The 33rd Annual Conference of the Japanese Society for Artificial Intelligence, 2019

階層型強化学習 MLSH における枝刈りによるサブポリシ数調整

A Sub-policy Pruning Method for Meta Learning Shared Hierarchies

Qing HONG *1*2 Yusuke TANIMURA*2*1 Hidemoto NAKADA*2*1

*1筑波大学
University of Tsukuba

*2産業技術総合研究所 人工知能研究センター
Artifical Intelligence Research Center, National Institute of Advanced Institute of Technology

Meta Learning Shared Hierarchies(MLSH) is a example of Hierarchical Reinforcement Learning.n Assuming some task distribution and reusing the knowledge learned on the preceding tasks, it aims to rapidly adjust to new tasks. It divides the policy into several sub-policies and master policy which choose one of the sub-policies, expecting each sub-policy is responsible for each sub-goal in the task. One of the drawbacks of this approach is that, in general, it is not possible know the number of sub-goals in unseen task distribution, hence, the number of sub-policies. With improper number of sub-policies, the performance of MLSH is degraded. To solve this problem, we propose a method to locate the proper number of sub-policies by pruning excessive sub-policies. The proposed method save resources and reduce the time the algorithm to converge. We test the method using 2D-bandit problem and demonstrate the effectiveness.

1. Introduction

Reinforcement learning takes long time to solve tasks with sparse rewards. Hierarchical reinforcement learning (HRL) is an approach to solve this problem reusing sub-policies learned on preceding similar tasks [Bakker 04]. It could improve the exploration efficiency [Simsek 04] or learning efficiency [Bacon 16].

Traditional HRL requires hand-crafted sub-goals to solve specific tasks with pre-trained sub-policies. The agent learn a policy to take action to receive an extrinsic reward first, then separate this action into several basic actions. With this approach it is difficult to obtain good performance in practical hierarchical problems. On the other hand, finds a useful sub-policies first and use such policies to solve sparse-reward problems is more efficient in practical hierarchical problems. Such methods are called meta learning [Finn 17].

Meta learning has a problem that it can hardly discover useful sub-policies for unseen problems. MLSH (Meta Learning Shared Hierarchies) [Frans 18] is a HRL algorithm which focuses on these problems. MLSH tries to solve a wide variety of tasks by using prior knowledge and primitive policies. The goal is to rapidly reach high reward on new tasks drawn from a certain task distribution. When related tasks contains multiple sub-goals, MLSH pre-trains multiple sub-policies correspond to the sub-goals and update them accordingly. However, in some complicated tasks, the numbers of goals are unpredictable. Too many sub-policies and too few sub-policies degrade the performance of MLSH in each way. To solve this problem, we propose a method to locate the proper number of sub-policies by pruning the excessive sub-policies.

2. Background

2.1 Meta Learning Shared Hierarchies

Meta Learning Shared Hierarchies is an end-to-end meta-learning approach that uses prior knowledge to solve unseen tasks.

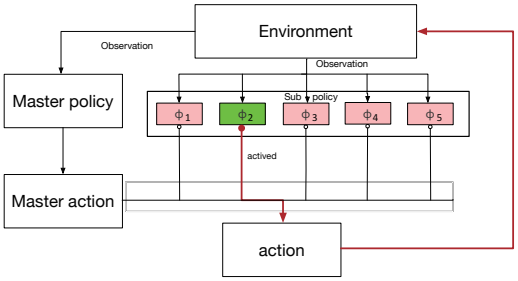

Figure 1: The Overview of MLSH

The goal of MLSH is to rapidly reach a high reward with similar tasks. By training master policy to activate one of the sub-policies that correspond to sub-goals, MLSH utilize the prior knowledge to solve tasks that share similar sub-goals. That is to say, MLSH create a probabilistic policy called master policy whose action is to choose one of the sub-policies. Master policy starts with randomly-initialized status and will be fine-tuned to each task. Then we optimize the probabilistic parameter based on the total reward from sub-policies and the master's action. At last, we update sub-policies and master policy jointly.

2.2 Simple 2D-Moving Bandits environment

We use a simple problem called 2D-moving bandid throughout this paper as an example problem. This is a simple bandit problem which contains different potential goals p(Figure 2). This environment generates one dots as our agent. Then randomly create two dots, and each dot is considered a potential goal. Our agent's goal is to reach them as close as possible. 2D-bandit problem is similar to the N-armed bandit problem. Each slot (dot) have different repay (reward depends on the rate of repaying and distance), our agent's goal is to find the highest total reward. In each iteration the position of dot and repay rate randomly be altered, our agent also needs to do such things again: 1. approach the ball. 2. confirm the reward and optimal the goal direction.

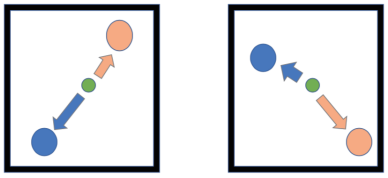

Figure 2: Simple 2D-Moving bandit problem: green dot denotes our agent, and other dots denote our potential goals.

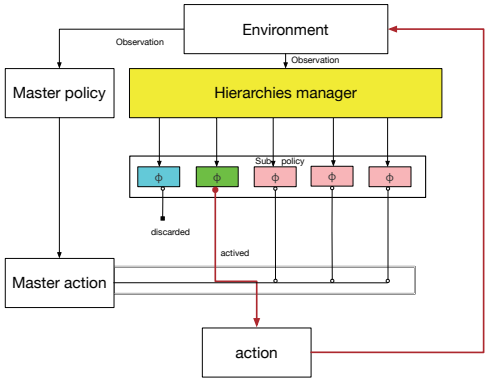

Figure 3: The Architecture of the proposed method, SPPM create hierarchies manager to formulate the policy which helps agent to locate the proper number of sub-policies.

This environment is easy to simplest hierarchical learning problem, and we will use it to help our research.

3. Sub Policy Pruning Method

We propose a method called Sub Policies Pruning Method (SPPM). The goal of SPPM is to find the proper number of sub-policies in the environment which contains an uncertain number of sub-goals. SPPM starts with enough number of sub-policies, observes the behavior of each sub-policy, and determines the proper number of sub-policies. A module called hierarchies manager is responsible for this procedure (Figure 3). With this method, we can save resources and improve learning speed.

Figure 4 shows the algorithm of SPPM. When the environment is unseen, and we do not know the number of sub-goals, we use SPPM to speed up the training. SSPM initialize an excessive number of sub-policies ϕ. To observe the behavior of each sub-policy our agent need a warm-up trial T to train the ϕ. T subjects to the complexity of the environment and it provided by our hierarchies manager. After warm-up trials, each sub-policy have a specific tendency, so we can remove the sub-policies with low-activity and replace the role of them with more active ones. In the next prune trial P, SPPM record the activated times A of each sub-policy selected by master-policy. If the A of ϕ lower than PL (Pruning threshold), the sub-policy will be pruned and will not be activated in the rest of trials. PT is provided by hierarchies manager based on the environment.

Sub Policies Pruning Method in MLSH
initialize ϕ
repeat
Initialize θ
Sample task M $\sim P_M$
For t=0,1,.....T(Test trial) do
Update ϕ to maximize expected return in W
Update θ to maximize expected return in U
end for
For P=0,1,.....P(Prune trial) do
Update ϕ to maximize expected return in W
Update θ to maximize expected return in U
If (ϕ_k|A(Activated times of ϕ_k)) < PT:
Prune ϕ_k
end if
end for
until convergence

Figure 4: The algorithm of SPPM.

3.1 When SPPM Conduct Pruning Method?

To estimate the actual value of each sub-policy , it is necessary to give enough time steps to train the sub-policies until they began to have redundancy. When the environment becomes complicated, the MLSH agent cost more time to get convergence. Hence, the pruning time point depends on the complexity of the environment. On the other hand, from the resource consumption point of view, we need to prune the unnecessary sub-policy before the convergence.

Figure 5 is the heat-map of similarity by using similarity equation:

$$\cos(\mathbf{A},\mathbf{B}) = \frac{\mathbf{AB}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n}\mathbf{A}_i\mathbf{B}_i}{\sqrt{\sum_{i=1}^{n}(\mathbf{A}_i)^2}\sqrt{\sum_{i=1}^{n}(\mathbf{B}_i)^2}} \quad (1)$$

The x-axis and y-axis denote sub-policies. Each cell stands for the 'similarity' of the corresponding two sub-policies. Larger value means that the two sub-policies behave similarly. With the training goes on, some of the sub-policies converge into the same behavior. Such sub-policies can be marked as redundant and one of them could be safely removed, since the other will fill in the role. We find that the behavior convergence after 50 trials. Conducting a pruning method after that may be a good time point for this case.

3.2 How SPPM Conduct Pruning Method?

Define the pruning rule to efficiently prune the redundant policies is a challenging problem. We found that pruning rule based on recent activated times is easier to obtain better performance (using moving average).

Find the best pruning threshold on a particular task is the second problem we need to consider. Pruning too much sub-policies leads to counterproductive effect. Figure 6 shows the moving average of activation ratio of each sub-policies. From the graph, we could figure out that pruning sub-policies that have activation ratio less than 50% or 60% of the most activated sub-policy will be moderate. Based on this observation, we define two threshold; aggressive threshold (50%) and conservative threshold (60%).

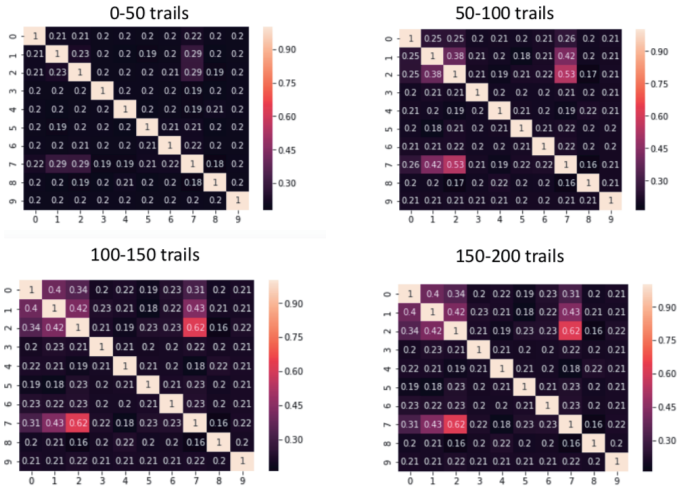

Figure 5: The heatmap of similarity using 10 sub-policies, the x-axis and y-axis denotes different sub-policies. the number denotes the similarity compare with 2sub-policy. The bigger the number is, the similar actions it take. From 50 trials some of the sub-policy start to get redundant.

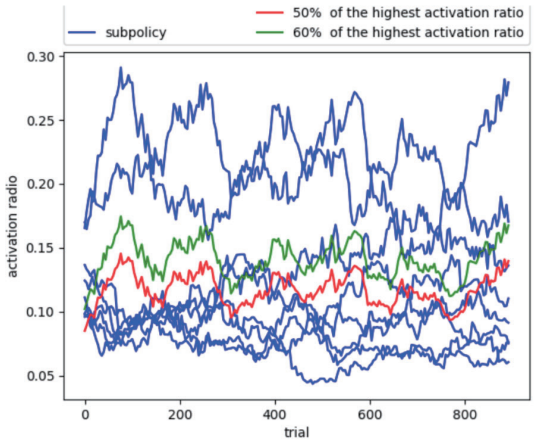

Figure 6: The result of 8sub-policies 2sub-goals. X-axis denotes the trials and Y-axis denotes the activation ratio. The red line denotes aggressive pruning threshold: less than 50% of highest activation ratio. The green line denotes conservative pruning threshold: less than 60% of highest activation ratio

4. Evaluation

To evaluate the proposed method, we conducted experiments using the 2D-moving bandits shown in 2.2 with only one sub-goal is designated as the real goal (other goals have 0 repay rate). Number of sub-goal is 2, hence the ideal number of sub-policies is also 2. As the parameter of SPPM, we employed two parameters; namely, 1) pruning threshold, 2) pruning start time. For pruning threshold, we tried aggressive (50%) and conservative (60%) setting. For start time, we tried 50 trials and 150 trials. After 50 trials is the time when the sub-policy behavior starts to converge. After 150 trials is the time just before whole algorithm reach the convergence. We started with 8 sub-policies.

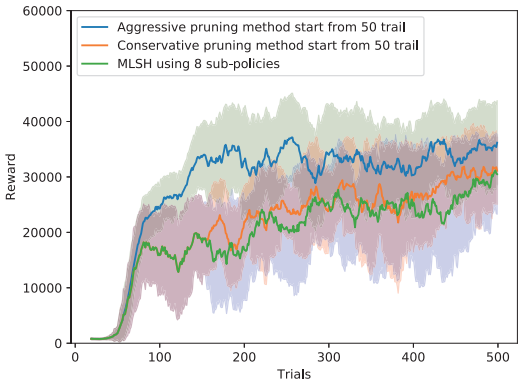

Figure 7: Comparison of aggressive pruning strategy with conservative pruning strategy with 8 sub-policies 2 sub-goals.

Figure 8 shows the results with 150 trials start and Figure 9 shows the results with 50 trials start. Left diagrams show the aggressive case while the right show the conservative case. Upper diagrams show the activated ratio of each sub-policies and lower diagrams show the reward. The blue vertical lines denote the trial we successful prune the sub-policies. When we start pruning after 150 trial (Figure 8), with aggressive pruning strategy, we successfully pruned 4 sub-policies at 150 trial and 1 sub-policy at 202 trial. With conservative pruning strategy, we could prunes 3 sub-policies at 297, 321 and 346 trial.

When we start pruning after 50 trial (Figure 9), with aggressive pruning strategy, we pruned 5 sub-policies at 27, 28, 32, 33, 38. With conservative strategy, we could prune 1 sub-policy at 153 trials

Figure 7 shows the comparison of different pruning strategies. Aggressive pruning method seems to have better performance than the conservative one. However, we have to note that there is a certain risk to prune too much sub-policies. For simple environment (the number of sub-goals is much smaller than number of sub-policies), starting to prune sub-policies at the early stage seems better. Aggressive pruning strategy seems to be better than conservative one.

5. Conclusion

In this paper, we proposed Sub Policies Pruning Method that locate proper number of sub-policies by starting with enough number of sub-policies and pruning the low-activity sub-policies during the training. We applied the proposed method to the 2D-bandid problem and confirmed the effectiveness of the method.

One of the problem with the proposed method is the tuning of the hyper parameters, namely, pruning threshold and pruning start time. Finding methods to automatically adjust these parameters is our future work.

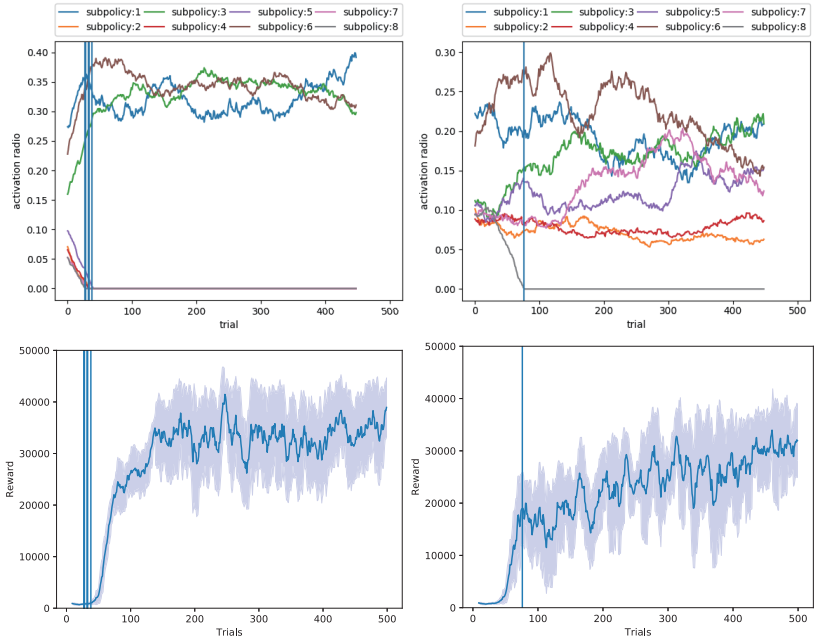

Figure 8: Evaluation Results: starts 150 trials. Left: aggressive, Right: conservative.

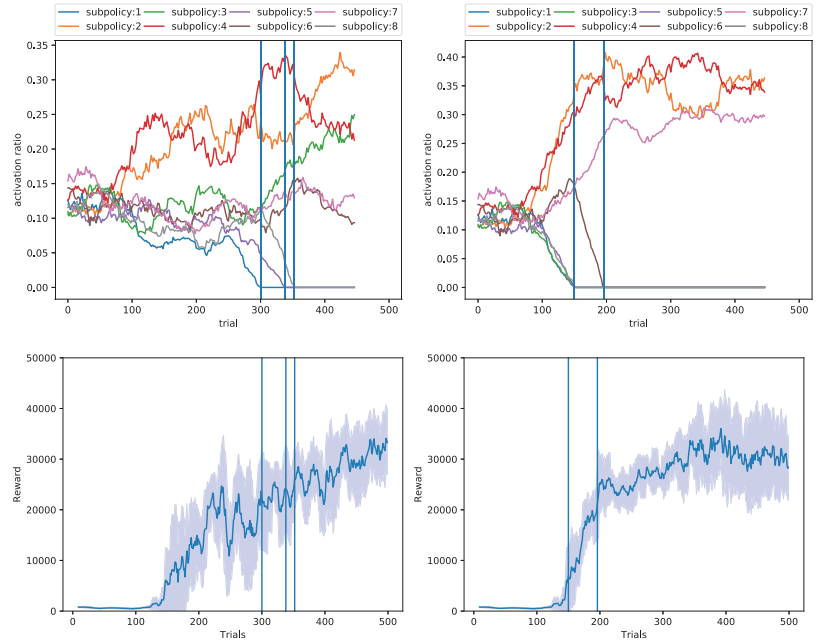

Figure 9: Evaluation Results: starts 50 trials. Left: aggressive, Right: conservative.

Acknowledgement

This paper is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO). This work was supported by JSPS KAKENHI Grant Number JP16K00116.

References

[Bacon 16] Bacon, P., et al.: The Option-Critic Architecture, in *CoRR*, p. abs/1609.05140 (2016)

[Bakker 04] Bakker, B., et al.: Hierarchical reinforcement learning with subpolicies specializing for learned subgoals, in *Nural Networks and Computational Intelligence*, p. abs/1703.03400 (2004)

[Finn 17] Finn, C., et al.: Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks, in *CoRR*, p. abs/1703.03400 (2017)

[Frans 18] Frans, K., et al.: Meta Learning Shared Hierarchies, in *International Conference on Learning Representations* (2018)

[Simsek 04] Simsek, O., et al.: Using relative novelty to identify to identify useful temporal abstractions in reinforcement learning, in *ICML* (2004)