Reinforcement learning from multiple experts demonstrations

Mao Li
University of York
York, United Kingdom
ml1480@york.ac.uk

Yi Wei
Shandong University
Jinan, China
weiyi.ve@mail.sdu.edu.cn

Daniel Kudenko
University of York
York, United Kingdom
National Research University Higher
School of Economics
St Petersburg, Russia
JetBrains Research
St Petersburg, Russia
daniel.kudenko@york.ac.uk

ABSTRACT

A major limitation of most Reinforcement Learning (RL) methods is low sample efficiency. One way to address this problem is to employ human expert demonstrations to speed up the RL process (Reinforcement Learning from Demonstration, or RLfD). The research so far has focused on demonstrations from a single expert, assuming that this expert is good at all aspects of a task. However, little attention has been given to the case where demonstrations are collected from multiple experts, whose expertise may vary on different aspects of the task. In such scenarios, it is likely that the demonstrations will contain conflicting advice in many parts of the state space. In this paper, we propose a two-level Q-learning algorithm, in which the RL agent not only learns the policy of deciding on the optimal action but also learns to select the most trustworthy expert according to the current state. Thus, our approach removes the traditional assumption that demonstrations come from one single source and are mostly conflict-free. We evaluate our technique on two different domains and the results show that the state-of-the-art RLfD baseline fails to converge or performs similarly to conventional Q-learning. In contrast, the performance level of our novel algorithm increases with more experts being involved in the learning process and the proposed approach has the capability to handle demonstration conflicts well.

KEYWORDS

reinforcement learning; Q-learning; Learning from demonstrations

1 INTRODUCTION

Reinforcement learning (RL) is a paradigm of how an agent can learn to maximise its cumulative reward from interactions with the environment. One limitation of most RL methods is their low sample efficiency. In other words, the number of interactions with the environment required to reach a specific performance level during the learning process is always very large.

Reinforcement learning from demonstration (RLfD) is an approach that employs expert demonstrations of solving the target task to guide the reinforcement learning agent (e.g. by biasing the exploration), in order to speed up the learning process and to improve sample efficiency.

The HAT [10] paper listed three techniques to inject human demonstrations into the RL process. (1)Reuse human's policy with a probability p. (2)Give the RL agent an extra reward if the agent makes the same action decision as the human's policy. (3)Extend the Q-table of an agent, adding "choose expert demonstration" into the Q-table as an action, which then becomes an action choice in the RL process.

The SAS algorithm [1] uses reward shaping to incorporate expert demonstrations and is able to ensure the convergence of RLfD. [5] extends RLfD to deep reinforcement learning. CHAT [11] improves HAT by using confidence measurements that avoid being misled by bad demonstrations, and can be considered state-of-the-art.

Overall, current RLfD techniques rely on the quality of the expert demonstrations. An accepted assumption of the above mentioned approaches is that the expert's policy on each state is consistent and beneficial. However, this assumption is rather strong. In most cases, demonstrations can be collected from multiple sources, such as multiple individuals' behaviour records using various heuristic rules. Moreover, the quality of these demonstrations is often imperfect. Thus conflicting advice suggested by different experts' demonstrations may occur in a large number of situations. In this paper, we propose a two-level Q-learning (TLQL) approach to deal with the challenge of conflicting domain knowledge among multiple experts' demonstrations. TLQL includes two Q-tables: a high-level Q-table and low-level Q-table. Compared with traditional Q-learning, it uses an additional Q-table to record the performance of experts in each state. During the RL process, TLQL keeps track of both action quality and the reliability of experts in each state, and updates two Q-tables simultaneously by using feedback signals (i.e. reward) from environment-agent interactions. As a result, TLQL overcomes the problems of learning from multiple demonstrations and thus performs better than state-of-the-art RLfD approaches.

Conflicting demonstrations may occur in various situations. In our experiments, we evaluated TLQL using a maze navigation and a coloured flag visiting domain. In the first domain, experts' demonstrations were conflicting because each expert is reliable in only a section of the maze. In the second domain, contradictions among experts' demonstrations resulted from the different goals of the experts. Each expert's goal was comprised of sub-tasks of the ultimate goal of the domain, and each expert is only good at achieving their individual goal. The empirical results demonstrate that TLQL outperforms Q-learning without prior knowledge and CHAT in several metrics (e.g. jumpstart and overall performance). Furthermore, we show that the TLQL agent performs better with an increasing number of experts.

2 PRELIMINARIES

2.1 Reinforcement learning

Reinforcement learning (RL) is a paradigm to compute an optimal decision sequence to maximise the cumulative reward in Markov decision processes (MDPs)[9]. An MDP is composed of five parts, noted as $\langle S, A, T, R, \gamma \rangle$: S is the set of states, each state representing a snapshot of the environment at a given time; A is the set of all actions that an agent can execute; T is the transition function indicating the probability of state s changing to state s' via action a. R is the reward function which denotes the reward (a positive number) or punishment (a negative number) signal for each agent-environment interaction; γ is the discount factor balancing short-term and long-term gains.

An RL agent observes a state from the environment and makes a decision based on the state. It will then receive a numerical reward signal, and the current state of the environment will be transitioned to another state. This process, named agent-environment interaction, generates a sample $\langle s_t, a_t, r_t, s_{t+1} \rangle$ at time step t. The RL agent uses Q-learning to update the Q-value $Q(s_t, a_t)$, which is an estimate of the expected discounted cumulative reward when executing action a_t in state s_t and then following the optimal policy.

In this paper, we focus on Q-learning[12], a model-free RL technique. Q–learning updates the Q-function based on the TD error between the estimated Q-value and the current experience sample and the estimated predicted Q-value of next state. For each sample, the algorithm updates Q-value as follows:

$$Q(s, a) \leftarrow Q(s, a) + \alpha(R + \gamma \max_{a'} Q(s', a') - Q(s, a))$$

2.2 Reinforcement Learning from Demonstration

In the RL process, the reward signal is often sparse and the feedback based on an episode is frequently delayed. In addition, the "curse of dimensionality" is an issue of large state spaces. As a result, the sample complexity of Q-learning is fairly high. Therefore, improving the sample efficiency, i.e. decreasing the number of experience samples necessary to learn the optimal policy, is essential. RL from Demonstration (RLfD) is an approach that aims at improving Q-learning's efficiency by incorporating expert demonstrations. In RLfD the RL agent utilises the reward signal from the environment as a ground truth and applies demonstration trajectories as heuristic suggestions to bias the exploration of the state space.

Early research of RLfD includes [8] using demonstrations to initialise Q-values to speed up the Q-learning process. [10] proposed human-agent transfer (HAT) to transmit human knowledge to an RL agent. HAT extracts demonstration's information via a classifier model such as a decision tree. The demonstration information is then integrated into the RL process. [10] lists three methods to inject human knowledge into RL process: value bonusextra action and probabilistic policy reuse. The Value Bonus approach gives the RL agent an extra reward when the agent makes the same action decision as the classifier. However, changing the reward directly may cause Q-learning to lose its convergence guarantees [7]. Brys et al. proposed the algorithm of Similarity Based Shaping (SBS) [1] to deal with this problem by employing potential-based reward shaping [13][2] into the value bonus approach. The SBS algorithm adopts the state-action Gaussian distance as a potential function to shape the reward function, which maintains the theoretical convergence guarantees.

In real world applications, demonstrations are usually from different experts and may conflict with each other, for example as a result of low-quality demonstrations. If we directly use SBS and ignore the emerging conflicts, SBS is not able to learn in scenarios, where the demonstrations contradict each other in a large proportion of states, as will be demonstrated later in the experiments section.

Our novel method can deal with conflicts in the demonstrations by learning a trust value of experts in each state. We employ a separate Q-table to maintain the value of $\langle state, experts \rangle$. The RL agent itself is also represented in the expert Q-table. Therefore, when the RL agent learns enough about the environment and surpasses the performance of other experts, it would stop using the information coming from demonstrations, and converge to the optimal policy in the same way as basic Q-learning.

2.3 Other Related Work

Hierarchical reinforcement learning (HRL) is another approach to deal with low sample efficiency of RL. HRL approaches such as HTD [6] and MAXQ [3] employ action abstractions and require the programmer to divide the task into sub-tasks. Consequently, the agent could learn polices on different levels of abstractions. A major difference between our proposed algorithm and hierarchical RL is that our approach focuses on learning trust values for different knowledge sources in different states and does not require to manually define abstractions.

Probabilistic policy reuse (PPR) [4] deals with low sample efficiency of RL by reusing sub-optimal policies to bias the exploration. PPR does this by reusing the action from the policy with the maximum performance. PRQ [4] applies the softmax exploration strategy to select the respective policy. Unlike our proposed approach, policy reuse assumes that the performance of each policy is known. Later research on policy reuse focuses on transferring policies from similar domains to speed up learning in the target domain.

3 LEARNING FROM MULTIPLE DEMONSTRATIONS

This section first introduces the phenomenon of conflicts between demonstrations from different sources. Following that, our two-level Q-learning algorithm (TLQL) is proposed to address this challenge.

3.1 Multiple domain knowledge sources

In many applications, such as training an agent to play chess from human demonstrations, the knowledge comes from different individuals with different demonstration trajectories, heuristic rules, etc. Each demonstration may therefore produce different actions and conflict with other demonstrations in some states.

Nevertheless in early stages of learning, it is beneficial for the RL agent to follow suggestions from experts rather than randomly explore without extra information. The problem is to determine which suggested action from different experts can be trusted in the case of conflicting advice. One feasible idea is to investigate each $\langle state, expert \rangle$ pair by trial and error. The proposed novel method

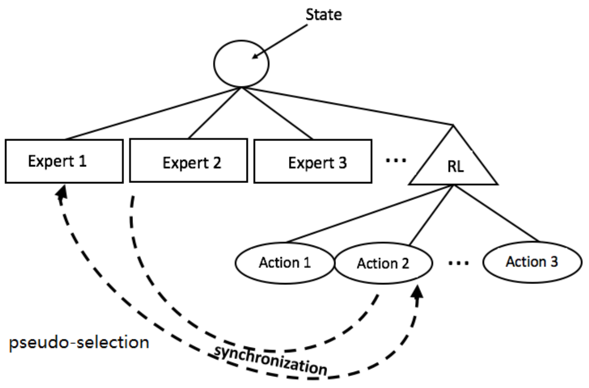

Figure 1: The structure of 2-level Q-learning

applies Q-learning to expert selection and learns a policy of assigning credit to experts. Combining both Q-learning of experts and Q-learning of actions simultaneously, the agent has the capability to effectively deal with conflicts and improve the sample efficiency of Q-learning even where demonstrations come from multiple conflicting sources.

3.2 Two-level structure of reinforcement learning

Figure 1 shows the structure of the proposed novel algorithm: two-level Q-learning (TLQL). This algorithm employs a low-level Q-table and a high-level Q-table. The high-level Q-table is used to store the value of $\langle state, expert \rangle$, representing the trust the RL agent has in the given expert, in the given state. The low-level Q-table is the same as in regular Q-learning, recording the Q-value of state-action pairs.

In the process of agent-environment interaction, the agent firstly observes the state of the environment and then selects an expert by an ϵ-greedy policy according to the high-level Q-table (the RL agent itself is also represented as an expert in this Q-table). Afterwards, it executes the action that the selected expert suggests, and finally receives the reward and next state feedback from the environment. The sample of the new algorithm is $\langle s, e, a, r, s^0 \rangle$, where e is the selected expert and s, a, r, and s^0, are same as in regular Q-learning, denoting current state, action, reward and the next state, respectively.

3.3 Synchronised Q-table updating

The TLQL algorithm updates information by exploring both experts and actions with an experience sample $\langle s, e, a, r, s^0 \rangle$. Firstly, the algorithm updates the low-level Q-value by $\langle s, a, r, s^0 \rangle$ in the same way as regular Q-learning does.

In order to share information between the high-level Q-table and the low-level Q-table, as well as make full use of the information of every sample, it is also necessary to update the low-level Q-table in addition to the high-level Q-table with the sample $\langle s, e = E, a, r, s^0 \rangle$. After updating the low-level Q-table, the value of the RL agent $highQ(s, RL)$ and $highQ(s, e_i) : e_i(s) = a$ is set as the Q-value of its policy $\max_a lowQ(s, a)$. Thus the high-Q value of the RL synchronises with $\max_a lowQ(s, a)$.

3.4 Pulling it all together

Algorithm 1 shows the pseudo code of TLQL, indicating how all components of figure 1 work on an algorithmic scale.

Algorithm 1 Two-level-Q-learning

```
procedure TWO-LEVEL-Q-LEARNING($\langle s_t, a_t, \widehat{q_t} \rangle$, $PQ$)
    Let $E$ be the set of experts, including the RL agent
    for all $s \in S, a \in A$ : $lowQ(s, a) \leftarrow 0$
    for all $s \in S, e \in E$ : $highQ(s, e) \leftarrow 0$
    for each episode do:
        Initialise $s_0$
        for each step in episode do:
            $\epsilon$-greedily choose expert $e$ from E using $highQ$
            action $a$ is the suggestion from expert $e$
            Take action $a$, observe $r$, $s^0$
            predicted $\leftarrow r + \gamma \max_{a^0} lowQ(s^0, a^0)$
            $\Delta_l = predicted - lowQ(s, a)$
            $lowQ(s, a) \leftarrow lowQ(s, a) + \alpha\Delta_l$
            highQ(s,RL) $\leftarrow \max_{a^0} lowQ(s, a^0)$
            if e=RL then
                for each expert e apart from RL do:
                    if $e(s) = a$ then
                        highQ(s,e) $\leftarrow \max_{a^0} lowQ(s, a^0)$
```

4 EXPERIMENTS

In this section, we present results of TLQL proposed in two domains: maze navigation and coloured flags visiting. To demonstrate that TLQL can significantly improve reinforcement learning by utilising demonstrations from multiple conflicting sources, we define several domain experts with different expertise in each domain.

In the domain of maze navigation, the maze is divided into three non-overlapping regions. Each expert is good at moving in a specific region and has no prior knowledge of other regions (note that this fact is not known to the experts and to the TLQL algorithm). In the domain of coloured flags visiting, the learning task and demonstration is more complicated. A training agent learns to finish a composite task in a grid world. Each expert is skilled in one sub-task and all sub-tasks have the same state space. Because each expert only concerns on its individual target, experts may make different decisions in a state. Thus many conflicted but local-optimal demonstrations will be recorded.

Unlike hierarchical RL, the partition of the domain is used to simulate the experts' different skills, the agent does not know the partition information in advance. The agent learns the strengths of each expert in high-level Q-learning.

In our experiments, we compared TLQL with two baselines: traditional reinforcement learning (RL) and confidence-based human-agent transfer (CHAT)[11]. The former one uses Q-learning approach without any prior knowledge. CHAT is the state-of-the-art approach of improving reinforcement learning from demonstration. As CHAT does not consider conflicting demonstrations caused by multiple domain knowledge sources, we adopt weighted random policy to make choices for CHAT when a contradiction occurs.

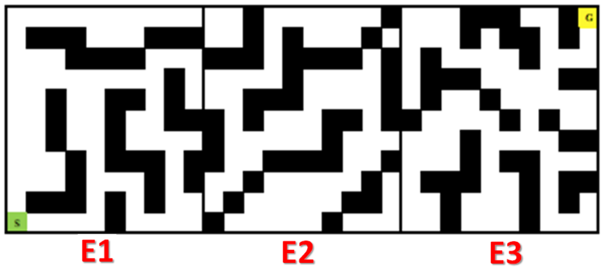

Figure 2: Maze with three areas. The initial state is marked 'S' in area 1 and the goal state is marked as 'G'in area 3. The goal of agent is to move from 'S' to 'G' as quickly as possible. Black grids are walls that the agent cannot go through.

Specifically, when multiple experts made different decisions in a state, each action is given a weight based on how many experts suggest this action. Then an action is chosen randomly based on these weight values: actions with higher weights will be chosen with higher probability.

All reported results in our experiments are averaged over 100 trials. All result figures display a 99% confidence interval to show statistical significance.

4.1 Maze navigation

The first experiment is with a maze environment as shown in Figure 2. The maze consists of 30*10 states. In each state, the agent has four available actions: up, down, left and right. It can move to one of the four directions as long as there is no obstacle in that particular direction. Furthermore, there is a probability of 0.1 that the agent fails to move toward its desired direction. The agent's goal is to reach the upper right corner of the maze as soon as possible starting from the bottom left corner. The immediate reward for the agent is 0, unless the agent arrives at the goal state, where it is +1. Each episode starts from the initial state S and ends with the goal state G. The parameter settings of Q-learning are the same for all approaches: learning rate α=0.01, discount factor γ=0.99, ϵ-greedy is adopted as the exploration strategy, where ϵ=0.1.

The demonstrations are collected from multiple experts. In this experiment, there are three experts E1, E2 and E3, each claiming that they have enough experience to complete the maze. As figure 2 shows, the maze has been divided into 3 areas. Each expert is only a master in one are of the maze. However, the RL agent does not know this in advance. We assume that E1, E2 and E3 know the optimal policies of area 1, area 2 and area 3 respectively. They can give optimal actions as the demonstration with a probability of 0.9 in their corresponding areas. Furthermore, because each expert knows nothing about the maze except their area of expertise, they move randomly in the other two areas. From the perspective of the learning agent, the experts' ability is unknown. Besides, the learning agent does not know the confidence of experts' demonstrations. When conflicting actions are suggested by different experts, the learning agent is unable to know whose demonstration is right, and learn it during the RL process.

For demonstration data collection, we generated 20 full demonstrations from each expert via behaviour simulations, and removed any duplicates. When applying TLQL to the maze navigation game with conflicting demonstrations, the high-level Q-learning is used to teach the agent which expert's demonstration is more reliable in each state. The low-level Q-learning teaches the agent to move in the optimal direction through its interactions with the maze.

Figure 3 illustrates the learning curves of TLQL and two other baselines: RL without prior knowledge and CHAT. TLQL and CHAT use demonstrations from three experts. Figure 3 shows the changes of cumulative reward. The figures clearly show that TLQL significantly outperforms RL and CHAT from several perspectives. Jumpstart is used to measure the average initial performance of the learning agent. A higher jumpstart performance means that the learning agent can benefit more from its prior knowledge in early stages of the learning. As TLQL can make good use of conflicting demonstrations to train the agent, its jumpstart performance is significantly better than the baselines. The overall performance is another metric which is measured by the area under the cumulative reward curve. Figure 3 indicates that no matter how many demonstrators are involved in the model, TLQL always performs better than RL and CHAT. In addition, as the agent trained by TLQL can achieve a relatively good performance very quickly, its asymptotic performance is superior to RL and CHAT.

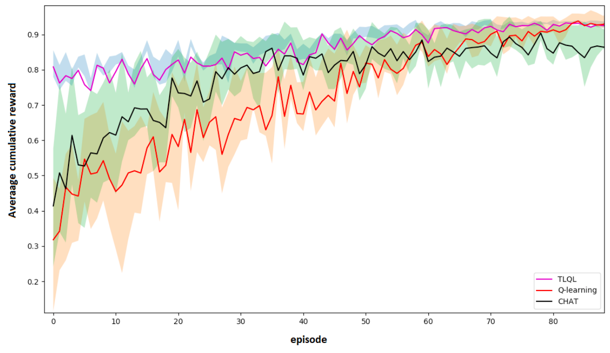

Figure 3: Performance comparison of TLQL and two baselines in the domain of maze navigation. The graph shows the average cumulative reward for an episode and the number of training episodes.

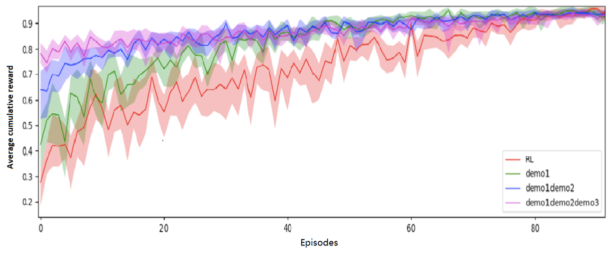

Figure 4: Comparison of TLQL with different number of experts (i.e. TLQL with expert 1, TLQL with expert 1 and expert 2, and TLQL with three experts) and RL in the domain of maze navigation

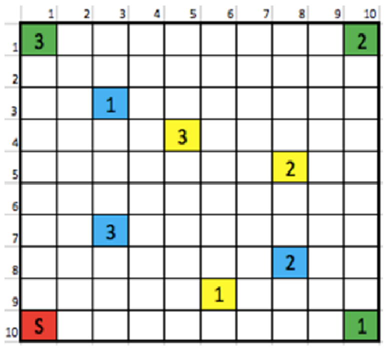
Figure 5: Coloured flags visiting problem domain. The learning agent's starting location is labelled with S. Its goal is to visit 9 coloured and numbered flags that are located in a 10*10 grid world.

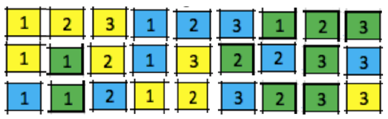
Figure 6: Three correct sequence of visiting flags

Moreover, we also compared the learning performance of TLQL with different demonstrators. Figure 4 shows that the training agent can perform better by increasing the number of demonstrators. Although more demonstrators imply more contradictions among demonstrations, TLQL can deal with conflicting demonstrations effectively. Therefore, the learning agent can gain more knowledge from the demonstrations involving more experts.

4.2 Coloured flags visiting

In the previous experiment, all experts shared the same goal (i.e. completing the maze) and they were skilled in different areas of the state space. However, domain knowledge conflicts may result from more complex scenarios. For example, in the Super Mario game collecting coins, killing enemies and moving towards the goal are three different tasks. Players perform all of these tasks to obtain a higher score, which is the ultimate goal of the game. Supposing there are three experts. Each of them is only good at achieving one specific task while ignoring the other tasks of the goal. In this case, experts with different knowledge may suggest different optimal actions for the same state. Our two-level Q-learning algorithm can also handle this kind of conflicting demonstration problem.

In our experiment, we chose coloured flags visiting as the domain for the aforementioned scenario. In coloured flags visiting an agent's goal is to visit all flags in a given order in a discrete 10*10 grid world. A picture of this domain is shown in Figure 5. There are a total of 9 flags of three different colours and labelled with digits. An agent starts from the "S" position which is located at the bottom left corner of the maze. At each time step, the agent can move in 8 directions: up, down, left, right, left up, left down, right up and right down. There is a probability of 0.1 that the agent's move in the desired direction is unsuccessful and the agent remains in its original

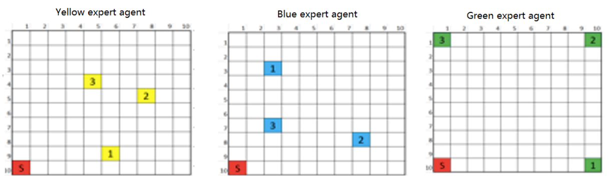

Figure 7: Three demonstrators and their view of the environment

position. The agent's goal is to visit all flags as soon as possible and also obey some rules: (1) the agent needs to visit all flags of the same colour in ascending order; (2) for flags with different colours, there is no order requirement; (3) incorrect (i.e. out of order) visits of flag positions are ignored and not taken into account. Figure 6 shows three correct examples satisfying the above rules. An entire episode is finished when the agent has visited all flags in the right order. At this time, the agent receives a reward equal to +1. No other rewards are given during the episode. Like in the first experiment, we set learning rate =0.01 and discount factor =0.99 for all approaches. We also adopt ϵ-greedy as the exploration strategy, where =0.1.

We define three experts (denoted by E1, E2 and E3) to provide demonstrations for playing this flag visiting game. E1, E2 and E3 are skilled in visiting yellow, blue and green flags respectively. Note that each expert's goal and the learning agent's goal are not the same. The learning agent is required to visit all flags while each expert only needs to visit the flags with one of the three colours. Thus every expert only focuses on how to complete its individual task as soon as possible rather than the ultimate goal of the game. Figure 7 depicts expert E1, E2 and E3 and the environment from their perspective.

20 demonstrations were generated from each of the three experts, with 10% random noise added in each. As E1, E2 and E3 have different goals, they are likely to take different actions for a given cell of the grid. From the perspective of the learning agent, all these suggested actions (i.e. demonstrations) could be helpful for achieving its goal. This is because each one of the suggested actions is an optimal choice for a specific task. Faced with the conflicting demonstrations from E1, E2 and E3, the learning agent needs to learn from these demonstrations and decide what action to take in each state. Incorporating noise into demonstrations, we assume that experts cannot give the optimal actions with a probability of 0.1, and instead propose a random action.

Performance comparisons regarding the cumulative reward and the number of steps of TLQL and the two baselines (i.e. regular Q-learning with no prior knowledge and CHAT) are shown in Figure 8. Like the results of experiment 1, learning curves of TLQL still outperform the curves of regular Q-learning and CHAT. Furthermore, Figure 9 shows that the learning performance becomes better with an increasing number of experts.

The reason for the much higher initial performance with three experts (than, e.g., with two experts) is that without having demonstrations of all three experts, there is a crucial part of the task information missing. Just using the advice of two experts is not sufficient to complete the overall task immediately. This is different from the maze navigation domain, where one expert alone can still help the agent to complete the overall task.