Automatic Subgoals Discovering Method With State Abstraction In Reinforcement Learning

Marek Lapko MAREK.LAPKO@GMAIL.COM
Rudolf Jaksa JAKSA@NEURON.TUKE.SK
Peter Sincak PETER.SINCAK@TUKE.SK
Department of Cybernetics and Artificial Intelligence, FEI, Technical University of Koice, Slovakia

Abstract

In this paper, we present an algorithm that automatically discovers *all* subgoals in reinforcement learning in a simple benchmark navigation task. The other advantage is that it considers an *state abstraction* for a subgoal representation. The aim of discovering subgoals is to speed up learning by allowing transfer of knowledge on various levels of abstraction.

1. Introduction

Reinforcement learning has proven to be a powerful tool in tasks, where (almost) all information we can provide learning agent is a goal to reach. As the agent is provided just with a little information, it takes a long time to learn, especially in the case when the agent is a real robot. One way, how we could speed up the learning, is to reuse the knowledge from previously learned tasks. The knowledge transfer in various levels of temporal abstraction allows us to transfer and learn how to use even parts of knowledge in a new task. In this paper, we focus on the problem, how to find subgoals in a task. These subgoals divide the task into the parts - subtasks. Every such a task could be assigned to an option (Sutton et al., 1999), which we can see as an action on the higher level of abstraction and which is easy to transfer (Lapko & Jaksa, 2009).

When discovering subgoals, we are looking for states, which we can see as a "funnels" in the state space (McGovern & Barto, 2001). In general, the funnel is a state (or a region of states), that lies on the path to the goal and it is critical to go through this state. There are several methods for finding such subgoals. In (Kretchmar et al., 2003), they proposed the algorithm that uses a combination of a frequency and a distance of a state from the start or from the end state. Subgoal discovered by algorithm in (Goel & Huber, 2003) is a state with a high gradient of a number of states leading to the state. The other approach to discover subgoals automatically is to use diverse density (McGovern & Barto, 2001).

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Reinforcement learning is a kind of learning, in which an agent, by interaction with an environment, tries to reach a goal. Usually, this problem is described as Markov Decision Process (MDP) (Sutton & Barto, 1998). An MDP is given by the tuple $\langle S,A,T,R \rangle$, where S is a set of states, that the agent can be in, A is a set of available actions, $T: S \times A \times S \rightarrow [0,1]$ is a transition function, which defines the probability of transition from the state $s \in S$ to the state $s' \in S$ when choosing action $a \in A$ in state s. $R : S \times A \rightarrow \mathcal{R}$ defines a reward, that the agent receives from the environment when choosing action a in state s. The goal is to find an optimal policy $\pi : S \times A \rightarrow [0,1]$.

Q-learning is one of the most used RL algorithms. The update rule for Q-learning is defined as:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t) \right] \tag{1}$$

where $Q(s_t,a_t)$ is the action-value function, which is a value of choosing action a_t in state s_t and following policy π. r_{t+1} is a actual reward given by the environment in time $t+1$.

2. Subgoal discovery method

Some of the important limitations of previously mentioned algorithms are: they start discovering subgoals after the agent have learned at least suboptimal policy;

they usually discover only first subgoal on the path; we have to set up a threshold for what is still the subgoal; we are looking just for specific states as subgoals.

The subgoal, our algorithm is to discover, is defined to be a state (abstract state) that occurs in every trajectory from the beginning of learning. After the first trial, every state is a candidate for the subgoal. After each following trial we have two sets: the set of candidate states and the set of states from just finished trajectory. The new set of candidate states will encompass just states that occur in both sets mentioned above - if there is a state in the set of candidates and a state in the last trajectory and they differ just in less than n parameters we will omit these dimensions and put this abstract state in the set of candidates.

Our algorithm consists of these steps:

1. if the current trajectory is the first trajectory, store it as the set of candidates C and go to the step 4.
2. store the current trajectory to P_{curr}
3. replace set C by set of states or abstract states up to predefined level of abstraction that occur in C and in P_{curr}
4. wait until end of the trial and go to the step 2.

3. Experimental results

All experiments were conducted in a simulated environment. The goal of the agent is to find a key (first subgoal) to open a door, go through the door (second subgoal) to the other room and find the goal position (final goal).

We used Q-learning algorithm to find optimal policy. The state space is represented by agent's position in the environment described by the x and y position, where size of the environment is 600×500 with discretization step 20 units in both dimensions, by the agent's orientation (an angle with the x axis) discretized by $\frac{\pi}{4}$ and the information whether we have key or not. We defined three discrete actions: turn right $(+\frac{\pi}{4})$, turn left $(-\frac{\pi}{4})$ and go straight. The agent get reward $+1.0$ when it reaches the goal, otherwise -0.01.

Our algorithm to discover subgoals started together with the learning algorithm. The only parameter of the algorithm, the level of abstraction, was set to 1, which means, that we allow only one dimension to be omitted. Our algorithm was able to discover all three subgoals[1] even before the optimal policy was learned. In conducted experiments, the number of trials needed to discover all subgoals ranged from 577 to 644. Discovered subgoals are $[5, 20, *, 0], [15, 10, *, 1]$ and $[25, 5, *, 1]$ (Fig. 1). This means that our algorithm was able to automatically set up abstraction in state space and discovered all subgoals represented by the abstract states.

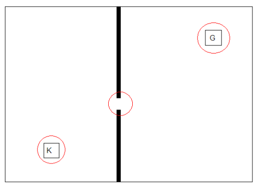

Figure 1. Discovered subgoals in simulated environment. Subgoals are circled in red. The key is in lower left corner. The door is a gap between two rectangular obstacles in the middle of the environment. In the upper right corner, there is a goal that the agent tries to reach.

In the experiments, we demonstrated that our algorithm can discover all subgoals in a simple navigation task and that it is able automatically work with the state abstraction when discovering subgoals. The other advantage of the algorithm is that it works in parallel with learning, it does not need optimal policy and it does not interfere with learning process. In real robots tasks, all these advantages are considered to be important.

In the future, we would like to focus on discovering subgoals in tasks with two different doors and two different keys. We also aim to consider the other forms of state abstractions and apply it to real robot applications with not so obvious subgoals.

References

Goel, S., & Huber, M. (2003). Subgoal Discovery for Hiearchical Reinforcement Learning Using Learned Policies.

Kretchmar, R., Feil, T., & Bansal, R. (2003). Improved automatic discovery of subgoals for options in hierarchical reinforcement learning. *Journal of Computer Science and Technology*, *3*, 9–14.

Lapko, M., & Jaksa, R. (2009). Reuse of knowledge within reinforcement learning in mobile robotics. *13th International Conference on Cognitive and Neural Systems 2009 (accepted).*

[1]we can consider task goal to be a subgoal

McGovern, A., & Barto, A. (2001). Automatic discovery of subgoals in reinforcement learning using diverse density. 361–368.

Sutton, R., Precup, D., & Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, *112*, 181–211.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction (adaptive computation and machine learning)*. The MIT Press.