The Knowledge Engineering Review, Vol. 34, e14, 1 of 12.
doi:10.1017/S0269888919000092

Two-level Q-learning: learning from conflict demonstrations

MAO LI[1], YI WEI[2] and DANIEL KUDENKO[1,3]

[1]*Computer Science Department, University of York, York, United Kingdom*
e-mail: ml1480@york.ac.uk
[2]*Computer Science Department, Shan dong University, Jinan, China*
e-mail: weiyi.ve@mail.sdu.edu.cn
[3]*JetBrains Research, St Petersburg, Russia*
e-mail: daniel.kudenko@york.ac.uk

Abstract

One way to address this low sample efficiency of reinforcement learning (RL) is to employ human expert demonstrations to speed up the RL process (RL from demonstration or RLfD). The research so far has focused on demonstrations from a single expert. 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. 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 three 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.

1 Introduction

Reinforcement learning (RL) is a paradigm of how an agent can learn to maximize 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 RL agent (e.g. by biasing the exploration), in order to speed up the learning process and to improve sample efficiency.

The human-agent transfer (HAT) (Taylor *et al*., 2011) 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, and (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 similarity-based shaping (SBS) algorithm (Brys *et al*., 2015) uses reward shaping to incorporate expert demonstrations and is able to ensure the convergence of RLfD. Hester *et al*. (2018) extend RLfD to deep reinforcement learning (DRL). Confidence-based human-agent transfer (CHAT) (Wang and Taylor, 2017) improves HAT 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 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, a coloured flag visiting domain and the Atari game of Pong. In the first domain, each expert is reliable in only a section of the maze. In the second domain, 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 last domain, Pong, is a popular benchmark domain for DRL. 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

RL is a paradigm to compute an optimal decision sequence to maximize the cumulative reward in Markov decision processes (MDPs) (Sutton and Barto, 1998). 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 (Watkins and Dayan, 1992), a model-free RL technique. Q-learning updates the Q-function based on the temporal difference (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-values as follows:

$$Q(s, a) \leftarrow Q(s, a) + \alpha(R + \gamma \max_{a'} Q(s', a') - Q(s, a)) \tag{1}$$

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, that is, decreasing the number of experience samples necessary to learn the optimal policy, is essential. RLfD is an approach that aims at improving Q-learning's efficiency by incorporating expert demonstrations. In RLfD, the RL agent utilizes 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.

https://doi.org/10.1017/S0269888919000092 Published online by Cambridge University Press

Early research of RLfD includes (Schaal, 1997) using demonstrations to initialize Q-values to speed up the Q-learning process. Taylor *et al*. (2011) proposed 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. Taylor *et al*. (2011) list three methods to inject human knowledge into RL process: value bonus, extra action and probabilistic policy reuse (PPR). 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 (Ng *et al*., 1999). Brys *et al*. proposed the algorithm of SBS (2015) to deal with this problem by employing potential-based reward shaping (Wiewiora *et al*., 2003; Devlin and Kudenko, 2012) 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 Section 4.

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 HTG (Kaelbling, 1993) and MAXQ (Dietterich, 2000) 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 HRL is that our approach focuses on learning trust values for different knowledge sources in different states and does not require manually defined abstractions.

PPR (Fernández and Veloso, 2006) 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. PPR (Fernández and Veloso, 2006) 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.

Deep Q-learning from demonstration (Hester *et al*., 2018) applied a small demonstration dataset to pre-train the Q-network before starting any interaction with the environment. Then it mixes TD-loss, supervised loss and the L2 regularization loss on Q-net to update the network. Potential conflicts in the demonstrations which is the key topic in this paper has been ignored.

3 Learning from multiple demonstrations

This section first introduces the phenomenon of conflicts between demonstrations from different sources. Following that, our TLQL algorithm is proposed to address this challenge.

3.1 Multiple domain knowledge sources

In many applications, such as trainingIn 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

https://doi.org/10.1017/S0269888919000092 Published online by Cambridge University Press

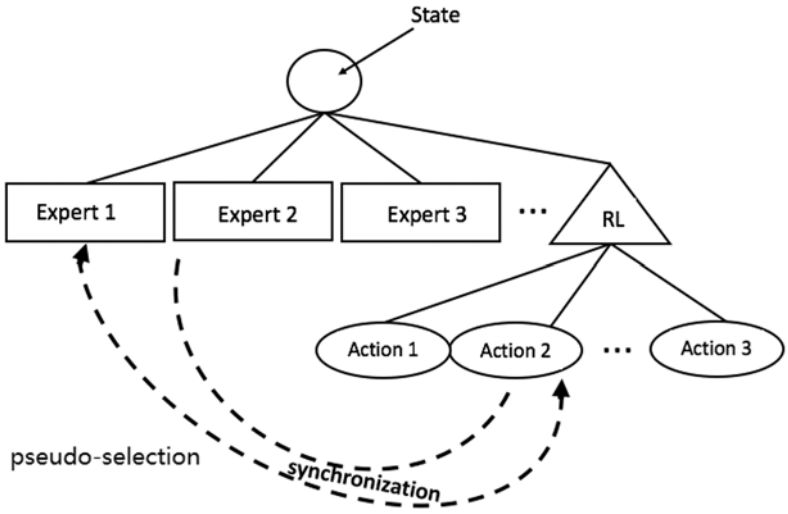

Figure 1 The structure of two-level Q-learning

suggested action from different experts can be trusted in which states, especially in the case of conflicting advice. One feasible idea is to investigate each ⟨*state*, *expert*⟩ pair by trial and error. The proposed novel method applies Q-learning to expert selection and learns a policy of assigning credit to experts. By combining both Q-learning of choosing expert advice and Q-learning of actions in a simultaneous fashion, the agent has the capability to effectively deal with conflicting demonstrations and improve the sample efficiency of Q-learning. In this paper, we propose such an approach which enables an agent to exploit demonstrations from multiple sources, even if they are conflicting.

3.2 Two-level structure of reinforcement learning

Figure 1 shows the structure of the proposed novel algorithm: TLQL. This algorithm employs a low-level Q-table (lowQ) and a high-level Q-table (highQ). The high-level Q-table is used to store the value of a ⟨*state*, *expert*⟩ pair, 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 via an ϵ-greedy policy according to the high-level Q-table (defined over state-expert pairs). Note that the RL agent itself is also represented as an expert in this Q-table.

An experience sample of this two-level algorithm is therefore $\langle s, e, a, r, s' \rangle$, where e is the selected expert and s, a, r and s' are same as in regular Q-learning, denoting current state, action, reward and the next state, respectively.

If the expert chosen is the same as the RL agent, then a normal Q-learning step is executed, that is, lowQ(s,a) is updated according to the regular Q update equation.

When an expert is chosen that is not the RL agent itself, the RL agent executes the action that the selected expert suggests, and receives the reward and observes the next state in the environment.

The TLQL algorithm updates information by exploring both experts and actions with an experience sample $\langle s, e, a, r, s' \rangle$. Firstly, the algorithm updates the low-level Q-value, that is, lowQ(s,a), using the experience tuple $\langle s, a, r, s' \rangle$ in the same way as regular Q-learning does.

In order to synchronize 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 at each learning step. If an expert suggested the same action as the one that has been executed (i.e. action a), then highQ(s,e) is assigned the value of the new lowQ(s,a).

Finally, highQ(s,RL), where RL represents the RL agent itself, is updated to $max_a lowQ(s, a)$.

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

https://doi.org/10.1017/S0269888919000092 Published online by Cambridge University Press

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 $\in \mathcal{S}$, $a \in \mathcal{A}$: $lowQ(s, a) \leftarrow 0$
for all s $\in \mathcal{S}$, $e \in E$: $highQ(s, e) \leftarrow 0$
for each episode **do**:
  Initialise s_0
for each step in episode **do**:
  ϵ-greedily choose e from $highQ$
  action a is the suggestion from expert e
  Take action a, observe r, s'
  predicted $\leftarrow r + \gamma \max_{a'} lowQ(s', a')$
  $\Delta_l = predicted - lowQ(s, a)$
  $lowQ(s, a) \leftarrow lowQ(s, a) + \alpha\Delta_l$
  $highQ(s, RL) \leftarrow \max_{a'} lowQ(s, a')$
  for each expert e in E **do**:
    if expert e suggest action a **them**
      $highQ(s, e) \leftarrow lowQ(s, a)$
    end if

4 Experiments

In this section, we present results of TLQL proposed in three domains: maze navigation, coloured flags visiting and Atari game Pong. To demonstrate that TLQL can significantly improve RL by utilizing 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 is only concerned with its individual target, experts may make different decisions in a state. Thus, many conflicted but local-optimal demonstrations will be recorded.

Unlike HRL, 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 RL and CHAT (Wang and Taylor, 2017). The former one uses Q-learning approach without any prior knowledge. CHAT is the state-of-the-art approach of improving RLfD. 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. 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

https://doi.org/10.1017/S0269888919000092 Published online by Cambridge University Press

Figure 2 Maze with three areas

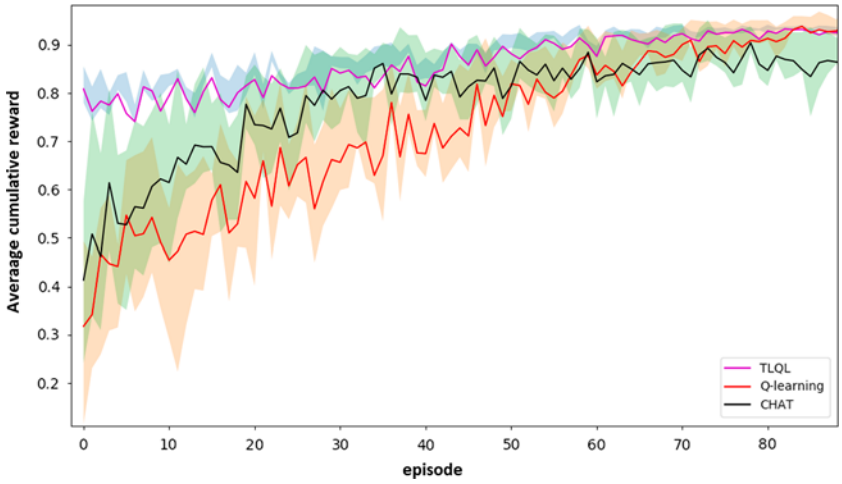

Figure 3 Performance comparison of two-level Q-learning and two baselines in the domain of maze navigation

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 towards 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 $\alpha = 0.01$, discount factor $\gamma = 0.99$, ϵ-greedy is adopted as the exploration strategy, where $\epsilon = 0.1$.

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 three areas. Each expert is only a master in one area 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 demonstrations from each expert via behaviour simulations of a complete episode 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

https://doi.org/10.1017/S0269888919000092 Published online by Cambridge University Press

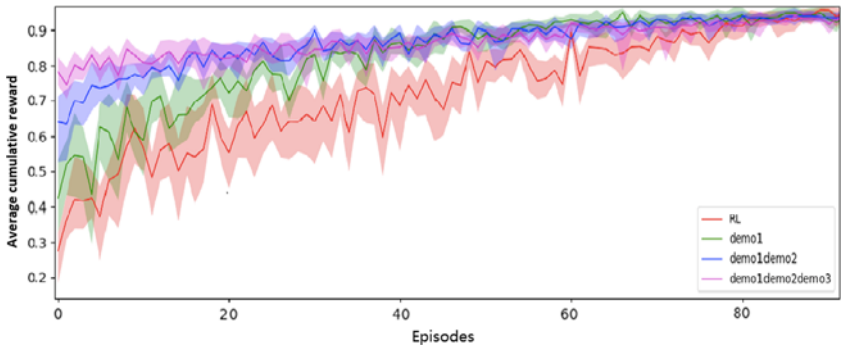

Figure 4 Comparison of two-level Q-learning with different number of experts (i.e. two-level Q-learning with expert 1, two-level Q-learning with expert 1 and expert 2, and two-level Q-learning with three experts) and reinforcement learning in the domain of maze navigation

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.

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 TLQL 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 nine 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 eight 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 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

https://doi.org/10.1017/S0269888919000092 Published online by Cambridge University Press

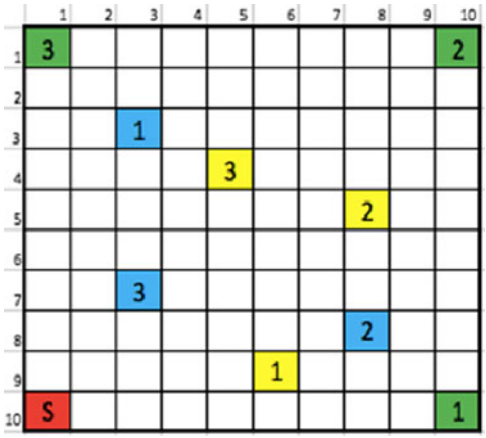

Figure 5 Coloured flags visiting problem domain

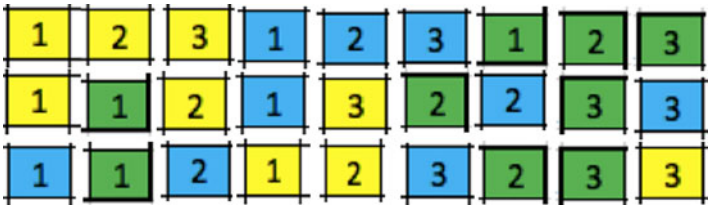

Figure 6 Three correct sequence of visiting flags

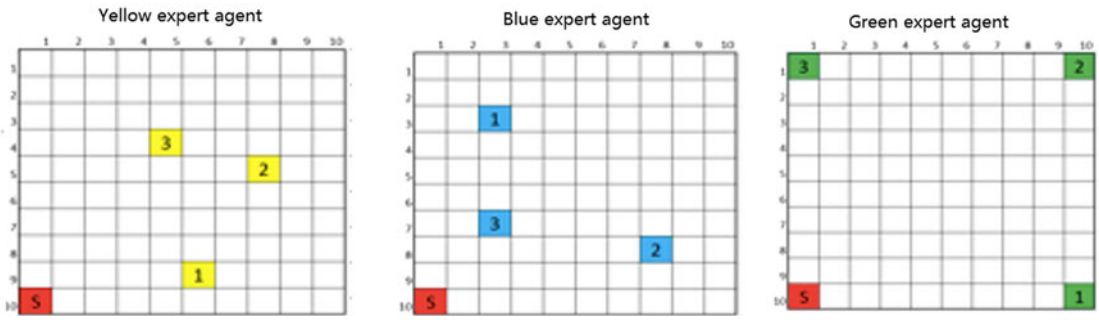

Figure 7 Three demonstrators and their view of the environment

the episode. The state information includes the position of the agent and how many flags were collected for each colour. Like in the first experiment, we set learning rate $\alpha = 0.01$ and discount factor $\gamma = 0.99$ for all approaches. We also adopt ϵ-greedy as the exploration strategy, where $\epsilon = 0.1$.

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.

Twenty demonstrations, each completing an episode, 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

https://doi.org/10.1017/S0269888919000092 Published online by Cambridge University Press

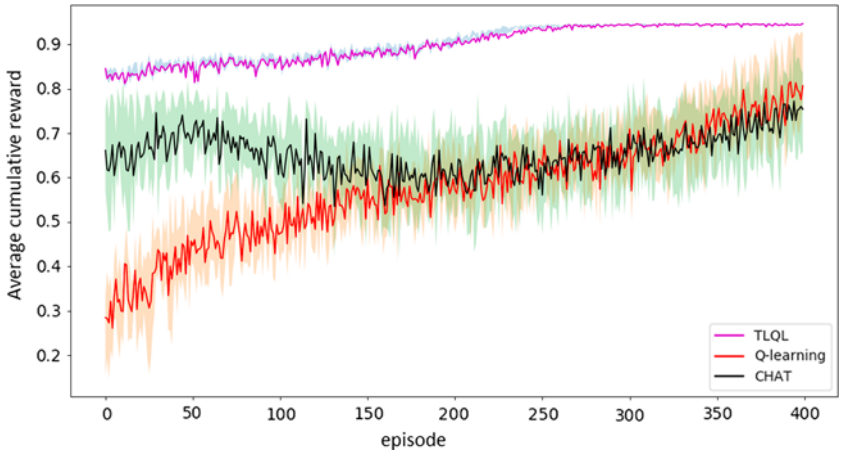

Figure 8 Performance comparison of two-level Q-learning and two baselines in the domain of coloured flags visiting

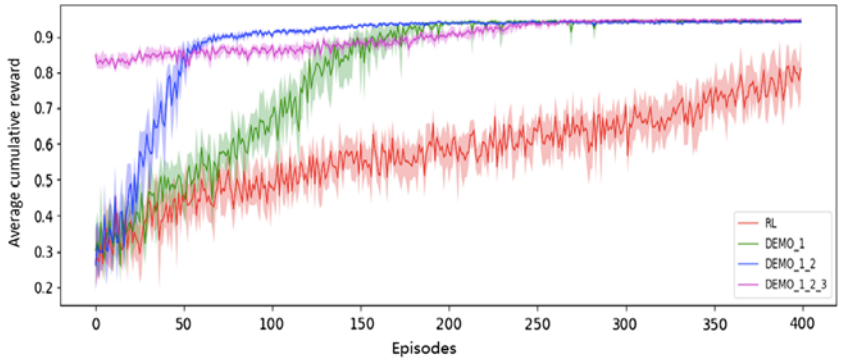

Figure 9 Comparison of two-level Q-learning with different number of experts and regular Q-learning in the domain of coloured flags visiting

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.

4.3 Pong

Since the deep Q-learning (DQN) (Mnih *et al.*, 2015) paper was published, Atari games became a popular benchmark for RL. We have tested the TLQL on the Pong (Figure 10) in the environment openAI gym (Brockman *et al.*, 2016). Figure 11 shows the structure of neural network used to train DQN.

We collected two demonstration sets from two imperfect agents: a human player and a rule-based agent. Each demonstration set contains 420 Pong games (i.e. complete episodes including 2k transitions). The human and rule-based agent show vastly different play behaviour. For each of these demonstration

https://doi.org/10.1017/S0269888919000092 Published online by Cambridge University Press

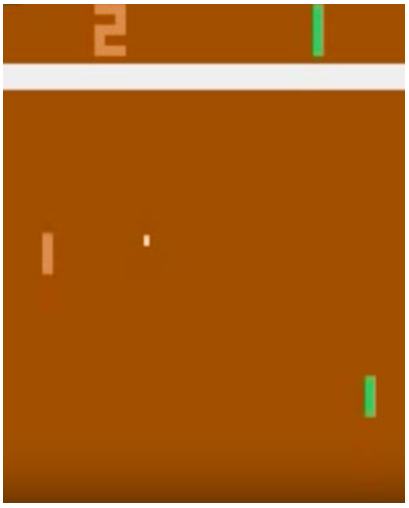

Figure 10 Pong: A game of Atari 2660

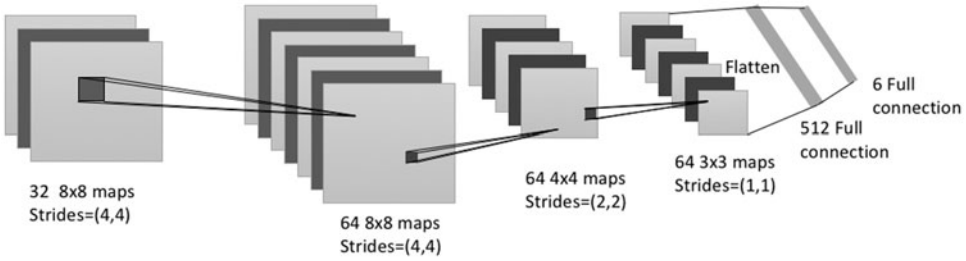

Figure 11 Structure of neural network of deep Q-learning and expert

sets, we used a convolutional neural networks (CNNs) with the structure shown in Figure 11 to learn a state to action mapping, resulting in a so-called human net and rule-based agent net. The average performance for rule-based agent and human player is -13 and -2 separately. In addition, we trained another CNN with the same structure from the union of both demonstration sets as a baseline.

We then used the human net and rule-based agent net in the TLQL approach as expert 1 and expert 2. Note that the experts (i.e. the CNNs) are now treated as oracles, unlike in the other two evaluation domains. In DRL setting, the high-level Q-table is a CNN whose input is an image and output are high Q-values (values of <*state, expert*>) of 3 experts: expert 1, expert 2 and DQN agent itself.

To trade off exploration and exploitation, we apply exponential decay on ϵ, a hyper-parameter of $\epsilon - greedy$. Equation (2) shows exponential decay of ϵ.

$$\epsilon = \frac{start - final}{e^{-\frac{ls}{ds}}} \tag{2}$$

- $start = 0.1$ is initial value of ϵ;
- $ds = 3000$ is number of steps using decay ϵ;
- $final = 0.0001$ is the ϵ value after $decay_steps$ steps;

We applied Adam (Kingma and Ba, 2014) as the optimizer and learning rate $lr = 0.0001$. Other hyper-parameters are

- batch size of memory replay: 32.
- Discount factor $\gamma = 0.99$.
- Replay memory size: 10,000.

Figure 12 shows the performance of DQN without demonstration, DQN with TLQL and DQN with CHAT. The results are similar to the ones in the maze and flag visiting domains. With TLQL, the agent

https://doi.org/10.1017/S0269888919000092 Published online by Cambridge University Press

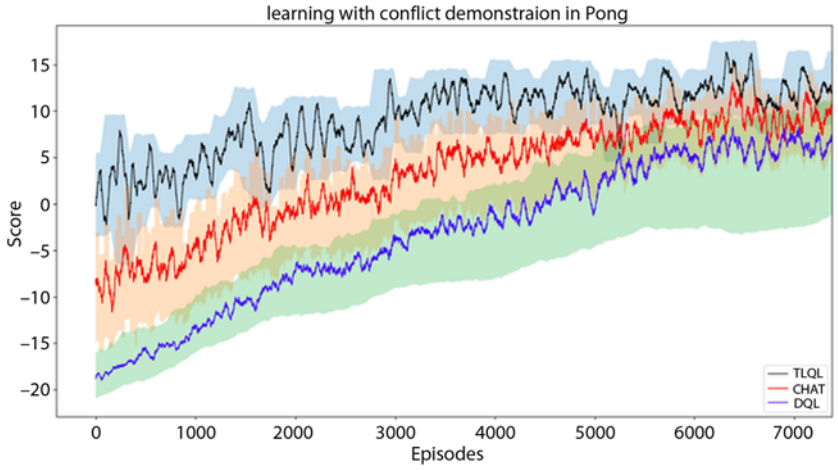

Figure 12 Result of Pong

can overcome the problem of conflicting demonstrations and speed up the learning process by learning when to trust each experts. Overall, TLQL significantly outperforms CHAT and the original DQN.

5 Conclusion

In this paper, we propose a novel algorithm, TLQL, that incorporates demonstrations from multiple experts with varying expertise into RL. The expert demonstrations are used to bias the exploration of the RL agent. The TLQL algorithm adds a Q-table to record the degree of trust in an expert for different states. The RL agent learns a policy of selecting an expert simultaneously to learning the policy for maximizing the cumulative reward. To keep the high-level and low-level Q-tables synchronized, we update both those to Q-tables in each agent–environment interaction.

Notably, the value of $\langle state, RL \rangle$ and values of $\langle state, otherexpert \rangle$ will be updated for each sample. The algorithm always keeps the value of $\langle state, RL \rangle$ equal to the low-level Q-table.

When using demonstrations from multiple experts, conflicting advice can often occur. In our experiments, we focused on two common reasons for such conflicts: (1) individuals are experts in different parts of state space; (2) individuals are skilled in different sub-tasks of the goal.

We evaluated our proposed algorithm in a maze navigation domain, a coloured flags visiting domain and the Atari game of Pong. The results of our experiments showed that TLQL significantly outperforms regular Q-learning without knowledge and the state-of-the-art CHAT algorithm in terms of jumpstart, overall performance and asymptotic performance.

Future work includes constructing a distance measure for conflicts. In this way, clusters of similar experts could be treated as one expert in the high-level Q-table. Applying clustering techniques could also provide a hyper-parameter, that is, the number of clusters, which can be used to trade off sample efficiency of RL and computational cost.

References

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J. & Zaremba, W. 2016. Openai gym. arXiv preprint arXiv:1606.01540.

Brys, T., Harutyunyan, A., Suay, H. B., Chernova, S., Taylor, M. E. & Nowé, A. 2015. Reinforcement learning from demonstration through shaping. In *Proceedings of the 24th International Conference on Artificial Intelligence*, 3352–3358. AAAI Press.

Devlin, S. & Kudenko, D. 2012. Dynamic potential-based reward shaping. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, 433–440. International Foundation for Autonomous Agents and Multiagent Systems.

Dietterich, T. G. 2000. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research(JAIR)* **13**, 227–303.

https://doi.org/10.1017/S0269888919000092 Published online by Cambridge University Press

Fernández, F. & Veloso, M. 2006. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, 720–727. ACM.

Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Dulac-Arnold, G., Osband, I., Agapiou, J. 2018. Deep Q-learning from demonstrations. In *Thirty-Second AAAI Conference on Artificial Intelligence 2018 Apr 29*.

Kaelbling, L. P. 1993. Hierarchical learning in stochastic domains: Preliminary results. In *Proceedings of the Tenth International Conference on Machine Learning*, **951**, 167–173.

Kingma, D. P. & Ba, J. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S. 2015. Human-level control through deep reinforcement learning. *Nature* **518**(7540), 529–533.

Ng, A. Y., Harada, D. & Russell, S. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, **99**, 278–287.

Schaal, S. 1997. Learning from demonstration. In *Proceedings of the 1997 Conference on Neural Information Processing Systems (NIPS97)*. Denver, CO, pp. 1040–1046.

Sutton, R. S. & Barto, A. G. 1998. *Reinforcement Learning: An Introduction*, **1**. MIT press.

Taylor, M. E., Suay, H. B. & Chernova, S. 2011. Integrating reinforcement learning with human demonstrations of varying ability. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, 617–624. International Foundation for Autonomous Agents and Multiagent Systems.

Wang, Z. & Taylor, M. E. 2017. Improving reinforcement learning with confidence-based demonstrations. In *Proceedings of the 26th International Conference on Artificial Intelligence (IJCAI)*.

Watkins, C. J. & Dayan, P. 1992. Q-learning. *Machine Learning* **8**(3–4), 279–292.

Wiewiora, E., Cottrell, G. W. & Elkan, C. 2003. Principled methods for advising reinforcement learning agents. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 792–799.

https://doi.org/10.1017/S0269888919000092 Published online by Cambridge University Press