Introspective Reinforcement Learning and Learning from Demonstration

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

Tim Brys
Vrije Universiteit Brussel
Brussels, Belgium
timbrys@vub.ac.be

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

Reinforcement learning is a paradigm to model how an autonomous agent learns to maximise its cumulative reward by interacting with the environment. One challenge faced by reinforcement learning agents is that in many environments the reward signal is sparse, leading to slow improvement of the agent's performance in early learning episodes. Potential-based reward shaping can help to resolve the aforementioned issue of sparse reward by incorporating an expert's domain knowledge into the learning through a potential function. Past work on reinforcement learning from demonstration directly mapped (sub-optimal) human expert demonstration to a potential function, which can speed up reinforcement learning. In this paper we propose an introspective reinforcement learning agent that significantly further speeds up the learning. An introspective reinforcement learning agent records its state-action decisions and experience during learning in a priority queue. Good quality decisions, according to a Monte Carlo estimation, will be kept in the queue, while poorer decisions will be rejected. The queue is then used as demonstration to speed up reinforcement learning via reward shaping. A human expert's demonstration can be used to initialise the priority queue before the learning process starts. Experimental validation in the 4-dimensional CartPole domain and the 27-dimensional Super Mario AI domain show that our approach significantly outperforms non-introspective reinforcement learning and state-of-the-art approaches in reinforcement learning from demonstration in both domains.

KEYWORDS

Q-learning, reinforcement learning, reward sharping, demonstrations

1 INTRODUCTION

One of the main challenges Reinforcement Learning [19] faces is dealing with the sparsity of rewards that is present in many tasks. A reinforcement learning agent will only very slowly learn to solve a task through trial-and-error if the feedback it receives in the form of rewards is given sparsely. This lack of feedback, and thus gradient, makes the agent explore the task uniformly at random, unless special mechanisms are implemented that guide the agent's exploration.

One way of tackling this problem that is often taken is to include prior (heuristic) knowledge that can bias the agent towards what are *a* priori believed to be good states, or good behaviour. Prior knowledge can come in the form of rules defined by a domain expert [8], demonstrations provided by such an expert [13], prior learning experiences of the agent in different tasks [20], etc.

Conversely, a second stream of thought, orthogonal to the former, tries to leverage internal knowledge and experiences in the current task to generate internal rewards to bias exploration in the absence of immediate external rewards. Approaches relying on intrinsic motivation [2, 12] attempt to leverage the ideas of curiosity and uncertainty to achieve more intelligent and effective ways of exploration.

This paper falls in the second category, as we develop a technique that allows the agent to bias its exploration by generating internal shaping rewards based on its own successful previous experiences in the task.

We evaluated our approach in the 4-dimensional CartPole domain and the 27-dimensional Super Mario AI domain, and the results show significant improvements in learning performance both when intrinsic RL is used on its own and when it is used in combination with human expert advice.

2 BACKGROUND

In this section we introduce the background knowledge that underpins the work presented in this paper.

2.1 Reinforcement Learning

Reinforcement learning (RL) [19] is a paradigm that describes how an agent can improve its behaviour based on reward and punishment received from interactions with its environment. Maximising the rewards accumulated over time equals solving the task by definition. We define an RL problem as a Markov Decision Process (MDP) $\langle S, A, T, \gamma, R \rangle$. S defines all environment states that can be observed, and A defines the actions an agent can take to affect this environment. T describes the dynamics of the system, defining the probability of the environment transitioning from state s to state s', given that action a was taken: $T(s'|s, a)$. R is the reward function that yields a numerical reward for every state transition, defining the task. Finally, γ is called the discounting factor, and defines how important long-term rewards are, i.e. how short- or far-sighted the agent is.

The goal of an RL agent is to find behaviour, i.e. a policy $\pi : S \rightarrow A$ that maximise the expected accumulation of rewards. Many approaches do not directly search the policy space, but instead estimate the quality of actions in each state, and derive a policy from these estimates. The quality of states and actions is defined by the Q-function: $Q : S \times A \rightarrow \mathbb{R}$. Temporal difference algorithms, such as Q-learning [23], approximate the true Q-function by incrementally updating their estimates:

$$Q(s, a) \leftarrow Q(s, a) + \alpha\delta$$

α is the learning rate, and δ is the temporal-difference error, the difference between the previous estimate and the target being tracked:

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

Q-learning is guaranteed to converge to the true Q-values, and thus an optimal policy, given certain assumptions such as a decreasing learning rate [22].

2.2 Reward Shaping

Vanilla versions of model-free RL algorithms typically require excessive amounts of interactions with the environment to learn high quality policies, because they do not incorporate prior knowledge to guide exploration. They must therefore rely on uniformly random exploration of the state-action space in order to stumble on the typically sparse rewards. Only after enough state transitions and their associated rewards have been observed can the agent start to exploit this knowledge by biasing its action selection towards what its estimates indicate are good actions. Incorporating prior knowledge to bias initial exploration is a successful way of reducing learning time in reinforcement learning, the most notable recent example being AlphaGO using expert demonstration data [15].

Reward shaping, derived from behavioural psychology, is a popular way of including prior knowledge in the learning process in order to alleviate the random exploration problem. It provides a learning agent with extra intermediate rewards, like a dog trainer would reward a dog for completing part of a task. This extra reward can enrich a sparse base reward signal (for example a signal that only gives non-zero feedback when the agent reaches the goal), providing the agent with useful gradient information. This shaping reward F is added to the environment's reward R to create a new composite reward signal that the agent uses for learning:

$$R_F(s, a, s') = R(s, a, s') + F(s, a, s')$$

Of course, since the reward function defines the task, modifying the reward function may modify the total order over policies, and make the agent converge to suboptimal policies (with respect to the environment's reward alone).

Following Ng et al. [10], we define a potential function $\Phi : S \rightarrow \mathbb{R}$ over the state space, and take the reward shaping function F as the difference between the new and old states' potential. Doing that provably maintains the total order over policies, and preserves any convergence guarantees [10]:

$$F(s, a, s') = \gamma\Phi(s') - \Phi(s)$$

Prior knowledge can be numerically encoded in the potential function Φ. The effect of applying such reward shaping, which is typically denser than the sparse environment reward, is that the agent is more likely to encounter non-zero rewards, and thus its exploration will move away from uniform random much earlier in learning. Instead, it will be biased towards states with high potentials. For example, using height as a potential function in Mountain Car [16] biases the agent to selecting actions that will increase height. Since the goal location in Mountain Car lies on top of a hill, shaping using this heuristic helps an agent solve that task faster.

The definition of F and Φ was extended by others ([5, 6, 24]) to include actions and time-steps, allowing for the incorporation of behavioural knowledge that reflects the quality of actions as well as states, and allowing the shaping to change over time:

$$F(s, a, t, s', a', t') = \gamma\Phi(s', a', t') - \Phi(s, a, t)$$

These extensions also preserve the total order over policies and therefore do not change the task, given Ng's original assumptions.

2.3 RL from demonstration

Reinforcement learning from demonstration is a term coined by Schaal [13] to denote the intersection between the fields of reinforcement learning and learning from demonstration. As in RL, an agent operating in a Learning from Demonstration (LfD) setting looks for a policy π that allows it to execute a task [1]. While in RL, the agent has a reward signal to evaluate behaviour, this objective evaluation metric is not present in the LfD setting (similarly for inverse reinforcement learning [11, 18]), and the agent must refer to expert demonstrations (sequences of state-action pairs $\{(s_0, a_0), \ldots, (s_n, a_n)\}$) of the task to derive a policy that mimicks and generalises these demonstrations. This constrains the LfD agent to the behaviour the expert demonstrator exhibits (however good or bad). On the other hand, LfD agents are typically much more sample efficient than RL agents, requiring orders of magnitude less demonstrator samples than an RL agent needs environment interactions to learn good behaviour.

A Reinforcement Learning from Demonstration agent, possessing both an objective reward signal and demonstrations, is able to combine the best from both fields, leveraging demonstrations to direct what would otherwise be uniform random exploration and thus speed up learning, while leaving the theoretical guarantees associated with reinforcement learning in place, as the objective reward signal allows such an agent to eventually outperform a potentially sub-optimal demonstrator.

Schaal [13] proposed two basic approaches to using demonstrations in a reinforcement learning setting that were later developed by other researchers. One is the generation of an initial value-function for Q-learning by using the demonstrations as passive learning experiences for the RL agent. This was later built upon by Smart and Kaelbling [17]. The other approach derives an initial policy from the demonstrations and uses that to kickstart the RL agent. This approach was picked up and further developed in [4, 18, 21]. In this paper, we mainly build upon and compare with the demonstration reward shaping work from [4].

In that work, demonstrations are encoded as a reward shaping function by defining a Gaussian similarity measure over the state space:

$$g(s, s^d, \Sigma) = e^{\left(-\frac{1}{2}(s-s^d)^T\Sigma^{-1}(s-s^d)\right)} \quad (1)$$

and subsequently incorporating it in the potential function as follows:

$$\Phi(s,a) = \max_{(s^d,a)} g(s, s^d, \Sigma) \tag{2}$$

thus creating a Gaussian landscape that rewards the agent for mimicking the demonstrator, but allows it to deviate from the demonstrator's example as well.

3 PRINCIPLES OF INTROSPECTIVE RL

This section will describe the methodology of using the agent's own experiences to shape the reward function in order to bias exploration and speed up reinforcement learning. Typically, in complex environments, rewards must be observed many times before the agent acquires a significant behavioural bias towards pursuing those rewards. In the Introspective Reinforcement Learning approach we propose, we leverage these experiences to include a more explicit bias, by shaping the reward function, rewarding current behaviour that is similar to past behaviour that led to rewards. A different perspective on this is to say that those previous experiences that led to rewards are task demonstrations (provided by the agent itself), which can be used to bias the agent's current exploration in the same fashion as external expert demonstrations would be used as described above in Section 2.3. If actual external expert demonstrations are available, these could even be used to initialised the introspective agent's bias.

3.1 Collecting the experiences

Introspective Reinforcement Learning extends RL by adding an experience filter module, and addresses the reward sparsity problem using dynamic reward shaping based on the filtered experiences. The experience filter collects the agent's exploratory behaviour that led to positive outcomes into a priority queue. These experiences are then immediately used as an exploratory bias to speed up the learning process.

Specifically, during a learning episode, every state-action-next state-reward tuple $((s_t, a_t, s_{t+1}, r_t))$ is stored in memory. At the end of the episode, the Q-value $\widehat{q}(s_t, a_t)$ for each state-action in this episode is Monte Carlo estimated until the final time step T:

$$\widehat{q}(s_t, a_t) = \sum_{k=t}^{T} \gamma^{k-t} r_t \tag{3}$$

The state-action pairs and their estimated Q-values are then stored in a priority queue with the estimated Q-values being the sort key of the queue. If the queue is full (defined by queue size parameter qs), only state-action pairs with a higher estimated Q-value than the smallest in the queue are added (and the smallest are consequently removed). If the queue is not full, all state-action pairs and their estimated Q-value are added.

Progressively, poorer Q-value elements of the queue will be removed, and experiences with higher estimated performance levels will remain. Thus, the exploratory bias induced by these experiences will progressively increase in quality.

Introspective RL is using prior experience to compute a reward shaping function, rather than using it to replay past experience directly [14]. As such, our approach is closer to the automatic generation of a reward shaping function. Also, our approach does not depend on using poor performing actions in the priority queue, which is sorted by Q value (and not TD error as in prioritised experience replay). Furthermore, dynamic reward shaping [5], as applied to experience reuse in our approach, has been proven to conserve the convergence guarantees of the RL algorithm used, in our case $Q(\lambda)$.

3.2 Defining the Potential Function

In the manner of [4], we encode state-action pairs as a potential function using a Gaussian similarity metric.[1] The assumption is that, if in the agent's past experience, certain state-action pairs led to high rewards, taking the same action in a similar state might lead to similarly high rewards.

When the potential function $\Phi(s,a)$ needs to be calculated in a certain state-action pair (s,a), first the agent must look in the priority queue for the recorded state-action pair that is most similar to the current, using Equation 1. Then the potential function is defined as follows:

$$\Phi(s,a) = \rho \max_{(s^d,a)} g(s, s^d, \Sigma)\widehat{q}(s^d, a) \tag{4}$$

which is a modification of Equation 2, incorporating the actual estimated quality of that state-action pair $\widehat{q}$, and a scaling factor ρ to control the strength of the exploratory bias induced.

Since the agent progressively collects more and more experiences, in principle of higher and higher quality, the potential function will change from episode to episode, making this potential function a dynamic one. Note that theoretical guarantees, i.e. that the optimal policy does not change, hold for dynamic potential-based advice [6], and that no modification to the policy needs to be made given that the initial Q-function is all zeroes.

3.3 Initialising the Priority Queue with Demonstrations of External Agents

Even though the introspective reinforcement learning idea is focused on using the agent's own experiences to bias its learning, it is also completely amenable to receiving an a priori-bias from demonstrations provided by an external agent. Such demonstrations can easily be incorporated by putting them through the same process of estimating Q-values and storing them in the priority queue as the experiences collected by the agent itself. When qualitatively good, these demonstrations can prevent the agent from initially filling the priority queue with whatever low-quality random trajectories it executes first, allowing it to be positively guided from the first episode.

Specifically, demonstration data from external agents will be collected as a set of episodes, i.e. sequences of state-action pairs that terminate in an end-state. Monte Carlo estimation of the Q-value $\widehat{q}(s_i, a_i)$ is performed, and the individual state-action pairs, augmented with the estimated Q-value are inserted into the priority queue. If the demonstration data does not include the reward signal, a viable solution may be to set the reward to 1 for all (s_n, a_n). The

[1] As opposed to the external expert demonstrations used in their work, we encode the agent's own filtered experiences in the potential function.

Reinforcement Learning from Demonstration work we base ourselves on successfully deals with this problem in the same way [4].

3.4 Pulling it together

The pseudo-code in Algorithms 1 and 2 describe a Q-learning agent employing the introspective technique to shape its exploration, showing how all components interact on an algorithmic scale.

Algorithm 1 Introspective Q-Learning

Require: discount factor γ, learning rate α, queue size qs

procedure INTROSPECTIVE Q-LEARNING
  initialise value-function Q to all 0.
  initialise the priority queue PQ
    (empty or with demonstrations)
  for each step of episode **do**
    s_t is initialised as the starting state
    choose a_t in s_t using π derived from Q
    SC = empty list ▷ SC collects all experience tuples (s,a,s',r) in an episode
    repeat
      perform action a_t
      observe reward r_t and new state s_{t+1}
      choose a_{t+1} in s_{t+1} using π derived from Q

$$\begin{aligned} Q(s_t,a_t) \leftarrow & Q(s_t,a_t) + \alpha(r_t \\ & + F(s_t,a_t,s_{t+1},a_{t+1}) \\ & + \gamma \max_b Q(s_{t+1},b) \\ & - Q(s_t,a_t)) \end{aligned} \tag{5}$$

      insert (s_t, a_t, s_{t+1}, r_t) to SC ▷ Collect steps
      $s_t \leftarrow s_{t+1}$
      $a_t \leftarrow a_{t+1}$
    until s_t is a terminal state
    for each (s_t, a_t, s_{t+1}, r_t) in SC **do**
      for i in $[0, 1, .., length(\text{SC}) - t]$ **do**
        $\widehat{q_t} \leftarrow \widehat{q_t} + \gamma^i r_{t+i}$
      end for
      FILTER($< s_t, a_t, \widehat{q_t} >, PQ$)
    end for
  end for
end procedure

4 EXPERIMENTAL VALIDATION

Two domains, CartPole and Super Mario, were selected to demonstrate the strength of the proposed approach in this paper. $Q(\lambda)$-learning and Reinforcement learning from Demonstration [4] were used as benchmarks to be compared with the learning curve of the proposed approach. In CartPole and Super Mario, ten and twenty trial demonstrations from human experts were used respectively to initialise the priority queue and shape the RLfD agent.

The results of all curves have passed a t-test, in which every 100 running results was averaged to be one sample and 30 samples were involved in the test. The t-test was conducted every hundredth step for the samples. All the presented empirical results are statistically different with $p < 0.05$.

Algorithm 2 Filter High Q-value Experience

Require: queue size qs

procedure FILTER($< s_t, a_t, \widehat{q_t} >, PQ$)
  if Size of $PQ < qs$ **then**
    Insert($PQ, < s_t, a_t, \widehat{q_t} >$)
  else
    $< s_e, a_e, \widehat{q_e} >\leftarrow$ the last element of PQ
    if $\widehat{q_t} > \widehat{q_e}$ **then**
      Remove $< s_e, a_e, \widehat{q_e} >$ from PQ
      Insert($PQ, < s_t, a_t, \widehat{q_t} >$)
    end if
  end if
end procedure

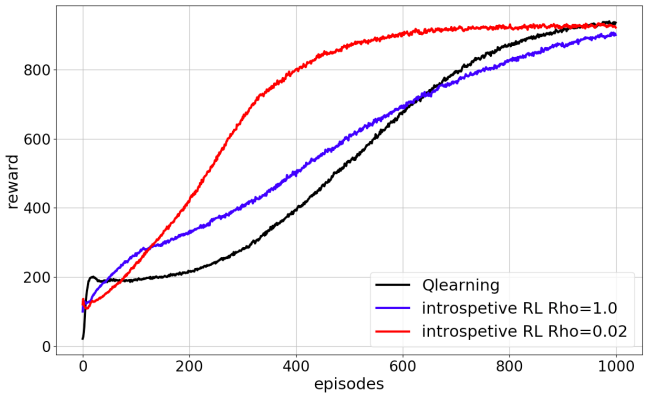

Figure 1: CartPole learning curves of Q(λ)-learning and Introspective RL without any external demonstrations

4.1 CartPole

In the CartPole domain [9], the RL agent controls a cart with a pole at its centre connected by a mechanical coupling. The RL agent learns to keep the pole balanced by pushing the cart to the left or the right. The observation consists of 4 features: the cart's velocity, its position, the angle of the pole, and its angular velocity.

In the experiments, Q(λ)-learning and reinforcement learning from demonstration, abbreviated as RLfD, were used as the benchmarks to be compared to introspective RL with and without demonstrations, to show the advantages of introspection. We set all parameters following [4] with whose RLfD method we will compare: learning rate $\alpha = \frac{0.25}{16}$, discount rate $\gamma = 1.0$, $\lambda = 0.25$ and an ϵ-greedy exploration strategy with $\epsilon = 0.05$. Tile coding was used as the function approximation with 16 10×10 tilings. Additionally, the potential-function scaling parameter $\rho = 0.2$.

Figure 1 shows the results of comparing plain Q(λ)-learning with introspective RL (without demonstrations). While both methods converge to optimal behaviour, the results show that introspection leads the agent to learn significantly faster than the regular Q(λ)-learning agent. Since the λ parameter chosen in [4] and adopted by us is quite low, we show experiments with two higher $\lambda \in$

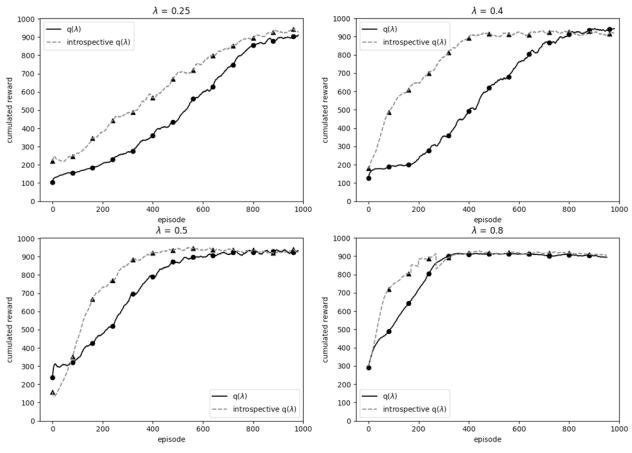

Figure 2: CartPole learning curves of Q(λ)-learning, and Introspective RL for $\lambda \in \{0.25, 0.4, , 0.5, 0.8\}$.

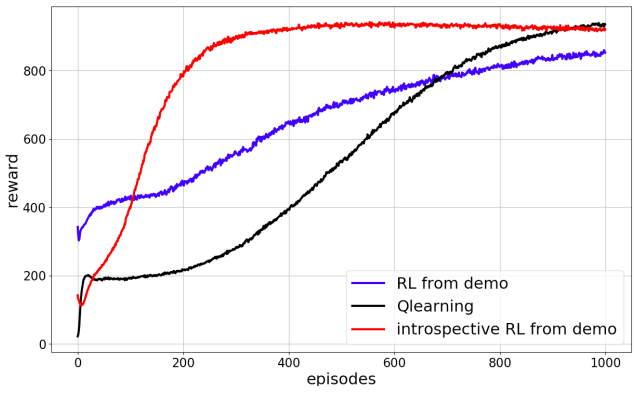

Figure 3: CartPole learning curves of Q(λ)-learning, Reinforcement learning from Demonstration, and Introspective RL with demonstrations

$\{0.4, 0.8\}$ in Figure 2. For higher λ, convergence for both methods is significantly faster, but the introspection advantage remains.

In Figure 3, we show the effect of external demonstrations on the learning processes. We compare the baseline $Q(\lambda)$-learning without demonstrations, with the RLfD approach provided with 10 different demonstrations, and our Introspective $Q(\lambda)$ agent using the same 10 demonstrations to seed the experience queue. The external demonstrations have sub-optimal performance between 450 to 700 steps of keeping the pole up. The results show that while leveraging demonstrations can help, giving RLfD a jumpstart compared to the vanilla $Q(\lambda)$-learner and the introspective agent, the introspective agent very quickly outperforms both other agents. Since the use of demonstrations and the introspective mechanism are orthogonal to each other, we see here that they can be combined to provide a more powerful learner. Furthermore, the introspective mechanism will replace the potentially sub-optimal external demonstrations' exploratory bias as the agent discovers better trajectories by itself. In the static RLfD case, the suboptimal bias remain throughout the learning process. Even though theoretical guarantees for convergence apply to both the dynamic introspective case as the static RLfD case, it is clearly better to dynamically change the bias to include higher and higher quality experiences.

4.2 Super Mario AI competition benchmark

Super Mario Bros is a famous 2-D side-scrolling video game first released by Nintendo in 1985. [7] converted this game into an AI algorithm competition benchmark. The goal of the game is for the agent to maximise its points. Points are earned for collecting coins, killing enemies, and finishing a game level, while points are subtracted for getting hurt and dying. The game ends when the agent is killed by an enemy, drops from a cliff, runs out of time, or finishes the level. The RL agent's reward corresponds to the points collected in the Super Mario game (e.g. for collecting coins or killing enemies) and to a large extent on being able to complete the level. Negative rewards are given for getting hurt by enemies or falling off a cliff.

The actions available to the Mario agent correspond to the buttons on the original game controller, which are (left, right, no direction), (jump, don't jump), and (run/fire, don't run/fire). One action from each of these groups can be taken simultaneously, resulting in 12 distinct combined actions. We use the same state-space as described in [3], which involves 27 discrete state features:

1 is Mario able to jump? (Boolean)
2 is Mario on the ground? (Boolean)
3 is Mario able to shoot fireballs? (Boolean)
4-5 Mario's direction in the horizontal and vertical planes ($\{-1, 0, 1\}$)
6-9 is there an obstacle in one of the four vertical grid cells in front of Mario? (Boolean)
10-17 is there an enemy within one grid cell removed from Mario in one of eight different directions (left, up-left, up, up-right, etc.) (Boolean)
18-25 as the previous, but for enemies within two to three grid cells. (Boolean)
26-27 the relative horizontal and vertical positions of the closest enemy ($(-10, 10)$, measured in grid cells, plus one value indicating an absence of enemies)

Similar to the CartPole domain, we used Q(λ)-learning and RLfD as benchmarks. The parameters were taken from [3], with the learning rate $\alpha = 0.001$, discount factor $\gamma = 0.9$, ϵ-greedy exploration with $\epsilon = 0.05$, $\lambda = 0.5$ with the additional $\sigma = 0.5$ for RLfD.

Figure 5 shows the learning curves of Q(λ)-learning and the introspective reinforcement learning agent without demonstrations in the Super Mario domain. We show results for two different values of the potential function scaling factor ρ. The results are similar to the CartPole domain in that the learning performance of the introspective reinforcement learning agent without demonstrations significantly improves over that of $Q(\lambda)$-learning, and reaches the asymptotic performance level earlier.

In another set of experiments, shown in Figure 6, 20 demonstration episodes from a human player (all with a performance score between 400 to 650) are used to initialise the priority queue for introspective RL. The results show that also in this highly complex

Figure 4: A Super Mario screen shot

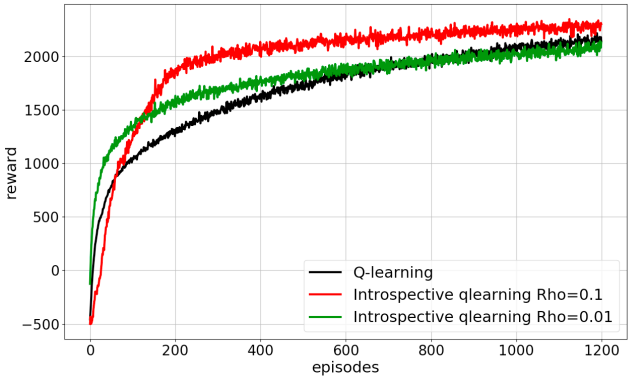

Figure 5: Super Mario Domain learning curves of Q(λ)-learning, and Introspective RL without demonstration

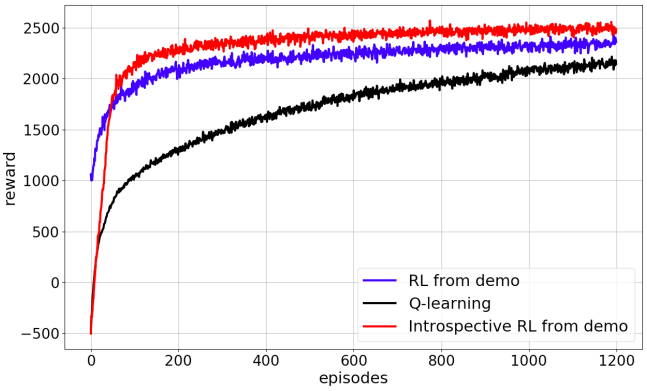

Figure 6: Super Mario Domain learning curves of Q(λ)-learning, RLfD, and Introspective RL with demonstration

domain, introspective RL with demonstrations outperforms both RLfD and regular $Q(\lambda)$-learning.

5 CONCLUSION AND FUTURE WORK

In this paper, we presented the idea of introspective reinforcement learning which takes inspiration from learning by demonstration. However, instead of using external expert demonstration to guide the learning, the reinforcement learning agent uses its own experiences that have produced high rewards in the past. The agent keeps a priority queue to record the most successful experiences as tuples of the form $\langle s, a, \widehat{q} \rangle$, where the Q value is obtained by Monte Carlo estimation. When the agent executes an action that is the same as the action of the most similar state in the priority queue, a reward based on the product of ρ, $\widehat{q}$, and the degree of similarity is added, where ρ is the hyper parameter that controls the weight of the reward shaping signal. Human experts' demonstrations can be used by injecting the corresponding state-action-q triples into the priority queue. In this way, introspective RL can be complementary to reinforcement learning from demonstration.

We empirically evaluated our introspective RL approach on two domains, namely the CartPole domain with 4 continuous features and the Super Mario domain with 27 discrete features. The results showed that the introspective reinforcement learning agent surpasses regular $Q(\lambda)$-learning performance significantly and reaches the asymptotic performance much earlier. Furthermore, introspective reinforcement learning with demonstrations significantly improves performance in both domains compared to state-of-the-art reinforcement learning from demonstration. The empirical results also show that while introspective RL can use up more computational time per learning step, this loss is counterbalanced by a reduced sample complexity, which is generally more critical than computational complexity.

In future work, we intend to apply introspective RL to deep Q learning. For example, the similarity of parameters in high layers of the Q-value neural network may be considered to represent the similarity of state in a high-level concept and could be used for reward shaping. Another promising area of investigation is the optimisation of the balance between the extra reward signals and the real rewards from the environment. A more sophisticated control of the parameter ρ (e.g. decay over time) could help to improve the learning performance.

REFERENCES

[1] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. 2009. A survey of robot learning from demonstration. *Robotics and autonomous systems* 57, 5 (2009), 469–483.

[2] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. 2016. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*. 1471–1479.

[3] Tim Brys. 2016. *Reinforcement Learning with Heuristic Information*. Ph.D. Dissertation. Vrije Universiteit Brussel.

[4] Tim Brys, Anna Harutyunyan, Halit Bener Suay, Sonia Chernova, Matthew E Taylor, and Ann Nowé. 2015. Reinforcement Learning from Demonstration through Shaping.. In *IJCAI*. 3352–3358.

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

[6] Anna Harutyunyan, Sam Devlin, Peter Vrancx, and Ann Nowé. 2015. Expressing arbitrary reward functions as potential-based advice. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*.

[7] Sergey Karakovskiy and Julian Togelius. 2012. The mario ai benchmark and competitions. *IEEE Transactions on Computational Intelligence and AI in Games* 4, 1 (2012), 55–67.

[8] Maja J Mataric. 1994. Reward functions for accelerated learning. In *Machine Learning: Proceedings of the Eleventh international conference*. 181–189.

[9] Donald Michie and Roger A Chambers. 1968. BOXES: An experiment in adaptive control. *Machine intelligence* 2, 2 (1968), 137–152.

[10] Andrew Y. Ng, Daishi Harada, and Stuart Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, Vol. 99. 278–287.

[11] Andrew Y Ng, Stuart J Russell, et al. 2000. Algorithms for inverse reinforcement learning.. In *Icml*. 663–670.

[12] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. 2017. Curiosity-driven exploration by self-supervised prediction. *arXiv preprint arXiv:1705.05363* (2017).

[13] Stefan Schaal. 1997. Learning from demonstration. *Advances in neural information processing systems* 9 (1997), 1040–1046.

[14] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2015. Prioritized experience replay. *arXiv preprint arXiv:1511.05952* (2015).

[15] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016), 484–489.

[16] Satinder P. Singh and Richard S. Sutton. 1996. Reinforcement learning with replacing eligibility traces. *Machine learning* 22, 1-3 (1996), 123–158.

[17] William D. Smart and Leslie Pack Kaelbling. 2002. Effective reinforcement learning for mobile robots. In *IEEE International Conference on Robotics and Automation*, Vol. 4. IEEE, 3404–3410.

[18] Halit Bener Suay, Tim Brys, Matthew E Taylor, and Sonia Chernova. 2016. Learning from demonstration for shaping through inverse reinforcement learning. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 429–437.

[19] R.S. Sutton and A.G. Barto. 1998. *Reinforcement learning: An introduction*. Vol. 1. Cambridge Univ Press.

[20] Matthew E Taylor and Peter Stone. 2009. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research* 10, Jul (2009), 1633–1685.

[21] Matthew E Taylor, Halit Bener Suay, and Sonia Chernova. 2011. Integrating reinforcement learning with human demonstrations of varying ability. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*. International Foundation for Autonomous Agents and Multiagent Systems, 617–624.

[22] John N. Tsitsiklis. 1994. Asynchronous stochastic approximation and Q-learning. *Machine Learning* 16, 3 (1994), 185–202.

[23] Christopher John Cornish Hellaby Watkins. 1989. *Learning from delayed rewards*. Ph.D. Dissertation. University of Cambridge.

[24] Eric Wiewiora, Garrison Cottrell, and Charles Elkan. 2003. Principled methods for advising reinforcement learning agents. In *ICML*. 792–799.