Averaged-DQN: Variance Reduction and Stablilization for Deep Reinforcement Learning

YIN-JUI, HSU
Institute of Network Engineering
National Chiao Tung University
z8872453@yahoo.com.tw

1 Introduction

The combination of non-linear function approximation and RL was considered unstable and was shown to diverge even in simple domains. The recent **Deep Q-Network (DQN)** algorithm was the first to successfully combine a powerful non-linear function approximation technique known as Deep Neural Network (DNN) together with the Q-learning algorithm. DQN presented a remarkably flexible and stable algorithm, showing success in the majority of games within the **Arcade Learning Environment (ALE)**.

But DQN still suffers from the issues that arise from the combination of Q-learning and function approximation. One of these issues is called *overestimation phenomena*. This paper suggests **Averaged-DQN**, based on averaging the K previously learned Q-values estimates, to reduce the target approximation error variance which leads to stability and improves results.

Compared to the prior works, the main contribution of this paper as follows:

- A novel extension to the DQN algorithm which stabilizes training, and improves the attained performance, by averaging over previously learned Q-values.
- Variance analysis that explains some of the DQN problems, and how the proposed exension toe the DQN algorithm addresses them.
- Experiments with three ALE games (*BREAKOUT*, *SEAQUEST*, and *ASTERIX*) and *Gridworld problem*, which demonstrating the favorable effect of the proposed scheme.

2 Problem Formulation

The Reinforcement Learning Framework operates as figure below:

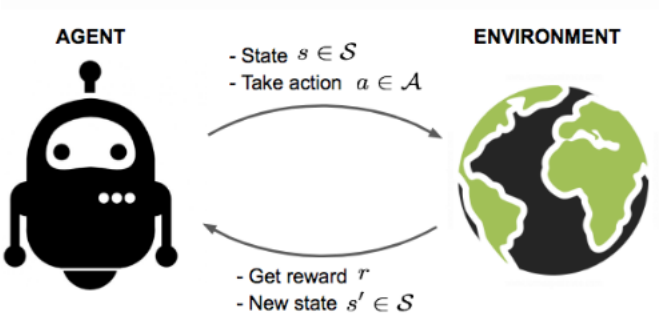

Figure 1: An agent is faced with a sequential decision making problem, where interaction with the environment takes place at discrete time steps ($t = 0, 1, ...$). At time t the agent observes state $s_t \in S$, selects an action $a_t \in A$, which results in a scalar reward $r_t \in R$, and a transition to a next state $s_{t+1} \in S$.

Preprint. Under review.

The state-action value function based on *Bellman Equation* is as follows, where $\gamma \in [0, 1]$ is the discount factor:

$$\begin{aligned} Q^{\pi}(s, a) &= \mathbb{E}[\, r_0 + \lambda Q^{\pi}(s^{'}, a^{'}) | s_0 = s, a_0 = a] \\ &= \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a] \end{aligned} \tag{1}$$

The optimal state-action value function is as follows:

$$\begin{aligned} Q^{*}(s, a) &= \max_{\pi} Q^{\pi}(s, a) \\ &= \mathbb{E}[\, r_0 + \lambda \max_{a^{'}} Q^{*}(s^{'}, a^{'}) | s_0 = s, a_0 = a] \end{aligned} \tag{2}$$

The state-action value function in Q-learning performs updates using the following update rule, where α is the learning rate:

$$Q(s, a) = Q(s, a) + \alpha(r + \gamma \max_{a^{'}} Q(s^{'}, a^{'}) - Q(s, a)) \tag{3}$$

The initial value of DQN target $r + \gamma \max_{a'} Q(s^{'}, a^{'})$ may be wrong, but after repeated iterations, the DQN target will become more and more accurate, so it is conceivable that if it is long enough, the state-action value function can be optimized to the optimal state-action value function, this is what Q-learning does.

The algorithm of Deep Q Networks(DQN) is as follows:

Algorithm 1 DQN

Initialize $Q(s, a; \theta)$ with random weights θ_0
Initialize Experience Replay (ER) buffer $\mathcal{B}$
Initialize exploration procedure *Explore(·)*
for $i = 1, 2, \ldots, N$ **do**
  $y^i_{s,a} = \mathbb{E}_{\mathcal{B}}\left[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) | \, s, a\right]$
  $\theta_i \approx \text{argmin}_{\theta} \, \mathbb{E}_{\mathcal{B}}\left[(y^i_{s,a} - Q(s, a; \theta))^2\right]$
  Explore(·), update $\mathcal{B}$
end for
output $Q^{\text{DQN}}(s, a; \theta_N)$

The DQN target $y^i_{s,a}$ is constructed by target network $Q(s, a; \theta_{i-1})$ and the expectation ($\mathbb{E}_{\beta}$) based on the sample districution of experience transitions in the ER buffer $(s, a, r, s^{'}) \sim \beta$. The DQN loss $y^i_{s,a} - Q(s, a; \theta_i)$ is minimized by the Stochastic Gradient Descent (SGD) variant, sampling mini-batches from the ER buffer.

The Δ_i is the error between optimal state-action value function and the state-action value function at iteration i, which can be decomposed as follows, which composed of *Target Approximation Error*, *Overestimation Error*, and *Optimality Difference*:

$$\begin{aligned} \Delta_i &= Q(s, a; \theta_i) - Q^{*}(s, a) \\ &= Q(s, a; \theta_i) - y^i_{s,a} + y^i_{s,a} - \hat{y}^i_{s,a} + \hat{y}^i_{s,a} - Q^{*}(s, a) \end{aligned} \tag{4}$$

Here $\hat{y}^i_{s,a}$ is the true target, and the definition is as follow:

$$\hat{y}^i_{s,a} = \mathbb{E}_{\beta}[\, r + \gamma \max_{a^{'}} y^{i-1}_{(} s^{'}, a^{'}) | s, a] \tag{5}$$

The definition of Target Approximation Error (TAE) $Z^i_{s,a}$ is as follows:

$$Z^i_{s,a} = Q(s,a;\theta_i) - y^i_{s,a} \tag{6}$$

The TAE is a result of several factors:

- The sub-optimal of θ_i due to inexact minimization.
- The limited representation power of a neural network (model error).
- The generalization error for unseen state-action pairs due to the finite size of the ER buffer.

In this paper, we hypothesize that the variability in DQN's performance is related to deviating form a steady-state policy induced by the TAE.

The definition of Overestimation Error $R^i_{s,a}$ is as follows:

$$R^i_{s,a} = y^i_{s,a} - \hat{y}^i_{s,a} \tag{7}$$

In this paper, we considered the TAE $Z^i_{s,a}$ as a random variable uniformly distributed in the interval $[-\epsilon, \epsilon]$. Due to the *max* operator in the DQN target $y^i_{s,a}$, the value of expected overestimation errors $\mathbb{E}_Z[R^i_{s,a}]$ are upper bounded by $\gamma\epsilon\frac{n-1}{n+1}$, where n is the number of applicable actions in state s. The derivation of $\mathbb{E}_Z[R^i_{s,a}]$ is as follows:

$$\begin{aligned} \mathbb{E}_Z[R^i_{s,a}] &= \mathbb{E}_Z[\max_{a'}[Z^{i-1}_{s',a'}]] \\ &= \gamma\epsilon\frac{n-1}{n+1} \end{aligned} \tag{8}$$

The overestimation error bias is uneven and is bigger in the following conditions:

- The states where the Q-values are similar for the different actions.
- The states which are the start of a long trajectory.

Due to the above mentioned overestimation error upper bound, the magnitude of the overestimation error bias is controlled by the variance of the TAE.

In this paper, we hypothesize the main cause for overestimation in DQN is the TAE variance.

The algorithm of Ensemble-DQN is as follows:

Algorithm 2 Ensemble DQN

Initialize K Q-networks $Q(s,a;\theta^k)$ with random weights θ^k_0 for $k \in \{1,\dots,K\}$
Initialize Experience Replay (ER) buffer $\mathcal{B}$
Initialize exploration procedure *Explore(·)*
for $i = 1, 2, \dots, N$ **do**
  $Q^E_{i-1}(s,a) = \frac{1}{K}\sum_{k=1}^{K} Q(s,a;\theta^k_{i-1})$
  $y^i_{s,a} = \mathbb{E}_{\mathcal{B}}\left[r + \gamma \max_{a'} Q^E_{i-1}(s',a'))|\, s,a\right]$
  for $k = 1, 2, \dots, K$ **do**
    $\theta^k_i \approx \text{argmin}_\theta\ \mathbb{E}_{\mathcal{B}}\left[(y^i_{s,a} - Q(s,a;\theta))^2\right]$
  end for
  Explore(·), update $\mathcal{B}$
end for
output $Q^E_N(s,a) = \frac{1}{K}\sum_{k=1}^{K} Q(s,a;\theta^k_i)$

Ensemble-DQN is a straightforward way to solve *K* DQN losses in parallel, then averages over the resulted Q-values estimates.

The algorithm of Averaged-DQN is as follows:

Algorithm 3 Averaged DQN

Initialize $Q(s,a;\theta)$ with random weights θ_0
Initialize Experience Replay (ER) buffer $\mathcal{B}$
Initialize exploration procedure *Explore*$(\cdot)$
for $i = 1, 2, \ldots, N$ **do**
  $Q^A_{i-1}(s,a) = \frac{1}{K}\sum_{k=1}^{K} Q(s,a;\theta_{i-k})$
  $y^i_{s,a} = \mathbb{E}_{\mathcal{B}}\left[r + \gamma \max_{a'} Q^A_{i-1}(s',a') | \, s,a\right]$
  $\theta_i \approx \text{argmin}_\theta \; \mathbb{E}_{\mathcal{B}}\left[(y^i_{s,a} - Q(s,a;\theta))^2\right]$
  Explore$(\cdot)$, update $\mathcal{B}$
end for
output $Q^A_N(s,a) = \frac{1}{K}\sum_{k=0}^{K-1} Q(s,a;\theta_{N-k})$

Averaged-DQN uses the *K* previously learned Q-values estimates to produce the current action-value estimate. The variance of TAE is reduced by Average-DQN, to stabilize the training process. The computational effort compared to DQN is, *K*-fold more forward passes through a Q-network while minimizing the DQN loss. The number of back-propagation updates remains the same as in DQN. The output of the algorithm is the average over the last *K* previously learned Q-networks.

The definition of **Parseval's Theorem** is as follows:
suppose f and g are two periodic continuous function with period 2π, and:

$$\begin{aligned} f(x) &\sim \sum_{-\infty}^{\infty} c_n e^{inx} \\ g(x) &\sim \sum_{-\infty}^{\infty} \gamma_n e^{inx} \end{aligned} \tag{9}$$

then:

$$\frac{1}{2\pi}\int_{-\pi}^{\pi} f(x) g^*(x) dx = \sum_{-\infty}^{\infty} c_n \gamma_n^* \tag{10}$$

3 Theoretical Analysis

To focus on the TAE, we eliminate the overestimation error by considering a fixed policy for updating the target values and a zero reward $r = 0$ everywhere.

Denote by Q_i the vector of value estimates at iteration i where the fixed action a is suppressed, and by Z_i the vector of corresponding TAEs. Then we can get the following equation, where $P \in \mathbb{R}_+^{S\times S}$ is the transition probabilities matrix for the given policy:

$$\begin{aligned} Q_i &= Z_i + y_i \\ &= Z_i + \gamma P \frac{1}{K}\sum_{k=0}^{K} Q_{i-k} \end{aligned} \tag{11}$$

To obtain an explicit comparison, this paper further specializes the model to an *M*-state unidirectional MDP as figure below:

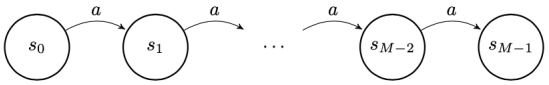

Figure 2: *M* states unidirectional MDP. A zero reward is obtained in any state.

Employing DQN on above mentioned MDP model, we can get the following derivation for $i > M$, where the last equality we have used the fact $y^j_{s_{M-1},a} = 0$ for all times j (since the s_{M-1} is the terminal state):

$$\begin{aligned} Q^{DQN}(s_0, a; \theta_i) &= Z^i_{s_0,a} + y^i_{s_0,a} \\ &= Z^i_{s_0,a} + \gamma Q^{DQN}(s_1, a; \theta_{i-1}) \\ &= Z^i_{s_0,a} + \gamma[\, Z^{i-1}_{s_1,a} + y^{i-1}_{s_1,a}] \\ &= Z^i_{s_0,a} + \gamma Z^{i-1}_{s_1,a} + ... + \gamma^{M-1} Z^{i-(M-1)}_{s_{M-1},a} \end{aligned} \tag{12}$$

Then the DQN variance can be derived as follows:

$$\mathrm{Var}[\, Q^{DQN}(s_0, a; \theta_i)] \,=\, \sum_{m=0}^{M-1} \gamma^{2m} \sigma^2_{s_m} \tag{13}$$

The TAE will be accumulated over the past DQN iteraitons on the update trajectory. Accumulation of TAE errors results in bigger variance with its associated adverse effect, which will induce the variability in DQN's performance.

Employing Ensemble-DQN on above mentioned MDP model, we can get the following derivation for $i > M$, where $\mathbb{E}[\, Z^{k,i}_{s,a}] \,= 0$, $\mathrm{Var}[\, Z^{k,i}_{s,a}] \,= \sigma^2_s$, for all $i \neq j$: $\mathrm{Cov}[\, Z^{k,i}_{s,a}, Z^{k',j}_{s',a}] \,= 0$, for all $k \neq k'$: $\mathrm{Cov}[\, Z^{k,i}_{s,a}, Z^{k',j}_{s',a}] \,= 0$, and the last equality we have used the fact $y^j_{s_{M-1},a} = 0$ for all times j (since the s_{M-1} is the terminal state):

$$\begin{aligned} Q^E_i(s_0, a) &= \frac{1}{K} \sum_{k=1}^{K} Q(s_0, a; \theta^k_i) \\ &= \frac{1}{K} \sum_{k=1}^{K} [\, Z^{k,i}_{s_0,a} + y^i_{s_0,a}] \\ &= \frac{1}{K} \sum_{k=1}^{K} [\, Z^{k,i}_{s_0,a}] \,+ y^i_{s_0,a} \\ &= \frac{1}{K} \sum_{k=1}^{K} [\, Z^{k,i}_{s_0,a}] \,+ \gamma Q^E_{i-1}(s_1, a) \\ &= \frac{1}{K} \sum_{k=1}^{K} [\, Z^{k,i}_{s_0,a}] \,+ \frac{\gamma}{K} \sum_{k=1}^{K} [\, Z^{k,i-1}_{s_1,a}] \,+ \gamma y^{i-1}_{s_2,a} \\ &= \sum_{m=0}^{M-1} \gamma^m \frac{1}{K} \sum_{k=1}^{K} Z^{k,i-m}_{s_m,a} \end{aligned} \tag{14}$$

Since the TAEs are uncorrelated by the assumption, the Ensemble-DQN variance then can be derived as follows:

$$\begin{aligned} \mathrm{Var}[\, Q^E_i(s_0, a)] \,&= \sum_{m=0}^{M-1} \frac{1}{K} \gamma^{2m} \sigma^2_{s_m} \\ &= \frac{1}{K} \mathrm{Var}[\, Q^{DQN}(s_0, a; \theta_i)] \end{aligned} \tag{15}$$

The Ensemble-DQN is a straightforward way to obtain a $\frac{1}{K}$ variance reduction, with a computational effort of K-fold learning problems, corresponding to DQN.

Employing Averaged-DQN on above mentioned MDP model, we can get the following derivation for $i > KM$, where $\mathbb{E}[\, Z^i_{s,a}] = 0$, $\mathrm{Var}[\, Z^i_{s,a}] = \sigma^2_s$, for all $i \neq j$: $\mathrm{Cov}[\, Z^i_{s,a}, Z^j_{s',a'}] = 0$, for all $ss \neq s'$: $\mathrm{Cov}[\, Z^i_{s,a}, Z^i_{s',a}] = 0$, and the last equality we have used the fact $y^j_{s_{M-1},a} = 0$ for all times j (since the s_{M-1} is the terminal state):

$$
\begin{aligned}
Q^A_i(s_0, a) &= \frac{1}{K}\sum_{k=1}^{K} Q(s_0, a; \theta_{i+1-k}) \\
&= \frac{1}{K}\sum_{k=1}^{K} [\, Z^{i+1-k}_{s_0,a} + y^{i+1-k}_{s_0,a}] \\
&= \frac{1}{K}\sum_{k=1}^{K} [\, Z^{i+1-k}_{s_0,a}] + \frac{\gamma}{K}\sum_{k=1}^{K} Q^A_{i-k}(s_1, a) \\
&= \frac{1}{K}\sum_{k=1}^{K} [\, Z^{i+1-k}_{s_0,a}] + \frac{\gamma}{K^2}\sum_{k=1}^{K}\sum_{k'=1}^{K} Q(s_1, a; \theta_{i+1-k-k'}) \\
&= \frac{1}{K}\sum_{j_1=1}^{K} Z^{i+1-j_1}_{s_0,a} + \frac{\gamma}{K}\sum_{j_1=1}^{K}\sum_{j_2=1}^{K} Z^{i+1-j_1-j_2 s_0,a}_{s_1,a} + ... + \frac{\gamma^{M-1}}{K^M}\sum_{j_1=1}^{K}\sum_{j_2=1}^{K}...\sum_{j_M=1}^{K} Z^{i+1-j_1-...-j_M}_{s_{M-1},a}
\end{aligned}
\tag{16}
$$

Since the TAEs in different states are uncorrelated, we can calculate the variance of each separately. For $L = 1, 2, ..., M$, the variance can be denoted as follows, where the Z_L, Z_{2L},...,Z_{KL} are independent and identically distributed TAE random variables, with $\mathbb{E}[\, Z_l] = 0$ and $\mathbb{E}[\, (Z_l)^2] = \sigma^2_Z$:

$$
\begin{aligned}
V_L &= \mathrm{Var}[\, \frac{1}{K^L}\sum_{i_1=1}^{K}\sum_{i_2=1}^{K}...\sum_{i_L=1}^{K} Z_{i_1+i_2+...+i_{i_L}}] \\
&= \frac{1}{K^{2L}}\mathbb{E}_Z[\, (\sum_{j=L}^{KL} n_{j,L} Z_j)^2] \\
&= \frac{\sigma^2_Z}{K^{2L}}\sum_{j=L}^{KL} (n_{j,L})^2
\end{aligned}
\tag{17}
$$

where $n_{j,L}$ is the number of times Z_j is counted in the multiple summation, and it can be noted recursively as follows:

$$
n_{j,L} = \sum_{i=1}^{K} n_{j-i,L-1} \tag{18}
$$

Since out final goal of this calculation is bounding the variance reduction coefficient, we turn to calculate the solution in the frequency domain where the bound can be easily obtained.

In order to perform the calculation in the frequency domain, we first introduce the following notation:

$$
u^K_j = \begin{cases} 1, & \text{if } j \in 1, ..., K \\ 0, & \text{otherwise} \end{cases}
$$

Then we can rewrite the recursive formula of $n_{j,L}$ as follow, where $*$ is the discrete convolution:

$$\begin{aligned} n_{j,L} &= \sum_{i=-\infty}^{\infty} n_{j-i,L-1} \cdot u_i^K \\ &= (n_{j-i,L-1} * u^K)_j \\ &= (u^K * u^K \ldots * u^K)_j \end{aligned} \tag{19}$$

In this paper, we denote the **Discrete Fourier Transform (DFT)** of $u^K = (u_n^K)_{n=0}^{N-1}$ as $U = (U_n)_{n=0}^{N-1}$, and using Parseval's Theorem to rewrite the variance as follows, where N is the length of the vectors u and U, and is taken larger enough so that the sum includes all nonzero elements of the convolution:

$$\begin{aligned} V_L &= \frac{\sigma_Z^2}{K^{2L}} \sum_{j=L}^{KL} (n_{j,L})^2 \\ &= \frac{\sigma_Z^2}{K^{2L}} \sum_{n=0}^{N-1} |(u^K * u^K \ldots * u^K)_n|^2 \\ &= \frac{\sigma_Z^2}{K^{2L}} \frac{1}{N} \sum_{n=0}^{N-1} |U_n|^{2L} \end{aligned} \tag{20}$$

In this paper, we denote the $D_{K,m} = \frac{1}{K^{2(m+1)}} \frac{1}{N} \sum_{n=0}^{N-1} |U_n|^{2(m+1)}$, then we can rewrite the variance of Averaged-DQN as follow:

$$\mathrm{Var}[\, Q_i^A(s_0, a)] \, = \sum_{m=0}^{M-1} D_{K,m} \gamma^{2m} \sigma_{s_m}^2 \tag{21}$$

In order to compare Averaged-DQN with Ensemble-DQn and traditional DQN, we bound $D_{K,m}$ with $K > 1$ and $m > 0$, then we can rewrite the $D_{K,m}$ as follows:

$$\begin{aligned} D_{K,m} &= \frac{1}{K^{2(m+1)}} \frac{1}{N} \sum_{n=0}^{N-1} |U_n|^{2(m+1)} \\ &= \frac{1}{N} \sum_{n=0}^{N-1} |\frac{U_n}{K}|^{2(m+1)} \\ &< \frac{1}{N} \sum_{n=0}^{N-1} |\frac{U_n}{K}|^2 \\ &= \frac{1}{K^2} \sum_{n=0}^{N-1} |u_n^K|^2 \\ &= \frac{1}{K} \end{aligned} \tag{22}$$

Then we can use the above result to rewrite the variance of Averaged-DQN as follow:

$$\begin{aligned} \mathrm{Var}[\, Q_i^A(s_0, a)] \, &< \mathrm{Var}[\, Q_i^E(s_0, a)] \\ &= \frac{1}{K} \mathrm{Var}[\, Q^{DQN}(s_0, a; \theta_i)] \end{aligned} \tag{23}$$

The above result shows that Averaged-DQN is theoretically more efficient in TAE variance reduction than Ensemble-DQN, and at least K times better than DQN.

4 Conclusion

Averaged-DQN algorithm can stabilizes training and improves performance by efficient TAE variance reduction.

Because the recently-learned state-action value estimates are likely to be better than older ones, therefore we have also considered a **recency-weighted average**. The recently-learned state-action value estimate has the larger weight, and the older state-action value estimate has the smaller weight. Using these weights to produce the current state-action value estimate.

Since the Averaged-DQN is a simple extension of DQN, it can be easily integrated with other DQN variants.

Since Averaged-DQN has variance reduction effect in the learning curve, a more systematic comparison between the different variants of DQN can be facilitated.

References

- Markov Decision Process(MDP)
- Deep Q-Learning
- Parseval's Theorem