Multi-task learning with modular reinforcement learning

Jianyong Xue, Frédéric Alexandre

Multi-task learning with modular reinforcement learning

Jianyong Xue[1,2,3][0000−0003−3245−8852] and Frédéric Alexandre[1,2,3][0000−0002−6113−1878]

[1] Inria Bordeaux Sud-Ouest, 33405 Talence, France
[2] LaBRI, Université de Bordeaux, Bordeaux INP
CNRS, UMR 5800, Talence, France
[3] Institut des Maladies Neurodégénératives, Université de Bordeaux
CNRS, UMR 5293, Bordeaux, France
{jianyong.xue, frederic.alexandre}@inria.fr

Abstract. The ability to learn compositional strategies in multi-task learning and to exert them appropriately is crucial to the development of artificial intelligence. However, there exist several challenges: (i) how to maintain the independence of modules in learning their own sub-tasks; (ii) how to avoid performance degradation in situations where modules' reward scales are incompatible; (iii) how to find the optimal composite policy for the entire set of tasks. In this paper, we introduce a Modular Reinforcement Learning (MRL) framework that coordinates the competition and the cooperation between separate modules. A selective update mechanism enables the learning system to align incomparable reward scales in different modules. Furthermore, the learning system follows a "joint policy" to calculate actions' preferences combined with their responsibility for the current task. We evaluate the effectiveness of our approach on a classic food-gathering and predator-avoidance task. Results show that our approach has better performance than previous MRL methods in learning separate strategies for sub-tasks, is robust to modules with incomparable reward scales, and maintains the independence of the learning in each module.

Keywords: Multi-task learning · Modular reinforcement learning · Incomparable reward scale · Compositionality policy · Model-based reinforcement learning.

1 Introduction

Multi-task learning is popularly observed in humans and several other intelligent animal species, and the ability to tackle with diverse tasks has gradually increased with the enrichment of skills that are learned from experience. For example, in a wild natural environment, most animals must pursue two tasks simultaneously: look for food and avoid predators. Meanwhile, long-time evolution has allowed animals to generate a more sensitive olfactory system to find food rapidly and to be more alert to predators, as well as to run faster.

A hallmark of multi-task learning benefits from the reusage of similar patterns underlying a set of regularities across tasks, in order to improve performance on any other single task [18,15]. The goal of multi-task learning is to find an optimal solution for solving the entire set of tasks in parallel. However, there exist several challenges: (i) how to maintain the independence of modules in learning their own sub-tasks; (ii) how to avoid performance degradation in situations where modules' reward scales are incompatible; (iii) how to find the optimal composite policy for the entire set of tasks.

These challenges have inspired the emergence of modular reinforcement learning (MRL) approaches, which decompose a multi-task problem into a collection of concurrently running RL modules, each of which learns a separate policy (or sub-policy) to solve a portion of the original problem [13,6]. Accordingly, these joint modules implicitly form a larger composite RL problem, the goal is to find the optimal policy for this composite RL [16]. Considering the performance degradation in the composability of modules that have incomparable reward scales, [13] introduced a special module named "command arbitrator", and proposed the architecture Arbi-Q to reformulate the MRL. Specifically, the arbitrator's policy assigns modular preferences given the observation of the state, then the selected module's preferred action will be used to interact with the environment. After each interaction, the state-abstraction function in the selected module transforms the world observation into a module-specific subset of the world states, in which modules are associated with the world that they operated in and not coupled to other modules or to an arbitrator.

Nevertheless, the Arbi-Q model has the following limitations: (i) all the modules need to be evaluated at each time-step without considering prior knowledge of modular selection, which slows down the learning process and the decision-making; (ii) learning system's action preference always comes from one single module, which may lead to dictatorship in decision-making; (iii) function approximation needs to be improved for complex continuous tasks (or the complex hybrid of discrete and continuous tasks).

Focused on these limitations, in this article, we adopt the idea of multiple model-based reinforcement learning from [3] and present a new MRL architecture named "Inverse Arbi-Q" to coordinate the competition and the cooperation between separate modules. A selective update mechanism enables the learning system to align incomparable reward scales in different modules. Furthermore, the learning system follows a "joint policy" to calculate actions' preferences combined with their responsibility for the current task.

2 Related work

Growing evidence from behavioral and biological neuroscience notes that most animals' brain parses ongoing behavior into discrete, bound segments [5]. Specifically, internal models in separated areas of the cerebellum are responsible for given tasks via a competitive and cooperative way [14,1]. In cases when a new task appears, multiple models learn it through competition, but only one or a

small part of them will be recruited for this task. Meanwhile, human behaviors are hierarchically organized: actions cohere into sub-task sequences, which fit together to achieve overall task goals [8,2].

In order to scale up the RL to address multiple goals at once, an intuitive approach is to train one RL module to handle each of the sub-goals, then the learned policies for each module can be used to fairly distribute control among the modules. However, these component policies are usually sub-optimal in the context of the composite tasks. Focusing on this problem, [16] replace the Q-learning rule within each module with a SARSA(0) learning rule, and propose a model of GM-Sarsa(0) for finding approximate solutions to multiple-goal RL problems. In order to better coordinate competitions between modules, the mixture of experts architecture from [11,7] introduced a learning system that is composed of a list of "experts" network, plus a gating network that determines which of the experts should be responsible for the training case. However, this modular architecture relies heavily on the performance of the gating network.

An alternative family of approaches [10,3,12] employs a state predictor within each expert (or module), which enables the environmental dynamics to be predictable. The controllers associated with these experts are decided by the prediction error: the one that has the smallest prediction error will be recruited at any particular moment. Nevertheless, models based on this idea mainly emphasize the competition between modules to choose one single expert that is the most suitable for a specific task (or sub-task), but the cooperation between modules as a compositional strategy for composable complex tasks is relatively lacking. As an improvement, [3] propose a model of multiple model-based reinforcement learning (MMBRL). In the MMBRL, a complex task is decomposed into multiple modules, particularly, the “responsibility signal" was introduced to weight the outputs of multiple modules, as well as to gate the learning of the prediction models and the RL controllers. However, the parallel update strategy degrades its performance when modules has incomparable reward scales, and the composite policy will be far from optimal as the complexity of the learning problem increases.

Generally, the original MRL supports not only the decomposition of complex tasks into modules, but also the composability of separately learned modules as new strategies for tasks that were never solved before [4,13]. Focusing on the optimality of the composite strategy for the entire task and the independence of learning in separate modules, [12] introduced the specific concept of “modular reward", which comes from the actual reward after each interaction plus a bonus for passing the task on a proper module. Specifically, this bonus is calculated from the modular value function and the temporal difference in the module gating signal, which propagates the reward toward the entire task achievement between modules. In situations where the tasks require to perform the sub-tasks concurrently, [4] propose an hierarchical RL approach to learn both compound and composable policies within the same learning process, by which exploiting the off-policy data generated by the compound policy. The results show that the experience collected with the compound policy permits not only to solve

the complex task but also to obtain useful composable policies that successfully perform in their corresponding sub-tasks.

Aiming at modules that have conflicting goals and misaligned reward scales, [6] presented the framework GRACIAS that assigns fine-grained importance to different modules and enables composable decision making based on modern deep RL methods. More precisely, the arbitrator learns the relative importance of each module in a given state, combined with action-value of the individual modules to calculate the joint action preferences. However, since the number of modules in this architecture is preset in advance, which will gradually limit its performance when the complexity of the task continues to increase. Moreover, when the arbitrator is trained with little prior knowledge, the performance will be initially far from optimal.

In the present work, the idea of Inverse Arbi-Q model is trying to address all the major concerns mentioned above. First, it's different from [7] where experts are determined by a gating network. Instead, modules in the present architecture learn their own behavioral policy and solve their own sub-tasks by competition. Second, it's also different from [13] where the action selection follows a "joint policy". Here action preferences come from the combination of all MBRL modules, rather than from one single preferred module. Furthermore, different from [3] where not all modules are involved in the update procedure after each interaction, instead, only the most related module will be targeted for amelioration. Moreover, the update of parameters in the selected module is gated by a dynamic signal, the "responsibility signal" [3,12] , which comes from the gaussian softmax function of prediction error in each module and determines how much the module is responsible for the current situation, rather than only according to a fixed parameter of learning rate as in [13,4] as we will explain in section 5.1.

3 Preliminaries

3.1 Reinforcement Learning

Reinforcement Learning (RL) paradigm focuses on learning an optimal policy π^* that enables the agent to gain the maximum accumulated rewards from future steps [17]. At each time step t, the agent stays in state $s_t \in S$ and follows the current policy $\pi(a|s)$ to execute an action $a_t \in A$; as a result, it receives an immediate feedback r_{t+1} from the environment and transitions stochastically to the next state $s_{t+1} \in S$. Note that performing this policy requires either knowing the underlying state transition model $P(s_{t+1}|s_t, a_t)$ and reward function $R(s_t, a_t)$, or estimating the state value $V(s_t)$ through rewards received in trajectories.

Based on differences in learning optimal policy, RL approaches could be divided into two categories: (a) model-free RL, which focuses on learning a policy or value function, and (b) model-based RL, which aims at learning a dynamic model and a reward function. In the model-free RL approaches, policy learning only focuses on interaction trajectories, and their parameters are mainly updated by the TD-error between state-values within two (or several) steps. Obviously,

it requires a large amount of samples to achieve good performance, and typically only learns a single task at a time; also it usually suffers from high sample complexity.

However, in the model-based RL system, a dynamic model is used to make predictions of the probabilities of transition to the next state s_{t+1}, and of a reward function to provide the expected values r', given the current state s_t and the action a_t selected by the current policy $\pi(a|s)$. Model-based reinforcement learning algorithms are generally regarded as being more sample efficient [9]. However, to achieve good sample efficiency, model-based RL algorithms conventionally use relatively simple function approximators, which fails to generalize well in complex tasks, or with probabilistic dynamics models in complicated and high-dimensional domains.

3.2 Modular reinforcement learning

The Modular Reinforcement Learning (MRL) framework is composed of several concurrently running RL modules, each of which learns a specific policy to solve a simpler sub-problem of the main RL problem. In particular, each module has access to the observation of the environment and shares a common action space [16]. During each interaction, all modules are required to execute the same action, while observing the state transition and a reward signal specific to the module [13]. The goal of MRL is to learn the optimal policy for the entire problem, in order to maximize the accumulative global reward in the long run.

Furthermore, action selection in MRL follows a "joint policy", in which action preferences come from the combination of all (or partially related) RL controllers in each module. At each time step, the candidate actions are evaluated by the weighted summation of their value expectations of future rewards from the RL controller in each module.

$$Q(s,a) = \sum_{i=1}^{n} Q_i(s,a) w_i \tag{1}$$

where the weight w_i assigned to each module indicates the responsibility of the module in the current task.

4 The Inverse Arbi-Q architecture

The basic idea of the Inverse Arbi-Q architecture lies in the learning system that decomposes a multi-task learning problem into a collection of RL modules, each of which learns a separate strategy for a specific sub-task concurrently. Figure 1 presents the overall organization of Inverse Arbi-Q architecture, which is composed of a list of n model-based RL (MBRL) modules. Each of MBRL modules shares a common inputs (state space and action space) and has an identical structure that consists of three components: the reward predictor, the RL controller and the state predictor, which provide the elementary realization of the module for learning its own behavioral policy and its own designated sub-task.

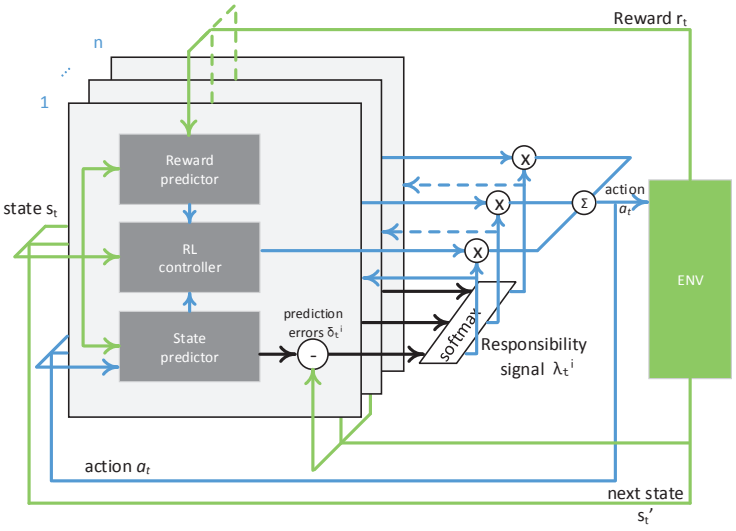


Fig. 1. Schematic diagram of Inverse Arbi-Q architecture.

5 Implementation

In this section, we explain the details for the implementation of Inverse Arbi-Q architecture. Subsection 5.1 introduces the state predictor and the calculation of responsibility signal for each module. Subsection 5.2 is about the reward predictor, we used a specific "modular reward" [12] for rewarding the module that is more suitable for the current sub-task, which encourages the architecture to be more concentrated to update to modules that better fit the current task. In subsections 5.3 and 5.4, we are more focused on the state-value function and the state-action function in the RL controller, as well as the action selection based on these two functions. Additionally, in order to make decisions more coherent, we adopted eligibility traces for each state, which are also introduced in this subsection.

5.1 State predictor

The dynamic function $F_i(s_{t+1}, s_t, a_t)$ in the state predictor of module i gives the probability distribution of the newly observed state s_{t+1} based on the previous state s_t and the action a_t performed by the agent at step t.

$$F_i(s_{t+1}, s_t, a_t : \theta_i) = P_i(s_{t+1}|s_t, a_t : \theta) \tag{2}$$

where $s_t \in \{1, \ldots, N\}$ and $a_t \in \{1, \ldots, M\}$ are discrete states and actions, $i \in \{1, \ldots, n\}$ is the index of modules. Specifically, this dynamic function is realized through a neural network with three fully-connected layers, where the vector of parameters θ represents the weights of the network. This network was

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 [9]. The initial parameters in this neural network were 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:

$$\delta^i_{st} = F_i(j, s_t, a_t : \theta_i) - c(j, s_{t+1}), j \in \{1, \ldots, N\} \tag{3}$$

$$c(j, s_{t+1}) = \begin{cases} 1 & j = s_{t+1}, \\ 0 & otherwise. \end{cases} \tag{4}$$

Along with state prediction errors in each module, the responsibility signal λ^i_t of each module for the current sub-task is formulated as the gaussian softmax function,

$$\lambda^i_t = \frac{\hat{\lambda^i_t} e^{-\frac{1}{2\sigma^2}{\delta^i_{st}}^2}}{\sum_{j=1}^{n} \hat{\lambda^i_t} e^{-\frac{1}{2\sigma^2}{\delta^j_{st}}^2}} \tag{5}$$

$$\hat{\lambda^i_t} = \frac{{\lambda^i_{t-1}}^{\rho}}{\sum_{j=1}^{n} {\lambda^j_{t-1}}^{\rho}} \tag{6}$$

where the responsibility predictor $\hat{\lambda^i_t}$ represents the prior knowledge or belief about module selection, which is used to maintain the temporal continuity of the selected module. Parameter ρ $(0 < \rho < 1)$ controls the effects of past module selections on current decision-making. Parameter vector θ is updated as follows:

$$\theta_i = \theta_i + \alpha \lambda^i_t \delta^i_{st} \frac{\partial F_i(s_{t+1}, s_t, a_t : \theta_i)}{\partial \theta_i} \tag{7}$$

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

5.2 Reward predictor

The reward function $R_i(r', s_t, a_t)$ provides an expected reward r'_t based on the previous state s_t and the action a_t selected at step t. Traditionally, parameters in the reward predictor are updated by the reward prediction error, which comes from the error between reward prediction r'_t and actually received immediate reward r_t after each interaction.

As described in the previous section, the responsibility signal represents how responsible the modules are for the current situation; the temporal difference of the responsibility signals between two steps could be used to reflect whether the current module is more sensitive than the previously selected one. Based on this idea, we utilize a "modular reward" [12], which comes from the immediately received reward and an extra bonus for rewarding the module that better fits the current sub-task. Specifically, this bonus is calculated from the conduct of modular value function and the temporal difference in the responsibility signal, which encourage the learning system to be more focused on the modules they

are responsible for, and lead to the optimality of the composite strategy for the entire task. Basically, the modular reward is calculated as:

$$\hat{r_t^i} = r_t + (\lambda_{t+1}^i - \lambda_t^i)V^i(s_{t+1}) \tag{8}$$

and we have the reward prediction error δ_{rt}^i as follows:

$$\delta_{rt}^i = \hat{r_t^i} - R_i(r', s_t, a_t) \tag{9}$$

Weighted by the responsibility signal and the learning rate, parameters in the reward predictor will be updated as follows:

$$R_i(r', s_t, a_t) = R_i(r', s_t, a_t) + \alpha\lambda_t^i\delta_{rt}^i \tag{10}$$

5.3 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 $Q^i(s_t, a_t)$. The value function $V^i(s_t)$ gives the reward expectation of state s_t at step t and the state-action-value function $Q^i(s_t, a_t)$ provides the reward expectation of action a_t in state s_t. Technically, the goal of reinforcement learning is to improve the policy so that a maximal accumulative reward will be obtained in the long run [17]. The basic strategy of reinforcement learning is to use this value function to estimate the state value under the current policy, and then to improve the policy based on the value function. We define the value function of the state $s(t)$ under the current policy π as:

$$\begin{aligned} V_\pi^i(s_t) &= E_\pi\left[\sum_{k=0}^{\infty}\gamma^k r(t+k)\right] \\ &= E_\pi(R_{t+1} + \gamma V_\pi^i(s_{t+1})|s_t) \end{aligned} \tag{11}$$

where γ is the discount factor that controls the effects of future rewards. Traditionally, $R_{t+1} + \gamma V_\pi^i(s_{t+1})$ is also called the "target value" of state s_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 [17] of value function:

$$V_\pi^i(s_t) = \sum_{a_t \in A}\pi_t^i(a_t|s_t)[R_i(r_t', s_t, a_t) + \gamma \sum_{s_{t+1}\in S} F_i(s_{t+1}, s_t, a_t : \theta_i)V_\pi^i(s_{t+1})] \tag{12}$$

In order to make decisions in each step more coherent with previous ones, we recorded the eligibility traces $e_t^i s$ for each state s, which are updated as:

$$e_t^i(s) = \begin{cases} \eta\gamma e_{t-1}^i(s) & if\ s \neq s_t \\ \eta\gamma e_{t-1}^i(s) + 1 & if\ s = s_t \end{cases} \tag{13}$$

The temporal-difference (TD) error δ_{vt}^i comes from the deviation between the target value and the evaluated value:

$$\delta_{vt}^i = \hat{r_t^i} + \gamma V^i(s_{t+1}) - V^i(s_t) \tag{14}$$

where $\hat{r_t^i}$ is the modular reward and the state value $V^i(s_t)$ will be updated with responsibility signal as:

$$V^i(s_t) = V^i(s_t) + \alpha\lambda_t^i\delta_{vt}^i e_t^i(s) \tag{15}$$

5.4 Action selection

At each step, the action preferences follow a "joint policy", in which state-action values derive from the combination of RL controllers in each module. Specifically, action is decided based on the weighted summation of its value expectation $Q^i(s_t, a_t)$ of future rewards in each module at state s_t. For each candidate action $a_j \in A, (j = 1, \ldots, M)$, their values are calculated as:

$$Q(s_t, a_j) = \sum_{i=1}^{n} \lambda_t^i [R_i(r', s_t, a_j) + \gamma \sum_{j \in X} F_i(s_j, s_t, a_j : \theta_i) V(s_j)] \tag{16}$$

where X is the set of possible states that the agent observes after taking action a_j in state s. In a greedy policy, the action that has the largest value will be selected:

$$a_t = arg \max_{a_j \in A} Q(s_t, a_j) \tag{17}$$

Furthermore, we use a softmax function to explore action space in a stochastic way, where the action a_t is selected by

$$P_i(a_j|s_t) = \frac{e^{\beta Q(s_t, a_j)}}{\sum_{k=1}^{M} e^{\beta Q(s_t, a_k)}} \tag{18}$$

where β is set as $trials/500$ and controls the stochasticity of action selection.

6 Simulation

6.1 Settings

In order to investigate the effectiveness of Inverse Arbi-Q architecture, we use a food-gathering and predator-avoidance task derived from [16,13]. In this carrot-rabbit-predator task, the environment is designed as a 5x5 grid world (as shown in Figure 2. The blue rectangle , purple triangle and green circle represent the rabbit, the predator, and the carrot respectively. Specifically, the rabbit searches carrots with one of eight possible one step actions: {north (N), east (E), south (S), west (W), northeast (NE), southeast (SE), northwest (NW), southwest (SW)} while avoiding being caught by the predator. When the rabbit moves in the location of any of the carrots, it receives a reward of 1.0 and a new carrot will be allocated to an otherwise position. In each step where the rabbit does not find any carrot, it gets a reward of -0.1 to represent increasing hunger. Meanwhile, the predator moves one step directly toward the rabbit while the rabbit moves two steps. If the rabbit avoids the predator, it will receive a reward of 0.5, and a reward of -1.0 if it is caught by the predator, which also means the termination of this trial. In this environment, the observation state of the rabbit and the predator are their absolute coordinates respectively.

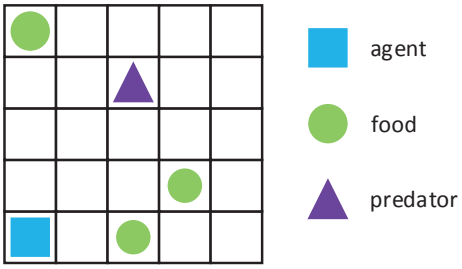
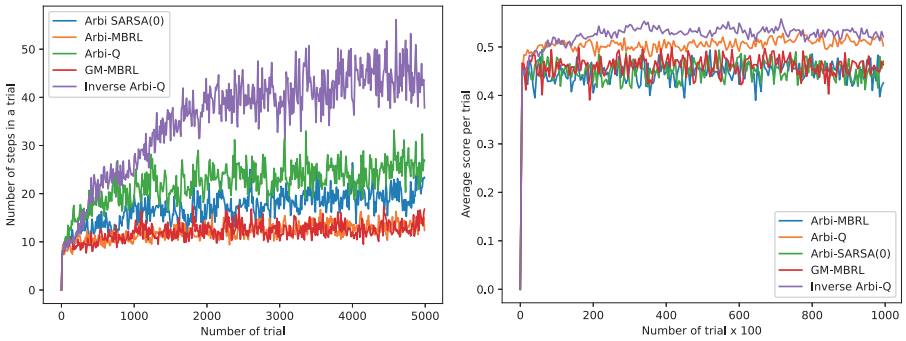


Fig. 2. The food-gathering and predator-avoidance world.

6.2 Results

At the begining, we'd like to compare the performance of Inverse Arbi-Q with previous MRL approaches. Fig. 3(a) shows the average number of steps during 5000 trial epochs in 20 simulation runs. It can be seen that the Inverse Arbi-Q model works significantly better than Arbi-* approaches (Arbi-Q, Arbi-MBRL and Arbi-SARSA(0)) and GM-MBRL.



(a) Averaged step per trial. (b) Averaged score per trial.

Fig. 3. Performance comparison with other MRL approaches.

Furthermore, we utilize another performance metric of "score" from [13] to compare Inverse Arbi-Q with different models. Specifically, the calculation of a score follows this method: for each time that the rabbit finds a carrot, the score is defined as 1.0, however, if it was caught by the predator, the score is 0.0, in situations that the rabbit avoids the predator while no carrot was eaten, the score is set as 0.5. Fig. 3(b) shows the scores between Inverse Arbi-Q with other approaches. We can find that Inverse Arbi-Q receives more scores that other models.

Moreover, we'd like to know the performance of Inver Arbi-Q with incomparable reward scales. Here we increase the avoidance reward by 10 times to observe the performance of Inverse Arbi-Q in the incomparable rewards. As shown in Fig. 4, Inverse Arbi-Q is robust across different incomparable reward scales.

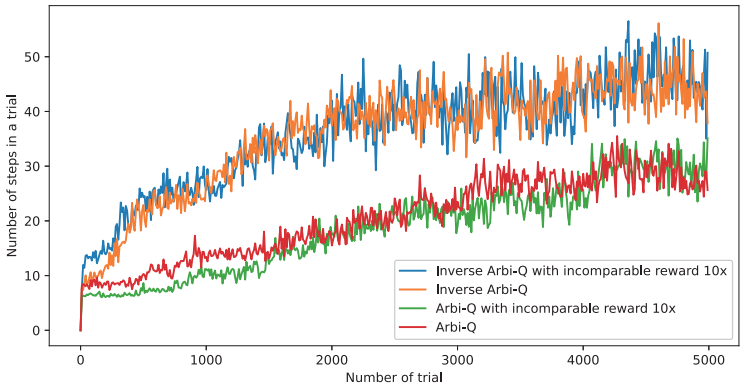


Fig. 4. Performance of Inverse Arbi-Q with incomparable reward scales.

7 Conclusions and future work

In summary, we present the simple yet effective architecture Inverse Arbi-Q to scale up the RL to learn compositional strategies and to exert them appropriately in multi-task learning. Particularly, the responsibility signal, the outcome of a gaussian softmax function of prediction error in each module, was used to weight the outputs of multiple modules, as well as to gate the learning of each module. Furthermore, action preferences are calculated by a "joint policy", which comes from the combination of state-action values in each module multiplied by its responsibility signal. Moreover, the selective update mechanism enables the learning system to align incomparable reward scales between different modules.

Our ultimate goal with modular reinforcement learning is to facilitate the integration of multiple modules into a developmental way, in order to learn compositional strategies, as well as to generate appropriate behaviors according to the specific contexts. Nevertheless, a number of questions stands out as important targets for the next stage of research, such as: (i) the improvement of planning strategy or the proactive way in the evaluation of modular reward; (ii) enabling module selection and responsibility calculation in high dimensionality situations; (iii) the design of bidirectional (top-down and bottom-up) solutions for task decomposition and compositionality. Inspirations from continual learning and meta-learning approaches will be a promising direction for learning tremendous tasks effectively without creating additional modules.

References

1. Bernard, J.A.: Don't forget the little brain: A framework for incorporating the cerebellum into the understanding of cognitive aging. Neuroscience & Biobehavioral Reviews p. 104639 (2022)
2. Botvinick, M.M.: Hierarchical models of behavior and prefrontal function. Trends in cognitive sciences **12**(5), 201–208 (2008)
3. Doya, K., Samejima, K., Katagiri, K.i., Kawato, M.: Multiple model-based reinforcement learning. Neural computation **14**(6), 1347–1369 (2002)
4. Esteban, D., Rozo, L., Caldwell, D.G.: Hierarchical reinforcement learning for concurrent discovery of compound and composable policies. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 1818–1825. IEEE (2019)
5. Gatti, D., Rinaldi, L., Ferreri, L., Vecchi, T.: The human cerebellum as a hub of the predictive brain. Brain Sciences **11**(11), 1492 (2021)
6. Gupta, V., Anand, D., Paruchuri, P., Kumar, A.: Action selection for composable modular deep reinforcement learning. In: Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems. pp. 565–573 (2021)
7. Jacobs, R.A., Jordan, M.I., Nowlan, S.J., Hinton, G.E.: Adaptive mixtures of local experts. Neural computation **3**(1), 79–87 (1991)
8. Logan, G.D., Crump, M.J.: Hierarchical control of cognitive processes: The case for skilled typewriting. In: Psychology of learning and motivation, vol. 54, pp. 1–27. Elsevier (2011)
9. Nagabandi, A., Kahn, G., Fearing, R.S., Levine, S.: Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In: 2018 IEEE International Conference on Robotics and Automation (ICRA). pp. 7559–7566. IEEE (2018)
10. Narendra, K.S., Balakrishnan, J., Ciliz, M.K.: Adaptation and learning using multiple models, switching, and tuning. IEEE control systems magazine **15**(3), 37–51 (1995)
11. Nowlan, S.J., Hinton, G.E.: Evaluation of adaptive mixtures of competing experts. In: NIPS. vol. 3, pp. 774–780 (1990)
12. Samejima, K., Doya, K., Kawato, M.: Inter-module credit assignment in modular reinforcement learning. Neural Networks **16**(7), 985–994 (2003)
13. Simpkins, C., Isbell, C.: Composable modular reinforcement learning. In: Proceedings of the AAAI conference on artificial intelligence. vol. 33, pp. 4975–4982 (2019)
14. Smith, B.J., Read, S.J.: Modeling incentive salience in pavlovian learning more parsimoniously using a multiple attribute model. Cognitive, Affective, & Behavioral Neuroscience pp. 1–14 (2021)
15. Sodhani, S., Zhang, A., Pineau, J.: Multi-task reinforcement learning with context-based representations. In: International Conference on Machine Learning. pp. 9767–9779. PMLR (2021)
16. Sprague, N., Ballard, D.: Multiple-goal reinforcement learning with modular sarsa (0) (2003)
17. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT press (2018)
18. Wang, J.X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J.Z., Munos, R., Blundell, C., Kumaran, D., Botvinick, M.: Learning to reinforcement learn. arXiv preprint arXiv:1611.05763 (2016)