Chapter 11

Reinforcement Learning

Contents

11.1 TD Learning

Algorithm 11.1 Tabular TD(0) for estimating v_π

function TD(π: the policy to be evaluated)
    Initialize $V(s)$ arbitrarily (e.g., V(s) = 0, $\forall s \in S^+$)
end function

- Model-free
- Off-policy
- Discrete action and state spaces

11.2 SARSA

- SARSA works by learning a state-action value function rather than a state value function like TD learning.
- SARSA explore the transitionsfrom one state-action pair to another state-action pair and learns the state-action value function.
- In on-policy learning, the optimal value function is learned from actions taken using the current policy.

- Update rule:

$$Q(S_t, A_t) = \tag{11.1}$$

- As in all on-policy methods, Q_n is continually esimated for the policy π, while the policy π is updated using an ϵ-greedy policy.

Algorithm 11.2 ϵ-Greedy Policy

$p \leftarrow$ Random
if $p < \epsilon$ **then**
    pull random action
else
    pull current-best action
end if

Algorithm 11.3 SARSA (on-policy TD control)

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\epsilon > 0$
Initialize $Q(s, a)$, for all $s \in S^+$, $a \in A(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$
loopfor each episode:
    Initialize S
    Choose A from S using policy derived from Q (e.g., ϵ-greedy)
end loop

- Model-free.
- On-policy.
- Discrete action and state spaces.

11.3 Q-Learning

- Q-Learning is very similar to SARSA. The major difference lies in the update rule of the Q-function.

11.4 Deep Q-Learning

- For tasks with continuous state space, updating the Q-function for all state-action pairs can be computationally inefficient and infeasible.
- Rather than using value learning to directly find the optimal Q-function, a function estimator can be used to estimate the optimal Q-function.

- ANNs are effective function estimators. Deep neural network (DNN) can be used to estimate the Q-function for each state-action pairs.
- DQN is trained using batch stochastic gradient updates and experience replay. Experience replay can interact with the environment to generate training data for the DQN.
- Experience replay is a technique where the agent stores a subset of its experiences $< s, a, r, s' >$ in a memory buffer and samples from this buffer to update the Q-function.
- Experience replay selects an ϵ-greedy action from the current state, executes it in the environment, and gets back a reward and the next state.
- If DQN was trained with single samples, each sample and the corresponding gradients
- ...
- Model-free.
- Off-policy.
- Continuous state space.
- Discrete action space.

11.5 Policy Gradient Methods

- Policy gradient methods learn a parameterized policy than can select actions without consulting a value function. The parameters of the policy are called policy weights.
- Policy gradient methods are methods for learning the policy weights using the gradient of some performance measure with respect to the policy weights.
- Policy gradient methods seek to maximize performance and so the policy weights are updated using gradient ascent.

11.6 Actor-Critic

- Actor-Critic method is a TD version of policy gradient. It sues two neural networks, one actor network and one critic network.
- The actor network decides what action should be taken and the critic network informs the actor network how good was the action and how it should update to improve.
- The learning of the actor network is based on policy gradient approach.
-