Deep imitation reinforcement learning with expert demonstration data

Menglong Yi 1, Xin Xu 1,2, Yujun Zeng 1, Seul Jung 3

1 College of Intelligence Science and Technology, National University of Defense Technology, Changsha, People’s Republic of China
2 Laboratory of Science and Technology on Integrated Logistics Support, National University of Defense Technology, Changsha, People’s Republic of China
3 Department of Mechatronics Engineering, Chungnam National University, Daejeon, Republic of Korea

E-mail: xuxin_mail@263.net

Abstract: In recent years, deep reinforcement learning (DRL) has made impressive achievements in many fields. However, existing DRL algorithms usually require a large amount of exploration to obtain a good action policy. In addition, in many complex situations, the reward function cannot be well designed to meet task requirements. These two problems will make it difficult for DRL to learn a good action policy within a relatively short period. The use of expert data can provide effective guidance and avoid unnecessary exploration. This study proposes a deep imitation reinforcement learning (DIRL) algorithm that uses a certain amount of expert demonstration data to speed up the training of DRL. In the proposed method, the learning agent imitates the expert’s action policy by learning from demonstration data. After imitation learning, DRL is used to optimise the action policy in a self-learning way. By experimental comparison on a video game called the Mario racing game, it is shown that the proposed DIRL algorithm with expert demonstration data can obtain much better performance than previous DRL algorithms without expert guidance.

1 Introduction

For a Markov decision process (MDP), a reinforcement learning [1] agent can continuously adjust its action strategy through the feedback from the environment so that an optimised policy can be obtained. When the input dimension is large, such as image inputs, a deep neural network [2] can be used to approximate the value function or policy. In recent years, various deep reinforcement learning (DRL) methods have been successfully developed and applied in playing games [3, 4] and robot control tasks [5, 6]. However, DRL usually requires a lot of data and time to learn a good policy. On the other hand, in some complex situations, the reward function is difficult to design. In many cases, we can only design relatively simple and fuzzy reward rules, and it is difficult to design a precise and perfect reward function. Without a good reward function, it is difficult for the algorithm to converge to the optimal strategy. It may even lead to learning failure.

In addition to DRL, supervised learning which uses expert data seems to be the solution to the problem. Someone such as NVidia, tried to use end-to-end deep learning, training convolutional neural network (CNN) to achieve end-to-end automatic driving [7]. However, the direct use of expert data for supervised learning requires a human to collect a large number of training samples. It is also necessary to ensure that the training samples are comprehensive and perfect which is unrealistic. Also the network cannot continue to evolve and improve through self-learning. Also there are also some methods that let an agent to imitate a human.

However these methods also have the same problem of end-to-end deep learning. Therefore, we consider the possibility of using expert data to guide the training of DRL and let the agent learn human strategies.

We propose a deep imitation reinforcement learning (DIRL) algorithm to improve DRL with expert demonstration data. In DIRL, when it almost reaches the expert level, it gradually transforms to an ordinary DRL algorithm and continues the self-learning process. To achieve this, we use the expert demonstration data to guide the training sample exploration of DRL and influence the reward function (Fig. 1).

Some experiments were conducted by using the DIRL method on the video game platform called the Mario racing game. Deep Q Network (DQN) was used as the baseline DRL algorithm to be compared with the DIRL algorithm. It can be found that the proposed DIRL algorithm converges much faster and the final policy of DIRL is better because of the expert guidance.

The structure of this paper is organised as follows. Section 2 will introduce the related works and Section 3 will describe the proposed DIRL algorithm. Finally, we will show the experimental results and obtain the conclusion.

2 Related works

There are many works about speeding up the training of DRL or using the expert data. Some people improved the DRL algorithm by using a more effective exploration strategy or changing the reward function, and others proposed the imitation learning algorithm to imitate human behaviour or used demonstrations to help the training of DRL.

2.1 Change exploration strategy

Many successful DRL algorithms use less effective explorations [8, 9]. In [10], the bootstrapped algorithm was proposed, using deep neural networks (DNN) for random sampling to improve the way of exploring the state space [10]. Lee and Chung [11] proposed a method to explore high-dimensional state space more efficiently through assigning exploration bonuses. Schaul et al. [12] selected better samples for training in the sample space, but in fact, these
methods are still based on the random exploration of improvements and the improvements are very limited.

2.2 Reward function with expert data

Human preferences have been used to set reward values rather than absolute values or used human feedback to train a reward function in case that it is difficult to define a reward function [13–15]. The reward function was trained to imitate human preferences, thereby finally the learning agent can obtain the strategy that is similar to human behaviours.

2.3 Imitation learning

Imitation learning is currently the hottest method to use expert data and learn human behaviour. The simplest imitation learning is supervised learning. For example, NVidia realises the unmanned driving with end to end control by collecting a large number of samples [7]. However this method is very limited, it requires a huge number of samples, and it needs to cover almost all possible situations, which is very difficult to accomplish. The generative adversarial nets (GAN) algorithm was proposed for imitation, with a generator for generating a sequence of actions, and a discriminator for distinguishing whether it is an expert operation [16]. Through the training of GAN, the motions produced by generators have become more and more close to expert movements. Some researchers applied imitation learning to robot control, allowing robots to perform very complex tasks [17–21] while some works directly let robots learn from humans by watching videos [22]. However the problem is that not all expert data are optimal, and it is often difficult to obtain sufficient expert data.

2.4 Use demonstrations to help DRL

The deep q-learning from demonstrations (DQFD) algorithm was proposed to use pre-training of expert data to speed up DQN training [23, 24]. The idea of the DQFD algorithm is similar to our work. However the DQFD algorithm only uses the pre-trained network to replace the Q network of DQN. It was suggested that the policy can be shaped by a human teacher [25] and some shaped reward function can be obtained by demonstration [26]. Some other researchers also used demonstration to solve the exploration problem of reinforcement learning (RL) algorithms [27].

3 Deep imitation RL algorithm

3.1 Framework of the DIRL algorithm

In this paper, we propose a DIRL algorithm to improve DRL with expert demonstration data. On the one hand, when it collects training samples, the algorithm will select better samples according to the expert data as much as possible, and explore the state space that is close to the expert behaviour. In this way, we avoid many unnecessary explorations; on the other hand, the reward function will have additional items related to the expert data. The closer to the expert behaviours, the greater the gain. With time, the algorithm will gradually degenerate into an ordinary DRL algorithm and continue self-learning. Fig. 2 shows the structure of the algorithm.

When we have some expert samples, we hope that we can use supervised learning to guide the training of DRL so that it can be better and faster to reach the level of experts and continue to improve through self-learning. The algorithm will try to explore the state space by the guidance of the expert data at the beginning, and learn from the expert's behaviour. After a certain period of time, the algorithm will gradually degenerate into an ordinary DRL algorithm. However, it will still retain a certain probability to obtain the expert sample, as the training pool will always have expert data. The algorithm is achieved mainly through two aspects:

i. The approach to collect training samples should be as close as possible to the expert behaviour: Normally, expert data are generally better data. Using information from expert data, we can make better training samples for exploration and reduce some unnecessary fail exploration. Compared to DQN's e-greedy method of obtaining training samples, this method does not randomly collect sample at a certain probability, but rather collects samples that are close to the expert's state – action space. Obviously, if we mix expert strategies with strategies based on Q-value greed selection, the effect will be better than the e-greedy method.

ii. When the behaviour is the same as the expert behaviour, the reward function gets extra incentive: In order to reach the expert level and to learn human behaviour faster, when actions obtained by the algorithm are consistent with expert behaviours, we allow the reward function to receive additional rewards. In many cases, it is difficult to obtain very accurate and perfect reward functions. Only simple and fuzzy reward rules can sometimes be used to judge whether the action is good or bad through the success or failure of the results. Adding additional items allows agents to have short-term and clear guidelines for rewards. It is wise to learn from experienced experts first. Of course, in order to continue learning and go beyond the experts, the proportion of this additional item will gradually decrease.

What we want to propose is actually an algorithm framework, in which the specific algorithm can be selected according to the actual situation. We will use DQN algorithm and CNN to perform the experiment and we will examine the performance of the algorithm by learning to play the Mario kart racing game. The computer uses the input image to learn how to drive the car to run on the track. In this paper, we will use expert data to train a simple
MDP may use (A denotes the action, and a system that can output expert action probability based on the expert networks to speed up the convergence of DQN through probability, which means the probability of obtaining end-to-end neural network as an expert network, and then use Fig. 4 end-to-end neural network as an expert network, and then use expert networks to speed up the convergence of DQN through influencing training sample generation and reward function.

We will use the DQN algorithm and CNN to experiment. For a MDP may use (S, A, P, R) to be represented, where S shows a state, A denotes the action, and P(S, |S, a) represents a state transition probability, which means the probability of obtaining S', when we get S, and action a. Also R represents the reward from the environment. DQN uses a network to estimate the value of each action. We can choose the action through the Q network. In this way, we can get the best strategies which can get most reward. In Q-learning, Q values are updated as follows: (see (1)). We can use neural networks to approximate the Q value function. The DQN training loss function is

\[
L(w) = E[(r + \gamma \max_a Q(s', a', w) - Q(s, a, w))^2]
\]

(2)

As we mentioned before, we mainly combine supervised learning and DRL with expert data through two aspects. That is, the training sample exploration method and the reward function expression form.

First of all, we need to have an expert evaluation system, that is, a system that can output expert action probability based on the input. With the input S, the expert system can select the action most likely to be selected by humans by using supervised learning. The training of this expert system can be completed with the sampled expert data. In this paper, we train a simple neural network as an expert prediction system. The network structure does not need to be very complicated. It only needs to be able to output the relatively accurate action label based on the expert data. In this paper, the neural network structure uses the NVidia CNN neural network [7].

The network structure is as follows (Fig. 3): the expert network contains nine layers, including a regular layer, five convolutional layers and three fully connected layers. We use convolution kernels in the first three convolutional layers 5 × 5, stepping is 2 × 2 step convolution, using convolution kernels in the last two layers 3 × 3 non-stepped convolutions. The output of this CNN network is a list of action probabilities which means the likelihood of each action. We will use this network as the basis for the output of expert behaviour. At first, when exploring the state space and putting training samples into the experience pool, the system will use the expert network to select the action with a certain probability, and this probability will continue to decrease as time goes by. However in the end, it will still maintain a relatively small probability using expert action to keep the expert data. Therefore, when we train DQN, the agent action selection includes the expert network selection method and \( \epsilon \)-greedy method of the ordinary DQN, and the selection basis is an attenuation probability coefficient alpha. In addition, because the expert data are not necessarily complete, the choice of whether to perform the output action of the expert network will be determined based on the output probability, that is, the credibility of the output action. In other words, if the probability of the expert network output action is very small, in other words, the expert network is not sure how to act, and we will select the optimal action through the \( Q \) network in the training pool.

For the reward function \( r \), we can add an additional compensation to speed up the learning from the experts. If the selected action is the same as the action chosen by the expert network, it will get the gain. In addition, the gain factor will also be multiplied by the attenuation coefficient, after a period of time. \( R \) will be degraded to a normal value:

\[
R = R_g + r \cdot \text{flagSame} \cdot R_g
\]

(3)

Feedback \( R_g \) indicates a normal reward \( r \) and \( \alpha \) indicates the attenuation coefficient. When the final selection of motion is the same as the expert motion, flagSame = 1, otherwise it is equal to 0. \( R_g \) indicates the gain of \( r \), you can set it by yourself. For most of the learning tasks, it is difficult for us to define the quality of each action accurately. In such a situation, it is difficult for an agent to learn optimal strategies quickly and easily, and usually requires a lot of exploration. Therefore, we believe that learning from experts will speed up the learning of DRL.

3.2 Detailed steps:
1. For each episode, we get observation \( S \) and calculate the expert action (Ae). We put the image inputs into the end-to-end expert net and we can get the expert action (Fig. 4).
2. The second step is exploring state space and sample the train data. We use expert net to guide this process by changing the exploration method and the reward function. Then, to reduce relevance, we put the train data into the experience pool. The process is shown in this figure (Fig. 5).
3. Then, we train the deep Q net. We sample a mini batch of train data from the experience pool \( D \) and use it to train the net. The loss function can be described by (2). In order to increase the stability of the training, we use two networks: the present Q network and target Q network which will be updated by the Q network every other time. The loss function can be changed as:

\[
y = \begin{cases} 
  r, & \text{end} \\
  r + \gamma \max_a \text{TargetQ}(s, a), & \text{normal} 
\end{cases}
\]

(4)

\[
Q(S, A) = Q(S, A) + \alpha(R_e + \gamma \max_a Q(S', a) - Q(S, A))
\]

(1)

J. Eng., 2018, Vol. 2018 Iss. 16, pp. 1567-1573
This is an open access article published by the IET under the Creative Commons Attribution License
(http://creativecommons.org/licenses/by/3.0/)
Loss = \mathbb{E}[(y - Q(s, a, w))^2] \quad (5)

Moreover, we can use the Adam optimiser to reduce the loss function.

4. Finally, we decay the probability \( \alpha \). We decay the parameter \( \alpha \) from 0.6 to 0.1. It means that when we get the train samples, there is a greater possibility to use expert net to get action at the start time. Also when we get the expert action, the reward function will also have an additional gain. However as time goes by, the probability will become small until it decays to 0.1. We decay \( \alpha \) so that the algorithm will change from learning from expert to normal DQN. What's more, we keep \( \alpha > 0.1 \) so that the experience pool will always have expert data.

To evaluate the performance of the algorithm, we record the result (the checkpoints it can get) every other time.

The detailed algorithm description can be found in Fig. 6.

4 Experimental results

4.1 Mario racing game environment

We registered Mario racing games in the gym and learned the racing game through the format of the gym (Fig. 7). The Mario racing game contains multiple maps, with a small map and the car’s position in the lower right corner, showing the number of laps at the top. Through the gym, we can return the game image as the input to the DRL algorithm. The same action output can also be applied to the environment through the gym. The Mario racing game environment is shown below:

- **The state space**: the original input to the 640*480*3 RGB picture, we convert it to a grayscale 64*64, and then use four consecutive frames as the input. After each action, the program updates the state based on the new returned state.
- **The action space**: the entire game can be completed by three actions: straight, left, and right, so we can simplify the action space to three dimensions, use [1,0,0] for straight, and [0,1,0] for left, [0,0,1] means turn right.
- **The reward function**: in this game, the status information we can get is very limited. We can get the position of the car in a mini map, so we can get rewards after it arrives at some specific location (checkpoint point). We divide the track roughly into 12 segments. We can get a reward after through a segment. Also it can get rewards after finishing a lap. When the car crashes which will be judged simply based on the image information, it receives a negative reward. In addition, when there is a normal form on the track, there will be a smaller negative reward, so that the car will not stay on the track, and it will run as fast as possible. The reward function table is shown in Table 1.

4.2 Experimental setup

First, we played the Mario racing game five times and recorded the image and corresponding human actions as the expert data. The expert data is not so sufficient and perfect but it is enough to train a simple deep neural network whose performance is ok. Then, we use the expert data to train the expert network. The accuracy of the last network training is probably more than 80%. The input of this network is image and the output is the probability of each action. After that, we sample the train data of DRL. When the agent explores the state-action space, it has probability \( \alpha \) to choose the expert actions instead of the \( \varepsilon \)-greedy method. The probability \( \alpha \) will decay from 0.6 to 0.1. Also the reward function will get an additional gain when the action of agent is the same as the expert network. We finally set this gain item to 4. Finally, we sample the train data from the experience pool and train the \( Q \) net.

We did some contrast experiments to evaluate our algorithm and the influence of each item in the algorithm. So, we also used DQN and the expert network that is an end-to-end deep neural network to learn to play this game. To see how the exploration method and reward function improve the DRL algorithm, we did an experiment by using DIRL without additional reward item and compared it with DIRL and DQN. What’s more, we set a different decay parameter \( \alpha \) to evaluate the influence of it and choose a better parameter.

4.3 Results

This paper uses the Mario racing game as an experimental platform. The most intuitive evaluation standard is to record how far the car can run after learning. The farther it runs, the better the learning result of the algorithm is. We use checkpoint that the agent will be able to get as the evaluation standard of algorithm.

To check the quality of training, we use the training network doing four experiments (4 episode) every 2000 steps (time step). We averaged the results of the four experiments as one assessment. Each ring has a 12-point check point. When the checkpoint point reaches 12 points or more, we think the result is relatively good. The middle section of the track is an underground passage, which is generally difficult to pass. Therefore, it can only reach about 6 in general.
4.3.1 DIRL versus DQN and supervised learning: We compare the ordinary DQN, DIRL algorithm and supervised learning algorithm. The DQN method uses Nature DQN algorithm which contains two Q networks. It uses a target Q network to improve stability and convergence. The DIRL method adds an expert module in original DQN. Also the supervised learning is actually the expert network which is an end-to-end deep neural network. The result is shown in Fig. 8.

Fig. 6 Algorithm 1: DIRL

Fig. 7 Mario racing game environment

4.3.1 DIRL versus DQN and supervised learning: We compare the ordinary DQN, DIRL algorithm and supervised learning algorithm. The DQN method uses Nature DQN algorithm which contains two Q networks. It uses a target Q network to improve
The figure of the blue dot is the result of the DIRL algorithm, which is the algorithm we proposed with a changing reward function and exploration method. The red dot indicates the result of the expert network experiment which uses the end-to-end deep neural network. Also, the green dot indicates normal DQN experiment results. Due to the underground passage, DQN or expert network is difficult to exceed six points check point. The expert network which is trained by the expert data can guarantee obtaining more than five point check points. It can also drive through underground passages regularly. We can see that it is almost stable in about six. The DQN algorithm is not so stable, because the ambiguity of the underground channel reward function and the similarity of the inner wall and surface make it difficult to steadily through the underground channel. Also we can find out that it need a lot of time to train. However, we can see that the DIRL algorithm will almost reach the level of experts after about 300–400 experiments and, subsequently, the performance is significantly better than end-to-end supervised learning and DQN. In the meantime, we can also know the influence of the exploration methods: the blue one is the result of DIRL; the red one is the result of supervised learning; the green one is the result of DQN.

Table 1

| Status          | Normal | Collision | Reaching checkpoint | Finishing a lap |
|-----------------|--------|-----------|---------------------|-----------------|
| awards          | 2      | 20        | 100                 | 150             |

Table 2

|                    | DIRL | DQN |
|--------------------|------|-----|
| 12 points          | 272  | 32  |
| 20 points          | 76   | 0   |
| First time getting 12 points | 80    | 120 |
| Most points        | 36   | 19  |

Fig. 8 Result of DIRL, DQN and supervised learning algorithm: the blue line is the result of DIRL algorithm; the red line is the result of supervised learning; the green line is the result of DQN.

Fig. 9 We did experiments to test the influence of reward bonus and exploration methods: the blue one is the result of DIRL; the red one is the result of DIRL without R bonus; the green line is the result of normal DQN algorithm.

Fig. 10 Experimental results with different decay coefficient α: the blue line is the result of the algorithm with α from 0.6 to 0.1; the red line is the result of the algorithm with α equalling 0.1; the green line indicates the result of the algorithm with α from 1 to 0.1.

4.3.2 Reward bonus experiment: We conduct a comparative experiment of reward functions. After removing the bonus gain item, we did an experiment and compared it with DQN and DIRL with the gain item. Do that and we will see the influence of reward bonus by comparing the experiment of DIRL and DIRL without R bonus. In the meantime, we can also know the influence of the exploration method by comparing the experiment of DIRL without R bonus and DQN.

It can be seen that when the bonus is existing, the algorithm can learn the behaviour of the experts more quickly. The results of the experiment are better than DIRL without gain bonus both in the early and late stages. At the same time, according to comparing the result of not using the bonus gain DIRL and DQN, we can also find that it can also achieve a very good performance improvement by only improving the exploration methods. According to the experiment and contrast, we can see that the changing of the exploration method and reward function can significantly improve the performance of the algorithm.

4.3.3 Decay coefficient experiment: For observing the influence of the decay coefficient, we set a comparative experiment. The decay coefficient decides the speed of degenerating from DIRL to original DQN and the number of expert data in the training pool. The experimental results are as follows (Fig. 9).

So, we set the experiment with decay coefficient from 1 to 0, decay coefficient from 0.6 to 0.1 and decay coefficient as 0.1. The experimental results are shown in Fig. 10.

We can see that when alpha decays from 1 to 0, the result (the number of checkpoints) is more variable. When α has been 1, the final result is pretty good even better than the result of α from 0.6 to 0.1, but the early learning speed of it is slower. Finally, we choose the alpha from 0.6 to 0.1, which is steadier and better.

5 Conclusion

In order to speed up the learning speed of DRL and improve the efficiency of learning, we propose to combine supervised learning and DRL when we have some expert demonstration data. The idea is to learn from expert behaviours via conducting effective explorations and changing the reward function. After reaching the level of experts, the learning agent will continue to improve through reinforcement learning and eventually exceed the experts.

Based on the above idea, this paper proposed a DIRL algorithm with expert demonstration data. At the beginning, we sampled some expert data by recording human control strategies. And we used expert data to train a convolutional neural network which can provide human decisions. Then, we collected the training samples...
by the expert net with a probability to imitate human behaviours. Also, the probability will decrease gradually so that the algorithm will degenerate into normal DRL gradually. It is worth mentioning that the expert samples will always be stored in the training pool. On the other hand, when the behaviour of the agent is consistent with the behaviour of the expert, the reward function becomes larger, prompting the agent to learn from the expert. Finally, we conducted experiments and compared the experimental results of the DQN algorithm. We also modified the parameters and conducted comparisonal experiments. We found that our DIRL algorithm can significantly improve the DRL performance.

6 Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grants U1564214, 61751311 and 61611540348.

7 References

[1] Sutton, R., Barto, A.: ‘Reinforcement learning: an introduction’ (MIT Press, Cambridge, 1998)
[2] Lecun, Y., Bengio, Y., Hinton, G.: ‘Deep learning’, Nature, 2015, 521, (7553), pp. 436–444
[3] Silver, D., Huang, A., Maddison, C.J., et al.: ‘Mastering the game of Go with deep neural networks and tree search’, Nature, 2016, 529, (7587), pp. 484–489
[4] Mnih, V., Kavukcuoglu, K., Silver, D., et al.: ‘Playing atari with deep reinforcement learning’. Proc. of Neural Information Processing Systems – Deep Learning Workshop, 2013
[5] Levine, S., Finn, C., Darrell, T., et al.: ‘End-to-end training of deep visuomotor policies’. J. Mach. Learn. Res., 2016, 17, (1), pp. 1334–1373
[6] Abbeel, P., Coates, A., Ng, A.Y.: ‘Autonomous helicopter aerobatics through apprenticeship learning’, Int. J. Robot. Res., 2010, 29, (13), pp. 1608–1639
[7] Bojarski, M., Del Testa, D., Dworakowski, D., et al.: ‘End to end learning for self-driving cars’, arXiv preprint, arXiv:1604.07316, 2016
[8] Osband, I., Russo, D., Van Roy, B.: ‘(More) efficient reinforcement learning via posterior sampling’, Adv. Neural Inf. Process. Syst., 2013, pp. 3003–3011
[9] Mnih, V., Kavukcuoglu, K., Silver, D., et al.: ‘Human-level control through deep reinforcement learning’, Nature, 2015, 518, (7540), p. 529
[10] Osband, I., Blundell, C., Pritzel, A., et al.: ‘Deep exploration via bootstrapped DQN’. Neural Information Processing Systems (NIPS), 2016, pp. 4026–4034
[11] Lee, S.G., Chung, T.C.: ‘A reinforcement learning algorithm using temporal difference error in ant model’. Int. Work-Conf. Artificial Neural Networks Springer Berlin Heidelberg, 2015, pp. 217–224
[12] Schaul, T., Quan, J., Antonoglou, I., et al.: ‘Prioritized experience replay’. Int. Conf. Learning Representations (ICLR), 2016
[13] Mnih, V., Kavukcuoglu, K., Silver, D., et al.: ‘Introducing human feedback with reinforcement learning’. Advances in Neural Information Processing Systems, 2013, pp. 2625–2633
[14] Ho, J., Ermon, S.: ‘Generative adversarial imitation learning’. Advances in Neural Information Processing Systems, 2016, pp. 4565–4573
[15] Silver, D., Huang, A., Maddison, C.J., et al.: ‘Mastering the game of Go with deep neural networks and tree search’, Nature, 2016, 529, (7587), pp. 484–489
[16] Duan, Y., Andrychowicz, M., Stadie, B., et al.: ‘One-shot imitation learning’, arXiv preprint arXiv:1703.07326, 2017
[17] Stadie, B.C., Abbeel, P., Sutskever, I.: ‘Third-person imitation learning’. Int. Conf. Learning Representations (ICLR), 2017
[18] Calinon, S., Guenter, F., Billard, A.: ‘On learning, representing, and generalizing a task in a humanoid robot’. IEEE Trans. Syst. Man Cybern., B, Cybern., 2007, 37, (2), pp. 286–298
[19] Doerr, A., Ratliff, N., Bohg, J., et al.: ‘Direct loss minimization inverse optimal control’. Proc. Int. Conf. Robotics: Science and Systems (RSS), 2015
[20] Sermanet, P., Xu, K., Levine, S.: ‘Unsupervised perceptual rewards for imitation learning’. Proc. Int. Conf. Robotics: Science and Systems (RSS), 2017
[21] Liu, Y.X., Gupta, J.K., Ermon, S.: ‘Model-free imitation learning with policy optimization’. Proc. AAAI, 2018
[22] Hester, T., Vecerik, M., Pietquin, O., et al.: ‘Deep Q-learning from demonstrations’. Proc. of AAAI, 2018
[23] Taylor, M., Suay, H.B., Chernova, S.: ‘Reinforcement learning with human demonstrations of varying ability’. Proc. Int. Conf. AAMAS, 2011, pp. 617–624
[24] Cederborg, T., Grover, I., Isbell, C.L., et al.: ‘Policy shaping with human teachers’. Proc. AAAI, 2015, pp. 3366–3372
[25] Brys, T., Harutyunyan, A., Suay, H.B., et al.: ‘Reinforcement learning from demonstration through human feedback’. Proc. AAAI, 2015, pp. 3352–3358
[26] Subramanian, K., Thomaz, A.L.: ‘Exploration from demonstration for interactive reinforcement learning’. Proc. Int. Conf. Autonomous Agents & Multiagent Systems, 2016, pp. 447–456