IIIT Hyderabad **Scribed By: Binu Jasim**
Topics in Machine Learning (CS975) **Roohi Nahid**
Monsoon 2017 **Lecture #22**
Instructor: Dr. Naresh Manwani **November 3, 2017**

Outline. *In this lecture we will see the proof of convergence of the Q-learning algorithm. We will also briefly discuss the Expected SARSA algorithm. Then we will learn about the maximization bias problem in Q-learning. Finally we will conclude with a heuristic, capable of reducing the maximization bias, called the Double Q-learning algorithm.*

1 Q-Learning

In the previous class, we have seen that the Q-learning is an off policy TD control algorithm. In contrast SARSA is an on policy TD control algorithm.

Algorithm 1: Q-Learning

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}$ arbitrarily.
foreach *episode* **do**
  Choose S (start state)
  repeat
    Choose action A using current policy (ϵ-greedy) derived from Q
    Take action A, observe next state S' and the reward R
    update: $Q(S,A) = (1-\alpha)Q(S,A) + \alpha[R + \gamma \max_{a'} Q(S',a')]$
    $S \leftarrow S'$
  until S *is terminal state*;
end

Note how the update rule of Q-learning (see Algorithm 1) differs from the update rule of SARSA ($Q(S,A) = (1-\alpha)Q(S,A) + \alpha[R + \gamma Q(S',A')]$). Q-learning estimates the return (total discounted future reward) for state-action pairs assuming a greedy policy were followed. But, the policy used to update Q-values (target policy) is not the same as the policy used to explore states (behavior policy), Q-learning is an off policy algorithm. The reason that SARSA is on-policy algorithm is that it updates its Q-values using the Q-value of the next state S' and the next action A' (from S') chosen using the same policy.

1.1 Q-Learning as Value Iteration

We can view the Q-Learning algorithm as a value iteration algorithm. We define a γ-contraction operator H corresponding to the update rules of Q-learning as follows. Let $d = |S| * |A|$, then $H : \mathbb{R}^d \to \mathbb{R}^d$ is defined as follows.

$$HQ(s,a) = \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma \max_{a'} Q(s',a')] \tag{1}$$

Show that. *H is a γ-contraction operator.*

Proof. We have to show that $||H(V_1) - H(V_2)||_\infty \leq \gamma ||HV_1 - HV_2||_\infty$ for any two vectors V_1 and V_2 in $\mathbb{R}^d$

$$||HQ_1 - HQ_2||_\infty = \max_{s,a} \left| \sum_{s'} P(s'|s,a)[r(s,a,s') - r(s,a,s') + \gamma(\max_{a'} Q_1(s',a') - \max_{a'} Q_2(s',a'))] \right|$$

$(\because ||V_1 - V_2||_\infty = \max_{i \in [d]} |V_1^i - V_2^i|$ where V^i denotes the i^{th} component of vector $V)$

$$\begin{aligned}
\therefore ||HQ_1 - HQ_2||_\infty &= \max_{s,a} \left| \gamma \sum_{s'} P(s'|s,a)[\max_{a'} Q_1(s',a') - \max_{a'} Q_2(s',a')] \right| \\
&\leq \max_{s,a} \gamma \sum_{s'} \left| P(s'|s,a)[\max_{a'} Q_1(s',a') - \max_{a'} Q_2(s',a')] \right| && (\because \text{triangle inequality}) \\
&= \max_{s,a} \gamma \sum_{s'} P(s'|s,a)[\left|\max_{a'} Q_1(s',a') - \max_{a'} Q_2(s',a')\right|] && (\because P(s'|s,a) \geq 0) \\
&\leq \max_{s,a} \gamma \sum_{s'} P(s'|s,a)[\max_{a'} \left| Q_1(s',a') - Q_2(s',a') \right|] && (\because \text{property of } \max) \\
&(\because |\max(U) - \max(V)| \leq \max(|U - V|) \text{ for } U, V \in \mathcal{R}^d) \\
&\leq \max_{s,a} \gamma \sum_{s'} P(s'|s,a)[\max_{a',s'} \left| Q_1(s',a') - Q_2(s',a') \right|] && (\because \max_{a'} f(a',s') \leq \max_{a',s'} f(a',s')) \\
&= \max_{s,a} \gamma \max_{a',s'} \left| Q_1(s',a') - Q_2(s',a') \right| \sum_{s'} P(s'|s,a)] && (\because \max_{a',s'} f(a',s') \text{ is a constant}) \\
&= \gamma ||Q_1 - Q_2||_\infty && \text{Hence the proof} \quad \square
\end{aligned}$$

1.2 A General Theorem for Stochastic Approximation

We will use the following general theorem of stochastic process convergence to show the convergence of Q-learning.

Theorem 1. *Let H be a γ-contraction mapping ($H : \mathbb{R}^N \to \mathbb{R}^N$) with fixed point x^*, and $(\boldsymbol{x}_t)$, $(\boldsymbol{w}_t)$ and $(\boldsymbol{\alpha}_t)$ be three sequences in $\mathbb{R}^N$ with*

$$\boldsymbol{x}_{t+1}(s) = \boldsymbol{x}_t(s) + \boldsymbol{\alpha}_t(s)[H(\boldsymbol{x}_t)(s) - \boldsymbol{x}_t(s) + \boldsymbol{w}_t(s)], \forall s \in [1, N] \tag{2}$$

Let $\mathcal{F}_t$ denotes the entire history for $t' \leq t$, that is: $\mathcal{F}_t = \{(\boldsymbol{x}_{t'})_{t' \leq t}, (\boldsymbol{w}_{t'})_{t' \leq t}, (\boldsymbol{\alpha}_{t'})_{t' \leq t}\}$ and assume that the following conditions are met:

- $\exists K_1, K_2 \in \mathcal{R} : \mathbb{E}[\boldsymbol{w}_t^2(s)|\mathcal{F}_t] \leq K_1 + K_2 \|x_t\|^2$ *for some norm* $\|.\|$
- $\mathbb{E}[\boldsymbol{w}_t(s)|\mathcal{F}_t] = 0$
- $\sum_t \boldsymbol{\alpha}_t(s) = \infty$ *and* $\sum_t \boldsymbol{\alpha}_t^2(s) < \infty$

then the sequence $\boldsymbol{x}_t$ converges almost surely to x^*.

We will not discuss the proof as it involves concepts like martingales which are beyond the scope of this course.

1.3 Convergence Proof of Q Learning

Theorem 2. *Consider a finite MDP. Assume that $\forall s \in S, \forall a \in A$*

$$\sum_t \alpha_t(s,a) = \infty \text{ and } \sum_t \alpha_t^2(s,a) < \infty \text{ and } \alpha_t(s,a) \in [0,1]$$

then the Q-learning algorithm converges to the optimal value Q^* with probability 1.

(Notice that the condition $\sum_t \alpha_t(s,a) = \infty$ implies that each state-action pair has to be visited infinitely many times)

Proof. Let $\{Q_t(s,a)\}_{t\geq 0}$ denote the sequence of state-action value functions generated by Q-learning algorithm.

$Q_{t+1}(s,a) = Q_t(s,a) + \alpha_t(s,a)[r(s,a) + \gamma \max\limits_{a'} Q_t(s', a') - Q_t(s,a)]$

Add and subtract $\mathbb{E}_{s' \sim Pr[.|s,a]}[\max\limits_{a'} Q_t(s', a')]$

(Where the expectation is over the transition probability $P(s'|s,a)$)

$$\begin{aligned} Q_{t+1}(s,a) =& Q_t(s,a) + \alpha_t(s,a)[r(s,a) + \gamma \mathbb{E}[\max_{a'} Q_t(s', a')] - Q_t(s,a)] \\ &+ \alpha_t(s,a)[\gamma \max_{a'} Q_t(s', a') - \gamma \mathbb{E}[\max_{a'} Q_t(s', a')]] \end{aligned}$$

(Note: $Q_t \in \mathbb{R}^d$ and $Q_t(s,a)$ defines a component of Q_t corresponding to $S_t = s, A_t = a$)

Comparing with Theorem 1, we define $\boldsymbol{w}_t(s') = \gamma[\max_{a'} Q_t(s', a') - \mathbb{E}[\max_{a'} Q_t(s', a')]]$
The condition $\mathbb{E}[\boldsymbol{w}_t(s')|\mathcal{F}_t] = 0$ is therefore satisfied.

$\therefore\ Q_{t+1}(s,a) = Q_t(s,a) + \alpha_t(s,a)[HQ_t(s,a) - Q_t(s,a) + \gamma w_t(s)]$
($\because$ from equation (1), $HQ_t(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim Pr[.|s,a]}[\max_{a'} Q_t(s', a')]$)

The conditions on α_t hold by assumption. Next we verify the first condition of the theorem 1.

$$\begin{aligned} |\boldsymbol{w}_t(s')| &\leq \gamma \max_{a'} |Q_t(s', a')| + \gamma \big|\mathbb{E}_{s' \sim Pr[.|s,a]}[\max_{a'} Q_t(s', a')]\big| \\ &\leq \gamma \max_{s', a'} |Q_t(s', a')| + \gamma \max_{s', a'} |Q_t(s', a')| \\ &= 2\gamma \max_{s', a'} |Q_t(s', a')| = 2\gamma \|Q_t\|_\infty \end{aligned}$$

$\therefore$ we get $\mathbb{E}[\boldsymbol{w}_t^2(s)|\mathbb{F}] \leq 4\gamma^2 \|Q_t\|_\infty^2$ ($\therefore$ by comparison with theorem 1, $K_1 = 0$, $K_2 = 4\gamma^2$ and the norm used here is the sup-norm).

So all the conditions for the theorem 1 are satisfied and hence Q-learning converges to Q^*. The algorithm can be viewed as a stochastic formulation of the value iteration algorithm we studied earlier in the course. $\square$

2 Expected SARSA Algorithm

Expected SARSA is another TD control algorithm. The Q values are updated by considering the expected Q values from the next state $S^{'}$ over all possible actions $a^{'}$, which is essentially the value of that state $S^{'}$. It's update rule is given below:

$$\begin{aligned} Q_t(S_t, A_t) &= (1-\alpha)Q_{t-1}(S_t, A_t) + \alpha[R_{t+1} + \gamma V_{t-1}(S_{t+1})] \\ &= (1-\alpha)Q_{t-1}(S_t, A_t) + \alpha[R_{t+1} + \gamma \sum_{a^{'} \in A} \pi(a|S_{t+1}) Q_{t+1}(S_{t+1}, a^{'})] \end{aligned} \tag{3}$$

Contrast this with SARSA, where the update rule considers the Q-value corresponding to only one action chosen by our policy (e.g.ϵ-greedy), or Q-learning, where only the greedily chosen action is considered for updating Q values. It is straight forward to prove the convergence of expected SARSA by appropriately defining a γ-contraction operator H. (Hint: Instead of $\max_{a'}$ in equation (1), use $\mathbb{E}_{a'}$).

Expected SARSA is an on-policy algorithm, because the behavior policy and the estimation policy are same(?[1]). Expected SARSA reduces variance because of the expectation over all possible actions. In contrast Q-learning is very aggressive and hence its variance is high, which can slow convergence. (Refer the slides for illustration of convergence of different algorithms on the cliff walking problem)

3 Double Q Learning

3.1 Maximization Bias

In Q-learning,the target policy is greedy policy given the current action values,which is defined with a max. In SARSA,the policy is often ϵ-greedy,which involves a maximum operation as well.In these algorithms,a maximum over estimated values is used implicitly as an estimate of the maximum value,which can lead to a significant positive bias. For example consider a single state s where there are many actions a whose true values, $q(s, a) = 0$.But the estimated values $Q(s, a)$ are uncertain and distributed some above and some below zero.The maximum of the true values is zero,but the maximum of the estimates is positive, which results in a positive bias.We call this as maximization bias.

[1] why Expected SARSA is on-policy algorithm?

3.2 Double Q-Learning

Double Q-learning is a heuristic algorithm proposed to counter the maximization bias issue. Double Q-learning stores two action value functions Q_1 and Q_2. Each Q function is updated with a value from other Q function of the next state.It is important that both Q function learn from separate set of experiences but to select an action to perform one can use both value functions. Therefore this algorithm is not less data efficient than Q-learning. Average of two Q values for each action is computed and ϵ-greedy exploration is performed. The algorithm for double Q-learning is given below.

Algorithm 2: Double Q-Learning

Initialize $Q_1(s, a)$ and $Q_2(s, a)$, $\forall s \in \mathcal{S}$, $a \in \mathcal{A}(s)$, arbitrarily.
Initialize $Q_1(terminal - state, .) = Q_2(terminal - state, .) = 0$.
foreach *episode* **do**
  Choose S (start state)
  repeat
    Choose action A from S using policy derived from Q_1 and Q_2 (e.g., ϵ-greedy in $Q_1 + Q_2$)
    Take action A, observe next state $S^{'}$ and the reward R
    if *Probability rand()* > 0.5 **then**
      $Q_1(S, A) \leftarrow Q_1(S, A) + \alpha(R + \gamma Q_2(S^{'}, \max_{a'} Q_1(S^{'}, a^{'}) - Q_1(S, A))$
    end
    else
      $Q_2(S, A) \leftarrow Q_2(S, A) + \alpha(R + \gamma Q_1(S^{'}, \max_{a'} Q_2(S^{'}, a^{'}) - Q_2(S, A))$
    end
    $S \leftarrow S^{'}$;
  until *S is terminal state*;
end

In the next class, we will see the multi step bootstrapping algorithm $TD(\lambda)$.

References

[1] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar *Foundations of Machine Learning*

[2] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*