Shared Learning in Ensemble Deep Q-Networks

Rakesh R Menon [*]
Indian Institute of Technology Madras
rakeshrmenon1995@gmail.com

Manu Srinath Halvagal[*]
Indian Institute of Technology Madras
mshalvagal@gmail.com

Balaraman Ravindran
Indian Institute of Technology Madras
ravi@cse.iitm.ac.in

ABSTRACT

The exploration-exploitation dilemma in Reinforcement Learning problems, although well-studied, is still a major challenge as existing techniques do not scale well to environments with large state spaces. Most deep RL algorithms use a naive extension of conventional exploration techniques like ϵ-greedy. However, recent work such as Bootstrapped DQN and A3C have introduced novel exploration strategies based on ensembles of value estimates that address these scalability issues to a large extent. In this work, we present an algorithm that takes advantage of the ensemble architecture of Bootstrapped DQN to further speed up learning by sharing useful learning progress amongst the bootstrapped value estimates.

Keywords

Co-operative learning, ensemble exploration, deep Q-networks

1. INTRODUCTION

Most deep Reinforcement Learning (RL) solutions still use extensions of conventional exploration strategies that have been well studied in bandit problems and simple MDPs. However, exploration in large state spaces needs to be more directed than is possible with these traditional exploration strategies such as ϵ-greedy. The recently proposed Bootstrapped DQN [9] offers a new exploration strategy that is capable of deep directed exploration and is better suited for deep RL problems.

Bootstrapped DQN works by learning multiple independent estimates of the action-value function and guiding action selection using a randomly selected estimate. The method relies on variability among the different action-value estimates (called heads) for effective and deep exploration. In Osband et al.[9], this variability is ensured through both selective masking of training examples as well as by the random initialization of network parameters of each head. However, while Bootstrapped DQN explores by keeping its estimates completely independent of each other, it is possible to speed up learning in heads that dither in the earlier learning stages by partially guiding them by using shared learning progress from more successful heads.

In this work, we present a method for the bootstrap heads to learn from each other in order to speed up learning in the initial learning stages. Periodically, the head with the highest Q-value is selected to provide information to the other heads. This information is passed in the form of action selection in a double Q-learning update [5]. We show the efficacy of our improved model on a toy MDP chain example, a small maze-like task (puddle world) and finally on Atari 2600 games using the ALE environment[3].

[*]Equal contribution

2. RELATED WORK

Prior work on exploration strategies in Reinforcement Learning have produced algorithms like R_{max} [4] and E^3 [7], which have near-optimal results and theoretical guarantees on MDP problems. However, such algorithms are intractable when it comes to exploration in domains with large state spaces. With the introduction of Deep Q-Networks (DQN) [8] for such domains, there was a need for better exploration strategies other than ϵ-greedy in order to perform human level control in these complex environments.

Some recent works have studied the use of count-based methods that incorporate exploration bonuses based on a count of the number of times a state-action pair is visited. These bonuses are meant to promote the visitation of state-action pairs that have been sparsely visited. Tang et al.[13] uses the technique of hashing, wherein a part of the continuous state space is hashed into a discrete state space and exploratory bonuses are given based on the visitation of the discretized space. Bellamare et al.[2] has tried to predict pseudo counts for state-action pairs using information theoretic approaches. This method has provided state-of-the-art performance on Montezuma's Revenge.

Another way of calculating exploratory bonuses involves the use of intrinsic motivation [10, 1] and its idea of state saliency. This state saliency idea has been exploited well in Stadie et al.[11] wherein a model prediction error is used for calculating the exploratory bonus in Atari Games. Variational Information Maximizing Exploration (VIME) [6] extends the idea of intrinsic motivation to environments with continuous state action spaces and encourages exploration by getting information about the environment dynamics.

Some of these methods seem to face an issue when it comes to dealing with sparse rewards.Bootstrapped DQN [9] was one of the first algorithms to introduce and show the effectiveness of ensemble learning of Q-functions towards deep exploration in Atari games. In this paper, we primarily focus on trying to speed up exploration by keeping track of and utilising the learning progress made by individual estimates in the Bootstrapped DQN architecture.

3. BACKGROUND

3.1 Q-learning

Reinforcement Learning deals with learning control policies in sequential decision making problems. A common approach is to learn an action-value function $Q(s, a) : S, A \rightarrow \mathbb{R}$ over the set of states S and actions A that can be taken from each state. The action-value function reflects the long-term profitability of taking any action. Q-learning [15] is one method to learn the optimal action-value function for an agent. The optimal policy can be achieved by behaving greedily with respect to the learnt action-value function in each state. The update rule for Q-learning is given as :

$$\begin{aligned} Q_{t+1}(s_t, a_t) =& Q_t(s_t, a_t) + \alpha(r_t + \gamma max_a Q_t(s_{t+1}, a) \\ & - Q_t(s_t, a_t)) \end{aligned}$$

where $Q_t(s, a)$ and $Q_{t+1}(s, a)$ are the action-value function estimates at times t and $t+1$ respectively, r_t is the reward obtained at time t for choosing action a_t in state s_t, α is the learning rate, and γ is the discount factor.

3.2 Double Q-learning

The max operator in the Q-learning update has been shown to produce overestimation of the action-value function. This comes about because a single estimator is used both for choosing the next action and for giving an estimate for the value of that action. Double Q-learning [5] reduces the overestimations by decoupling the action selection and value estimation by training two estimators $Q^A(s, a)$ and $Q^B(s, a)$. The update rule for double Q-learning is given as:

$$\begin{aligned} Q^A_{t+1}(s_t, a_t) =& Q^A_t(s_t, a_t) \\ & + \alpha(r_t + \gamma Q^B_t(s_{t+1}, argmax_a Q^A_t(s_t, a)) \\ & - Q^A_t(s_t, a_t)) \end{aligned}$$

3.3 Deep Q-Networks (DQN)

In environments with large state spaces, it is not possible to learn values for every possible state-action pair. The need for generalising from experience of a small subset of the state space to give useful approximations of the action-value function becomes a key issue [12]. Neural networks, while attractive as potential action-value function approximators, were known to be unstable or even to diverge on Reinforcement Learning problems. Mnih et al.[8] successfully overcame these problems with two crucial ideas: *replay memory* and *target networks*. The parametrised action-value function was trained using the expected squared TD-error as the loss signal given to the neural network.

$$L_i(\theta_i) = \mathbb{E}_{s,a,r,s'}[((r + \gamma max_{a'} Q(s, a'; \theta_i^-) - Q(s, a; \theta_i))^2]$$

Here, $L_i(\theta_i)$ is the loss function, θ_i is the current parameters of the network and θ_i^- is the parameters of the target network. The target network is needed in order to provide stationary targets so as to ensure stable training. The purpose of the replay memory is to reduce the correlation of the samples provided to the network during training.

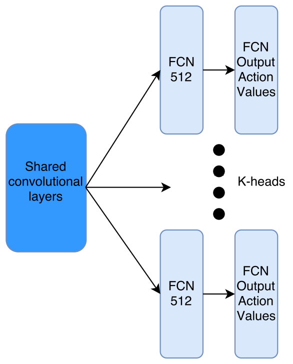

Figure 1: Bootstrapped DQN architecture

3.4 Bootstrapped DQN

The key idea behind exploration with ensembles is that although all the action-value estimators may converge to the same policy eventually, a large number of exploratory actions are taken as a consequence of different estimators deciding the behaviour in each episode during the initial phase of learning. Bootstrapped DQN [9] algorithm is implemented by adding multiple head networks which branch out from the output of the CNN as shown in Figure 1. The outputs from each head represent different independent estimates of the action-value function, which promotes exploration. Surprisingly, random initialization of the heads alone was seen to be sufficient to ensure the required variability. The author(s) claim that when a new action is sampled by one of the heads of the network, the TD bootstrapping update can propagate signals through the target network to drive exploration.

4. SHARED LEARNING

With the bootstrap exploration framework, we have k different action-value function estimates. While there is no communication between heads in the original Bootstrapped DQN, that extent of independence between the heads might be unnecessary. Our work investigates if some amount of guidance of dithering heads by other more successful heads would speed up learning in the overall ensemble. This would be in some sense an online transfer of learned knowledge. In our formulation, the extent of this sharing of experience can be varied by varying the probability with which a guided update is made instead of the regular DQN update.

The technique is formulated as follows. At regular intervals, a locally best estimator is selected on the basis of a metric that measures learning progress to a reasonable extent. Until the next such selection, this head guides the learning of the other heads through the double Q-learning update rule as given below.

$$\begin{aligned} Q^i_{t+1}(s_t, a_t) =& Q^i_t(s_t, a_t) \\ & + \alpha(r_t + \gamma Q^j_t(s_{t+1}, argmax_a Q^{best}_t(s_t, a)) \\ & - Q^i_t(s_t, a_t)) \end{aligned}$$

The metric that is chosen for measuring learning progress can be either cumulative TD-error or the action-value function estimates. The extent of such guidance can be varied by making it a stochastic decision whether to use guided target values or whether to use the normal update rule.

In this way, the ensemble agent can explore and understand parts of the state space which were rewarding initially really well. We hypothesize that using this method should be able to speed up learning of all the estimators at the expense of a slight decrease in exploration.

5. EXPERIMENTS

5.1 MDP

We present a toy chain MDP experiment to demonstrate the improved exploration of the new approach. The MDP consists of k states, arranged sequentially. The agent always starts from state 2. From each state *four* actions are allowed namely, go left, go right, do nothing or jump to state 1 incurring a reward of -10 and terminate the episode. The episode ends when the agent has reached the kth state upon which it receives a reward of +10.

We used normal Q-learning(with ϵ-greedy) and double Q-learning(with ϵ-greedy) as baselines to compare the performance of a bootstrap agent (5 heads) with and without shared learning. The figures below show the result for a 35 and 50 state MDP.

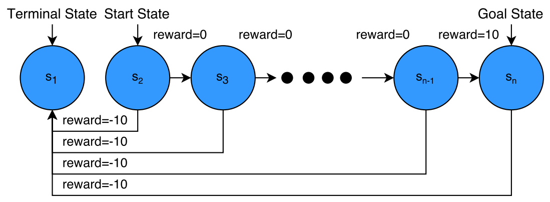

Figure 2: MDP Chain Structure with n-states

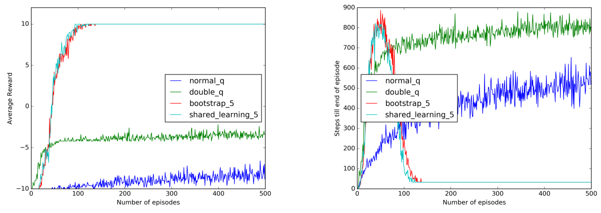

Figure 3: MDP results showing average reward obtained per episode and number of steps taken until completion for 35-state chain MDP

Solving the above environment requires deep exploration, especially with a larger number of states. This is illustrated by the fact that Q-learning and double Q-learning, with ϵ-greedy exploration, are unable to solve large MDP chains (beyond 20 states). Bootstrap DQN (adapted for tabular settings) showed how deep exploration can be performed by its algorithm and its benefits can be seen in the figures. We observe that the speedup due to shared learning becomes more crucial with increasing chain length. The performance of both Bootstrapped DQN and *shared learning* would probably be identical on this problem until the first time the goal state is reached. It is at this point that sharing

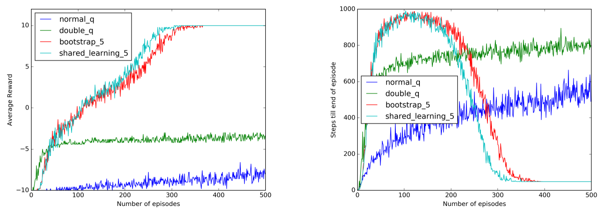

Figure 4: MDP results showing average reward obtained per episode and number of steps taken until completion for 50-state chain MDP

learned experience becomes vital so that knowledge of the goal state propagates out to every head. The faster this happens, the faster the entire network learns. This is the reason why *shared learning* outperforms the bootstrap algorithm on larger MDP chains. The fact that our algorithm is still capable of performing deep exploration shows that sharing does not take away from the diversity among the estimates, which is what drives the exploration in Bootstrapped DQN.

5.2 Puddle World

The puddle world is a typical grid world, with 4 stochastic actions. The actions might result in movement in a direction other than the one intended with a probability of 0.1. For example, if the selected action is North, it will transition to the cell one above the agent's current position with probability 0.9. It will transition to one of the other neighbouring cells with probability 0.1/3. Transitions that take it off the grid will not result in any change.

There is also a gentle Westerly blowing, that will push the agent one additional cell to the East, regardless of the effect of the action you took, with a probability of 0.5.

The episodes start with the agent in one of the start states in the first column, with equal probability. We inspect two variants of the problem, A(0,11) and C(6,7). In each of these, the goal is in the square marked with the respective alphabet as shown in Figure 5. There is a puddle in the middle of the grid world, which the agent should avoid. Every transition into a puddle cell, gives a negative reward, depending on the depth of the puddle at that point, as indicated in the figure, and a reward of +10 is given on reaching the goal. The environment used for these experiments do

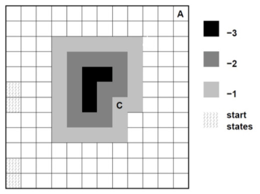

Figure 5: Layout of Puddle World

not necessarily require deep exploration as grid size is very small. This is why normal and double Q-learning, with ϵ-greedy, are able to complete the task successfully. However, what seems to be more interesting is the fact that Bootstrap DQN (adapted for tabular settings) is able to outperform the

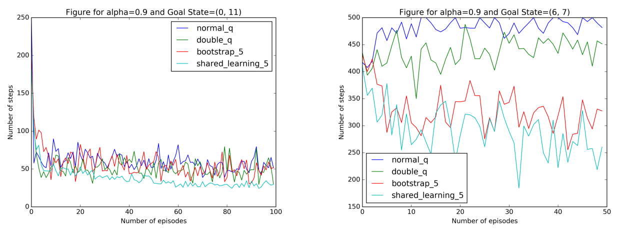

Figure 6: Puddle World results for Shared Learning and Bootstrap

above mentioned algorithms in a small world as well. This shows the effectiveness of ensemble exploration strategies for environments of all scales. Furthermore, shared learning seems to help the exploration from dithering to uninformative states. This can be seen more evidently in the case of goal C(6,7) wherein, bootstrap DQN was seen to marginally unlearn while the shared learning algorithm seems to help the agent in continuing to learn.

5.3 Arcade Learning Environment

We evaluate our model on 3 Atari games on the Arcade Learning Environment [3]. The games chosen for comparison were Frostbite (Figure 9) and Hero (Figure 10). These games have been chosen in order to compare with the results shown in Osband et al.[9]. The network architecture is same as that of Bootstrapped DQN and consists of a shared convolutional layer followed by 10 bootstrap heads to provide action-value function estimates. Gradient normalisation of $1/K$ (K is the number of bootstrap heads) was also applied to the network with no masking of heads. The learning progress for the Atari games have been measured, at intervals of 100 steps, using the Q-values. The head corresponding to the maximum Q-value is chosen to be the best head that is suitable for information transfer.

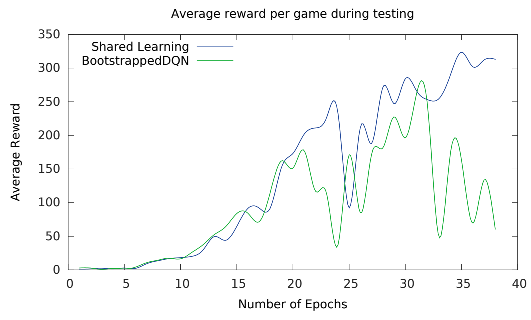

Figure 7: Average Reward on Breakout after 35 epochs of training

Table 1 shows the cumulative rewards obtained on the Atari games Pong and Qbert.

We have not presented the performance of DQN [8] and double DQN [14], since it was shown in Osband et al.[9] that Bootstrapped DQN clearly outperforms them. Our results show an increased cumulative reward during learning, while

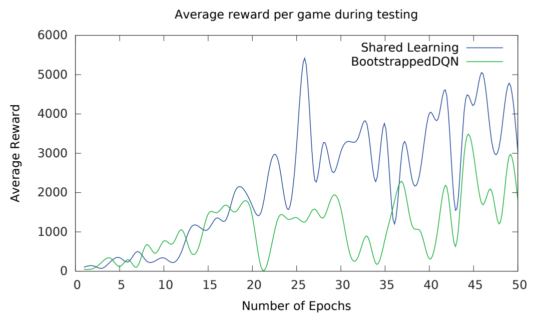

Figure 8: Average Reward on Seaquest after 50 epochs of training

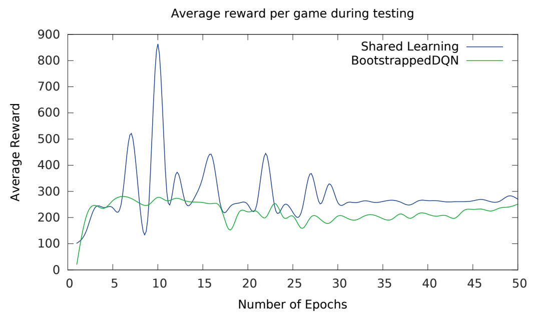

Figure 9: Average Reward on Frostbite after 50 epochs of training

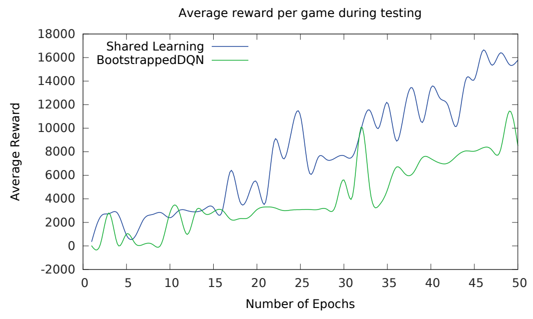

Figure 10: Average Reward on Hero after 50 epochs of training

Cumulative Reward on Atari Games during testing		
Game	Shared Learning Score	Bootstrapped DQN Score
Qbert	**39001.6661**	15286.0259
Pong	**-44.9735**	-80.4237

Table 1: 20 epoch cumulative scores

Bootstrapped DQN appears to dither to a larger extent. We are currently studying this behavior for all 60 games as well as for a larger number of epochs.

6. CONCLUSION AND FUTURE WORK

In this work, we have presented a method to speed up Bootstrapped DQN based on shared learning progress, without affecting its deep exploration capabilities. We have also introduced the notion of online transfer of knowledge, which we would like to investigate further on a wider array of AI tasks. While we have presented the sharing of learning through the double Q-learning update in the form of action-selection, we would like to look into other means of sharing knowledge via attention.

REFERENCES

[1] A. G. Barto. Intrinsic motivation and reinforcement learning. In *Intrinsically motivated learning in natural and artificial systems*, pages 17–47. Springer, 2013.

[2] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016.

[3] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Intell. Res.(JAIR)*, 47:253–279, 2013.

[4] R. I. Brafman and M. Tennenholtz. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct):213–231, 2002.

[5] H. V. Hasselt. Double q-learning. In *Advances in Neural Information Processing Systems*, pages 2613–2621, 2010.

[6] R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117, 2016.

[7] M. Kearns and D. Koller. Efficient reinforcement learning in factored mdps. In *IJCAI*, volume 16, pages 740–747, 1999.

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

[9] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy. Deep exploration via bootstrapped dqn. In *Advances In Neural Information Processing Systems*, pages 4026–4034, 2016.

[10] S. P. Singh, A. G. Barto, and N. Chentanez. Intrinsically motivated reinforcement learning. In *NIPS*, volume 17, pages 1281–1288, 2004.

[11] B. C. Stadie, S. Levine, and P. Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *CoRR*, abs/1507.00814, 2015.

[12] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[13] H. Tang, R. Houthooft, D. Foote, A. Stooke, X. Chen, Y. Duan, J. Schulman, F. D. Turck, and P. Abbeel. #exploration: A study of count-based exploration for deep reinforcement learning. *CoRR*, abs/1611.04717, 2016.

[14] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *AAAI*, pages 2094–2100, 2016.

[15] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.