Escaping Groundhog Day

Abstract

The dominant approaches to reinforcement learning rely on a fixed state-action space and reward function that the agent is trying to maximize. During training, the agent is repeatedly reset to a predefined initial state or set of initial states. For example, in the classic RL Mountain Car domain, the agent starts at some point in the valley, continues until it reaches the top of the valley and then resets to somewhere else in the same valley. Learning in this regime is akin to the learning problem faced by Bill Murray in the 1993 movie *Groundhog Day* in which he repeatedly relives the same day, until he discovers the optimal policy and escapes to the next day. In a more realistic formulation for an RL agent, every day is a new day that may have similarities to the previous day, but the agent never encounters the same state twice. This formulation is a natural fit for robotics problems in which a robot is placed in a room in which it has never previously been, but has seen similar rooms with similar objects in the past. We formalize this problem as learning to act in a *domain*. A domain is defined by the tuple (X, A, P), where X is a state representation, A is an action set, and P is a probability distribution of MDPs with different state spaces, reward functions, and transition dynamics, but the same state representation X and action set A. The agent samples an MDP from some unknown distribution of related MDPs and must learn to behave well under the distribution of MDPs. The agent observes samples from the MDP distribution and at test time is evaluated on a new set of MDP samples drawn from the same distribution, focusing the evaluation on the agent's ability to generalize.

Keywords: Meta-learning, transfer learning, learning to plan,

1 Introduction

The dominant approaches to reinforcement learning rely on a fixed state-action space and reward function that the agent is trying to maximize. Often during training, the agent is reset to a predefined initial state or set of initial states. For example, in the classic RL Mountain Car problem [2], the agent is tasked with driving an underpowered car out of a valley and the agent learns to do so over a series of episodes. At the start of an episode, the agent is placed in some random location in the valley; the episode ends when the agent drives out of the valley. After the episode ends a new episode begins and learning continues. Learning in this problem, and many other RL problems, is akin to the learning problem faced by Bill Murray in the 1993 movie *Groundhog Day* in which he repeatedly—and unrealistically—relives the same day, until he discovers the optimal policy and escapes to the next day. In a more realistic formulation for an RL agent, every day is a new day that may have similarities to the previous day, but the agent never encounters the same state twice. This formulation is a natural fit for robotics problems in which a robot is placed in a room it has never previously seen, but has seen similar rooms with similar objects in the past.

We formalize this problem as learning to act in a *domain*. A domain is defined by the tuple (X, A, P), where X is a state representation, A is an action set, and P is a probability distribution of MDPs with different state spaces, reward functions, and transition dynamics, but the same state representation X and action set A. In principle, P could have such a large variation that nothing learned in one MDP drawn from it is useful for another; however, we are interested in domains in which there strong commonalities. For example, in the real world, no two rooms may be the same, but physical and social constraints entail similarity in the mechanics of objects, such as light switches and door knobs; we would like are agent to learn in this type of scenario.

There are two types of learning problems that can be addressed in this setting: (1) learning to plan and (2) learning to learn. In learning to plan, the agent always knows the transition dynamics for each MDP; however, the agent can exploit its knowledge from solving previous MDPs from the distribution to more efficiently find good solutions in new MDPs. In learning to learn, the agent does not know the transition dynamics for each MDP, but can use knowledge from learning in previous MDPs to learn a portable model or bias its exploration through the state space.

By formalizing the learning problem in this way, we are able to focus research on generalization to states never previously seen rather than overfitting to a single set of states and reward function. Focusing on generalization suggests a set of evaluation metrics in which at training time the agent learns from i.i.d. samples from the MDP distribution and at test time is evaluated on a new set of i.i.d. MDP samples drawn from the same distribution. In this work, we formally describe our learning problem and evaluation metrics for it, present two approaches that address it, and review how existing work fits in with our problem formulation.

2 Problem Definition

Classic reinforcement learning (RL) problems are formulated as a trying to learn a (near-)optimal policy—a mapping from states to actions—in a Markov decision process (MDP). An MDP is defined by the tuple (S, A, T, R), where S is the state space, a set of states in which the agent can make decisions; A is the set of actions the agent can take; $T(s'|s, a)$ is the transition dynamics, which specifies the probability of transitioning to state s' after taking action a in state s; and $R(s, a, s')$ is the reward function, which specifies the reward received by the agent for taking action a in state s and then transitioning to state s'. Although there are other formulations, a policy $\pi : S \rightarrow A$ is typically considered optimal if following it maximizes the expected discounted future reward from each state $E\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s, \pi\right]$, where $\gamma \in [0, 1]$ is a discount factor specifying how much the agent prefers immediate rewards over more distant rewards; r_t is the reward received at time t; and s is the state at time zero. Learning is performed using observations of interactions with the MDP environment. Specifically, the agent is placed in some initial state of the MDP, acts for some fixed amount of time, or until they research a terminal state of the MDP, and then is reset to another initial state of the MDP to repeat the process. The sequence of states and action between an initial state and being reset to another initial state define an *episode* of learning.

We extend this typical learning problem to learning how to act overtime in a *domain*. A domain is defined by the tuple (X, A, P), where X is a state representation, A is an action set, and P is a probability distribution of MDPs with different state spaces, reward functions, and transition dynamics, but the same state representation X and action set A. There are multiple different forms that a state representation can take. In many classic RL problems, state representations are a fixed vector of numeric state variables. Since we are interested in learning over a distribution of MDPs, this representation may be too limiting since different MDPs may have different numbers of state variables. For example, in robotics domains, it is typical for the number of objects and people in an environment to vary between problems. Instead, representations like object-oriented Markov decision processes (OO-MDPs) [1] may be more appropriate for expressing a range of environments. In an OO-MDP, states are represented by a collection of objects, each that belong to a specific object class and are defined by their own state variables. As a consequence, a disperse set of environments with different numbers of different objects can all be unified under a common representation.

Learning in a domain is performed using observations from a sequence of MDPs drawn from the domain's MDP distribution. That is, first an MDP $m \sim P$ is drawn from the distribution and the agent is placed in an initial state $s^m \in S^m$, where S^m is the state space of MDP m. The agent is then allowed to learn and act for some fixed amount of time in m, after which the process will repeat in a new MDP drawn from P. In the extreme case, the agent will only be allowed one learning episode in each MDP sample.

Depending on what the agent is allowed to know, different variants of this learning problem arise. If the agent knows the transition dynamics and reward function for each MDP in the domain, then learning problem is a *learning to plan* problem. If the agent does not know either the transition dynamics or reward function, then it is a *learning to learn* problem.

- overall problem definition
- OO-MDP plug for representation
- planning instantiation (true transition dynamics provided)
- RL instantiation (true transition dynamics unknown)
- evaluation metrics

3 Approaches

3.1 Goal-directed Action Priors (Affordances)

3.2 Meta-RL

4 Related Work

4.1 Model-based RL

Learning a portable model (or model prior) over the distribution of MDPs is extremely powerful because it allows the agent plan out solutions to each novel situation without much needed exploration. Only tells part of the story, because it defers to a good planner being available, which may be difficult for complex domains. Therefore, model-learning transitions into the learning to plan problem.

4.2 Bayesian RL

Directly reasons over the distribution of MDPs and performs Bayes optimal learning. However, planning in this paradigm is extremely complex and does not scale well. Like Model-based RL could substantially benefit from learning to plan.

4.3 Transfer Learning

Most RL transfer work focuses on transferring knowledge from a single or set of source tasks to a single target task. Here we are interested in generalized performance across a distribution of tasks.

4.4 Skill learning

Most option learning work is limited to the same state space with different reward functions and there have been some negative results for learning with options. Current open problem with some success is learning portable options. State space abstraction+skill learning promising.

5 Conclusion

References

[1] C. Diuk, A. Cohen, and M.L. Littman. An object-oriented representation for efficient reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, pages 240–247. ACM, 2008.

[2] S.P. Singh and R.S. Sutton. Reinforcement learning with replacing eligibility traces. *Machine learning*, 22(1):123–158, 1996.