in standard GTD, our approach rapidly learns a high performance policy by building upon the agent's previously learned knowledge. Our preliminary results on controlling different mountain car tasks demonstrates that GTD-ELLA significantly improves learning over standard GTD RL.

Reinforcement Learning

In a reinforcement learning problem, an agent must decide how to sequentially select actions to maximize its expected return. Such problems are typically formalized as a Markov decision process (MDP) $\langle \mathcal{X}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where $\mathcal{X}$ is the (potentially infinite) set of states, $\mathcal{A}$ is the set of possible actions that the agent may execute, $\mathcal{P} : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \to [0, 1]$ is a state transition probability function describing the task dynamics, $\mathcal{R} : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \to \mathbb{R}$ is the reward function measuring the performance of the agent, and $\gamma \in [0, 1)$ is the discount factor. A policy $\pi : \mathcal{X} \times \mathcal{A} \to [0, 1]$ specifies a probability distribution over state-action pairs, where $\pi(\boldsymbol{x}, \boldsymbol{a})$ represents the probability of selecting action $\boldsymbol{a}$ in state $\boldsymbol{x}$. The goal of an RL agent is to find an optimal policy $\pi^\star$ that maximizes the total expected discounted reward.

Gradient Temporal-Difference Learning

In GTD (Sutton, Szepesvári, and Maei 2008), the value function is approximated by a linear combination of a parameter vector $\boldsymbol{\theta} \in \mathbb{R}^n$ and a set of state basis functions $\boldsymbol{\Phi} : \mathcal{X} \to \mathbb{R}^n$. In the independently and identically distributed (i.i.d.) formulation, states $\boldsymbol{x}_k$ are assumed to have been generated according to some distribution. Applying an action $\boldsymbol{a}_k$ in a state $\boldsymbol{x}_k$ generates a corresponding successor state $\boldsymbol{x}'_k$ and a reward $\mathcal{R}_k$. The value function is then estimated from the set $\{(\boldsymbol{\Phi}(\boldsymbol{x}_k), \boldsymbol{\Phi}(\boldsymbol{x}'_k), \mathcal{R}_k)\}_{k=1,2,\ldots}$. Let $\boldsymbol{\Phi} = \boldsymbol{\Phi}(\boldsymbol{x}_k)$ and $\boldsymbol{\Phi}' = \boldsymbol{\Phi}(\boldsymbol{x}'_k)$. GTD minimizes the L_2 norm of the temporal-difference error:

$$\mathcal{J}(\boldsymbol{\theta}) = \mathbb{E}\left[\delta\boldsymbol{\Phi}\right]^\mathsf{T} \mathbb{E}\left[\delta\boldsymbol{\Phi}\right] \tag{1}$$

by following the gradient of the objective function:

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) = -2\mathbb{E}\left[\boldsymbol{\Phi}(\boldsymbol{\Phi} - \gamma\boldsymbol{\Phi}')^\mathsf{T}\right]^\mathsf{T} \mathbb{E}\left[\delta\boldsymbol{\Phi}\right] \quad , \tag{2}$$

where $\delta = \mathcal{R}_k + \gamma\boldsymbol{\theta}^\mathsf{T}\boldsymbol{\Phi}(\boldsymbol{x}'_k) - \boldsymbol{\theta}^\mathsf{T}\boldsymbol{\Phi}(\boldsymbol{x}_k)$.

Problem Definition

We focus on the online MTL setting in which the agent is required to learn a series of RL tasks $\mathcal{Z}^{(1)}, \ldots, \mathcal{Z}^{(T_{\max})}$ over its lifetime. Each task t is an MDP $\mathcal{Z}^{(t)} = \langle \mathcal{X}^{(t)}, \mathcal{A}^{(t)}, \mathcal{P}^{(t)}, \mathcal{R}^{(t)}, \gamma^{(t)} \rangle$. The agent learns the tasks consecutively, acquiring multiple trajectories within each task before moving to the next. The tasks may be interleaved, providing the agent the opportunity to revisit earlier tasks for further experience, but the agent has no control over the task order. We assume that *a priori* the agent does not know the total number of tasks $T_{\max}$, their distribution, or the task order. The goal is to learn a set of *optimal* value functions $\boldsymbol{V}^\star = \left\{V^\star_{\boldsymbol{\theta}^{(1)}}, \ldots, V^\star_{\boldsymbol{\theta}^{(T_{\max})}}\right\}$ with corresponding parameters $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(T_{\max})}$.[1]

[1] In this work, we consider the model-based RL setting. Extending GTD-ELLA to the model-free scenario is fairly straightforward, and we leave this to future work.

Approach

Our approach maintains a library of k latent components $\boldsymbol{L} \in \mathbb{R}^{d \times k}$, which is shared among all tasks and forms a basis for the task models. The parameter vectors can then be represented as $\boldsymbol{\theta}^{(t)} = \boldsymbol{L}\boldsymbol{s}^{(t)}$, where $\boldsymbol{s}^{(t)} \in \mathbb{R}^k$ are the coefficients over the basis $\boldsymbol{L}$ for task t. The $\boldsymbol{s}^{(t)}$'s are encouraged to be sparse to ensure that each basis component represents a cohesive chunk of knowledge. Given T tasks, we can represent the MTL problem via the following objective function:

$$e_T(\boldsymbol{L}) = \frac{1}{T}\sum_{t=1}^{T} \min_{\boldsymbol{s}^{(t)}} \left[\mathcal{J}\left(\boldsymbol{\theta}^{(t)}\right) + \mu \left\|\mathbf{s}^{(t)}\right\|_1 \right] + \lambda \|\boldsymbol{L}\|_{\mathsf{F}}^2 \ , \quad (3)$$

where the L_1 norm of $\boldsymbol{s}^{(t)}$ is used to approximate the true vector sparsity. Solving Equation 3 is computationally expensive due to two inefficiencies: *a*) the explicit dependence of Equation 3 on *all* available trajectories, and *b*) the exhaustive evaluation of a single candidate $\boldsymbol{L}$ that requires the optimization of all $\boldsymbol{s}^{(t)}$'s.

Eliminating Dependence On All Trajectories

As in ELLA, to eliminate the inefficiency related to the dependence on all available trajectories, we approximate Equation 3 by performing a second-order Taylor expansion of $\mathcal{J}\left(\boldsymbol{\theta}^{(t)}\right)$ around the optimal single-task solution $\boldsymbol{\alpha}^{(t)}$:

$$\hat{e}_T(\boldsymbol{L}) = \frac{1}{T}\sum_{t=1}^{T} \min_{\boldsymbol{s}^{(t)}} \left\{ \left\|\boldsymbol{\alpha}^{(t)} - \boldsymbol{L}\boldsymbol{s}^{(t)}\right\|_{\boldsymbol{\Gamma}^{(t)}}^2 + \mu \left\|\boldsymbol{s}^{(t)}\right\|_1 \right\} + \lambda ||\boldsymbol{L}||_{\mathsf{F}}^2 \ ,$$

where

$$\boldsymbol{\alpha}^{(t)} = \arg\min_{\boldsymbol{\theta}} \mathcal{J}\left(\boldsymbol{\theta}^{(t)}\right) \quad \boldsymbol{\Gamma}^{(t)} = \nabla_{\boldsymbol{\theta}^{(t)},\boldsymbol{\theta}^{(t)}} \mathcal{J}\left(\boldsymbol{\theta}^{(t)}\right) \ .$$

Solving $\arg\min_{\boldsymbol{\theta}} \mathcal{J}\left(\boldsymbol{\theta}^{(t)}\right)$ is typically difficult in RL settings, requiring a substantial amount of experience. We remedy this problem by following the gradient of $\mathcal{J}\left(\boldsymbol{\theta}^{(t)}\right)$. Crucially, we have removed the dependence of the optimization on all trajectories.

Eliminating the Reoptimization of Other Tasks

The second inefficiency arises from the need to reoptimize all learned models by recomputing the $\boldsymbol{s}^{(t)}$'s when learning each new task. We modify Equation 3 to eliminate this minimization over all $\boldsymbol{s}^{(t)}$'s by optimizing each task-specific projection only when training on task t, without updating them when training on other tasks. This leads to the following update equations for learning the model for each new task t, which approximates the result of Equation 3:

$$\boldsymbol{s}^{(t)} \leftarrow \arg\min_{\boldsymbol{s}^{(t)}} l\left(\boldsymbol{L}_m, \boldsymbol{s}^{(t)}, \boldsymbol{\alpha}^{(t)}, \boldsymbol{\Gamma}^{(t)}\right) \quad (4)$$

$$\boldsymbol{L}_{m+1} \leftarrow \arg\min_{\boldsymbol{L}} \frac{1}{T}\sum_{t=1}^{T} l\left(\boldsymbol{L}, \boldsymbol{s}^{(t)}, \boldsymbol{\alpha}^{(t)}, \boldsymbol{\Gamma}^{(t)}\right) + \lambda \|\boldsymbol{L}\|_{\mathsf{F}}^2 \quad (5)$$

where $l\left(\boldsymbol{L}, \boldsymbol{s}^{(t)}, \boldsymbol{\alpha}^{(t)}, \boldsymbol{\Gamma}^{(t)}\right) = \mu||\boldsymbol{s}||_1 + ||\boldsymbol{\alpha} - \boldsymbol{L}\boldsymbol{s}||_{\boldsymbol{\Gamma}}^2$ and $\boldsymbol{L}_m$ corresponds to the value of the latent basis at the m^{th} iteration. This formulation can be solved efficiently (Ruvolo and Eaton 2013), and together, Equations 4 and 5 constitute the core of GTD-ELLA.

Preliminary Results

We evaluated GTD-ELLA on multiple tasks from the mountain car (MC) domain. In MC, the state is given by the position and the velocity of the car, which was represented by 6 radial basis functions that were linearly spaced across both dimensions. The position was bounded between -1.2 and 0.6, while the velocity was bounded between -0.07 and 0.07. Rewards of -1 were given in all states, with exception of the goal which gave a reward of 0. We generated 75 tasks by randomizing the valley's slope, resulting in a shift in the valley's position. These shifts were critical to ensure that the optimal solution for one task is suboptimal in another. We trained GTD-ELLA on different numbers of tasks to learn $\boldsymbol{L}$, observing either 10, 30, or 50 tasks to learn the latent bases. Evaluation was then conducted on 25 unobserved MC tasks using either GTD-ELLA or standard GTD(0).

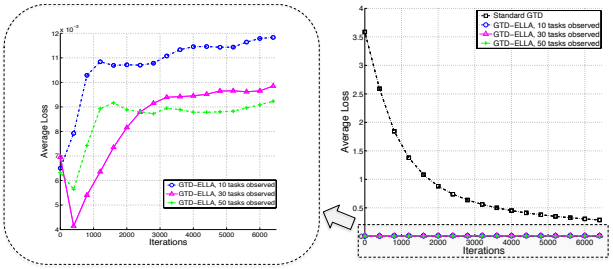

Figure 1: Comparison of GTD-ELLA versus standard GTD(0), showing that GTD-ELLA attains significantly lower average loss over 25 test tasks. The left graph zooms in on the cluttered region of the graph on the right.

Figure 1 shows that GTD-ELLA significantly improves RL performance when training on new tasks. Further, as the agent learns more tasks, its overall performance improves. Although successful, this work is still in progress. In the future, we plan to extend the approach to model-free RL, as well as allow for more differences between the tasks.

Acknowledgements

This work was partially supported by ONR grant #N00014-11-1-0139 and AFOSR grant #FA8750-14-1-0069.

References

Lazaric, A., and Ghavamzadeh, M. 2010. Bayesian multi-task reinforcement learning. In *Proceedings of the 27th International Conference on Machine Learning*, 599–606.

Ruvolo, P., and Eaton, E. 2013. ELLA: An efficient lifelong learning algorithm. In *Proceedings of the 30th International Conference on Machine Learning*, 507–515.

Sutton, R. S.; Szepesvári, C.; and Maei, H. R. 2008. A convergent O(n) algorithm for off-policy temporal-difference learning with linear function approximation. In *Neural Information Processing Systems* 21, 1609–1616.

Wilson, A.; Fern, A.; Ray, S.; and Tadepalli, P. 2007. Multi-task reinforcement learning: A hierarchical Bayesian approach. In *Proceedings of the 24th International Conference on Machine Learning*, 1015–1022.