Deep Recurrent Q-Learning for Partially Observable Environments using Advantage Prioritized Experience Replay

David Betancourt
School of Computational Science and Engineering
Georgia Tech
david.betancourt@gatech.edu

Abstract

This project presents and end-to-end method to train reinforcement learning agents using deep recurrent networks, while prioritizing the replay memory experiences with importance sampling. The measure to prioritize the experience is an energy spectral density combination of the Advantage function and the TD-error. Experiments with the partially observable Atari game `Frostbite` show that the method outperforms the DRQN without prioritized experience replay of [2] and converges to the optimal policy faster than the state-of-the art for the single-agent case in [7].

1 Introduction

Significant advances in deep reinforcement learning have been made over the last two years in the areas of games, natural language, and visual dialog. Deep recurrent Q-networks (DRQN) developed for partial observability use an RNN on top of CNNs in order to solve Partially Observable Markov Decision Processes (POMDPs) for the Atari benchmarks [2]. DRQN outperformed the vanilla, but yet effective, Deep Q-Network (DQN) [6] and human players in games with *induced*[1] partial observability. However, the DRQN has shortcomings, such as using the same experience replay heuristics as in DQN, where the experience transitions are sampled uniformly at random from the experience replay buffer. Such approach has been shown infective in [4] where DQN was investigated in a multi-agent setting for partially observable Markov Games. A superior sampling approach for experience replay was presented by [7] where experiences were replayed for DQN based on the magnitude of their importance, using Temporal-Difference (T-D) loss and importance sampling.

The goal of this project is to present an end-to-end method to train a neural Q-agent to act in a partially observable environment but that builds on the shortcomings of [2], using a new prioritized importance replay method, similar to the presented in [7] but more adept to POMDPs.

2 Related Work

The most well-known contemporary approach to integrating deep neural networks with reinforcement learning was developed in [6] and coined as deep Q-Learning (DQN). Other approaches have also been explored in deep reinforcement learning, such as Policy Gradient and Actor-Critic methods, which also include deep networks to estimate state-action pairs. All these methods make assumptions which essentially make them more suitable to solve Markov Decision Processes (MDPs).

[1] The partial observability in [2] was induced by flickering the screen of the Atari games. This was because in the DQN of [6] the last 4 frames of the game were taken as state input thereby converting most POMDP games to MPD.

In the area of POMDPs, the first to develop a deep learning approach was [2]. Their approach replaced the first fully connected layers used in [6] with the recurrency of LSTMs in order to capture the hidden states of the network to estimate the observation probability function.

3 Background

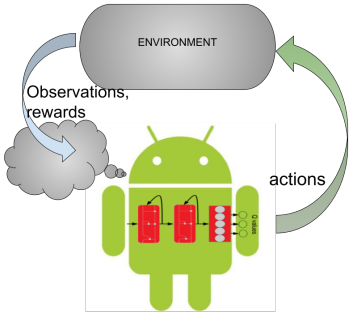

Figure 1: POMDP diagram.

MPDs and POMDPs In most real-world applications, the Markov Property does not hold because the agent cannot observe the full state of the environment. When this happens, the MPD becomes a POMDP and the agent only receives observations $o \in \mathcal{Z}$, instead of complete states $s \in \mathcal{S}$. A POMDP is defined by the following experience tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{Z}, \mathcal{O}\}$. At each time step t, the environment is in state $s_t \in \mathcal{S}$, and the agent takes an action $a_t \in \mathcal{A}$ according to a policy $\boldsymbol{\pi} : \mathcal{O} \times \mathcal{A} \longmapsto [0, 1]$, which causes the environment to transition to the new state s_{t+1} according to the transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A}$, the agent then receives an observation $o_t \in \mathcal{Z}$, which depends on the new state of the environment with probability $\mathcal{O}(o_t|s_{t+1}, a_t)$. The agent then receives rewards according to its actions and the state $r : \mathcal{S} \times \mathcal{A} \longmapsto \Re$. As in MDPs, the goal of the agent is to maximize the expected future cumulative discounted rewards $\mathcal{R} = \mathbb{E}\big(\sum_{t=0}^{T} r^t \gamma^t\big)$, where γ is the discount factor.

Deep Q Networks (DQN) Q-Learning is a model-free off-policy Temporal-Difference algorithm [9] that estimates the expected future cumulative discounted rewards of an MDP by an agent that executes an action a_t in state s_t. Q-Learning uses a value function for policy estimation $\boldsymbol{\pi}$ as $Q_{\boldsymbol{\pi}}(s, a) = \mathbb{E}[R|s, a]$. The Q function can be written recursively as $Q_{\boldsymbol{\pi}}(s, a) = \mathbb{E}_{s'}\big[r(s, a) + \gamma \mathbb{E}_{a' \sim \boldsymbol{\pi}}[Q_{\boldsymbol{\pi}}(s', a')]\big]$.

In DQN, the value function is evaluated by a deep neural network.

Double Q-Learning Tabular Q-learning and vanilla DQN implementations tend to over-estimate the Q values, which is known as the overoptimism problem, because it uses the same network to select actions a and to compute value function due to the actions. Thus, a common solution employed for DQN known as Double DQN [11], is to have two separate networks: one *online/current* network selects the actions (i.e. determines the policy), while the other *targets* network computes the value function. The Double DQN algorithm is now de rigueur for DQN implementations. Hence, (Double) DQN learns the action-value function Q^* for the optimal policy by minimizing the following loss:

$$\mathcal{L}(\Theta) = \mathbb{E}_{s,a,r,s'}[(Q^*(s, a|\Theta) - y_{target})^2], \quad \text{where} \quad y_{target} = r + \gamma \max_{a'} \hat{Q}(s', a'|\Theta^-), \quad (1)$$

where $\hat{Q}$ is the target value function whose parameters update the value of the new actions selected by the online network.

Dueling Networks The dueling networks architecture [12] is based on the fact that most deep RL implementations do not use a specialized deep neural network architecture for RL. The dueling network architecture has two streams for the value $V(s)$ and advantage functions $A(s, a)$ to estimate the Q-values $Q(s, a)$. The advantage function gives a relative measure of the importance of each action, and is obtained as follows

$$A(s, a) = Q(s, a) - V(s) \quad (2)$$

Deep Recurrent Q Networks (DRQN) Deep recurrent Q-networks (DRQN) developed in [2] use an LSTM on top of CNNs in order to solve POMDPs for Atari games. DRQN outperformed DQN and human players in Atari benchmarks with partial observability. Essentially, the deep RNN is used to extract complex abstractions as temporal data representations. In particular, the Q function estimates $Q(h_t, a_t)$, where h_t is hidden state determined by the RNN, as $h_t = RNN(h_{t-1}, o_t)$. The

RNN can be any recurrent variant, in this paper we choose LSTM in order to match the benchmarks of previous works [2].

$$\mathcal{L}(\Theta) = \mathbb{E}_{<o,h,a,r,o',h'>}[(Q^*(h_t, a|\Theta) - y_{target})^2], \quad \text{where} \quad y_{target} = r + \gamma \max_{a'} \hat{Q}(h_{t+1}, a'|\Theta^-), \tag{3}$$

Visual Attention A attention mechanisms, such as was presented in [5] for policy gradient methods, and in [8] for DRQN, are another way to solve the POMDPs.

Experience Replay The DQN implementations of [6, 5] and also DRQN of [2] use experience replay in order to make the agent reuse experiences from past transitions. This helps to solve the problem of having strongly correlated successive transitions by replaying experiences from a buffer and also helps the agent replay transitions that might otherwise be forgotten. With experience replay, the transitions are uniformly randomly sampled from the replay buffer. The problem with this sampling approach is that important and less important transitions have an equal chance of survival.

Prioritized Experience Replay Prioritized experience replay developed in [7], uses the difference in Temporal-Difference (TD) error and importance sampling to prioritized the order in which experiences are replayed. The work in [7] outperformed previous DQN implementations in almost all the games in the Atari benchmark. This work seeks to implement prioritized experience replay for a DRQN using a different measure for prioritization than TD, along with a variant of importance sampling.

4 Methods

4.1 Deep Recurrent Q-Agent for POMDPs

The architecture for the DRQN used in this project is shown in Fig. 2. Essentially, the architecture of [2] is taken as a starting point, then the reinforcement learning algorithm is modified to consist of: (1) double DQN, (2) multi-step bootstrap targets, (3) prioritized experience replay, all within a (4) dueling network architecture. An attention mechanism that uses all the aforementioned network enhancement except recurrency, is used for comparison.

4.2 Prioritized Experience Replay with Importance Sampling

The key contribution in this project is developing a prioritized experience replay (PER) method that performs better than the uniform experience replay used for POMDPs in [2]. Moreover, the PER method used in [7] and even in its recent multi-agent distributed extension [3] use temporal difference (TD) error to prioritized the experiences. However, TD error is not a good measure when there is inherent partial observability and it ignores stochasticity in transitions and/or rewards. Thus, the proposed measure is to obtain the importance weights by using both the advantage function $A(s, a)$ and the TD-error δ_t, and combining both signals in a energy spectral density fashion giving the ρ measure as

$$\rho_t = \Big[\frac{1}{2}(\delta_t + A(s, a))\Big]^2 \tag{4}$$

The intuition behind the proposed IS weight measure ρ is that combining the advantage function with the TD-error offers for redundancy, for when one or the other might be missed and therefore not replayed, and replayed more often if they have similar magnitude at the same time step. If the TD-error measures 'how surprising' [7] a transition is, and the advantage function measures the importance of each action, then the advantage function by itself would seem to give a better measure for POMDPs.

Moreover, the advantage function and TD-error can be used interchangeably in the gradient estimation of policy gradient methods $g = \mathbb{E}[\sum_{t=0}^{\infty} \Psi_t \nabla_\theta \log \boldsymbol{\pi}_\theta(a_t|s_t)]$, where Ψ_t can be, among other functions, the Q function, the advantage function, or the TD-error [3]. Yet, these functions are not identical but the advantage function offers the least variance [10, 3].

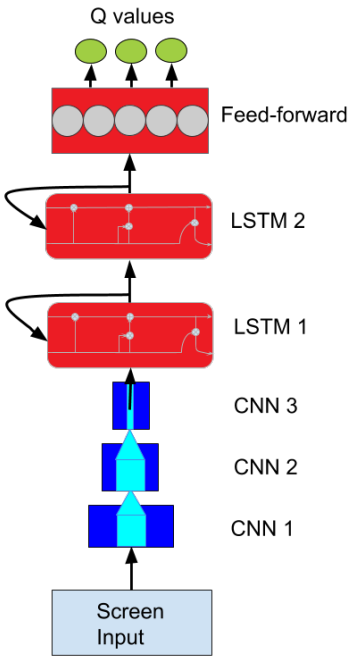

Figure 2: Architecture for DRQN 'OBSQR' network

4.3 The Agent: Advantage-Prioritized Experience Deep Recurrent Q-Network

The developed reinforcement learning algorithm for this project uses the learning architecture of Section 4.1 with the experience replay approach of Section 4.2. Together, this approach is referred to as OBSQR.

5 Experiments

5.1 Environments: Atari Games

The OpenAI [1] is a framework for reinforcement learning research which provides different environments that provide the 'world' abstraction for the RL problem, thus allowing researchers to focus on developing different new agents. OpenAI also provides a way for the community to benchmark problems, thus allowing to measure the gains of one's research. Chief among these environments is Arcade Learning Environment (ALE), which has a library of Atari 2600 games.

While working on this project, an interesting fact came up about the degree of partial observability of the Atari games. It was found that out of all 49 games in the Atari suite, the one that presented the lowest score with DQN [6] was `Frostbite`—the scores with DQN are almost insignificant. Indeed, the game which had the highest improvement with the DRQN [2] was `Frostbite`. Moreover, the DQN-PER versions [7, 12] showed improvement over uniform-replay DQN's (all these are *non-recurrent*), hence motivating the examination of this game further. In [2] it became necessary to artificially flicker the screen of several games in an attempt to turn them to an POMDP because they didn't have the benefit of hindsight to choose `Frostbite`. In fact, in [7], a discussion regarding the measure for importance weight prioritization, led to conclude that possibly the reason that TD-error performed well was that most games were nearly-deterministic. In addition to finding one interesting case where to test my proposed method, there were also computing expense considerations to keep the number of test cases at a minimum. However, for future benchmarking of POMDPs, it will be necessary to choose a different benchmark where there are more POMDP environments.

5.2 The Agent: OBSQR

The developed agent OBSQR is tested in the Atari `Frostbite` environment. The experimental set-up is as follows: The environment provides 10 frames which are processes by the double recurrent Q-network, with the LSTM layer on top of the DQN convolutional layers, processing the higher-level abstractions of the input. The training was done to match the parameters from [2]. The policy and its evaluation are determined through the online and target networks, respectively. The maximum number of episodes is 20k and the replay memory has a capacity for 10^6 samples. The advantage values are obtained from the dueling network, while the TD-error values are obtained from the online network. The policy converges at around 7k episodes. The prioritized replay memory was implemented using the proportional SumTree structure described in [7].

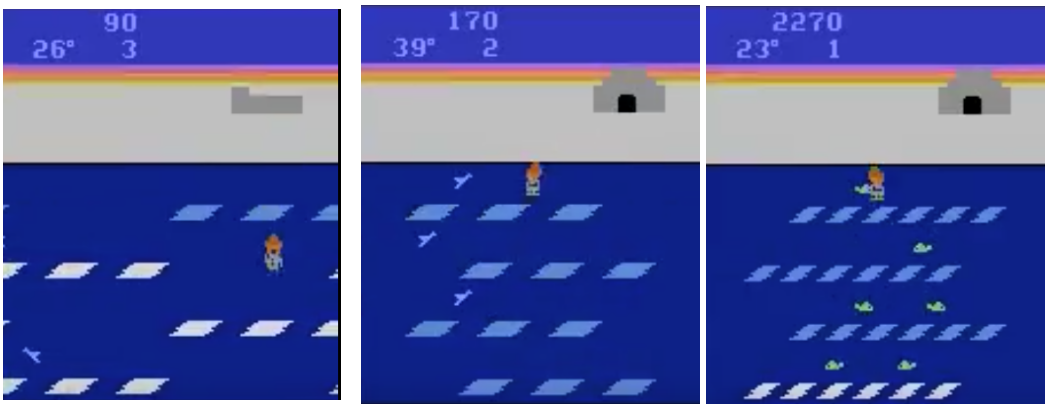

Figure 3: Game renderings at various stages. Left: First stage, mostly random. Center: Second stage, learns to jump and not drown. Right: Final stage, learns to get inside the igloo. DQN never discovers final stage.

5.3 Results

Results for the average reward per episode using OBSQR are shown in this section and compared with other methods. The results show that OBSQR outperforms not only DQN as expected but also DRQN [2] and the PER-DQN of [7].

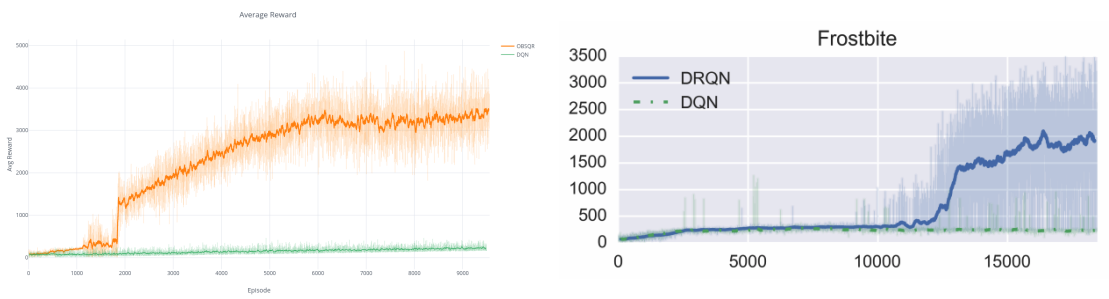

Figure 4: Final Results. Left: OBSQR converges at around 7k episodes for Frostbite for a maximum average reward of 3451. Right: DRQN from [2] with uniform experience replay converges at around 12k episodes to a lower reward value of 2875

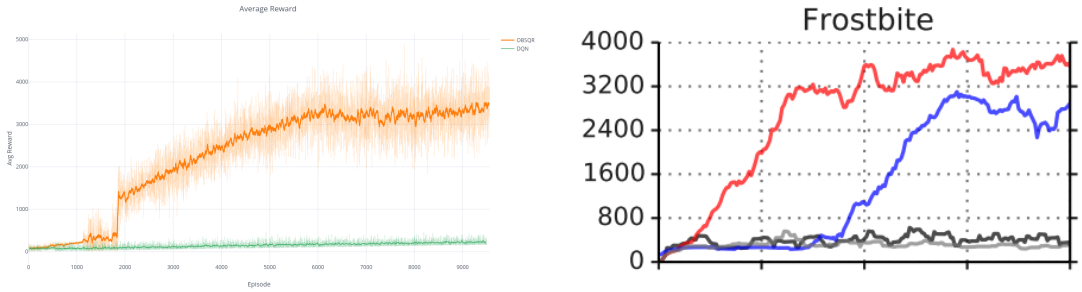

Figure 5: Final Results. Left: OBSQR converges faster than TD-error PER approach from [7]. TD-error PER approach converges to about the same value but at a lower rate—blue and red lines are for rank-based and proporsional based-PER, respectively. (each tickmark is 25 hrs, not episodes. OBSQR reaches convergence in about 20 hrs vs. 50 hrs for [7])

6 Conclusions and Future Work

This project showed the great benefit of using importance sampling in deep reinforcement learning (Prioritized Experience Replay). It allows for better policies and faster convergence, than with uniform replay memory. Moreover, it shows that for POMDPs, using the combination of the Advantage function and TD-error helps the agent find the optimal policy faster. Future work should include the applicability to other POMDP environments and a more specialized network architecture.

References

[1] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[2] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. *CoRR, abs/1507.06527*, 2015.

[3] Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado van Hasselt, and David Silver. Distributed prioritized experience replay. *arXiv preprint arXiv:1803.00933*, 2018.

[4] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pages 6382–6393, 2017.

[5] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212, 2014.

[6] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

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

[8] Ivan Sorokin, Alexey Seleznev, Mikhail Pavlov, Aleksandr Fedorov, and Anastasiia Ignateva. Deep attention recurrent q-network. *arXiv preprint arXiv:1512.01693*, 2015.

[9] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[10] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[11] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI*, volume 16, pages 2094–2100, 2016.

[12] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.