CSE585 Term Project

Merve Noyan

May 2021

1 Introduction to Reinforcement Learning

Reinforcement learning is a domain of machine learning that is concerned with decision making given an environment. Reinforcement learning problems consist of agent(s) and an environment they take action in. Agents take an action, observe the impact of that action on that environment and try to maximize their gains based on it. Unlike supervised learning, reinforcement learning does not need labelled inputs. Some of the reinforcement learning problems require feature recognition from existing data, (e.g. actions of an autonomous car based on image recognition) however, the goal is to come up with a decision process, also called "a policy", a function that maps an observation obtained from its environment to an action.

Multi-Agent Reinforcement Learning (MARL) is a problem that consists of an environment with multiple agents, where they learn simultaneously. It is a domain that heavily utilizes machine learning, game theory and optimization. Just like reinforcement learning, one can control multiple agents where each agent can learn and come up with an optimal policy to maximize their rewards, or they can share one policy by communicating each other synchronously. In this problem, each agent use a different policy to maximize their own reward instead of sharing one policy.

2 Value Based Reinforcement Learning

Value based reinforcement learning methods involve learning values of state-actions pairs. Agents needs to explore each and every state and take action and observe these values and then exploit. The policies of an agent in value-based reinforcement learning methods is to optimize the value itself. This function is called the V-value function and is defined as follows,

$$\begin{aligned} V^{\pi}(s) &\equiv r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ... \\ &\equiv \sum_{i=0}^{\infty} \gamma^i r_{t+i} \end{aligned}$$

$r_t, r_{t+1}, \ldots$ are rewards gained by the agent, following policy π starting at state s. Gamma is the discount factor, a multiplier of how the agent considers the importance of the future rewards. V-value function corresponds to the expected discounted reward when agent is in the state s and it follows the policy pi.

3 Q-Learning

Q-learning is based on agent taking an action that has the highest Q-value for that state. The V*(s) is the maximum of the Q*(s,a) over all the Q values

of the actions an agent can take in each possible state s. Q-value is expected discounted gains of an agent in a state s and action a taken. Each state holds multiple Q-values for each of the action that can be taken for that state. The relationship between Q*(s,a) and V*(s) is as follows:

$$V*(s) = max_a Q*(s,a) \tag{1}$$

Bellman optimality equation provides a dynamic programming approach to calculate Q-values recursively. .

$$Q^*(s,a) \;=\; \Big[r_{t+1}+\gamma \max_{a'} Q^*(s_{t+1},a')\Big| s_t = s, a_t = a\Big] \;=\; \sum_{s'} {}^{a}_{ss'}\big({}^{a}_{ss'}+\gamma \max_{a'} Q^*(s',a')\big)$$

For environments with discrete state space, Q-tables are used. Q-tables hold each Q-value for every state-action pair. When the state space is too large or continuous, deep Q-networks are used to approximate Q-values for a given state. Before every training, Q-table is either randomly initialized or initialized with zeros, and in every episode, Q-table is updated. For Q-table updates, exploration is required. Pseudocode for the Q-table learning for multi-agent reinforcement learning problem is given in the section "Environment and Agents".

4 Problem

4.1 Environment

The environment is a square-shaped grid world that consist of 64 grids. By the start of the training, runner starts from the block on the top left and the chasers start from two blocks at bottom right.

The chasers get one point when they get two blocks close to the runner, two points when they get one block close to the runner, 10 points if they catch the runner. The runner gets 2 points if it is at least 5 blocks away from the runners and -10 points if it gets caught. The game is played for 1000 episodes, 100 turns in each episode, learning rate is 0.1 and gamma is 0.9.

4.2 Agents and Q-table Learning

Each agent has a separate Q-table and the agents take their actions simultaneously, and the Q-tables are updated for the state they land in, rather than sharing the same Q-table. Q-table is hold as a dictionary, each key in the dictionary corresponds to states and the values are the Q-values of each action. State is hold for combination of three values, for chasers, state consists of a tuple (x coordinate of chaser, y coordinate of chaser, the Manhattan distance between

the runner and the chaser), and for the runner, state is (x coordinate of runner, y coordinate of runner, min(distance between runner and the first chaser, distance between runner and the second chaser). The value of the dictionary is a list of Q-values for each action: up, right, down and left. Thus, for every state, action values are kept and used when the agent lands in that state. Instead of defining an epsilon value for exploration-exploitation trade-off, first 10 turns in each episode are solely dedicated for the exploration.

Algorithm 1 Q-table Learning

```
for each agent do
    initialize Q-table with random values
end for
for each episode do
    Initialize agents in their starting blocks
    for each exploration steps do
        for each agent do
            Take a random step
        end for
        Calculate cumulative future reward
        Calculate Q-values
        Update Q-table
    end for
    for rest of the episodes do
        for each agent do
            In each state, take the action with the highest Q-value
        end for
        Calculate cumulative future reward
        Calculate Q-values
        Update Q-table
    end for
end for
```

4.3 Findings

The code is ran for four times for different discount rate and exploration steps configurations. The below table shows the configurations and the wins of both sides.

Exploration Steps	γ	Runner Win	Chaser Win
10	0.9	21	79
10	0.1	19	81
0	0.9	13	87
0	0.1	12	88

Having exploration steps is necessary and has more effect on the wins compared to the discount rate. On the other side as the game begins, runner starts

gaining rewards and refuses to move unless the chasers move, and discount rate has a dramatic effect on runner's wins as well.