Reinforcement Learning With Emma Brunskill

Yaro Kazakov

September 26, 2024

Abstract

These are study notes to prepare myself for PhD applications.

1 Lecture 6 - Monte Carlo and TD Control

2 Control in MC and TD

Find how good a policy is and learn the best policy having no model of the environment.

Our assumption for this lecture is that MDP model is unknown but can be sampled.

MDP model is known but it is computationally infeasible.

1. On-policy learning
   (a) Direct Experience
   (b) Learn to estimate and evaluate a policy using experience obtained from following that policy
2. Off-policy learning
   (a) Learn to estimate and evaluate a policy using experience gathered from following a different policy

Recall that the standard Policy Iteration required the model of the environment (dynamics + reward). What can we do instead?:

We can try on-policy MC control:

1. Initialize G(s,a) = 0, N(s,a) = 0, $Q^{\pi}(s,a) = 0$
2. Using policy π sample episode $i = s_1, a_1, r_2, s_2, ...$
3. $G_{i,t} = r_{i,t} + \gamma * r_{i,t+1} + \gamma^2 * r_{i,t+2}...$
4. For each state,action visited in episode i:
   (a) For first or every time t that (s,a) is visited in episode i
      i. $N(s,a) = N(s,a) + 1, G(s,a) = G(s,a)G_{i,t}$
      ii. Update estimate $Q^{\pi}(s,a) = G(s,a)/N(s,a)$
      i. Update new policy $\pi_{i+1}(s) = argmaxQ^{\pi_i}(s,a)$

The problem is that **There is NO Exploration** and the algorithm will probably get stuck with a greedy action on every step. Introduce Exploration through:

1. $\epsilon-$greedy algos
2. Optimistic initialization

In general we want to:

1. Try all (s,a) pairs but then follow π
2. We want to ensure resulting Q_π is good enough so that policy improvement is a monotonic operator.

Epsilon-greedy ensures exploration and essentially adds randomness when selecting an action at s.

An alternative to on-policy MC control is TD control.

TD is slightly different.

We update tuples based on the next tuple only. We don't have to wait until the end of episode to calculate the rewards. We also **don't have to wait until the end to CHANGE HOW we're acting in the world**.

3 SARSA

SARSA (State Action reward next state action) algorithm:

1. Set initial $\epsilon-$greedy policy randomly, t= 0, initial state = s_0
2. Take a_t from π_t. Sample action from policy
3. Observe (s_t, r_{t+1})
4. loop
   (a) Then Take another action a_{t+1} from $\pi(s_{t+1})$
   (b) Observe (r_{t+1}, s_{t+2})
   (c) Update Q given (s_t, a_t, r_t, s_{t+1}, a_{t+1}).

   $Q(s_t, a_t) < -Q(s_t, a_t) + (\alpha)(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$ Q-function is not just for one policy.
   (d) Perform Policy Improvement:
   (e) $\pi(s_t) = argmaxQ(s_t, a_t)$ with prob 1- epsilon. Else random.
   (f) t= t + 1
5. End Loop

In essence:

1. We initialize the policy π.
   (a) Repeat
   (b) Policy evaluation: compute Q^π using temporal difference updating with epsilon greedy
   (c) Policy improvement: Same as Monte carlo policy improvement, set π to epsilon-greedy.

Note how similar this is to Q-learning.

4 Convergence of SARSA

SARSA for finite-state and finite-action MDPs converges to the optimal action-value, $Q(s,a) \rightarrow Q^*(s,a)$, under the following conditions:

1. The policy sequence $\pi_t(a|s)$ satisfies the condition of GLIE (Greedy in the Limit with Infinite Exploration).

2. The step-sizes α_t satisfy the Robbins-Monro sequence such that:

$$\sum_{t=1}^{\infty} \alpha_t = \infty \quad \text{and} \quad \sum_{t=1}^{\infty} \alpha_t^2 < \infty$$

For example $\alpha_t = 1/T$ satisfies the above condition.

5 Q-learning

The only difference between SARSA and Q-learning is in the $Q(s_t, a_t)$.

$Q(s_t, a_t) < -Q(s_t, a_t) + (1-\alpha)(r_t + \gamma max_{a_{t+1}}(Q(s_{t+1}, a_{t+1})) - Q(s_t, a_t))$.

SARSA is updating on policy (generally you want to do Q-learning). Sutton shows with the cliff walk example that sometimes when you can have loads of negative outcomes. Max() can be too optimistic instead of realistic.

Sutton also shows how SARSA can do better in early samples. SARSA is realistic as opposed to optimistic.

Convergence?: depends on the learning rate α

Q-learning is OFF-POLICY TD Control and SARSA is ON-POLICY TD Control

SARSA is on-policy because it updates the Q-value based on the action actually taken by the agent following its current policy. The update rule is:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma Q(s', a') - Q(s,a)\right)$$

where a' is the next action chosen by the current policy.

Q-learning is off-policy because it updates the Q-value using the maximum future reward, regardless of the action taken. The update rule is:

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

It evaluates the optimal policy even when following an exploratory policy.