ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

Developmental Modular Reinforcement Learning

Jianyong Xue[1,2,3] and Frédéric Alexandre[1,2,3]

1- Inria Bordeaux Sud-Ouest - Talence - France

2- LaBRI - Universite de Bordeaux - Talence - France

3- Institut des Maladies Neurodégénératives - Bordeaux - France

Abstract. In this article, we propose a modular reinforcement learning (MRL) architecture that coordinates the competition and the cooperation between modules, and inspires, in a developmental approach, the generation of new modules in cases where new goals have been detected. We evaluate the effectiveness of our approach in a multiple-goal torus grid world. Results show that our approach has better performance than previous MRL methods in learning separate strategies for sub-goals, and reusing them for solving task-specific or unseen multi-goal problems, as well as maintaining the independence of the learning in each module.

1 Introduction

Traditional reinforcement learning algorithms have been proven successful in solving complex, single-goal tasks; yet they perform badly in multi-tasks and non-stationary environments with hidden states. Modular reinforcement learning (MRL) provides one solution that decomposes a complex problem into a collection of concurrently running RL modules, each of which learns a separate policy to solve a portion of the original problem.

MRL has been studied not only for the decomposition of complex tasks into modules, but also for the reusability of separately learned modules in new strategies for other tasks that have never been solved before [1]. Focusing both on the optimality of the composite strategy for the entire task and on the independence of learning in separate modules, [2] introduced a specific concept of "modular reward" to propagate local rewards toward the entire task achievement between modules. Considering the performance degradation in the composability of modules that have incomparable reward scales, [1] proposed a model, Arbi-Q, that recruits an additional module of *command arbitrator* to assign probabilities to the selection of a module for each state; then the selected module's preferred action becomes the agent's action in that state. However, one limitation of Arbi-Q is that the agent's action always comes from one of the modules.

In this article, we adopt the idea of multiple model-based reinforcement learning from [3] and present a new MRL architecture that addresses all the concerns mentioned above by borrowing principles and mechanisms from the mentioned models. In addition, a particular component, i.e. the "coordination module" is introduced, to mediate the competition and the cooperation between different modules, and to inspire the learning system to create new modules in circumstances where current modules are not capable of handling unseen tasks.

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1.
Available from http://www.i6doc.com/en/.

2 The developmental MRL architecture

Figure 1 shows the overall organization of the proposed Developmental MRL (DMRL) architecture, which is composed of a coordination module plus a list of n model-based RL (MBRL) modules. Each MBRL module shares the same inputs and has an identical structure that consists of three components: the reward predictor, the RL controller and the state predictor [3]. The coordination module will be activated to create new modules whenever the summation of errors from each state predictor reaches a threshold. Furthermore, learning in each component is gated by a "responsibility signal" [3], which comes from the gaussian softmax function of prediction errors and determines how much the modules are responsible for the current task.

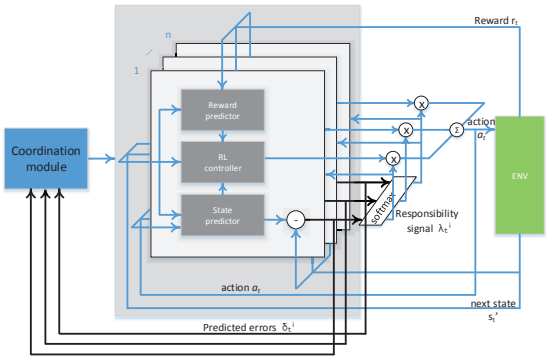

Fig. 1: . Schematic diagram of the DMRL structure.

2.1 State predictor and the responsibility signal

The state prediction function $F_i(s_{t+1}, s_t, a_t : \theta)$ gives the probability distribution of the newly observed state $s_{t+1} \in S$ based on the previous state $s_t \in S$ and the action $a_t \in A$ performed by the agent at step t, where S and A are the set of state space and action space, $i \in \{1, \ldots, n\}$ is the index of modules. Specifically, the dynamic function is realized through a three-layers neural network, where the vector of parameters θ represents the weights of the network. This network is trained by a collection of pairs of inputs (s_t, a_t) and their corresponding output labels s'_{t+1} from trajectories $\tau = \{s_0, a_0, \ldots, s_{T-2}, a_{T-2}, s_{T-1}\}$ of length T [4]. Initial parameters in this neural network are set as a small random value with uniform distribution between 0.0 and 1.0. With the newly observed state s_{t+1} , the state prediction error δ^i_{st} of module i is calculated as follows:

$$\delta^i_{st} = F_i(j, s_t, a_t : \theta_i) - c(j, s_{t+1}), j \in S \tag{1}$$

where $c(j, s_{t+1})$ is 1 if $j = s_{t+1}$ and 0 otherwise. Along with state prediction errors, the responsibility signal λ^i_t for each module is formulated as an output of

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

the gaussian softmax function:

$$\lambda_t^i = \frac{\hat{\lambda_t^i} e^{-\frac{1}{2\sigma^2}{\delta_{st}^i}^2}}{\sum_{j=1}^n \hat{\lambda_t^i} e^{-\frac{1}{2\sigma^2}{\delta_{st}^j}^2}}, \quad \hat{\lambda_t^i} = \frac{{\lambda_{t-1}^i}^\rho}{\sum_{j=1}^n {\lambda_{t-1}^j}^\rho} \tag{2}$$

where $\hat{\lambda_t^i}$ represents the prior knowledge about module selection, which was used to maintain the temporal continuity of the selected module [3]. Parameter σ controls the sensitivity of responsibility signals to state prediction errors, and ρ $(0 < \rho < 1)$ indicates the effects of past module selections on current decision-making. The parameter vector θ is updated as follows:

$$\theta_i = \theta_i + \alpha \lambda_t^i \delta_{st}^i \frac{\partial F_i(s_{t+1}, s_t, a_t : \theta_i)}{\partial \theta_i} \tag{3}$$

where $0 < \alpha < 1$ is the learning rate.

2.2 Reward predictor

The reward predictor $R_i(r', s_t, a_t)$ provides an expected reward r_t' based on the previous state s_t and the action a_t selected at step t. The *modular reward* $\hat{r_t^i}$ is formed as the sum of immediately received reward plus an extra bonus for rewarding the module that better fits the current sub-goal. Specifically, this bonus is calculated from the product of modular value function and the temporal difference in the responsibility signals; then we have the reward prediction error:

$$\delta_{rt}^i = \hat{r_t^i} - R_i(r', s_t, a_t), \quad \hat{r_t^i} = r_t + (\lambda_{t+1}^i - \lambda_t^i) V^i(s_{t+1}) \tag{4}$$

therefore, parameters in the reward predictor are updated by:

$$R_i(r', s_t, a_t) = R_i(r', s_t, a_t) + \alpha \lambda_t^i \delta_{rt}^i \tag{5}$$

2.3 RL controller

The RL controller in module i provides two functions: the state-value function $V^i(s_t)$ and the state-action-value function $Q^i(s_t, a_t)$. Combined with the state transition function $F_i(s_{t+1}, s_t, a_t)$ and the reward function $R_i(r', s_t, a_t)$, we have the full Bellman equation [5] for the value function:

$$V_\pi^i(s_t) = \sum_{a_t \in A} \pi_t^i(a_t|s_t)[R_i(r_t', s_t, a_t) + \gamma \sum_{s_{t+1} \in S} F_i(s_{t+1}, s_t, a_t : \theta_i) V_\pi^i(s_{t+1})] \tag{6}$$

where γ is the discount factor that controls the effects of future rewards. Furthermore, we recorded the eligibility traces $e_t^i s$ for each state s, which are updated as:

$$e_t^i(s) = \begin{cases} \eta\gamma e_{t-1}^i(s) & if\ s \neq s_t \\ \eta\gamma e_{t-1}^i(s) + 1 & if\ s = s_t \end{cases} \tag{7}$$

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

then the state value $V^i(s_t)$ will be updated by:

$$V^i(s_t) = V^i(s_t) + \alpha\lambda_t^i\delta_{vt}^i e_t^i(s), \quad \delta_{vt}^i = \hat{r_t^i} + \gamma V^i(s_{t+1}) - V^i(s_t) \tag{8}$$

where δ_{vt}^i is the temporal-difference (TD) error with modular reward $\hat{r_t^i}$.

2.4 Action selection

For each candidate action $a_j \in A, (j = 1, \ldots, M)$, their preferences are calculated by:

$$Q(s_t, a_j) = \sum_{i=1}^{n} \lambda_t^i [R_i(r', s_t, a_j) + \gamma \sum_{j \in X} F_i(s_j, s_t, a_j : \theta_i) V(s_j)] \tag{9}$$

where X is the set of possible states that the agent observed after taking action a_j in state s_t. We use a softmax function as a stochastic version of the greedy policy, where the action a_t is selected by

$$P_i(a_j|s_t) = \frac{e^{\beta Q(s_t, a_j)}}{\sum_{k=1}^{M} e^{\beta Q(s_t, a_k)}} \tag{10}$$

where β was set as $trials/500$, which controls the stochasticity of action selection.

2.5 The coordination module

The coordination module (CM) is designed to (a) prevent learned modules for specific subs-tasks from being modified, and (b) inspire the learning system to create new modules in circumstances where current modules are not capable of handling tasks that they never experienced before. In order to keep modules independent in learning their own policies, the CM modulates the sensibility of responsibility signals to smaller predictions errors by adjusting the value of the parameter σ, as:

$$\sigma_t = -\sum_{i=1}^{n} log(\delta_{st}^i) \tag{11}$$

In situations where the system detects a new sub-task that has never been solved before, or when current modules are not capable of solving them, the *watch_list* records the sum of prediction errors in κ steps and calculates the average value $\overline{se_t}$ to compare with the threshold. When $\overline{se_t}$ reaches the threshold, the coordination module will directly duplicate the module that has the least prediction error from current list of modules, and then train it through trajectories observed in experimental data.

3 Experiments

3.1 Settings

We investigate the performance of DMRL in a torus grid world derived from [3]. In this hunter-prey world, the hunter agent chooses one of five possible actions:

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1.
Available from http://www.i6doc.com/en/.

{north (N), east (E), south (S), west (W), stay}, and tries to catch multiple preys in a 7x7 torus grid world (as shown in Figure 2). Meanwhile, preys move in either one of four directions: $\{NE, NW, SE, SW\}$, which are represented as four different kinds of targets: $G = \{g_1, g_2, g_3, g_4\}$.

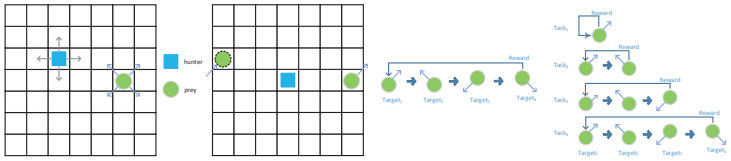

Fig. 2: The pursuit experiment in the torus grid world.

The experiment is divided into two parts: (a) all of 4 targets appear cyclically in a fixed order, which is $G_1 = (g_1, g_2, g_3, g_4)$ (as shown in Fig. 2). At the beginning of each trial, the hunter and the prey are placed at a random position and the prey's movement follows the first target g_1. When the hunter catches the prey, another prey appears at another position and moves in the direction of the next target. A reward $r = 10$ is given only when the last target g_4 is caught, which terminates this trial; a cost for each movement is set as -0.01. (b) Four different tasks are designed in the order of increasing complexity: $T = \{T_1 = (g_1), T_2 = (g_1, g_2), T_3 = (g_1, g_2, g_3), T_4 = (g_1, g_2, g_3, g_4)\}$.

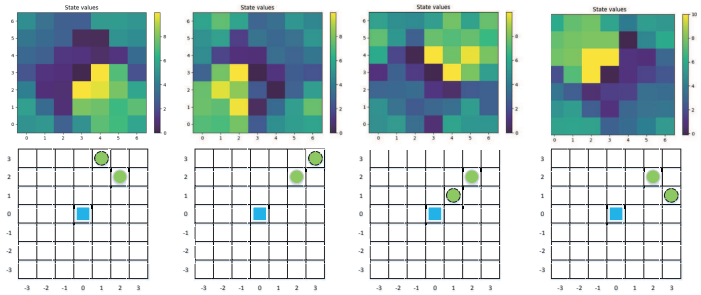

Fig. 3: Value functions and prediction models learned by the DMRL.

3.2 Results

In the first experiment, we compare the performance of DMRL with other MRL approaches. The left three figures in Fig. 4 show the average number of steps, the averaged reward, and the total value of states respectively during 2000 trial epochs in 20 simulation runs. It can be seen that the DMRL works significantly better than Arbi-Q and Inverse_Arbi-Q (an adaptation we made, where action preferences are evaluated by a joint policy, instead of coming from a single module); also it converges faster than the MMBRL and MMBRL with modular reward (MR). Fig. 3 shows examples of the value functions and the prediction

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1.
Available from http://www.i6doc.com/en/.

models learned by the DMRL. It can be seen that modules 1, 2, 3, 4 are specialized for the prey moving to the NW, NE, SW, and SE, respectively. The landscapes of the value functions are coherent with preys' movement directions.

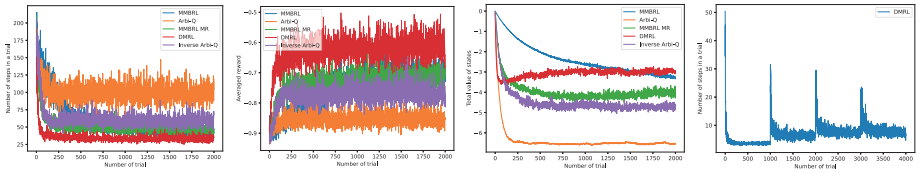

Fig. 4: Performance of DMRL compared with previous MRL approaches.

In the second experiment which investigates the increasing complexity and which is reported on in the figure on the far right of Fig. 4, the DMRL gradually learns modules as new sub-problems appear in the tasks, and the amplitude of the spikes representing the maximum of the error decreases as the complexity of the tasks increases, which indicates that the architecture is capable of reusing previously learned modules for solving new problems.

4 Conclusion and future work

In summary, we present the simple yet effective architecture DMRL to enable evolving multiple-goal tasks. Specifically, the responsibility signal was used to gate the learning of components in each module, and a particular coordination module is introduced to mediate the competition and the cooperation between different modules, and inspire the learning system to create new modules as needed. Nevertheless, a number of questions stands out as important targets for the next stage of research, such as (i) the planning strategy or the proactive way in modular reward; (ii) the top-down and bottom-up approaches for task decomposition. Inspirations from continual learning approaches will be a promising direction for learning tremendous task effectively without creating discrete modules.

References

[1] Christopher Simpkins and Charles Isbell. Composable modular reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4975–4982, 2019.

[2] Kazuyuki Samejima, Kenji Doya, and Mitsuo Kawato. Inter-module credit assignment in modular reinforcement learning. *Neural Networks*, 16(7):985–994, 2003.

[3] Kenji Doya, Kazuyuki Samejima, Ken-ichi Katagiri, and Mitsuo Kawato. Multiple model-based reinforcement learning. *Neural computation*, 14(6):1347–1369, 2002.

[4] Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7559–7566. IEEE, 2018.

[5] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.