Reinforcement Learning for Large and Variable Scale Problems using Improvement Based Rewards

Abhik Ray , **Richa Verma**[*] and **Harshad Khadilkar**
TCS Research
{ray.abhik, richa.verma4, harshad.khadilkar}@tcs.com,

Abstract

We analyze the convergence properties and potential advantages of an improvement based reward function. The proposed approach masks the actual rewards generated by the environment from the agent, instead providing a fixed positive reward for continuous improvement over past reward values and negative reward for failing to do so. The key advantage of this approach is to make the learned values or policies (as applicable) independent of the actual scale or complexity of the problem, allowing us to transfer learning across a wide range of small and large scale instances of the same problem with minimal re-learning. We follow a theoretical treatment of the approach with tests on benchmark problems, and also cite cases where it has been successfully used on real-world, large scale problems.

1 Introduction

A majority of reinforcement learning (RL) studies in literature focus on gameplay [Mnih *et al.*, 2015; Van Hasselt *et al.*, 2016]. The scale of such problems, measured in terms of size of the state space, can be very large. Consequently, most studies report results using significant hardware resources and training time [Mnih *et al.*, 2016; Silver *et al.*, 2017]. There is also a growing body of literature on applications in operations research [Gambardella and Dorigo, 1995; Zhang and Dietterich, 1995], robotics [Gu *et al.*, 2017], and networked systems [O'Neill *et al.*, 2010]. The challenge of scaling of the instance sizes in such problems is just as acute; however, there is sometimes the added complexity of limited hardware availability. Furthermore, there is a unique challenge in such problems, which is that of *variable* scale. We refer to problems such as job shop scheduling [Zhang and Dietterich, 1995], where the number of jobs and machines may vary from one instance to another. Were an RL algorithm to be deployed to solve such problems in real-time, it would not only need to be scalable, but in a sense also *scale-free*.

One approach to generalising learning across instances of variable scale within the same class of problems, is to mask the actual reward function. We describe a threshold-based reward for RL algorithms, which provides positive reward when the agent improves on its own past performance, and negative reward otherwise. This reward mechanism can be used in conjunction with any standard RL algorithm (value or policy based) without additional changes.

We have used this approach in practical applications, including railway scheduling [Khadilkar, 2019] and container loading for ships [Verma *et al.*, 2019]. The railway scheduling problem solves instances with 450 trains passing through 50 stations, while the container loading problem solves instances with 1500 slots on a ship, to be filled from a pool of 100,000 containers in the yard. In both cases, a constant sized state and action space was devised in addition to the threshold based reward. This allowed us to reuse the models on instances of different scale, for example across different railway networks or across ships with different container capacities. The true optimal reward varied from one problem instance to another, but the threshold based reward could distinguish between *good* and *bad* outcomes.

In this paper, the reward function is analysed from the perspective of convergence, and we prove that this is guaranteed for a general RL algorithm as long as the original algorithm (for example, Q-Learning) is guaranteed to converge to the optimal value or policy. As far as we are aware, this is the first theoretical proof of convergence of this type of method to an optimal policy as the algorithm without the tweaked reward. We perform empirical tests on the multi-armed bandit problem as well as the sequential decision-making environment 'Gridworld'. The development of the proof in Section 3 focuses on Q-Learning for illustration, but the same logical steps can be applied to any standard RL algorithm.

2 Description of methodology

Consider an episodic Markov Decision Process (MDP) specified by the standard tuple $< \mathcal{S}, \mathcal{A}, \mathcal{R}, P >$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{R}$ is the set of possible rewards, and P is the transition function. We assume the existence of a reinforcement learning algorithm for learning the optimal mapping $\mathcal{S} \to \mathcal{A}$. For illustration, we will assume that the algorithm to be used is Q-Learning [Sutton and Barto, 2018]. Typically, the reward structure is a natural consequence of the problem from which the MDP was derived. For example, in the popular environment Gridworld, the task is to navigate through a 2-D grid towards a goal state. The

[*]Contact Author

most common reward structure in this problem is to provide a small negative step reward for every action that does not end in the goal state, and a large positive terminal reward for reaching the goal state. It follows that the value of optimal reward depends on both the values of the step and terminal rewards, as well as on the size of the grid. In this paper, we replace the rewards $\mathcal{R}$ completely by a threshold based binary episodic reward structure $r_t : \mathcal{S}, \mathcal{A}, \mathcal{S} \rightarrow \mathbb{R}$:

$$r_t(s_k, a_k, s_{k+1}) = \begin{cases} +z_p, & s_{k+1} \in \mathcal{T} \ \ \& \ \ G_t \geq \rho_t \\ -z_n, & s_{k+1} \in \mathcal{T} \ \ \& \ \ G_t < \rho_t \\ 0, & \text{otherwise} \end{cases}$$

where k is the step within an episode, t is the number of the episode, $\mathcal{T}$ is the set of terminal states, G_t is the return for episode t, ρ_t is the performance threshold at episode t, z_p and z_n are the magnitudes of the positive and negative terminal rewards respectively. Note that the return G_t is based on the original reward structure of the MDP. If the original step reward at k is R_k, then $G_t = \sum_{k=0}^{T} R_k$. The net effect of the reward structure is to provide a positive terminal reward z_p if $G_t \geq \rho_t$. The threshold itself is updated using the relation,

$$\rho_{t+1} = \begin{cases} \rho_t + \beta_t\left(\frac{\sum_{x=1}^{t} G_x}{t} - \rho_t\right) & \text{if q-values converged} \\ \rho_t & \text{otherwise} \end{cases},$$

where $\beta_t \in (0, 1)$ is the step size and is assumed to be externally defined according to a fixed schedule.

Training process: We assume that the q-values must converge for every value of the threshold before its own value is updated. The threshold updates and q-value updates thus happen alternately. If the initial threshold value is very low, it is fairly easy for the algorithm to achieve positive terminal reward, and a large proportion of state-action pairs converge to positive q-values. During the next threshold update, the high average returns G_x since the last update result in an increase in value of ρ_t. The threshold thus acts as a lagging performance measure of the algorithm over the training history. Practically, we find in Section 4 that we can afford to update the threshold after every episode instead of waiting for the q-values to converge every time. The algorithm is said to have converged when both the threshold and the q-values converge. A schematic of the procedure is shown in Figure 1. The original rewards R only affect the returns G, which in turn are used to update the threshold ρ. At the end of each episode, the current return and threshold values are used to compute the new rewards r, which implicitly or explicitly drive the policy π.

$$\begin{tikzcd}
G \arrow[rr, "update"] \arrow[drr] & & \rho \arrow[d] \\
R \arrow[u, "sum"] & & r \arrow[dl, "update"] \\
& \pi \arrow[ul, "action"] &
\end{tikzcd}$$

Figure 1: Training process at a fixed t

3 Convergence

At time step t, we can use any reinforcement learning algorithm proven to converge to an optimal policy (or even much milder conditions as shown in section 2.2) and let it converge for the new reward structure r with threshold ρ_t. For example, with Q-Learning it is known that Q values converge to Q^* under certain (mild) conditions [Jaakkola *et al.*, 1994].

3.1 Proof

Let $\rho^* = \mathbb{E}_{\pi^*}[G] = V^*(s_0)$ be the maximum expected return from the start state for the original reward structure.

Case 1: $\mathbb{E}[\rho_t] < \rho^*$

There exists a policy for which $\mathbb{E}[G] = \rho^*$, therefore there exist policy for which $\mathbb{E}[G_t] > \mathbb{E}[\rho_t]$. If we let the RL algorithm converge, $\mathbb{E}[G_t] > \mathbb{E}[\rho_t]$

$$\begin{aligned} \rho_{t+1} &= \rho_t + \beta_t(G_t - \rho_t) \\ \mathbb{E}[\rho_{t+1}] &= \mathbb{E}[\rho_t] + \beta_t \mathbb{E}[G_t - \rho_t] \\ \mathbb{E}[\rho_{t+1}] &> \mathbb{E}[\rho_t] \end{aligned} \tag{1}$$

$$\{\text{Since } \beta_t > 0 \text{ and } \mathbb{E}[G_t] > \mathbb{E}[\rho_t]\}$$

$$\begin{aligned} \rho_{t+1} &= (1 - \beta_t)\rho_t + \beta_t G_t \\ \mathbb{E}[\rho_{t+1}] &= (1 - \beta_t)\mathbb{E}[\rho_t] + \beta_t \mathbb{E}[G_t] \\ &< (1 - \beta_t)\rho^* + \beta_t \rho^* \\ \mathbb{E}[\rho_{t+1}] &< \rho^* \end{aligned} \tag{2}$$

$$\{\text{Since } \rho^* \geq \mathbb{E}[G_t] \text{ by definition}\}$$

From (3) and (4):

$$\mathbb{E}[\rho_t] < \mathbb{E}[\rho_{t+1}] < \rho^* \tag{3}$$

Case 2: $\mathbb{E}[\rho_t] > \rho^*$

There exists no policy for which $\mathbb{E}[G_t] \geq \mathbb{E}[\rho_t]$ since by definition of ρ^*, it is the maximum expected return. If we let the RL algorithm converge, $\mathbb{E}[G_t] < \mathbb{E}[\rho_t]$

$$\begin{aligned} \rho_{t+1} &= \rho_t + \beta_t(G_t - \rho_t) \\ \mathbb{E}[\rho_{t+1}] &= \mathbb{E}[\rho_t] + \mathbb{E}[\beta_t(G_t - \rho_t)] \\ \mathbb{E}[\rho_{t+1}] &< \mathbb{E}[\rho_t] \end{aligned} \tag{4}$$

$$\{\text{Since } \beta_t > 0 \text{ and } \mathbb{E}[G_t] < \mathbb{E}[\rho_t]\}$$

Case 3: $\mathbb{E}[\rho_t] = \rho^*$

There exists a policy for which $\mathbb{E}[G] = \rho^*$, therefore for the same policy $\mathbb{E}[G_t] = \mathbb{E}[\rho_t] = \rho^*$. If we let the RL algorithm converge, $\mathbb{E}[G_t] = \mathbb{E}[\rho_t]$

$$\begin{aligned} \rho_{t+1} &= \rho_t + \beta_t(G_t - \rho_t) \\ \mathbb{E}[\rho_{t+1}] &= \mathbb{E}[\rho_t] + \mathbb{E}[\beta_t(G_t - \rho_t)] \\ \mathbb{E}[\rho_{t+1}] &= \mathbb{E}[\rho_t] = \rho^* \end{aligned} \tag{5}$$

$$\{\text{Since } \mathbb{E}[G_t - \rho_t] = 0\}$$

Hence proved,

$$\begin{aligned}
\mathbb{E}[\rho_t] < \rho^* &\implies \mathbb{E}[\rho_t] < \mathbb{E}[\rho_{t+1}] < \rho^* \\
\mathbb{E}[\rho_t] > \rho^* &\implies \mathbb{E}[\rho_{t+1}] < \mathbb{E}[\rho_t] \\
\mathbb{E}[\rho_t] = \rho^* &\implies \mathbb{E}[\rho_{t+1}] = \mathbb{E}[\rho_t] = \rho^*
\end{aligned}$$

Therefore $\rho \to \rho^*$ and the optimal policy for the new reward structure when $\rho = \rho^*$ is an optimal policy for the original MDP since the definition of optimal policy is one that attains maximum expected reward.

3.2 Notes on the proof

- Convergence to optimal policy is not required after each threshold update. Should just be sufficient for:

  $$\begin{gathered}
  \mathbb{E}[G_t] > \mathbb{E}[\rho_t] \text{ for } \mathbb{E}[\rho_t] < \rho^* \\
  \mathbb{E}[G_t] < \mathbb{E}[\rho_t] \text{ for } \mathbb{E}[\rho_t] > \rho^* \\
  \mathbb{E}[G_t] = \mathbb{E}[\rho_t] \text{ for } \mathbb{E}[\rho_t] = \rho^*
  \end{gathered}$$

  Which is practically not difficult for any RL algorithm as the following note shows.
- At time step t, let the threshold and the policy be such that $\mathbb{E}[G_t] > \mathbb{E}[\rho_t]$. The change in threshold is just $\beta_t(G_t - \rho_t)$, so it should not take many episodes of training to get a policy that achieves $\mathbb{E}[G_{t+1}] \geq \mathbb{E}[\rho_{t+1}]$. Practically updating threshold after every episode also works.
- Step in the direction of optimality for the new reward structure r_t is a step in the direction of optimality for the original reward structure $\mathcal{R}$.

4 Results

4.1 Multi-Armed Bandit

The 10-armed Testbed [Sutton and Barto, 2018] was used to test the convergence of ϵ-Greedy, Softmax and UCB with the threshold based reward. The experiments show that the average reward and the percentage of times optimal action is taken for the threshold based reward structure are the same as that of the original reward structure on average across a range of parameters, as well as the best parameters found after a grid search. The number of training steps taken to converge to the optimal value is also the same on average. The metric used for the average reward obtained after convergence is the average reward of the last 100 steps out of the 2000 training steps. The metric used to test the time take to converge is the average rewards of all the 2000 training steps.
Figure 2 shows a run comparing the original reward structure with threshold based reward structure using epsilon-greedy ($\epsilon = 1$ and ϵ-decay $= 0.995$) with $z_p = 1$ and $z_n = 1$. The threshold is updated after every episode (and not waiting for convergence after every threshold update).

4.2 Gridworld

A variable size Gridworld environment was set up with a variable reward structure with a negative step reward, a positive reward for reaching the gold, and a negative reward for reaching a state with a bomb. The initial position, the gold position and bomb position are all randomly set. An episode terminates after reaching the gold, bomb or after 100 steps. The experiments show that the threshold based reward structure converges to the same average reward as the original one on average across various agent and environment parameters and the best agent parameters found using a grid search. Again, the convergence rate is also the same on average. The metric used for the average reward obtained after convergence is the average reward of the last 100 steps out of the 2000 training steps. The metric used to test the time take to converge is the average rewards of all the 2000 training steps.

Figure 3 shows a run for a 15x15 Gridworld with the initial position as (0,0), the bomb placed at (7,7) and the gold placed at (14,14). $\gamma = 0.9$, $\epsilon = 1$, ϵ-decay $= 0.995$, $z_p = 1$, $z_n = 1$, learning rate for the original reward structure is 0.2 and the learning rate for the threshold based q-values as well as the threshold update is 0.05. $success - ratio = 0.9$ which means that instead of the return having to be greater than or equal to the threshold every time, it is sufficient to do better than 0.9 of the current threshold value to get a positive reward. Initially there is a -0.01 penalty for every step, a -1 penalty for reaching the bomb and a $+1$ reward for reaching the gold. At step 1000, this reward structure is changed (while maintaining the q-values) to a -0.08 penalty for every step, a -5 penalty for reaching the bomb and a $+10$ reward for reaching the gold. One of the potential advantages of the threshold based reward structure is that it adjusts to the non linearly re-scaled rewards quickly and smoothly. The intuition behind it is that it is able to distinguish between $good$ and bad decisions and outcomes in a scale invariant way.

Further in our research, we plan to test the potential advantages of using a threshold-based reward, in terms of stability and performance for additional RL algorithms (especially recent advances such as DDPG, TRPO, and PPO). We also want to use the technique along with curriculum learning [Yoshua Bengio, 2009] especially in partially observed environments where the state and action space size remains the same (local) but the environment complexity (scale, observability, stochasticity, non-linearity) is scaled up. We would also like to explore the effect of parameters such as z_p and z_n, the value and schedules for β_t, the number of episodes per threshold update, the value and schedules for $success{-}ratio$, and the initial value of the threshold (and optimistic starts). Finally, there remains the possibility of including intermediate rewards and non-episodic MDPs within the threshold-based setup.

References

[Gambardella and Dorigo, 1995] Luca M Gambardella and Marco Dorigo. Ant-q: A reinforcement learning approach to the traveling salesman problem. In *Machine Learning Proceedings 1995*, pages 252–260. Elsevier, 1995.

[Gu *et al.*, 2017] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017.

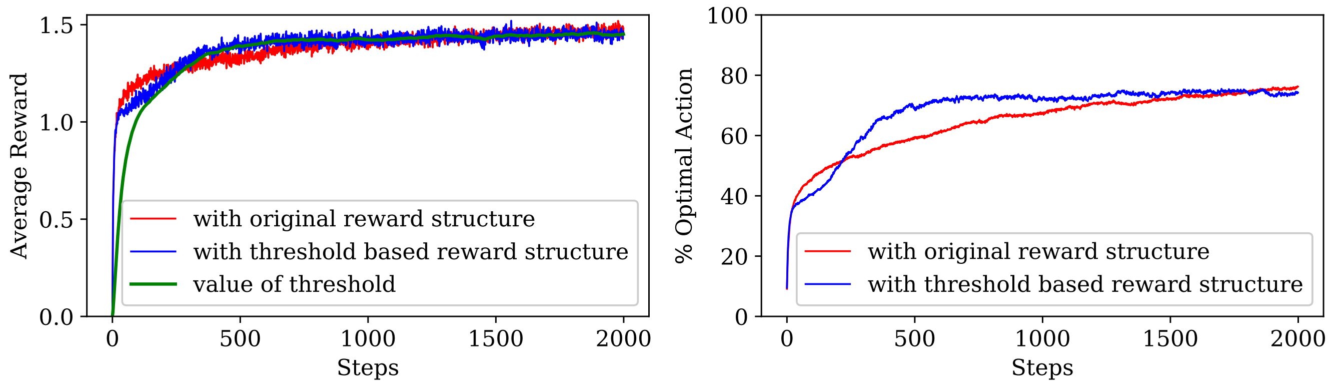

Figure 2: 10-armed Testbed [Sutton and Barto, 2018] using epsilon-greedy.

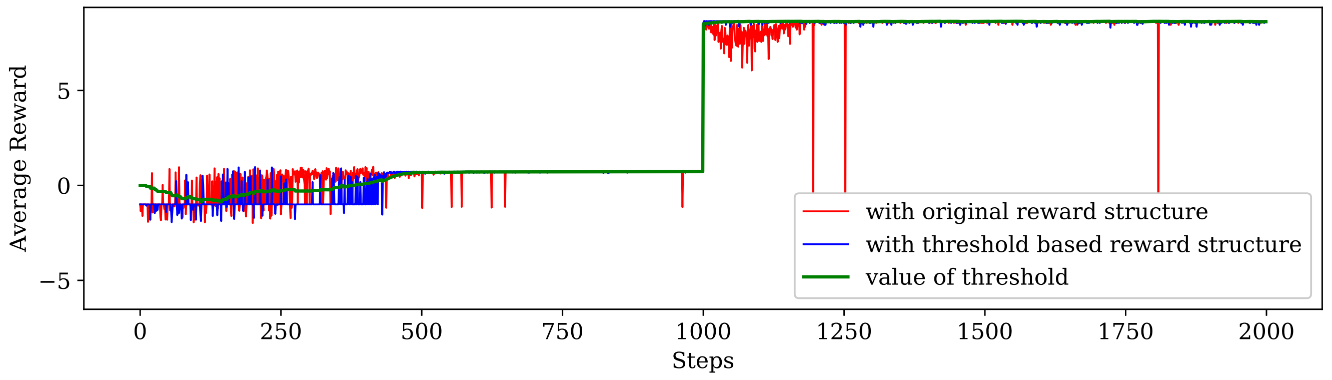

Figure 3: 15x15 Gridworld with rewards scaled non-linearly at step 1000. Using Q-Table with epsilon-greedy.

[Jaakkola *et al.*, 1994] Tommi Jaakkola, Michael Jordan, and Satinder Singh. On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6:1185–1201, 11 1994.

[Khadilkar, 2019] Harshad Khadilkar. A scalable RL algorithm for scheduling railway lines. *IEEE Transactions on Intelligent Transportation Systems*, 20(2):727–736, 2019.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep RL. *Nature*, 518(7540):529, 2015.

[Mnih *et al.*, 2016] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep RL. In *International conference on machine learning*, pages 1928–1937, 2016.

[O'Neill *et al.*, 2010] Daniel O'Neill, Marco Levorato, Andrea Goldsmith, and Urbashi Mitra. Residential demand response using RL. In *IEEE International Conference on Smart Grid Communications*, pages 409–414. IEEE, 2010.

[Silver *et al.*, 2017] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[Sutton and Barto, 2018] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 2nd edition, 2018.

[Van Hasselt *et al.*, 2016] Hado Van Hasselt, Arthur Guez, and David Silver. Deep RL with double q-learning. In *AAAI*, volume 2, page 5. Phoenix, AZ, 2016.

[Verma *et al.*, 2019] Richa Verma, Sarmimala Saikia, Harshad Khadilkar, Puneet Agarwal, Ashwin Srinivasan, and Gautam Shroff. An RL framework for container selection and ship load sequencing in ports. In *International conf. on autonomous agents and multi agent systems*, 2019.

[Yoshua Bengio, 2009] Ronan Collobert Ronan Collobert Yoshua Bengio, Jérôme Louradour. Curriculum learning. In *ICML '09 Proceedings of the 26th Annual International Conference on Machine Learning*, pages 41–48, 2009.

[Zhang and Dietterich, 1995] Wei Zhang and Thomas G Dietterich. An RL approach to job-shop scheduling. In *IJCAI*, volume 95, pages 1114–1120. Citeseer, 1995.