Value-based Reinforcement Learning

Some Discussions

Kan Ren

Apex Data and Knowledge Management Lab
Shanghai Jiao Tong University

Aug. 3 2017

Outline

Outline

1. Revision of Value-based RL
   - Dynamic Programming(omitted)
   - Monte-carlo Method(omitted)
   - TD: Sarsa and Q-learning
2. Deep Q-network
   - Nature DQN
   - Several Imrovements
3. Issues in Q-learning
   - Overestimation
   - Double Q-learning
   - Averaged Q-learning
4. Convergence of Tabular TD
   - Sarsa
   - Q-learning (TBE)

Outline

Outline

1. Revision of Value-based RL
   - Dynamic Programming(omitted)
   - Monte-carlo Method(omitted)
   - TD: Sarsa and Q-learning
2. Deep Q-network
   - Nature DQN
   - Several Imrovements
3. Issues in Q-learning
   - Overestimation
   - Double Q-learning
   - Averaged Q-learning
4. Convergence of Tabular TD
   - Sarsa
   - Q-learning (TBE)

Sarsa & Q-learning

Algorithm

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

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
  Choose A from S using policy derived from Q (e.g., ϵ-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ϵ-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

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'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
    $S \leftarrow S'$
  until S is terminal

Difference

- Exploration
  - Sarsa: on-policy
  - Q-learning: off-policy
- Update Rule
  - Sarsa

$$
\begin{aligned}
&Choose\ A'\ from\ S'\ using\ policy\ derived\ from\ Q\ (e.g.\epsilon-greedy)\\
&Q(S,A) \leftarrow Q(S,A) + \alpha[r + \gamma Q(S',A') - Q(S,A)]
\end{aligned}
$$

  - Q-learning

$$
Q(S,A) \leftarrow Q(S,A) + \alpha[r + \gamma \max_a Q(S',a) - Q(S,A)]
$$

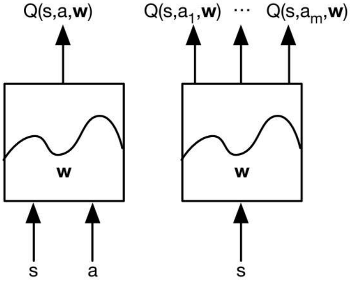
Outline

Outline

Q-networks

Represent value function by Q-network with weights w

$$Q(s, a; w) \approx Q^*(s, a) \ . \tag{1}$$



Deep Q-network

Refer to D. Silver's slides P31 - P45.

Duelling network

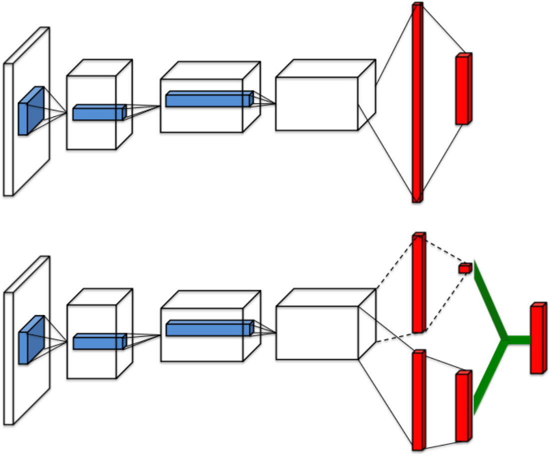


Figure: Duelling network: split Q-network into two channels

Outline

Overestimation

Preliminaries

Recall that

$$Q(s,a) \longleftarrow r_s^a + \gamma \max_{\hat{a}} Q(s', \hat{a}) \quad (2)$$

Repeated application of this update equation eventually yields Q-values that give rise to a policy which maximizes the expected cumulative discounted reward[1] in the look-up table case.

The max operation may cause some problems under the approximation scenario.

[1] C. J. C. H.Watkins, Learning from Delayed Rewards. PhD thesis, Kings College, Cambridge, England, 1989.

Overestimation

Assume $Q^{approx}(\cdot)$ representing implicit target values Q^{target}, corrupted by a noise term Y such that

$$Q^{approx}(s', \hat{a}) = Q^{target}(s', \hat{a}) + Y_{s'}^{\hat{a}}$$

$$\begin{aligned} Z_s &\overset{def}{=} r_s^a + \gamma \max_{\hat{a}} Q^{approx}(s', \hat{a}) - \left(r_s^a + \gamma \max_{\hat{a}} Q^{target}(s', \hat{a}) \right) \\ &= \gamma \left(\max_{\hat{a}} Q^{approx}(s', \hat{a}) - \max_{\hat{a}} Q^{target}(s', \hat{a}) \right) \end{aligned} \tag{3}$$

The key observation is

$$E[Y_{s'}^{\hat{a}}] = 0, \ \forall \hat{a} \overset{often}{\Longrightarrow} E[Z_s] > 0 \ .$$

Expectation of Z

Lemma

Let n denote the number of actions applicable at state s'. If all n actions share the same target Q-value, i.e., $\exists q : \forall \hat{a} : q = Q^{target}(s', \hat{a})$, then the average overestimation $E[Z_s]$ is γc with $c \overset{def}{=} \epsilon \frac{n-1}{n+1}$.

The proof can be referred to the paper[2].

Corollary

$0 \leq E[Z_s] \leq \gamma c$ with $c = \epsilon \frac{n-1}{n+1}$.

[2] Thrun S, Schwartz A. Issues in using function approximation for reinforcement learning[C] Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum. 1993.

Bounds for Expected Failure of Q-learning

Simple Assumptions

- There is a set of goal states;
- Positive reward r_{goal} is only recieved upon entering a goal state;
- $r_{goal} = 1$;
- The state transition function is deterministic.

One *necessary* condition for the success of Q-learning is that the sequence of Q-values $Q(s_i, a_i)$ is monotonically increasing in i:

$$Q(s_i, a_i) \leq Q(s_{i+1}, a_{i+1}), \textit{for all } i \in \{0, \dots, L-1\} \tag{4}$$

Bounds for Expected Failure of Q-learning

Simple Assumptions

Case 1: the learner *always* overestimates Q-values by γc.

Theorem

If there is maximal, repeated overestimation of magnitude γc along an optimal path, Q-learning is expected to fail to learn an optimal policy if $\gamma > \frac{1}{1+c}$.

Case 2: Assume that Q-learning managed to learn the *last* L-1 Q-values of this optimal path correctly.

- Q-values are given by iteratively discounting the final reward with the *distance* to the goal state, i.e., $Q(s_{L-i}, a_{L-i}) = \gamma^i$ for $i \in \{1, \ldots, L-1\}$.
- *Correct* Q-value $Q^{correct}(s_0, a_0)$ is γ^L.
- In order to maintain monotonicity of Q, we need to make sure that

$$\gamma^{L-1} - \gamma^L \geq \gamma c \ . \tag{5}$$

Theorem

Under the conditions above, Q-learning is expected to fail if

$$\gamma^{L-1} - \gamma^L < \gamma c \ . \tag{6}$$

Theorem

Under the conditions above, Q-learning is expected to fail if

$$\epsilon > \frac{n+1}{n-1} \cdot \frac{(L-2)^{L-2}}{(L-1)^{L-1}} \,. \tag{7}$$

Outline

Double Q-learning

Preliminaries

- a set of random variables $X = \{X_i, \ldots, X_M\}$ Our interest is that

$$\max_i E[X_i] \;, \tag{8}$$

  which is in the Q-learning update rule.
- $S = \cup_{i=1}^{M} S_i$ where S_i is the subset contains samples for the variable X_i and each $s \in S_i$ is i.i.d.
- $E[X_i] = E[\mu_i] \approx \mu_i(S) \overset{def}{=} \frac{1}{|S_i|} \sum_{s \in S_i} s$, where μ_i is an unbiased estimate for the value of $E[X_i]$.
- f_i^{μ} is PDF and F_i^{μ} is CDF of X_i.

$$\max_i E[X_i] = \max_i \int_{-\infty}^{\infty} x \; f_i^{\mu}(x) dx \;.$$

Double Q-learning

Single Estimator

An obvious way to approximate the value in Eq. (8) is

$$\max_i E[X_i] = \max_i E[\mu_i] \approx \max_i \mu_i(S) \ . \tag{9}$$

- Assume the maximal estimator $\max_i \mu_i(S)$ is distributed as PDF f_{max}^{μ}.
- $f_{max}^{\mu} \neq f_i^{\mu}$ but f_{max}^{μ} is dependent on f_i^{μ}.

Double Q-learning

Single Estimator

An obvious way to approximate the value in Eq. (8) is

$$\max_i E[X_i] = \max_i E[\mu_i] \approx \max_i \mu_i(S) \ . \tag{9}$$

- Assume the maximal estimator $\max_i \mu_i(S)$ is distributed as PDF f^{μ}_{max}.
- $f^{\mu}_{max} \neq f^{\mu}_i$ but f^{μ}_{max} is dependent on f^{μ}_i.
- CDF $F^{\mu}_{max}(x) \stackrel{def}{=} P(\max_i \mu_i \leq x) = \prod_{i=1}^{M} P(\mu_i \leq x) \stackrel{def}{=} \prod_{i=1}^{M} F^{\mu}_i(x)$.

Double Q-learning

Biased Estimation of $E[X_i]$

- The value $\max_i \mu_i(S)$ is an unbiased estimate for $E[\max_j \mu_j]$.

$$
\begin{aligned}
E[\max_i \mu_i] &= \int_{-\infty}^{\infty} x \; f_{max}^{\mu}(x) \\
&= \int_{-\infty}^{\infty} x \frac{d}{dx} \prod_{i=1}^{M} F_i^{\mu}(x) dx \\
&= \sum_{j}^{M} \int_{-\infty}^{\infty} x \; f_j^{\mu}(x) \prod_{i \neq j}^{M} F_i^{\mu}(x) dx \; .
\end{aligned}
\tag{10}
$$

- $E[\max_i \mu_i]$ is not the same as $\max_i E[X_i]$.

Double Q-learning

Double Estimators

- Two sets of estimators: $\mu^A = \mu_1^A, \ldots, \mu_M^A$, $\mu^b = \mu_1^B, \ldots, \mu_M^B$.
- Two subsets of samples: $S = S^A \cup S^B$, $S^A \cap S^B = \emptyset$
- $\mu_i^A(S) \stackrel{def}{=} \frac{1}{|S_i^A|} \sum_{s \in S_i^A} s$, $\mu_i^B(S) \stackrel{def}{=} \frac{1}{|S_i^B|} \sum_{s \in S_i^B} s$.

- Both μ_i^A and μ_i^B are unbiased if we assume proper split on the sample set S.
- $Max^A(S) \stackrel{def}{=} \{j | \max_i \mu_i^A(S)\}$.
- Since $\mu_i^B(S)$ is an independent, unbiased set of estimators, we have $E[\mu_j^B(S)] = E[X_j]$ for all j including $j \in Max^A$. We can pick a^* such that $\mu_{a^*}^A \stackrel{def}{=} \max_i \mu_i^A(S)$. So that

$$\max_i E[X_i] = \max_i E[\mu_i^B] \approx \mu_{a^*}^B \ . \tag{11}$$

Double Q-learning

Difference Between Single/Double Estimators

$$\begin{aligned} P(j = a^*) &= \int_{-\infty}^{\infty} P(\mu_j^A = x) \prod_{i \neq j}^{M} P(\mu_j^A < x) dx \\ &\stackrel{def}{=} \int_{-\infty}^{\infty} f_j^A(x) \prod_{i \neq j}^{M} F_i^A(x) dx \end{aligned} \tag{12}$$

$$\sum_{j}^{M} P(j = a^*) E[\mu_j^B] = \sum_{j}^{M} E[\mu_j^B] \int_{-\infty}^{\infty} f_j^A(x) \prod_{i \neq j}^{M} F_i^A(x) dx \ . \tag{13}$$

Recall Eq. (10) of single estimator that

$$E[\max_i \mu_i] = \int_{-\infty}^{\infty} x \ f_{max}^{\mu}(x) = \sum_{j}^{M} \int_{-\infty}^{\infty} x \ f_j^{\mu}(x) \prod_{i \neq j}^{M} F_i^{\mu}(x) dx \ .$$

Double Q-learning

Algorithm[3,4]

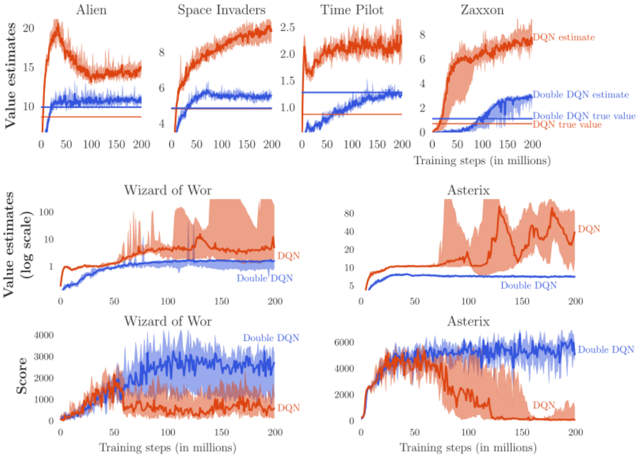
Algorithm 1 Double Q-learning

Initialize Q^A,Q^B,s
repeat
  Choose a, based on $Q^A(s,\cdot)$ and $Q^B(s,\cdot)$, observe r, s'
  Choose (e.g. random) either UPDATE(A) or UPDATE(B)
  if UPDATE(A) **then**
    Define $a^* = \arg\max_a Q^A(s',a)$
    $Q^A(s,a) \leftarrow Q^A(s,a) + \alpha(s,a)\left(r + \gamma Q^B(s',a^*) - Q^A(s,a)\right)$
  else if UPDATE(B) **then**
    Define $b^* = \arg\max_a Q^B(s',a)$
    $Q^B(s,a) \leftarrow Q^B(s,a) + \alpha(s,a)(r + \gamma Q^A(s',b^*) - Q^B(s,a))$
  end if
  $s \leftarrow s'$
until end

[3] Hasselt H V. Double Q-learning[C] Advances in Neural Information Processing Systems. 2010: 2613-2621.
[4] Van Hasselt H, Guez A, Silver D. Deep Reinforcement Learning with Double Q-Learning[C] AAAI. 2016: 2094-2100.

Double Q-learning

Performance



Outline

Averaged Deep Q-Network

- Double Q-learning aims to correct the *overestimation* of natural Q-learning.
- Averaged DQN focus on variance reduction and stabilization.

Averaged Deep Q-Network

Revision of DQN[5]

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*$(\cdot)$
4: **for** $i = 1, 2, \ldots, N$ **do**
5: $\quad y^i_{s,a} = \mathbb{E}_{\mathcal{B}}\left[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) | \, s, a\right]$
6: $\quad \theta_i \approx \operatorname{argmin}_\theta \mathbb{E}_{\mathcal{B}}\left[(y^i_{s,a} - Q(s, a; \theta))^2\right]$
7: $\quad$ *Explore*$(\cdot)$, update $\mathcal{B}$
8: **end for**

output $Q^{\mathrm{DQN}}(s, a; \theta_N)$

[5] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529-533.

Averaged Deep Q-Network

Algorithm[6]

Algorithm 2 Averaged DQN

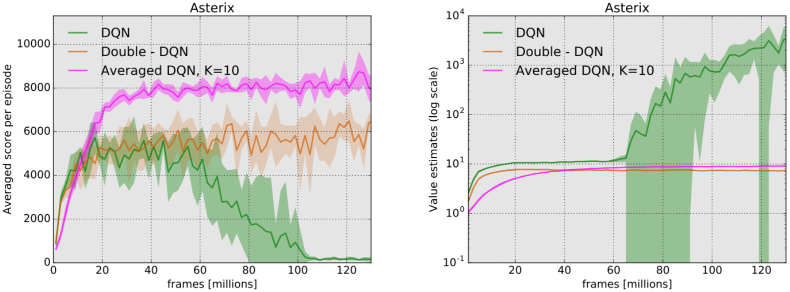
1: Initialize $Q(s, a; \theta)$ with random weights θ_0
2: Initialize Experience Replay (ER) buffer $\mathcal{B}$
3: Initialize exploration procedure $Explore(\cdot)$
4: **for** $i = 1, 2, \dots, N$ **do**
5: $\quad Q^A_{i-1}(s, a) = \frac{1}{K} \sum_{k=1}^{K} Q(s, a; \theta_{i-k})$
6: $\quad y^i_{s,a}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

output $Q^A_N(s, a) = \frac{1}{K} \sum_{k=0}^{K-1} Q(s, a; \theta_{N-k})$

[6] Anschel O, Baram N, Shimkin N. Averaged-DQN: Variance Reduction and Stabilization for Deep Reinforcement Learning[C] International Conference on Machine Learning. 2017: 176-185.

Averaged Deep Q-Network

Performance



Averaged Deep Q-Network

Error Analysis

Let $Q(s, a; \theta_i)$ be the value function of DQN at iteration i,

$$\begin{aligned} \Delta_i &= Q(s, a; \theta_i) - Q^*(s, a) \\ &= \underbrace{Q(s, a; \theta_i) - y^i_{s,a}}_{\text{Target Apprixmation Error}} + \underbrace{y^i_{s,a} - \hat{y}^i_{s,a}}_{\text{Overestimation Error}} \\ &\quad + \underbrace{\hat{y}^i_{s,a} - Q^*(s, a)}_{OptimalityDifference} . \end{aligned} \tag{14}$$

Here $y^i_{s,a}$ is the *DQN target*, and $\hat{y}^i_{s,a}$ is the *true target*, such that

$$\begin{aligned} y^i_{s,a} &= E_{\mathcal{B}}\left[r + \gamma \max_{a'} Q(s', a'; \theta_i i - 1) | s, a \right] , \\ \hat{y}^i_{s,a} &= E_{\mathcal{B}}\left[r + \gamma \max_{a'} (\hat{y}^{i-1}_{s',a'}) | s, a \right] . \end{aligned} \tag{15}$$

Averaged Deep Q-Network

Background and Related Work

Define $Z^i_{s,a}$ as TAE (Target Approximation Error) and $R^i_{s,a}$ as overestimation error.

$$\begin{aligned} Z^i_{s,a} &= Q(s, a; \theta_i) - y^i_{s,a} \ , \\ R^i_{s,a} &= y^i_{s,a} - \hat{y}^i_{s,a} \ . \end{aligned} \tag{16}$$

In Thrun & Schwartz (1993), $Z^i_{s,a}$ is considered as a random variable uniformly distributed error in $[-\epsilon, \epsilon]$ and

$$E_z[R^i_{s,a}] = \gamma E_z[\max_{a'}[Z^{i-1}_{s',a'}]] = \gamma\epsilon\frac{n-1}{n+1} \ . \tag{17}$$

In Double Q-learning paper, the author replaces *positive* bias with a *negative* one.

Averaged Deep Q-Network

TAE Variance Reduction

Assume that

$$\begin{aligned} &E[Z^i_{s,a}] = 0,\ \ Var[Z^i_{s,a}] = \sigma^2_s, \\ &\text{for } i \neq j, Cov[Z^i_{s,a}, Z^j_{s',a'}] = 0. \end{aligned} \tag{18}$$

We consider a fixed policy for updating the target values, and conveniently consider a zero reward $r = 0$ everywhere since it has no effect on variance calculations.

Averaged Deep Q-Network

TAE Variance Reduction (cont.)

Consider M-state unidirectional MDP as

$$
\begin{aligned}
Q^{DQN}(s_0, a; \theta_i) &= Z^i_{s_0,a} + y^i_{s_0,a} \\
&= Z^i_{s_0,a} + \gamma Q(s_1, a; \theta_{i-1}) \\
&= Z^i_{s_0,a} + \gamma[Z^{i-1}_{s_1,a} + y^{i-1}_{s_1,a}] = \ldots = \\
&= Z^i_{s_0,a} + \gamma Z^{i-1}_{s_1,a} + \ldots + \gamma^{M-1} Z^{i-(M-1)}_{s_{M-1},a}
\end{aligned} \tag{19}
$$

Since for $i \neq j$, $Cov[Z^i_{s,a}, Z^j_{s',a'}] = 0$, we have

$$
Var[Q^{DQN}(s_0, a; \theta_i)] = \sum_{m=0}^{M-1} \gamma^{2m} \sigma^2_{s_m} \ . \tag{20}
$$

Averaged Deep Q-Network

TAE Variance Reduction (cont.)

For Averaged DQN,

$$Q_i = Z_i + \gamma P \frac{1}{K} \sum_{k=1}^{K} Q_{i-k} \ , \tag{21}$$

where $P \in \mathbb{R}_{+}^{S \times S}$ is the transition probabilities matrix for the given policy. Recall that $Z_{s,a}^{i} = Q(s, a; \theta_i) - y_{s,a}^{i}$.

Averaged Deep Q-Network

Ensemble DQN

Algorithm 3 Ensemble DQN

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

Averaged Deep Q-Network

Ensemble DQN Variance

For $i > M$,

$$\begin{aligned} Q_i^E(s_0, a) &= \sum_{m=0}^{M-1} \gamma^m \frac{1}{K} \sum_{k=1}^{K} Z_{s_m,a}^{k,i-m} \\ Var[Q_i^E(s_0, a)] &= \sum_{m=0}^{M-1} \frac{1}{K} \gamma^{2m} \sigma_{s_m}^2 \\ &= \frac{1}{K} Var[Q^{DQN}(s_0, a; \theta_i)] \end{aligned} \tag{22}$$

Averaged Deep Q-Network

Averaged DQN Variance

For $i > KM$,

$$Var[Q_i^A(s_0, a)] = \sum_{m=0}^{M-1} D_{K,m} \gamma^{2m} \sigma_{s_m}^2 \ , \tag{23}$$

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

$$\begin{aligned} Var[Q_i^A(s_0, a)] &< Var[Q_i^E(s_0, a)] \\ &= \frac{1}{K} Var[Q^{DQN}(s_0, a; \theta_i)] \ . \end{aligned} \tag{24}$$

Outline

1. Revision of Value-based RL
   - Dynamic Programming(omitted)
   - Monte-carlo Method(omitted)
   - TD: Sarsa and Q-learning
2. Deep Q-network
   - Nature DQN
   - Several Imrovements
3. Issues in Q-learning
   - Overestimation
   - Double Q-learning
   - Averaged Q-learning
4. Convergence of Tabular TD
   - Sarsa
   - Q-learning (TBE)

Convergence of Sarsa(0)

Convergence of Random Iterative Process

Lemma

A random iterative process

$$\Delta_{t+1}(x) = (1 - \alpha_t(x))\Delta_t(x) + \alpha_t(x)F_t(x), \ x \in X, \ t = 0, 1, 2, \ldots \quad (25)$$

converges to zero w.p.1 if the following properties hold:

- *1. the set of possible states X is finite.*
- *2. $0 \leq \alpha_t(x) \leq 1, \sum_t \alpha_t(x) = \infty, \sum_t \alpha_t^2(x) < \infty$ w.p.1, where the probability is over the learning rates α_t.*
- *3. $\|E[F_t(\cdot)|P_t]\|_W \leq \kappa\|\Delta_t\|_W + c_t$, where $\kappa \in [0, 1)$ and c_t converges to zero w.p.1.*
- *4. $Var[F_t(x)] \leq K(1 + \|\Delta_t\|_W)^2$, where K is some constant.*

Here P_t is an increasing sequence of σ-fields that includes the past of the process. In particular we assume that $\alpha_t, \Delta_t, F_{t-1} \in P_t$. The notation $\|\cdot\|_W$ refers to some (fixed) weighted maximum norm.

Convergence of Sarsa(0)

Theorem

In finite state-action MDPs, the Q_t values computed by the Sarsa(0) rule

$$\begin{aligned} Q_{t+1}(s_t, a_t) &= Q_t(s_t, a_t) + \alpha_t(s_t, a_t)[r_t + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)] \\ &= (1 - \alpha(s_t, a_t))Q_t(s_t, a_t) + \alpha_t(s_t, a_t)[r_t + \gamma Q_t(s_{t+1}, a_{t+1})] \ . \end{aligned}$$

converges to Q^* and the learning policy π_t converges to an optimal policy π^* if the learning policy is GLIE with these additional conditions are satisfied

- *1. The Q values are stored in a lookup table.*
- *2. The learning rates satisfy* $0 \leq \alpha_t(s_t, a_t) \leq 1, \ \sum_t \alpha_t(s_t, a_t) = \infty, \sum_t \alpha_t^2(s_t, a_t) < \infty$ *and* $\alpha_t(s_t, a_t) = 0$ *unless* $(s, a) = (s_t, a_t)$.
- *3.* $Var[r(s, a)] < \infty$.

Convergence of Sarsa(0)

- $x \stackrel{def}{=} (s_t, a_t)$.
- $\Delta_t \stackrel{def}{=} Q_t(s,a) - Q^*(s,a)$.

So we get

$$\begin{aligned}\Delta_{t+1}(s_t,a_t) &= Q_{t+1}(s_t,a_t) - Q^*(s,a) \\ &= (1-\alpha(s_t,a_t))\Delta_t(s_t,a_t) + \alpha_t(s_t,a_t)F_t(s_t,a_t).\end{aligned} \tag{26}$$

where

$$\begin{aligned}F_t(s_t,a_t) &= r_t + \gamma \max_{a'} Q_t(s_{t+1},a') - Q^*(s_t,a_t) \\ &\quad + \gamma\left[Q_t(s_{t+1},a_{t+1}) - \max_{a'} Q_t(s_{t+1},a')\right] \\ &\stackrel{def}{=} r_t + \gamma \max_{a'} Q_t(s_{t+1},a') - Q^*(s_t,a_t) + C_t(Q) \\ &\stackrel{def}{=} F_t^Q(s_t,a_t) + C_t(s_t,a_t)\end{aligned} \tag{27}$$

Outline