Weekly report of lessons

Name: Suryam Arnav Kalra
Roll No: 19CS30050
The week: 1st November 2021 to 5th November 2021

The topics covered:

- Learning with a critic
- A simple learner: K-arm bandit
- Nondeterministic: K-arm bandit
- More complex environment
- Learning a Markov Decision Process
- Learning a policy and optimal policy
- Model based learning
- Value iteration algorithm
- Policy iteration algorithm
- Temporal difference algorithm
- Deterministic environment
- Nondeterministic environment
- Large number of states and actions

Summary topic wise:

- Learning with a critic: A critic is not a teacher but only evaluates the past performances (does not tell us what to do). A learner (agent) interacts with the environment and gets a reward/penalty and tries to reach a goal using a series of actions.
- A simple learner: K-arm bandit: The action is to choose and pull one of the K levers, and win a reward. The task is to decide which lever to pull to maximize the reward. If this were supervised learning, then the teacher would tell us the correct class, namely, the lever leading to maximum earning. In this case of reinforcement learning, we can only try different levers and keep track of the best.
- Nondeterministic: K-arm bandit: The amount of the reward is defined by the probability distribution $p(r \mid a)$. Let $Q_t(a)$ be the estimate of the value of action a at time t. It is an average of all rewards received when action a was chosen before time t. Delta Rule: $Q_{t+1}(a) \leftarrow Q_t(a) + \eta[r_{t+1}(a) - Q_t(a)]$. Choose $a* \ if \ Q(a*) = max\ Q(a)$
- More Complex environment: There could be multiple states of the environment, action also affects the next state. Presence of nondeterministic rewards and delay in the rewards.
- Learning a MDP: The reward and next state are sampled from their respective probability distributions, $p(r_{t+1}|s_t, a_t)$ and $p(s_{t+1}|s_t, a_t)$. The sequence of actions from the start to terminal state is episode or trial. A mapping from the states of the environment to actions is a policy.
- Learning a policy and optimal policy: For each policy π, there is a $V^{\pi}(s_t)$, and we want to find the optimal policy π∗ such that $V^*(s_t) = max\ V^{\pi}(s_t)$. $V^*(s_t) = max_{a_t}\ E[r_{t+1}] + \gamma \sum_{s_{t+1}} p(s_{t+1}|s_t, a_t)V^*(s_{t+1})$. $To\ get$ $\pi^*(s_t)$: Choose a* providing $V^*(s_t)$ OR choose a* if Q*(s_t, a*) = max Q*(s_t, a)
- Model based learning: We can directly solve for the optimal value function and policy using a dynamic programming algorithm using two approaches (value iteration and policy iteration algorithm).
- Value iteration algorithm: To find the optimal policy, we can use the optimal value function, and this is an iterative algorithm that has been shown to converge to the correct

V $*$ values. We say that the values converged if the maximum value difference between two iterations is less than a certain threshold δ.

- Policy iteration algorithm: In policy iteration, we store and update the policy rather than doing this indirectly over the values. The idea is to start with a policy and improve it repeatedly until there is no change. The value function can be calculated by solving for the linear equations. This step is guaranteed to improve the policy, and when no improvement is possible, the policy is guaranteed to be optimal
- Temporal difference algorithm: When we explore and get to see the value of the next state and reward, we use this information to update the value of the temporal current state. These algorithms are called temporal difference algorithms difference because what we do is look at the difference between our current estimate of the value of a state and the discounted value of the next state and the reward received.
- Deterministic environment: For any state-action pair, there is a single reward and next state possible $Q(s_t, a_t) = r_{t+1} + \gamma\, max_{a_{t+1}}\, Q(s_{t+1}, a_{t+1})$. We then update the value of previous action as $Q'(s_t, a_t) \leftarrow r_{t+1} + \gamma\, max_{a_{t+1}}\, Q'(s_{t+1}, a_{t+1})$. The later updates are more reliable and converge when all pairs are stable.
- Nondeterministic environment: In this case, the reward or next state varies for a state-action pair and we can keep a running average of values Q-learning: $Q'(s_t, a_t) \leftarrow Q'(s_t, a_t) + \eta(r_{t+1} + \gamma\, max_{a_{t+1}}\, Q'(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$. We can also choose the next action randomly using an ∈-greedy sampling: actions providing higher values would have higher probability.
- Large number of states and actions: In this case the result is not feasible through tabular search and we can use regression to predict Q values given the current value, reward and next state. It requires supervisory information or labels.

Concepts challenging to comprehend:

Iteration algorithms and policy learning are a little bit challenging to comprehend.

Interesting and exciting concepts:

Reinforcement learning as a whole is quite interesting and exciting to learn.

Concepts not understood:

After going through the book and the video lectures the concepts are understood.

Any novel idea of yours out of the lessons:

I think that since in reinforcement learning we are talking about states and rewards, we can model it using finite state automata (DFA for a deterministic environment or an NFA for a nondeterministic environment) or maybe using mealy/moore machines. We can store the states as nodes in the automata and the termination state would be the final states in the automaton. We can then simulate the automaton and update the states and actions using the formulas mentioned above and maybe arrive at an optimal solution faster.