2P4-3

ドメイン非依存強化学習エージェントのための冗長なアクションの検出手法

陣内 佑 *1 Alex Fukunaga*2
Yuu Jinnai Alex Fukunaga

*1*2東京大学大学院総合文化研究科 Graduate School of Arts and Sciences, The Univeristy of Tokyo
*1理化学研究所革新知能統合研究センター Center for Advanced Intelligence Project, RIKEN

One of the goal of reinforcement learning research is to build a single architecture which can successfully learn control policies in a wide range of different environments. However, the downside of such a multi-purpose architecture is that it often adds a redundancy to solve each problem compared to a problem specific architecture. Specifically, an agent has to be able to apply all actions required in any of the environments. Previous works tackle this problem by transfer learning and weight actions successful in previous tasks using action prior. In this paper, we propose a method to learn actions which tend to be successful in a present task.

1. Introduction

One of the major challenge of artificial intelligence is to develop agents capable of solving a variety of problems and domains without domain-specific engineering.

Several frameworks have proposed to evaluate the general competency of the agents such as the General Game Playing [Genesereth 05], the General Video Game Playing [Levine 13], International Planning Competition [Coles 12], Reinforcement Learning Competition [Whiteson 10].

The Arcade Learning Environment (ALE) [Bellemare 13] is widely used as a benchmark platform for domain-independent agents based on reinforcement learning and classic planning.

To solve a wide range of unknown problems, an agent has to be capable of applying a variety of actions. However, not all actions are needed to solve a specific problem. One of the difficulity a domain-independent agent faces on unknown environment is that it has no clue of which action is relevant to compete in the environment. Pruning redundant actions is investigated in classic planning with transparent domain model which a precise dynamics of the world is given to the agent at the beginning [Taylor 93]. However, such methods require a domain model and are not applicable to a black-box domain where an agent has no access to the description of the domain. Recently [Jinnai 17] proposed a method to prune dominated action sequence in online black-box planning, and successfully improved the performance of black-box planning in the ALE [Jinnai 17].

On the other hand, there are no method to prune redundant actions in the action set of the domain-independent agent in reinforcement learning to our knowledge. Previous research sidestepped this issue by either using all the available actions or using an action set hand-coded for each task, which is contrary to the goal of domain-independent agent.

In this paper, we investigate duplicate avoidance strategies in reinforcement learning settings for black-box domains. Specifically, we seek to eliminate actions (and sequences of actions) which are dominated by others (and lead to duplicate states). For example, in the ALE, which simulates an Atari 2600 game machine, 18 actions are always available (the joystick has 9 states – up/down/left/right/4 diagonals/"neutral", and the "fire" button has 2 states, 9x2=18). Previous work in search-based planning for the ALE treats all 18 actions as applicable at every state [Lipovetzky 15, Shleyfman 16]. However, in any particular game, many of these 18 actions are dominated ("useless"): First, some actions are trivially dominated because they are completely ignored, or the program always treats them as being equivalent to other actions (e.g., in some games, the state of the "fire" button is irrelevant). Second, some actions are *conditionally* dominated because, in a given context, the action results in the same state as another action (e.g., in a maze-based game, if the agent is stuck against a wall to the left, then the "left" action is useless because (in some games) it results in the same state as "no action"). More generally, *sequences of actions* can be useless. For example, some actions can have *cooldown periods*, i.e, after action a is used, executing a again has no effect for the next t seconds (e.g. firing missiles in shooting games).

If the domain is transparent (e.g. PDDL is available to the agent), dominated actions can be detected trivially by analyzing domain models, and dominated action sequences can be pruned using methods such as duplicate action sequence detection [Taylor 93], symmetry detection [Fox 99, Pochter 11], and strong stubborn sets [Wehrle 13]. However, in black-box domain, pruning dominated actions and action sequences is nontrivial because we can not be certain whether an action is truly dominated, or merely appears to be dominated due to the context provided by the current game state.

2. Background

The reinforcement learning (RL) problem an agent interacts with an unknown environment and attempts to maximize a reward [Kaelbling 96]. The agent and environment interact at each of a sequence of discrete time steps, $t = 0, 1, 2, 3, ...$ At each time step t, the agent recieves some representation of the environment's state, $s_t \in S$, where S is the set of possible states, and on that basis selects an action $a_t \in A$, where A is the set of actions available to the agent. An agent may have prior knowledge of $A(s_t)$, the set of actions valid at state s_t. One time step later, in part as a consequence of its action, the agent receives a numerical reward $r_{t+1} \in R$ and finds itself in a new state s_{t+1}.

In particular, a reinforcement learning task that satisfies the Markov property is called a Markov decision process (MDP). If

the state and action spaces are finite, then it is called a finite MDP. A finite MDP model is defined by its state and action sets and by the one-step dynamics of the environment. The state-transition probabilities are: $p(s'|s,a) \approx Prs_{t+1} = s'|s_t = s, a_t = a$

One of the goal of reinforcement learning is to build a single architecture applicable to a wide range of environments. To achieve this, an agent needs to equipped with In reinforcement learning, a major approach to cope with the large action set is *action prior*.

2.1 Action Prior

Previous work addressed the problem of large action sets by transfer learning. An action prior is one way to use knowledge acquired in other tasks or domains [Taylor 09]. An agent counts how many times the action was chosen for the state in previous trajectories. If an action is chosen more frequently, then use the action in the present task more often, on the assumption that the optimal action in previous tasks is more likely to be useful in the present task. [Sherstov 05] formalized a transfer learning from a task to a task within a domain.

2.2 Dominated Action Sequence Pruning

Pruning dominated actions or irrelevant actions in the action set is a common practice in classic planning.

Taylor and Korf ([Taylor 93]) proposed a standard method for dominated action sequence elimination in deterministic domains with transparent models, based on the following criterion for determining dominance: An action sequence S1 dominates S2 if and only if (1) $Cost(S1) \leq Cost(S2)$, (2) $S1$ is applicable whenever $S2$ is applicable, and (3) Applying $S1$ and $S2$ to state s always result in the same resulting state s'.

However, in model-free environment, pruning dominated actions and action sequences is nontrivial because we can not be certain whether an action is truly dominated, or merely appears to be dominated due to the context provided by the current game state.

Recently [Jinnai 17] proposed Dominated Action Sequence Avoidance (DASA), a method to learn dominated action sequences dominated in classical planning in black-box environment (model-free environment). DASA learns dominated action sequences in the course of the planning using previously explored state transitions. DASA successfully reduced the number of node generation in Arcade Learning Environment on online planning setting [Bellemare 13].

3. Dominated Action Sequence Avoidance for RL

Previous work on RL relies on reward to evaluate the effectiveness of actions. However, reward by itself is not sufficient to solve duplicated or dominated actions problem. In fact, many previous works sidestep this issue by using either entire action set or hand-coded action set for each task.

We propose an approach which applies Dominated Action Sequence Avoidance to RL so that an agent can avoid actions which are likely to generate duplicated states. The pseudocode of the framework is given in Algorithm 1. An agent periodically runs tree search to learn dominated actions. Based on that, $\theta(a)$ is set. During RL episodes, an agent applies action a with probability of $\theta(a)\pi(s,a)$ instead of $\pi(s,a)$ so that it avoids dominated actions in addition to actions with lesser reward.

Algorithm 1: Reinforcement Learning with Action pruning framework

Require: pruning method D and a pruning condition
Initialize $\pi(s,a)$ arbitrarily;
Initialize $\theta(a) = 0(a \in A)$;
for *every episode* $k = 1...K$ **do**
  if $Trigger$ **then**
    $\theta(a) \leftarrow D$;
  Choose initial state s;
  repeat
    $a \leftarrow a \in A$ with probability of $\theta(a)\pi(s,a)$;
    Take action a, observe r, s';
    Update $\pi(s,a)$;
    $s \leftarrow s'$;
  until s *is terminal*;
return $\pi(s,a)$;

4. Conclusions and Future work

Previous wors in RL coped with large action set problem by pruning actions which tend to give lesser reward. This approach does not prune duplicated or dominated actions. In this paper, we proposed a method to avoid dominated action sequences in RL. Our method seeks to avoid actions which tend to generate duplicated states in addition to actions with lesser reward.

We are currently evaluating the proposed method in a SARSA [Sutton 98] RL-agent in the Arcade Learning Environment.

References

[Bellemare 13] Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M.: The Arcade Learning Environment: An Evaluation Platform for General Agents, *Journal of Artificial Intelligence Research*, Vol. 47, pp. 253–279 (2013)

[Coles 12] Coles, A., Coles, A., Olaya, A. G., Jiménez, S., López, C. L., Sanner, S., and Yoon, S.: A survey of the seventh international planning competition, *AI Magazine*, Vol. 33, No. 1, pp. 83–88 (2012)

[Fox 99] Fox, M. and Long, D.: The detection and exploitation of symmetry in planning problems, in *Proc. IJCAI*, Vol. 2, pp. 956–961 (1999)

[Genesereth 05] Genesereth, M., Love, N., and Pell, B.: General game playing: Overview of the AAAI competition, *AI magazine*, Vol. 26, No. 2, p. 62 (2005)

[Jinnai 17] Jinnai, Y. and Fukunaga, A.: Learning to Prune Dominated Action Sequences in Online Black-box Domain, in *Proc. AAAI* (2017)

[Kaelbling 96] Kaelbling, L. P., Littman, M. L., and Moore, A. W.: Reinforcement learning: A survey, *Journal of artificial intelligence research*, Vol. 4, pp. 237–285 (1996)

[Levine 13] Levine, J., Congdon, C. B., Ebner, M., Kendall, G., Lucas, S. M., Miikkulainen, R., Schaul, T., and Thompson, T.: General video game playing (2013)

[Lipovetzky 15] Lipovetzky, N., Ramirez, M., and Geffner, H.: Classical planning with simulators: Results on the Atari video games, in *Proc. IJCAI*, pp. 1610–1616 (2015)

[Pochter 11] Pochter, N., Zohar, A., and Rosenschein, J. S.: Exploiting Problem Symmetries in State-Based Planners, in *Proc. AAAI*, pp. 1004–1009 (2011)

[Sherstov 05] Sherstov, A. A. and Stone, P.: Improving action selection in MDP's via knowledge transfer, in *Proc. AAAI*, Vol. 5, pp. 1024–1029 (2005)

[Shleyfman 16] Shleyfman, A., Tuisov, A., and Domshlak, C.: Blind Search for Atari-Like Online Planning Revisited, in *Proc. IJCAI*, pp. 3251–3257 (2016)

[Sutton 98] Sutton, R. S. and Barto, A. G.: *Reinforcement learning: An introduction*, Vol. 1, MIT press Cambridge (1998)

[Taylor 93] Taylor, L. A. and Korf, R. E.: Pruning Duplicate Nodes in Depth-First Search, in *Proc. AAAI*, pp. 756–761 (1993)

[Taylor 09] Taylor, M. E. and Stone, P.: Transfer learning for reinforcement learning domains: A survey, *Journal of Machine Learning Research*, Vol. 10, No. Jul, pp. 1633–1685 (2009)

[Wehrle 13] Wehrle, M., Helmert, M., Alkhazraji, Y., and Mattmüller, R.: The Relative Pruning Power of Strong Stubborn Sets and Expansion Core, *Proc. ICAPS*, pp. 1–9 (2013)

[Whiteson 10] Whiteson, S., Tanner, B., White, A., et al.: The reinforcement learning competitions, *AI Magazine*, Vol. 31, No. 2, pp. 81–94 (2010)