Machine Learning (CS60050) – Weekly Report

Kaushal Banthia (19CS10039)

Week 12: 3rd – 5th November, 2021

Topics Covered:

- Learning with a critic
- K-Arm Bandit
- Markov Decision Process (MDP)
- Model Based Learning
- Temporal difference algorithm
- Deterministic and Non-Deterministic Environment

Summary (Topic Wise):

- <u>Learning with a critic</u>

  - A critic is not a teacher as it does not tell us what to do. It only evaluates past performances. Also, the feedback is scarce and delayed
  - The learner (Agent) interacts with an environment (at some state), which may change its state. It gets a reward / penalty sometimes (by the critic) for the steps it takes. In the end, it tries to reach a goal (a state).
  - The learner learns a series of actions to reach the goal. It does this by maximizing the total rewards from any state.

- <u>K-Arm Bandit</u>

  - Simple Learner:
    - There are K-levers in a simple machine with one state, where $Q(a)$ is the value of action a. Initially $Q(a) = 0 \ \forall \ a$. Store r_a after each $a, Q(a) = r_0$
    - The action (a) is to pull a lever to win a reward (r_0).
    - The task is to decide which lever to pull to maximize the reward.
    - Supervised Learning works to give the correct class, namely, the lever leading to maximum earning labeled by a teacher.
    - Reinforcement learning works by trying out different levers and keeping track of the best.
    - Choose $a^* \ if \ Q(a^*) = \max_a Q(a)$
  - Non deterministic:
    - Here, the action(a) is to pull a lever to win a reward r with probability $p(r|a)$
    - The task is to decide which lever to pull to maximize the reward.
    - Reinforcement learning tries different levers and keeps track of the best.
    - $Q_t(a)$: Estimate of the Value of action a at time $t \rightarrow$ an average of all the past rewards when a was chosen.
    - Delta Rule: $Q_{t+1}(a) \leftarrow Q_t(a) + \eta[r_{t+1}(a) - Q_t(a)]$, where η is the learning factor that decreases with time.
    - Choose $a^* \ if \ Q(a^*) = \max_a Q(a)$
  - In a more complex environment, there are multiple states and the action also affects the next state. The agent senses state (s_{t+1}) after an action (a_t) and may

get a reward (r_{t+1}). The non-deterministic reward is $p(r|s_i, a_j)$ and we have to learn $Q(s_i, a_j)$, which is the value of taking action a_j in state s_i. The rewards maybe delayed and thus, there is a need for the immediate estimation of a prospective reward.

- <u>Markov Decision Process (MDP)</u>

  - An agent takes an action $a_t \in A$ at state $s_i \in S$ at discrete time t
  - Its goal is reaching a terminal state and then it remains there for any action with probability 1 without any reward.
  - The sequence of actions from the start to the terminal state is called an episode or a trial, while the mapping from the states of the environment to actions $\pi: S \to A$ is called a policy.
  - Value of a policy π: $V^{\pi}(s_t) = E[\, r_{t+1} + r_{t+2} + \, \dots \, + r_{t+T}]$
  - Discounted value with infinite horizon is $V^{\pi}(s_t) = E[\, r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots \,]$
  - Optimal policy π^*: $V^*(s_t) = \max\limits_{\pi}[V^{\pi}(s_t)]$
  - As an alternative to this, we can learn $Q(s_t, a_t)$, which is value at state with action.
  - $V^*(s_t) = \max\limits_{a}[Q(s_t, a_t)] = \max\limits_{a} E[\, r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots \,]$
    $= \max\limits_{a} E[\, r_{t+1} + \gamma V^{\pi}(s_{t+1})]$
  - Optimal Policy is

$$V^*(s_t) = \max_{a_t}\left(E[r_{t+1}] + \gamma \sum_{s_{t+1}} P(s_{t+1}|s_t, a_t) V^*(s_{t+1})\right)$$

$$Q^*(s_t, a_t) = \max_{a_t}\left(E[r_{t+1}] + \gamma \sum_{s_{t+1}} P(s_{t+1}|s_t, a_t) \max_{a_{t+1}} (Q^*(s_{t+1}, a_{t+1}))\right)$$

  $\pi^*(s_t)$: Choose a^*, providing $V^*(s_t)$ or $\pi^*(s_t)$: Choose a^*, if $Q^*(s_t, a^*) = \max\limits_{a}(s_t, a)$

- <u>Model Based Learning</u>

  - The models: $p(r_{t+1}|s_t, a_t)$ *and* $p(a_{t+1}|s_t, a_t)$ are known.
  - Directly solve for the optimal value function and policy using dynamic programming. There are two approaches for this, namely, Value iteration algorithm and policy iteration algorithm.

    - Value Iteration Algorithm:

```
Initialize V(s) to arbitrary values
    Repeat
        For all s ∈ S
            For all a ∈ A
            Q(s,a) = max_a(E[r|s,a] + γ Σ_s' P(s'|s,a)V(s')
                V(s) ← max_a Q(s,a)
            π*(s) = argmax_a( E[r|s,a] + γ Σ_s' P(s'|s,a)V(s')
    Until V(s) converge
```

    - Policy Iteration Algorithm (Update policy directly from intermediate values):

```
Initialize π to arbitrary values
    Repeat
        π ← π'
        Compute value using π
```

$V^{\pi}(s) = \max_a (E[r|s,\pi(s)] + \gamma \sum_{s'} P(s'|s,\pi(s)) V^{\pi}(s'))$

Improve the policy at each stage

$\pi(s) = argmax_a (E[r|s,a] + \gamma \sum_{s'} P(s'|s,a) V(s'))$

Until $\pi = \pi'$

- Temporal difference algorithm

  - No prior knowledge of model. No $p(r_{t+1}|s_t, a_t)$ *and* $p(a_{t+1}|s_t, a_t)$ required.
  - It requires exploration of the environment to query the model to see the value of the next state and reward.
  - Uses this information to update the value of the current state
  - It is called as temporal difference algorithm, as it examines the difference between the current estimate of the value and the discounted value of the next state and the reward received.

- Deterministic and Non-Deterministic Environment

  - Deterministic Environment:
    - For any state-action (s_t, a_t) power, a single reward (r_{t+1}) and state transition (s_{t+1}) are possible. $Q(s_t, a_t) = (r_{t+1} + \gamma \max_{a_{t+1}} (Q(s_{t+1}, a_{t+1})))$
    - Update at every exploration by adding immediate reward with discounted estimate of the next state-action pair.
    - Later updates are more reliable
    - Converges when all the pairs are stable (little changes with iteration).

  - Nondeterministic Environment:
    - Varying reward or next state for a state-action pair.
    - Keeps a running average of values (Q-Learning)
    - Choose next action randomly (ε-Greedy sampling)
    - Sample an action uniformly with a probability ε initially.
    - Actions providing higher values would have higher probability.
    - Softmax over Q-values done using a temperature variable (T) and at every iteration, T increases, favoring higher Q-values.
    - Large number of states and actions are not feasible through tabular search. Use of regression is done to predict Q-values, given the current value, the reward and the next state. This requires supervisory information or labels.

Concepts Challenging to Comprehend: None yet.

Interesting and Exciting Concepts: K-Arm Bandit and Temporal difference algorithm.

Concepts not understood: None yet.

A novel idea: Instead of just choosing the best lever for the K-Arm Bandit, we could also keep a track of the second-best lever, third-best lever, ..., and so on, for the Bandit. This can be done by keeping note of the $Q(a)$ values for all the levers and then sorting them in descending order. This way, if the first-best output might not have been the one that we had desired (or due to any other such reasons), we can try out the second-best lever, third-best lever, ... and so on. This gives us more flexibility and control over our model!