Automatic Curriculum Generation

Shuo YANG

December 4, 2017

Abstract

This report mainly focuses on the empirical result on the automatic curriculum generation. The modified algorithm could be used to generate curriculum according to the agent's current learning state and its learning ability without having to specify the set of possible sub-tasks. The new method follows the ways of simplifying a task proposed in Sanmit's paper [2]. The algorithm will use several domain-specific operators to simplify the target task. Then the new tasks, say sub-tasks, will be used to construct the curriculum according to the agent's current learning state and its learning ability. This procedure will be used iteratively until the agent is able to solve the target task. The empirical results have demonstrated the great efficiency improvement by using the automatically generated curriculum both in normal reward setting and sparse reward setting. The generation algorithm and the setting up of the test domain will be discussed in detail.

1 Experiment Environment

1.1 Description of the domain

The experiments are carried out in a grid world domain. The world consists of one or several rooms, which can contain 5 types of objects, keys, locks, beacons, pits and doors. The final goal of the agent is unlocking the "master lock". Before that, it has to travel all over the grid world to collect all the necessary keys.

Keys are items that the agent can pick up by moving to them and executing a pickup action. These are used to unlock locks.

Locks are typically linked to a set of keys and a door. When the agent has picked up keys associating with a lock, the lock could then be unlocked if the agent moves to the lock and execute the unlock action. When a lock is unlocked, the corresponding door would then be passable.

Doors are set to the connections of different rooms. Some of them are linked with locks. When the lock is still locked, the agent could not pass through the door and enter another room.

Pits are set inside rooms. When the agent move into a pit, the episode is terminated.

Beacons are the landmarks placed on the corners of pits.

1.2 Agent

Agent Types and Action Space

There are three different kinds of agents: Basic Agent, Rope Agent, Jump Agent. The basic agent can walk in 4 different directions and has two other actions: pick up and unlock. The rope agent will have 4 additional actions, which are to use a rope in one of the four directions. Doing so opens a path across a pit if one is present. The jump agent could move 2 grids in a step, this gives it the ability to jump over a pit and also move faster.

State Representation and Learning Algorithm

The three agents share the same state representation but have different action types. For agent's perception, each state will be converted into a feature vector containing the information of the distance to different objects. Instead of using the raw distance to an object, the feature would be the ordering of distance in 4 different directions. For example, if the distance to an object is [4.24, 5.66, 5.83, 4.47], then the corresponding feature would be [1, 3, 4, 2]. The agent will consider each of the nearest pit, beacon, key, lock and all the doors , and concatenate them to construct a feature

vector. If no such object is in the room, the feature would be [5, 5, 5, 5]. If the agent is standing on the object, which makes the four direction has the same distance, the feature would be [0, 0, 0, 0]. All the three agents are using the Q-learning Algorithm. The Q value will be stored in a table according to the state feature.

1.3 Target Task

In the experiments, the environment is set to have 3 rooms, with 3 keys in the middle room and 1 key in each side room. There are 2 locks in the environment, a sub-lock and a master lock. The door to the left room is dependent on the sub-lock. The agent needs to collect all 3 keys in the middle room to unlock the sub-lock and travel 3 rooms to collect all the keys so that it could unlock the master lock. The environment is shown in figure 1.

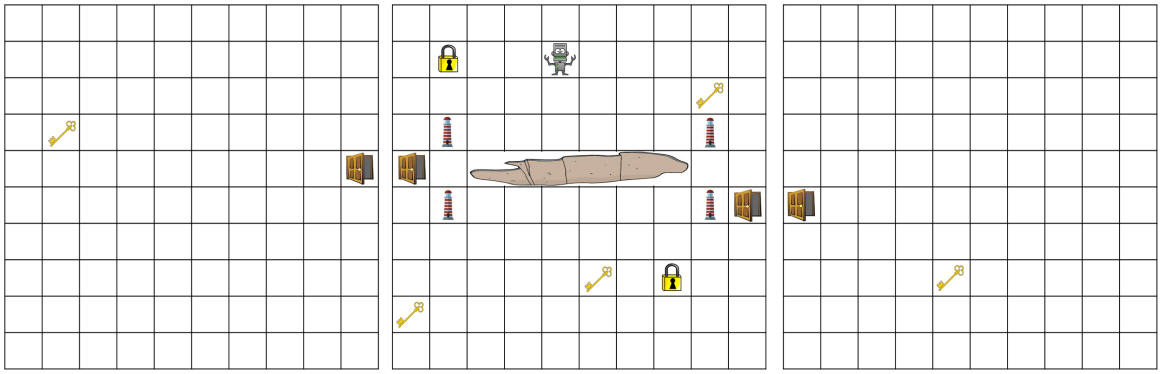


Figure 1: Target Task

2 Simplify Operator

In paper [1], different ways to simplify a task is discussed in detail. Here we adopted the Task Dimension Simplification, Promising Initialization, Task-based subgoals to generate subtasks. We designed several operators to simplify the target task. The operators are listed below.

2.1 Reduce Key

Task Dimension Simplification & Task-based subgoals

this operator can be applied for several times. For the first time, it will keep removing keys until there is only one key left in each room. For the second time, it will remove the key in the middle room, and for the last time, it will remove the keys in both side rooms. A lock is automatically set unlockable if the corresponding keys are all removed.

2.2 Reduce Lock Operator

Task Dimension Simplification & Task-based subgoals

This operator can be applied for 2 times. Firstly, it will remove all the sub-locks. At that time, the door to the left room, which depends on the sub-lock, is automatically opened. Secondly, it will remove the master lock. After that, there will be no lock in the domain, so the termination function will be changed into picking up all the keys.

2.3 Remove Pits

Task Dimension Simplification

This operator can only be applied once. It will remove all the pits from the task.

2.4 Remove Left Room

Task Dimension Simplification

This operator can be applied for only 1 time. It will remove the left room as well as the key in the left room. Simplify the task into a two-room task.

2.5 Remove Right Room

Task Dimension Simplification

this operator is quite similar to the 'remove left room', but it will remove the right room.

2.6 Better Initial position operator

Promising Initialization

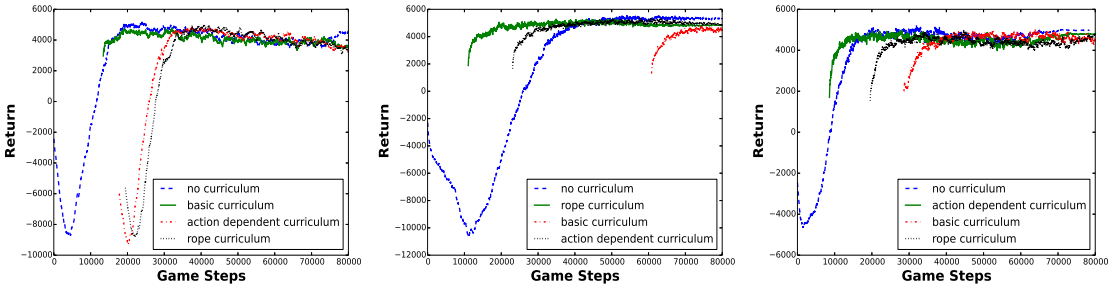
this operator is only applicable after the task is simplified by #4 operator or #5 operator. Without applying this operator, the agent will always be born in the middle of the middle room, which means the agent has to find a way go into the side room and come out of the side room. After applying this operator, the agent will be born in the side room, it will only need to find the way out which simplify the task a lot.

3 Result

The following results are gained by using the methods mentioned above. All the sub-tasks are generated by the operators defined above and the selecting of sub-task is the same as the one described in Sanmits' paper [2]. We evaluated the performance of each agent on the target task using no curriculum, a curriculum tailored for that specific agent and curricula tailored for each of the other two agents. Each curve below is averaged over 500 runs and is offset to reflect the training in the curriculum.

3.1 Normal Reward Setting

In this reward setting, picking up a key would have a reward of 500, unlocking a lock would have a reward of 700 and unlocking the master lock would have a reward of 4000. Each step will have a -10 reward, and fall into a pit will cause a -2000 reward.



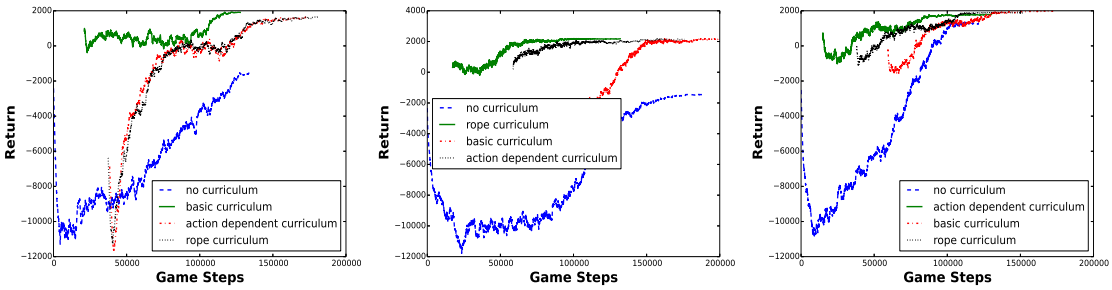
(a) Basic Agent (b) Rope Agent (c) Jump Agent

Figure 2: Experiment Result under Normal Reward Setting

The results here showed that training by the curriculum customized for an agent provides the best benefit. Training on the curriculum other than the one created for the agent may lead to an efficiency loss.

3.2 Sparse Reward Setting

In this reward setting, the agent will only get the reward for unlocking the master lock or fall into a pit for +4000 and -2000. Other than that, it will only receive a step reward of -10. This makes the task much harder to learn because unless the agent learns how to solve the target task, it won't be able to know picking up a key or unlock a sub-lock is good.



(a) Basic Agent (b) Rope Agent (c) Jump Agent

Figure 3: Experiment Result under Sparse Reward Setting

The results here also showed that training by the curriculum customized for an agent provides the best benefit. But now, the curriculum is significantly better than the benchmark, no curriculum. This means by the automatically generated sub-tasks, the curriculum could serve as a ladder for the agent to conquer the target task. This experiment has also been carried out with fixed sub-tasks set and it is notable that using the automatically generated sub-tasks, the learning process is actually faster. This indicates that the operators generated some source tasks which are not in the fixed sub-tasks set are more suitable for the agents to learn.

References

[1] Narvekar, S., Sinapov, J., Leonetti, M., and Stone, P. Source task creation for curriculum learning. In *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016)* (May 2016).

[2] Narvekar, S., Sinapov, J., and Stone, P. Autonomous task sequencing for customized curriculum design in reinforcement learning. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)* (August 2017).