Deep Learning Papers Review: Striving for Simplicity in Off-policy Deep Reinforcement Learning

Seungwon Kim
Incheon International Airport Corporation
Georgia Institute of Technology
skim3222@gatech.edu

1 Background

Off-policy Q-learning allows two different policies: behavior policy and target policy. As in figure 1, behavior policy selects an action, whereas the target policy is used to estimate the next state-action value. On the other hand, on-policy learning uses the same policy for choosing the action to execute and estimating the next state-action value. Additionally, the term off-policy can be used to indicate pure exploratory random behavior, which is literally off from the current policy, whereas on-policy can suggest following current policy, which means acting greedily or choosing the action using epsilon-greedy strategy.

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state},\cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$
    $S \leftarrow S'$;
  until S is terminal

Figure 1: Off-policy Q-learning

Deep Q-learning (DQN) algorithm is labeled as off-policy RL algorithm but it is considered as "online" rather than "offline" since it exploits near on-policy behavior such as epsilon-greedy with an experience replay. Training DQN in offline setting, i.e. batch setting, which means using data that have been already logged, doesn't work well due to the extrapolation error.

For episode = 1, M **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For t = 1,T **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $\left(\phi_t, a_t, r_t, \phi_{t+1}\right)$ in D
    Sample random minibatch of transitions $\left(\phi_j, a_j, r_j, \phi_{j+1}\right)$ from D
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \bar{Q}\left(\phi_{j+1}, a'; \theta^-\right) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$ with respect to the network parameters θ
    Every C steps reset $\bar{Q} = Q$
  End For
End For

Figure 2: DQN in online setting

Off-policy RL algorithms such as Q-learning are more practical and efficient to handle real-world problems with already logged data than on-policy algorithms such as policy gradient since in principle, off-policy algorithm can learn from any data collected by any policy. But off-policy RL algorithms are unstable and there is no guarantee that it converges especially when using neural networks for function approximation.

To make stable off-policy DQN, here have been various advances (see figure 3) such as distributional RL: C51 or quantile regression DQN (QR-DQN). Both C51 and QR-DQN obatin state-of-the-art performance on Atrai 2600 games by estimating the distribution of return using support with assigned probabilities rather than estimating expected return.

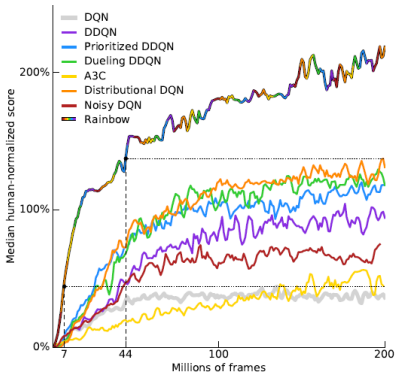

Figure 3: Rainbow DQN

2 Main point, Summary of this paper

This paper investigates training DQN with the batch setting, i.e. " Is it possible to make successful agents using completely on offline logged data?" It presents the performance of distributional RL, nature DQN(2015 Mnih et al) on Atrai 2600 games in both batch setting(offline setting) and online setting. Surprisingly, offline QR-DQN outperforms online C51 and online DQN, which is contrary to recent work.

This paper also tries to make RL algorithms as simple as possible. In spite of recent advances in DQN, those might lead to complex hypothesis and it is more likely to coincidentally fits data than simple hypothesis does (Occam's razor). It is important to ask " Are those advances really necessary? " The paper proposes Ensemble DQN and Random Ensemble Mixture (REM) DQN, which are way simpler than distributional RL, and it shows that REM DQN outperforms C51 and QR-DQN in the offline setting and online REM DQN performs comparably with online QR-DQN.

3 Proposed Architecture

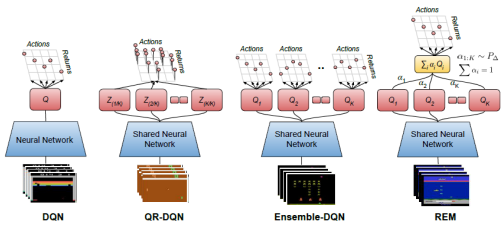

Figure 4: Architecutre

DQN, QR-DQN, Ensemble-DQN, and REM DQN use the same network architecture but DQN outputs state-action value Q(s,a) and QR-DQN outputs K *|A| number of Z to represent the distribution

of return rather than just estimating the expected value of the return. Ensemble-DQN is a simple extension of DQN, which uses an ensemble of Q(s,a) (see below). Each random Q-head with its target is used to compute Bellman error respectively as shown in below loss function.

REM-DQN uses a convex combination of Q values to estimate Q-values (see below). Each coefficient alpha is randomly drawn from a categorical distribution.

4 Experiments

Experiments are performed with the same number of Q-heads at 200 and the same value of other parameters. Experience replay consists of 50 million tuples of (observation, action, reward, next observation), which have been logged by Nature DQN(Mnih et al 2015). Offline QR-DQN outperforms online C51 and online DQN. REM DQN outperforms C51 and QR-DQN in the offline setting and online REM DQN performs comparably with online QR-DQN.

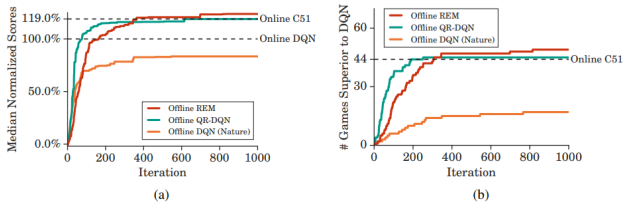

Figure 1: (a) Median normalized online evaluation scores averaged over 5 runs across 60 Atari 2600 games of offline agents trained using the DQN replay dataset (b) Number of games where an offline agent achieves a higher score than fully trained online DQN (Nature) as a function of training time. The online agents are trained for 200 iterations where each iteration corresponds to 1 million game frames.

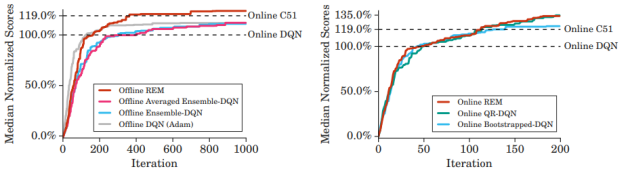

Figure 4: Normalized scores averaged over 5 runs across 60 Atari 2600 games of (a) offline agents trained using the DQN replay dataset, and (b) agents trained online for 200 million game frames.

Table 1: Median normalized best evaluation scores (averaged over 5 runs) across 60 Atari 2600 games, measured as percentages and number of games where an agent achieves better scores than a fully trained online DQN (Nature) agent. The offline RL agents are trained using the DQN replay dataset for five times as many gradient updates as the online DQN agent. The online RL agents are trained for 200 million frames (standard protocol).

Offline agent	Median	> DQN
DQN (Adam)	111.9%	41
Ensemble-DQN	111.0%	39
Averaged Ensemble-DQN	112.1%	43
QR-DQN	118.9%	45
REM	**123.8%**	**49**

Online agent	Median	> DQN
C51	119.0%	44
Bootstrapped-DQN	122.8%	**53**
QR-DQN	**134.8%**	52
REM	134.0%	52

5 Contribution

This paper shows that it is possible to successfully train DQN in batch setting, which uses a fixed dataset without interacting with environments. Using a common logged dataset can improve the reproducibility of off-policy RL algorithms, accelerating research by serving a testbed similar to Imagenet dataset in computer vision.

References

[1] Amy Greenwald, Keith Hall, and Roberto Serrano. Correlated q-learning. In *ICML*, volume 3, pages 242–249, 2003.