EE675 - Introduction to Reinforcement Learning

Lecture 18: SARSA and Q-Learning

27th March 2023

Lecturer: Subrahmanya Swamy Peruru *Scribes: Lavesh Mangal,Gaurang Dangayach*

1 Recap

In last lecture, we looked at the algorithms for TD Control, SARSA, Expected SARSA and Q-Learning. We give a brief summary as in previous lecture of these methods.

1.1 TD Control

TD based control algorithm is an algorithm used to find optimal policy based on difference between expected and actual reward, Previously in DP based methods we've seen the generalized policy iteration where we can approximately compute V_π and improve on the current policy.

Given a policy π, we can estimate q_π using DP by the equation

$$q_\pi(s,a) = R_s^a + \gamma \Sigma_{s'} V_\pi(s') P_{ss'}^a = R_s^a + \gamma \Sigma_{s'} \Sigma_a \pi(a|s') q_\pi(s'|a') P_{ss'}^a \tag{1}$$

We don't know the dynamics of the MDP since we are working in model-free setting. So we work with Q-functions instead of value functions.

1.2 SARSA

The SARSA (State-Action-Reward-State-Action) algorithm is a TD-based control algorithm that learns an action-value function $Q(s,a)$ for each state-action pair.

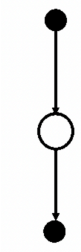
Recall that the TD target for SARSA can be expressed as:

$$\text{TD target} = R + \gamma Q(s',a') \tag{2}$$

Where the TD target is the target value for the current state-action pair (s,a), R is the immediate reward obtained from the current state-action pair, γ is the discount factor, s' is the next state, and a' is the next action that the agent will take based on the policy being followed.

SARSA finds an optimal policy by updating the value function of a state-action pair based on the rewards received for a sequence of state-action pairs. SARSA finds an optimal policy by updating $Q(s,a)$ based on the values of the current state-action (s,a) and the next state-action (s',a') as follows:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(R + \gamma Q(s',a') - Q(s,a)\right) \tag{3}$$



Sarsa

SARSA is an on-policy algorithm, which means that it updates the action-value function and chooses the next action using the same policy. By iteratively updating the Q-value function using the TD target i.e. the expected future reward, SARSA learns to choose the best action in a given state, under the current policy. The algorithm is given as follows:

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

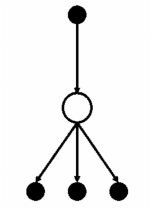
Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop 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

1.3 Expected SARSA

SARSA and Expected SARSA differ in choice of next action, where SARSA chooses the next action by the current policy whereas Expected SARSA averages over all possible actions by using the expected value of action .
The Expected SARSA algorithm updates the estimate of the action-value function $Q(s, a)$ using the following update rule:

$$Q(s, a) \leftarrow Q(s, a) + \alpha(r + \gamma \sum_{a'} \pi(a'|s')Q(s', a') - Q(s, a)) \tag{4}$$


Expected Sarsa

1.4 Q-Learning

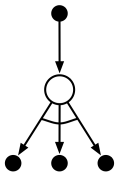
Q-Learning is an off-policy, sample based method. In Q-learning, we learn the optimal action-value function $Q^*(s, a)$ by updating the action-values based on sample transitions within the environment, unlike value iteration which keeps a complete history of all states and actions over all iterations. The updates are processed according to the following equation:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left(r + \gamma \max_{a'} Q(s', a') - Q(s, a)\right) \tag{5}$$

where $Q(s, a)$ is the action-value of taking action a in state s at time t, α is the learning rate , r is the reward received at time $t + 1$, γ is the discount factor, and the action a takes s to the state s'.

The Q-value of a state-action pair is updated using the difference between Q-learning target and current estimate of that pair:

$$\text{Q-learning Target} = r + \gamma \max_{a'} Q(s', a') \tag{6}$$



Q-learning

To generate the sample transitions, the agent is made to interact with the environment through a behavior policy. The behavior policy should be such that it covers all possible state-action pairs. An example of such a behavior policy is an ϵ-greedy policy which takes the action with the highest Q-value with probability $(1 - \epsilon)$ and chooses a random action with probability ϵ.

Note that this behavior policy is distinct from the target policy which is the policy being learned. In Q-Learning, the target policy is the greedy policy that always chooses the action with maximal Q-value. Using a different behavior and target may allow for better exploration and is known as off-policy learning.

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

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\textit{terminal}, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop 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

Here, S is the space of all possible states, A is the set of all possible actions, and $Q(s, a)$ is the Q-value for state s and action a.

2 Maximisation Bias

Many of the algorithms that we used contained the use of greedy policy which involves a maximisation operation. This can lead to a significant positive bias.

To understand this with the help of an example, consider 10 random variables $\{X_i\}_{i=1}^{10}$ such that each X_i has mean $\mathbb{E}[X_i] = 0$. Suppose we estimate the means$\mathbb{E}[X_i]$ with $\hat{\mu}_1, \hat{\mu}_2, \ldots \hat{\mu}_{10}$ by taking 100 samples of each random variable. Notice that $\max_{i \in [10]}(\hat{\mu}_i)$ is most likely positive even though we sampled from mean 0 RVs.

This problem arises due to the fact that we are using the same samples for determining the best arm (in case of bandits problem) and estimating mean. Hence, to solve this we consider two sets of samples, one for finding out the best arm and another set for estimating the mean of that arm. This is the idea of *double learning*.

In the example we considered, to solve the bias, we first take equal number of samples from all of the random variables, say 50. Let the obtained sample means be $\nu_1, \nu_2, \nu_3 \ldots$. Then, let

$$i^* = \arg\max_{i \in [10]} \nu_i$$

Then, we further sample say 50 values from X_{i^*} and generate the sample mean $\hat{\mu}^*$ from this sample.

3 Double Q-Learning

In Q-learning, greedy policy is involved and hence there is a maximization bias to it. To avoid this, we take into account the idea of 'double learning' as explained above. It involves taking two estimates of Q-value and updating them randomly and symmetrically. A detailed algorithm is as follows:-

Double Q-learning, for estimating $Q_1 \approx Q_2 \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q_1(s, a)$ and $Q_2(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, such that $Q(\textit{terminal}, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using the policy ε-greedy in $Q_1 + Q_2$
    Take action A, observe R, S'
    With 0.5 probabilility:
      $Q_1(S, A) \leftarrow Q_1(S, A) + \alpha\Big(R + \gamma Q_2\big(S', \arg\max_a Q_1(S', a)\big) - Q_1(S, A)\Big)$
    else:
      $Q_2(S, A) \leftarrow Q_2(S, A) + \alpha\Big(R + \gamma Q_1\big(S', \arg\max_a Q_2(S', a)\big) - Q_2(S, A)\Big)$
    $S \leftarrow S'$
  until S is terminal

The material is primarily based on the content from [1]

References

[1] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.