A Note on Averaged-DQN: Variance Reduction and Stabilization for Deep Reinforcement Learning

Lin-Jyun-LI
Department of Computer Science
National Chiao Tung University
dominatorjohnny.cs08g@nctu.edu.tw

1 Introduction

In Reinforcement learning(RL) an agent seek an optimal policy for a sequential decision making problem, the optimal policy, is to maximize expected discounted cumulative reward $R_t = \sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'}$, where $\gamma \in [0, 1]$. There are lots of algorithms to deal with RL problem, and the algorithm can also classify into policy optimization(eg.A2C(Mnih et al. [2016]),DDPG(Lillicrap et al. [2015]) and value-based(eg.Sarsa(Rummery and Niranjan [1994],Q-learning(Watkins and Dayan [1992],Double Q-learning(Hasselt [2010]),DQN(Mnih et al. [2013]).

Deep Q-Network(DQN) was the first successfully combine a power-ful non-linear VFA technique known as DNN together with Q-learning algorithm. DQN increasing training stability and convergence issue by using 2 network(evaluation network and target network), and use a Experience Replay buffer(ER)(Lin [1993]) to deal with off-policy exploration. Also, there are some other paper to additional modifications and extensions to the basic algorithm further increased training stability. Schaul et al. [2015] suggested sophisticated ER sampling strategy.

- The main research challenges tackled by the paper : However, there is a issues that arise from the combination of Q-learning and function approximation,which called "overestimation phenomena"(Thrun and Schwartz [1993a]). This phenomena happen because in the update(we will explain each term later in section 2):

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q\left(s', a'\right) - Q(s,a) \right] \tag{1}$$

  maximization bias can occur under greedy policy w.r.t the estimated Q function. However,van Hasselt et al. [2015] suggested a Double-DQN method that uses the idea behind Double Q-learning algorithm, also showed that Q-learning overestimation do occur in practice. Which change the update target value from

$$Y_t^{\mathrm{DQN}} \equiv R_{t+1} + \gamma \max Q\left(s_{t+1}, a; \boldsymbol{\theta}_t'\right) \tag{2}$$

  to

$$Y_t^{\text{Double } Q} \equiv R_{t+1} + \gamma Q\left(S_{t+1}, \operatorname{argmax} Q\left(s_{t+1}, a; \boldsymbol{\theta}_t\right); \boldsymbol{\theta}_t'\right) \tag{3}$$

  (θ_t' is target network, θ_t is the evaluation network)
- The main research challenges tackled by the paper: This paper suggests a different solution to the overestimation phenomena, named Average-DQN, based on averaging previously learned Q-valued estimates.
- The main contributions of the paper (compared to the prior works): The main contributions of this paper as follows:
  - Average-DQN:A novel and simple extension to the DQN algorithm which stabilize training, and improves the attained performance, by averaging over previously learned Q values.

Preprint. Under review.

- Have shown both in theory and practice that Average-DQN can deal with overestimate problem and also shown the major problem in DQN and how to addresses them.
- Experiments with several ALE games demonstrating the favorable effect of the proposed scheme

- Your personal perspective on the proposed method: I think the proposed method is easy to implement and modify from DQN algorithm, and also has better solution compare to Dobule-DQN, also has a good potential to apply (average method from previous learned Q-valued estimates)on the other on-policy method to stabilize these algorithm.

2 Problem Formulation

2.1 Reinforcement learning and Q-learning

In this section , we first identify the RL problem and Q-learning update. In RL, there is a environment which agent take a action, and interact with the environment take place at discrete time steps($t = 0, 1, \ldots$) to get the next observe state. That is , At time t the agent observes state $s_t \in S$ and select a action $a_t \in A$, and get the reward $r_t \in \mathbb{R}$,and transition to next state $s_{t+1} \in S$. We consider infinite horizon problems with a discounted cumulative reward $R_t = \sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'}$from time t with discount factor $\gamma \in [0, 1]$.The goal of the agent is to find the optimal policy $\pi : S \rightarrow A$ that maximize the expected discounted cumulative reward. Next we define the value function:

$$Q^{\pi}(s, a) = \mathbb{E}^{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a\right] \tag{4}$$

Our goal is to find the optimal value function denoted as $Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a)$,with optimal policy π^*, which can obtain by $\pi^*(s) \in \operatorname{argmax}_a Q^*(s, a)$ Recall from section 1, the equation (1)we have a Q-learning update rule:

$$Q(s, a) \leftarrow Q(s, a) + \alpha\left[r + \gamma \max_{a'} Q\left(s', a'\right) - Q(s, a)\right] \tag{1}$$

where s' is the next state after applying acion a in the state s, r is the immediate reward observed for action a at state s, γ is the discount factor, and α is the learning rate. However, if the state is large, we need to maintain a large look-up table with all possible state-action pairs values in memory which is impractical. Thus a common way to do is to use function approximation parameterized by θ, such that $Q(s, a) \approx Q(s, a; \theta)$

2.2 DQN algorithm

DQN algorithm combine the idea of the Q-learning and function approximation with new concept "target network and Experience Replay buffer(ER buffer)" to stablize the learning . we capture the algorithm from DQN in Algorithm with slightly different(compare with the latest DQN algorithm in Mnih et al. [2015],the target network update every C step instead of the previous iteration parameters of action-value networkθ_{i-1}).

Algorithm 1 DQN

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

the target value $y^i_{s,a}$(line 5) are constructed using a target-network$Q\left(s, a; \theta_{i-1}\right)$(using the previous iteration parameters θ_{i-1})where the expectation $(\mathbb{E}_{\mathcal{B}})$ is taken w.r.t. the sample distribution of

experience transitions in the ER buffer $(s, a, r, s') \sim \mathcal{B}$. The DQN loss(line 6) is minimized using a SGD update, sampling mini-batch from the ER buffer, and the $Explore(\cdot)$ means the $\epsilon - greedy$ strategy to interact with the environment.

2.3 Ensemble DQN

Next we discuss the Ensemble DQN, which is a straightforward way and modify some line of DQN by parallel to reduce the variance.

Algorithm 2 Ensemble DQN

1: Initialize K Q-networks $Q\left(s, a; \theta^k\right)$ with random weights θ_0^k for $k \in \{1, \ldots, K\}$
2: Initialize Experience Replay (ER) buffer $\mathcal{B}$
3: Initialize exploration procedure Explore $(\cdot)$
4: for $i = 1, 2, \ldots, N$ do
5: $\quad Q_{i-1}^E(s, a) = \frac{1}{K} \sum_{k=1}^K Q\left(s, a; \theta_{i-1}^k\right)$
6: $\quad y_{s,a}^i = \mathbb{E}_{\mathcal{B}}\left[r + \gamma \max_{a'} Q_{i-1}^E\left(s', a'\right) | s, a\right]$
7: $\quad$ for $k = 1, 2, \ldots, K$ do
8: $\quad\quad \theta_i^k \approx \operatorname{argmin}_\theta \mathbb{E}_{\mathcal{B}}\left[\left(y_{s,a}^i - Q(s, a; \theta)\right)^2\right]$
9: $\quad$ end for
10: $\quad Explore(.)$, update $\mathcal{B}$
11: end for
output $Q_N^E(s, a) = \frac{1}{K} \sum_{k=1}^K Q\left(s, a; \theta_i^k\right)$

In line 5 of Algorithm 2 , the $Q_i^E\left(s_0, a\right)$ is compute over K Q-network ,and calculate the target value $y_{s,a}^i$, and solves K DQN losses in parallel. However, this scheme is computationally demanding , and need to have well calculate hardware and software in parallel,in next section we talk about Average-DQN, which is better and easy to use.

2.4 Average-DQN

In this section we show Average-DQN algorithm which is well-performance in the experiments and theory.

Algorithm 3 Averaged DQN

1: Initialize $Q(s, a; \theta)$ with random weights θ_0
2: Initialize Experience Replay (ER) buffer $\mathcal{B}$
3: Initialize exploration procedure Explore $(\cdot)$
4: for $i = 1, 2, \ldots, N$ do
5: $\quad Q_{i-1}^A(s, a) = \frac{1}{K} \sum_{k=1}^K Q\left(s, a; \theta_{i-k}\right)$
6: $\quad y_{s,a}^i = \mathbb{E}_{\mathcal{B}}\left[r + \gamma \max_{a'} Q_{i-1}^A\left(s', a'\right) | s, a\right]$
7: $\quad \theta_i \approx \operatorname{argmin}_\theta \mathbb{E}_{\mathcal{B}}\left[\left(y_{s,a}^i - Q(s, a; \theta)\right)^2\right]$
8: $\quad$ Explore $\cdot$, update $\mathcal{B}$
9: end for
output $Q_N^A(s, a) = \frac{1}{K} \sum_{k=0}^{K-1} Q\left(s, a; \theta_{N-k}\right)$

The Averaged-DQN algorithm (Algorithm 3) is an extension of the DQN algorithm. Averaged-DQN uses the K previously learned Q-values estimates to produce the current action-value estimate (line 5),The Averaged-DQN algorithm stabilizes the training process (see Figure 1), by reducing the variance of target approximation error(we will prove this in section 3). The computational effort compared to DQN is, K-fold more forward passes through a Q-network while minimizing the DQN loss (line 7). The number of back-propagation updates (which is the most demanding computational element), remains the same as in DQN(compare to Ensemble DQN algorithm, Ensemble computational cost is high). The output of the algorithm is the average over the last K previously learned Q-networks.

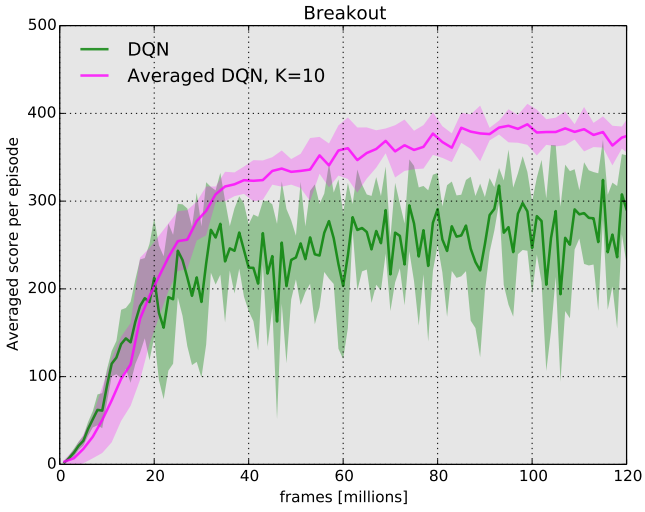

Figure 1: DQN and Averaged-DQN performance in the Atari game of BREAKOUT. The bold lines are averages over seven independent learning trials

3 Theoretical Analysis

In this section , we will discuss the theoretical prove of Average-DQN, which reduce the variance by theory prove. Before analysis the variance of DQN and Average-DQN, we first denote the notation $\Delta_i = Q(s,a;\theta_i) - Q^*(s,a)$, which means the difference between optimal Q-value and approximate Q-value. Also , decompose it as follows:

$$\begin{aligned}\Delta_i &= Q(s,a;\theta_i) - Q^*(s,a) \\ &= \underbrace{Q(s,a;\theta_i) - y^i_{s,a}}_{\text{Target Approximation Error}} + \underbrace{y^i_{s,a} - \hat{y}^i_{s,a}}_{\text{Overestimation Error}} + \underbrace{\hat{y}^i_{s,a} - Q^*(s,a)}_{\text{Optimality Difference}}\end{aligned}$$

where $y^i_{s,a} = \mathbb{E}_{\mathcal{B}}\left[r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) \,|\, s,a\right]$ is the DQN target

and $\hat{y}^i_{s,a} = \mathbb{E}_{\mathcal{B}}\left[r + \gamma \max_{a'} Q\left(y^{i-1}_{s',a'}\right) \,|\, s,a\right]$ is the "true" target.(this notation is different from Thrun and Schwartz [1993b], but the $y^{i-1}_{s,a}$ can consider close to true Q-value because it is expectation over Explore Buffer $\mathcal{B}$,so can apply similar as Thrun's paper ,but still we can see the Optimality difference here in notation) The equation of $\hat{y}^i_{s,a}$ still remain a problem that the author here use the $y^{i-1}_{s',a'}$ here on the right hand side when calculate the Q-value, instead of using $y^i_{s',a'}$, I think using the i is intuition and will not violate the rule in paper, because the y^i's value is calculate by target-network θ_{i-1}.

In the subsection, we will talk about the Target Approximation Error(TAE) and Overestimation Error more detail .

3.1 Target Approximation Error(TAE)

TAE: $Z^i_{s,a} = Q(s,a;\theta_i) - y^i_{s,a}$ TAE is the error in mininizing step ,TAE will be effect by three factor:1.minimization residual 2. the model error 3. generation error for unseen state-action pairs The TAE may cause the policy into sub-optimal policy. For example $y^i_{s,a} = \hat{y}^i_{s,a} = Q^*(s,a)$

$$\begin{aligned}\arg\max_a \left[Q(s,a;\theta_i)\right] &\neq \arg\max_a \left[Q(s,a;\theta_i) - Z^i_{s,a}\right] \\ &= \arg\max_a \left[y^i_{s,a}\right] \\ &= \arg\max_a Q^*(s,a) \qquad \text{(which lead to the left hand side sub-optimal policy)}\end{aligned} \tag{5}$$

3.2 Overestimation Error

The work in Thrun and Schwartz [1993b] has analysis the overestimate error, now we give a clear prove of this

Lemma 1 *The expect overestimate error* $\mathbb{E}_z[R^i_{s,a}] = \mathbb{E}[y^i_{s,a} - \hat{y}^i_{s,a}]$ *are upper bounded by* $\gamma\epsilon\frac{n-1}{n+1}$

Where n is the number of applicable action in state s, and the upper bounded happen when all Q values are equal no matter chose which action.

Proof of Lemma 1 We consider TAE $Z^i_{s,a}$ as a random variable uniformly distributed in the interval $[-\epsilon, \epsilon]$, then the overestimate error can calculate by following.

$$\begin{aligned}\mathbb{E}_z[R^i_{s,a}] &= \mathbb{E}_z[y^i_{s,a} - \hat{y}^i_{s,a}] \\ &= \mathbb{E}_z[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) - r - \gamma \max_{a'}(y^{i-1}_{s',a'})] \\ &= \gamma\mathbb{E}_z[\max_{a'}[Z^{i-1}_{s',a'}]]\end{aligned} \tag{6}$$

let f(x) denote the density of variable $Z^{i-1}_{s',a'}$,that is $f(x) \overset{\text{def}}{=} \text{Prob}\left[Z^{i-1}_{s',a'} = x\right] = \frac{1}{2\epsilon}$ Then

$$\begin{aligned}\gamma\mathbb{E}_z[\max_{a'}[Z^{i-1}_{s',a'}]] &= \gamma\int_{-\infty}^{\infty} xn \underbrace{f(x)}_{\text{Prob}[Z=x]} \underbrace{(\int_{-\infty}^{x} f(a)da)^{n-1}}_{\text{Prob}[Z\leq x]} dx \\ &= \gamma n \int_{-\epsilon}^{\epsilon} x\frac{1}{2\epsilon}(\frac{1}{2} + \frac{x}{2\epsilon})^{n-1}dx \\ &= \gamma n \int_0^1 (2\epsilon y - \epsilon)y^{n-1}dy \qquad \text{(by substituting } y = \tfrac{1}{2} + \tfrac{x}{2\epsilon}\text{)} \\ &= \gamma n\epsilon \int_0^1 2y^n - y^{n-1}dy = \gamma n\epsilon(\frac{2}{n+1} - \frac{1}{n}) = \gamma\epsilon\frac{n-1}{n+1} \qquad \text{(prove lemma 1 done)}\end{aligned} \tag{7}$$

The intuition behind lemma1 is, due to the TAE noise, some of the Q-values might be too small, while others might be too large. The max operator, showever, always picks the largest value, making it particularly sensitive to overestimation, and TAE give a positive bias that can cause asymptotically sub-optimal policies.

As we can see, the overestimation error is controll by TAE variance , and the Dobule Q-learning and its DQN implementation(Double-DQN) has solve by change the positive bias into negative one , but may lead to under-estimation(see figure2). The solution in this paper is, reduce the variance of the TAE, that is , Average-DQN scheme.

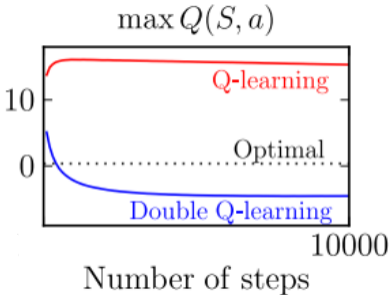

Figure 2: A example in Double Q Learning paper the maximal value in start state, Grid World with 9 states and 4 actions, reward at the terminal state :+5, reward at any non-terminal state:-12 or +10(random)

3.3 TAE Variance Reduction for Average-DQN and Ensemble DQN analysis

To identify the reduce in Average-DQN and Ensemble-DQN , we first discuss the DQN TAE variance. We make the assume similar as Thrun and Schwartz [1993b]. Suppose TAE $Z^i_{s,a}$ is a random process such that $\mathbb{E}[Z^i_{s,a}] = 0$,$\mathrm{Var}[Z^i_{s,a}] = \sigma^2_s$, and for i $\neq$ j : $Cov[Z^i_{s,a}, Z^j_{s\prime,a\prime}] = 0$. Moreover, to focus on the TAE Variance, we consider a fixed policy for updating the target values. Also, We consider reward r = 0 everywhere which has no effect on variance calculation and is easier.

3.3.1 DQN Variance

Consider a Markov Decision Process(MDP) in Figure 3. We use DQN on this MDP model to get lemma 2(by $i > M$)

Lemma 2 *for the MDP in Figure 3 we can get Variance of DQN,* $\mathrm{Var}\left[Q^{\mathrm{DQN}}(s_0, a; \theta_i)\right] = \sum_{m=0}^{M-1} \gamma^{2m} \sigma^2_{s_m}$

Proof of Lemma 2

$$\begin{aligned} Q^{\mathrm{DQN}}(s_0, a; \theta_i) &= Z^i_{s_0,a} + y^i_{s_0,a} && \text{(by definition)} \\ &= Z^i_{s_0,a} + \gamma Q(s_1, a; \theta_{i-1}) && \text{(r=0,thus vanish)} \\ &= Z^i_{s_0,a} + \gamma\left[Z^{i-1}_{s_1,a} + y^{i-1}_{s_1,a}\right] = \cdots = && \text{(expand more iteration)} \\ &= Z^{i_1}_{s_0,a} + \gamma Z^{i-1}_{s_1,a} + \cdots + \gamma^{(M-1)} Z^{i-(M-1)}_{s_{M-1},a} \end{aligned} \tag{8}$$

where the last equality we can used $y^j_{M-1,a} = 0$ because r=0 and no next state(terminal state).Therefore, we get

$$\mathrm{Var}\left[Q^{\mathrm{DQN}}(s_0, a; \theta_i)\right] = \sum_{m=0}^{M-1} \gamma^{2m} \sigma^2_{s_m} (\gamma^{2m} \text{because it is a constant})$$

In the following Average-DQN variance and Ensemble DQN, we will use the DQN baseline to compare.

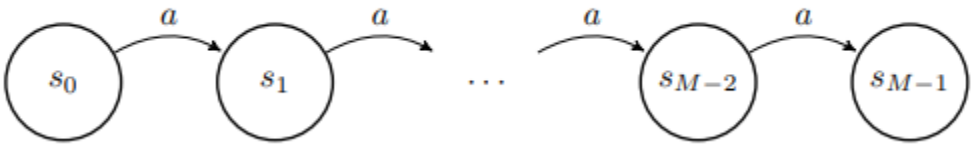

Figure 3: M states unidirectional MDP, The process starts at state s_0, then in each time step moves to the right, until the terminal state s_{M-1} is reached. A zero reward is obtained in any state.

3.3.2 Ensemble DQN Variance

In Algorithm 2 , we employed on Figure 3, we get lemma 3.

Lemma 3 *for $i > M$,assume that:* $\mathbb{E}\left[Z^{k,i}_{s,a}\right] = 0$,$\mathrm{Var}\left[Z^{k,i}_{s,a}\right] = \sigma^2_s$, *for i$\neq$j :*$\mathrm{Cov}\left[Z^{k,i}_{s,a}, Z^{k',j}_{s',a}\right] = 0$,*and for $k \neq k'$:* $\mathrm{Cov}\left[Z^{k,i}_{s,a}, Z^{k',j}_{s',a}\right] = 0$, *then we get*

$$\mathrm{Var}\left[Q^E_i(s_0, a)\right] = \frac{1}{K} \mathrm{Var}\left[Q^{\mathrm{DQN}}(s_0, a; \theta_i)\right] \tag{9}$$

Proof of Lemma 3

$Q_i^E(s_0,a) =$

$= \frac{1}{K}\sum_{k=1}^{K} Q\left(s_0,a;\theta_i^k\right)$

$= \frac{1}{K}\sum_{k=1}^{K}\left[Z_{s_0,a}^{k,i} + y_{s_0,a}^{i}\right]$ (by algorithm 2 line 5)

$= \frac{1}{K}\sum_{k=1}^{K}\left[Z_{s_0,a}^{k,i}\right] + y_{s_0,a}^{i}$ (y is a constant here)

$= \frac{1}{K}\sum_{k=1}^{K}\left[Z_{s_0,a}^{k,i}\right] + \gamma Q_{i-1}^E(s_1,a)$ (Due to the fact in algorithm 2 line6 , and r=0 & max over one action

$= \frac{1}{K}\sum_{k=1}^{K}\left[Z_{s_0,a}^{k,i}\right] + \frac{\gamma}{K}\sum_{k=1}^{K}\left[Z_{s_1,a}^{k,i-1}\right] + \gamma y_{s_1,a}^{i-1}$(typo in paper)

By continue until the terminal state $y_{s_{M-1},a}^{j} = 0$,we can get,

$$Q_i^E(s_0,a) = \sum_{m=0}^{M-1}\gamma^m \frac{1}{K}\sum_{k=1}^{K} Z_{s_m,a}^{k,i-m}$$

Since TAE are uncorrelated by assumption, we can add the variance separately , for the variance of first term in $Q_i^E(s_0,a)$, we can get $\mathrm{Var}(\frac{1}{k}\sum_{k=1}^{K}\left[Z_{s_0,a}^{k,i}\right]) = \frac{1}{k^2}k\sigma_{s_0}^2 = \frac{1}{k}\sigma_{s_0}^2$, thus we can collect all the term into

$$\mathrm{Var}\left[Q_i^E(s_0,a)\right] = \sum_{m=0}^{M-1}\frac{1}{K}\gamma^{2m}\sigma_{s_m}^2$$

,the prove done.

3.3.3 Average-DQN Variance

Lemma 4 *for $i > KM$.assume that:*$\mathbb{E}\left[Z_{s,a}^i\right] = 0$,$\mathrm{Var}\left[Z_{s,a}^i\right] = \sigma_s^2$, *for i≠j :*$\mathrm{Cov}\left[Z_{s,a}^i, Z_{s',a}^j\right] = 0$,*and for $s \neq s'$:* $\mathrm{Cov}\left[Z_{s,a}^i, Z_{s',a}^j\right] = 0$, *we get the Average DQN Variance :*

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

where $D_{K,m} = \frac{1}{N}\sum_{n=0}^{N-1}|Un/K|^{2(m+1)}$, with $U = (U_n)_{n=0}^{N-1}$ denoting a Discrete Fourier Transform(DFT) of a rectangle pulse, and $|Un/K|^{2(m+1)} \leq 1$.

Proof of Lemma 4

By the equation in Algorithm 3 :

$Q_i^A(s_0,a) =$

$= \frac{1}{K}\sum_{k=1}^{K} Q\left(s_0,a;\theta_{i+1-k}\right)$

$= \frac{1}{K}\sum_{k=1}^{K}\left[Z_{s_0,a}^{i+1-k} + y_{s_0,a}^{i+1-k}\right]$

$= \frac{1}{K}\sum_{k=1}^{K}\left[Z_{s_0,a}^{i+1-k}\right] + \frac{\gamma}{K}\sum_{k=1}^{k} Q_{i-k}^A(s_1,a)$(Due to the fact in algorithm 3 line6 , and r=0 & max over one action)

$= \frac{1}{K}\sum_{k=1}^{K}\left[Z_{s_0,a}^{i+1-k}\right] + \frac{\gamma}{K^2}\sum_{k=1}^{K}\sum_{k'=1}^{K} Q(s_1,a;\theta_{i+1-k-k'})$(expand one term).

By continue expanding $Q(s_1,a;\theta_{i+1-k-k'})$ as above, and until the final state , we get :

$Q_i^A(s_0,a) = \frac{1}{K}\sum_{k=1}^{K} Z_{s_0,a}^{i+1-k} + \frac{\gamma}{K^2}\sum_{k=1}^{K}\sum_{k\prime=1}^{K} Z_{s_1,a}^{i+1-k-k\prime}$

$+\cdots+\frac{\gamma^{M-1}}{K^M}\sum_{j_1=1}^{K}\sum_{j_2=1}^{K}\cdots\sum_{j_M=1}^{K} Z_{s_{M-1},a}^{i+1-j_1-\cdots-j_M}$(change k and $k\prime$ into j_1 and j_2 to make it clear) (10)

Since the TAEs in different states are uncorrelated by assumption ,and the sum are uncorrelated ,thus can calculate the variance of each separately. For $L = 1, \cdots, M$, we denote a V_L means the each term in the right hand side of equation 10:

$V_L = Var[\frac{1}{K^L} \sum_{i_1=1}^{K} \sum_{i_2=1}^{K} \cdots \sum_{i_L=1}^{K} Z_{i_1+i_2+\cdots+i_L}]$,

to simplify the notation, paper assume $Z_L, Z_{L+1}, \cdots, Z_{K \cdot L}$ (here $Z_L, Z_{L+1}, \cdots$ is correspond to the term of all possible value of $i_1 + i_2 + \cdot + i_L$ above) are independent distribution(i.i.d.) TAEs random variable with $\mathbb{E}[Z_l] = 0$ and $\mathbb{E}[(Z_l)^2] = \sigma_z^2$. Thus we can have

$$\begin{aligned} V_L &= \frac{1}{K^{2L}} \mathbb{E}_z[(\sum_{j=L}^{KL} n_{j,L} Z_j)^2] \\ &= \frac{\sigma_z^2}{K^{2L}} \sum_{j=L}^{KL} (n_{j,L})^2 \end{aligned}$$

Where $n_{j,L}$ is the number of times Z_j is counted in the multiple summation. So the problem is to evaluating the $n_{j,L}$, where $j \in \{L, \cdots, KL\}$, the calculation can be done by recursively, by

$$n_{j,L} = \sum_{i=1}^{K} n_{j-i,L-1} \text{(find by extract one term out ,and find the remaining sum to j-i)}$$

Next we use frequency domain to bound the variance reduction

Denote frequency:

$$u_j^K = \begin{cases} 1 \text{ if } j \in \{1, \ldots, K\} \\ 0 \text{ otherwise} \end{cases}$$

and $n_{j,1} = u_j^K$ which is obviously (base case), and we rewrite the recursive formula

$$\begin{aligned} n_{j,L} &= \sum_{i=-\infty}^{\infty} n_{j-i,L-1} \cdot u_i^K \\ &\equiv (n_{j-1,L-1} * u^K)_j \qquad \text{by definition from wiki:} [(f*g)[n] = \textstyle\sum_{m=-M}^{M} f[n-m]g[m]] \\ &= \underbrace{\left(u^K * u^K \ldots * u^K\right)_j}_{L \text{ times}} \qquad \text{(expand the term L times)} \end{aligned}$$

Continue, we use 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 we can get

$$\begin{aligned} V_L &= \frac{\sigma^2}{K^{2L}} \sum_{n=0}^{N-1} |(u^K * u^K \ldots * u^K)_n|^2 \\ &= \frac{\sigma^2}{K^{2L}} \frac{1}{N} \sum_{n=0}^{N-1} |U_n|^{2L} \qquad \text{(see eq(12) for detail)} \end{aligned} \tag{11}$$

N is the length of the vector u and U and is taken large enough so that the sum includes all non-zero elements of the convolution

Proof of Parseval's theorem

here we denote $|(u^K * u^K \dots * u^K)_n|$ as $x[n]$ and $W_N = e^{j\frac{2\pi}{n}}$

$$\begin{aligned}
&\textstyle\sum_{n=0}^{N-1} |x[n]|^2 \\
&= \textstyle\sum_{n=0}^{N-1} x[n]x^*[n] \\
&= \textstyle\sum_{n=0}^{N-1} x[n] \left[\frac{1}{N}\sum_{n'=0}^{N-1} U[n']W_N^{-n'n}\right] \\
&= \textstyle\frac{1}{N}\sum_{n'=0}^{N-1} U[n'] \left[\sum_{n=0}^{N-1} x[n]W_N^{-n'n}\right] \\
&= \textstyle\frac{1}{N}\sum_{n'=0}^{N-1} U[n]U^*[n] \\
&= \textstyle\frac{1}{N}\sum_{n=0}^{N-1} |U[n]|^2
\end{aligned} \tag{12}$$

By (11) to make it clear we define $D_{K,m} = \frac{1}{K^{2(m+1)}}\frac{1}{N}\sum_{n=0}^{N-1}|U_n|^{2(m+1)}$, then we can derive Average-DQN variance as:

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

Next we bound the $D_{K,m}$ to compare with DQN and Ensemble-DQN

For $K > 1, m > 0$:

$$\begin{aligned}
D_{K,m} &= \frac{1}{K^{2(m+1)}}\frac{1}{N}\sum_{n=0}^{N-1}|U_n|^{2(m+1)} \\
&= \frac{1}{N}\sum_{n=0}^{N-1}|U_n/K|^{2(m+1)} \\
&< \frac{1}{N}\sum_{n=0}^{N-1}|U_n/K|^2 \qquad \text{(by the fact } \tfrac{1}{K}|U_n| \le 1\text{)} \\
&= \frac{1}{K^2}\sum_{n=0}^{N-1}\left|u_n^K\right|^2 \\
&= \frac{1}{K}
\end{aligned}$$

As we can see , the variance is smaller than Ensemble-DQN and Ensemble-DQN is smaller than DQN , we can see that Average-DQN is the best , which reduce much more variance.

This end the proof of lemma 4

4 Conclusion

In this report , at first ,we identify the issue of overestimate, then discuss the Average-DQN algorithm and Ensemble algorithm and theory prove (with much more clear and detail compare to the paper), which prove that this algorithm indeed reduce the variance and also have well performance in implementation (can see the paper for implementation part).

However, there is some assumption, such as the covariance zero assumption, seems unreasonable because the variance of the next iteration is truly some relate to the previous one, I think we might assume that the covariance is under some constant and get more reasonable prove. And we can also see in figure 4 , the Average DQN seems to have underestimate problem in the first 500 iteration, but in theory prove, this situation shouldn't happen(because this method reduce the variance instead of give negative bias on Q-value), which is a contradiction. However, despite those deficiency, the

experiment is still perform very well compare to the DQN, this algorithm truly make good solution in average score. I think the potential of this paper is very high, ex: We can try to find what is the best "K" we need to average in the iteration(also can tune K to be different in every iteration) , or even we can design a dynamic "K" search , let the "K" automatically tune in different environment and perform the best result. Also , can use the average idea into lots of DQN variants , and also other RL method to see if there is a improve.

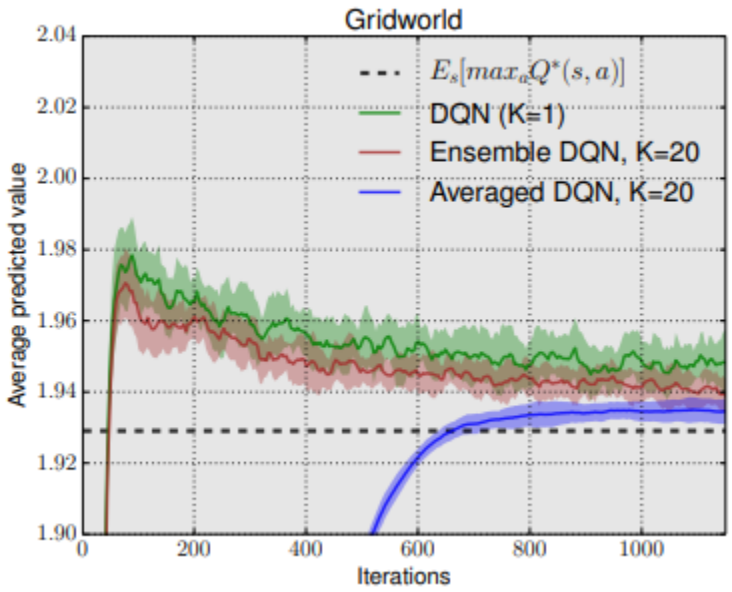

Figure 4: As we can see, in the Average DQN line, in the first iteration , it underestimate the Q value, which is not aware in theory prove

Also there are lots of research deal with the overestimate problem,Fujimoto et al. [2018] inspired by this paper to identify the overestimate in the Actor-Critic algorithm , and Jin et al. [2017] use some modify of Average-DQN to defined a new advantage function to minimize the regret just like the idea here. Gruslys et al. [2017] also do the algorithm by combines multiple algorithmic and architectural contributions to produce an agent with higher sample-efficiency.

I think this idea can use not only on RL range but also can use in some machine learning and supervise learning , which average the model seems to be good when facing some unstable problem. Like the research I studied before McMahan et al. [2016] also using the average idea to deal with the federating learning problem.

References

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning, 2016.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2015.

G. Rummery and Mahesan Niranjan. On-line q-learning using connectionist systems. *Technical Report CUED/F-INFENG/TR 166*, 11 1994.

Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, May 1992. ISSN 1573-0565. doi: 10.1007/BF00992698. URL `https://doi.org/10.1007/BF00992698`.

Hado V. Hasselt. Double q-learning. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 2613–2621. Curran Associates, Inc., 2010. URL `http://papers.nips.cc/paper/3964-double-q-learning.pdf`.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. URL http://arxiv.org/abs/1312.5602.

L. J. Lin. *Reinforcement Learning for Robots Using Neural Networks*. PhD thesis, Carnegie Mellon University, Pittsburgh, January 1993.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay, 2015. URL http://arxiv.org/abs/1511.05952. cite arxiv:1511.05952Comment: Published at ICLR 2016.

Sebastian Thrun and Anton Schwartz. Issues in Using Function Approximation for Reinforcement Learning. In M. Mozer, P. Smolensky, D. Touretzky, J. Elman, and A. Weigend, editors, *Proceedings of the 1993 Connectionist Models Summer School*, Hillsdale, NJ, 1993a. Lawrence Erlbaum. URL citeseer.ist.psu.edu/thrun93issues.html.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning, 2015.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. ISSN 00280836. URL http://dx.doi.org/10.1038/nature14236.

Sebastian Thrun and Anton Schwartz. Issues in using function approximation for reinforcement learning. In *In Proceedings of the Fourth Connectionist Models Summer School*. Erlbaum, 1993b.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods, 2018.

Peter Jin, Kurt Keutzer, and Sergey Levine. Regret minimization for partially observable deep reinforcement learning, 2017.

Audrunas Gruslys, Will Dabney, Mohammad Gheshlaghi Azar, Bilal Piot, Marc Bellemare, and Remi Munos. The reactor: A fast and sample-efficient actor-critic agent for reinforcement learning, 2017.

H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data, 2016.