Agent-based Reinforcement Learning in Colonel Blotto

Joseph Christian G. Noel

1 Introduction

In this paper we compare a reinforcement learning agent against a baseline random agent. It is shown that a reinforcement learning agent using the Q-Learning method handily beats basic baseline agents in Colonel Blotto.

2 Colonel Blotto

2.1 Definition

Colonel Blotto is a constant-sum game where two or more players distribute resources (or troops) over several fronts in a battlefield. A player wins or loses a front depending on whether they have allocated more or less resources to it than their opponent has, and the winner of the game is who has won the most fronts.

2.2 Strategy

3 Reinforcement Learning

3.1 Definition

Reinforcement Learning (RL) is a form of agent-based modeling. In RL an agent learns by performing actions which changes the state of an environment. After each action, the agent may also receive a ”reward” whose value depends on how close the agent is to what it wants to achieve. The goal for the agent is to maximize the cumulative reward that it receives over all the actions that it takes. After a series of actions, the agent eventually reaches a goal state or terminal state, which signifies the end of an episode. The environment is then reset and the agent starts again from an initial state and the process then repeats itself.

Formally, an RL model is a set of states S, a set of actions A, and transition rules between states depending on the action taken. For state $s \in S$ at time t, an agent performs an action $a \in A$, moves to a new state s' and receives a reward r_t. The goal of the agent is to maximize the expected reward Rt,

$$R_t = \sum_{k=0}^{\infty} \lambda^k r_{t+k} \tag{1}$$

where $0 \leq \gamma \leq 1$ is a discount factor for handling infinite horizons.

$\pi(s, a)$ is a probability mapping of an agent taking action a while in state s. A proper policy is one where there is a non-zero probability of reaching a terminal state. There is always an optimal policy π^* that is better than or equal to all other policies when it comes maximizing the cumulative rewards.

3.2 Markov Decision Process

A common formulation of the Reinforcement Learning problem is as a Markov Decision Process (MDP). In this class of RL problems, the information from the history of all states, actions, and rewards before time t is encapsulate in the current state at time t. This is the Markov property, and tasks which exhibit this property are called Markov decision processes. Formally, the Markov property states that for transition probability function Pr,

$$Pr(s_{t+1}, r_{t+1}|s_t, a_t) = Pr(s_{t+1}, r_{t+1}|s_t, a_t, r_t, s_{t-1}, a_{t-1}, r_{t-1}, ..., s_0, a_0, r_0) \tag{2}$$

3.3 Value Function

The state-action value function Q is the estimate of the expected rewards the agent will receive by being at state s, at time t, taking action a, and then following policy π from $t + 1$ onwards.

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}(R_t|s_t, a_t) \tag{3}$$

The reinforcement learning problem can therefore be reduced to finding the optimal policy π^* by simply choosing the action a that maximizes $Q(s, a)$ for all $s \in S$

$$\pi^*(s) = \arg\max_a Q(s, a) \tag{4}$$

3.4 Q-Learning

Q-Learning is one of the basic reinforcement learning methods. It approximates the state-action value function $Q(s, a)$ using the Bellman equations, which allows it to define $Q(s, a)$ in a recursive manner. At each time step Q-Learning approximates the optimal policy in the following manner:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \tag{5}$$

where $\alpha > 0$ is the learning rate, and $0 \leq \gamma \leq 1$ is again the discount factor.

4 Experiments

We now show how how a Q-Learning agent performs against a random agent in Colonel Blotto.

4.1 Experimental Setup

We setup a 2-player Colonel Blotto game with 3 fields and 10 coins per player. This provides a total of 66 possible actions a (ways to distribute the 10 coins on the 3 fields) that the players can choose from. The players aim to distribute their 10 coins among the 3 fields in such a way that their coins are greater than their opponent's coins in each field. The winner of the game is the player that wins more fields than their opponent. A single episode in reinforcement learning constitutes one game. We count how many games each player has won over time and show the results.

Our reinforcement learning agent uses the Q-Learning algorithm for approximating the optimal policy. We use $\alpha = 0.1$ as the learning rate and $\gamma = 1$ as the discount factor. We also set $\epsilon = 0.2$ as the exploration rate for the agent.

For the opponent, we use a RandomAgent that will select one of the 66 possible actions at random with equal probability for all of them.

4.2 Results

We run 1000 games of Colone Blotto to see the performance of the reinforcement learning agent. As shown in Figure 1, it takes almost 200 games before the RL agent really begins to start defeating the random agent regularly. In fact, if we look at the just the first 100 games (Figure 2), both agents win games at practically similar rates. However after 1000 games, the improved win rate of the RL agent can already be clearly seen.

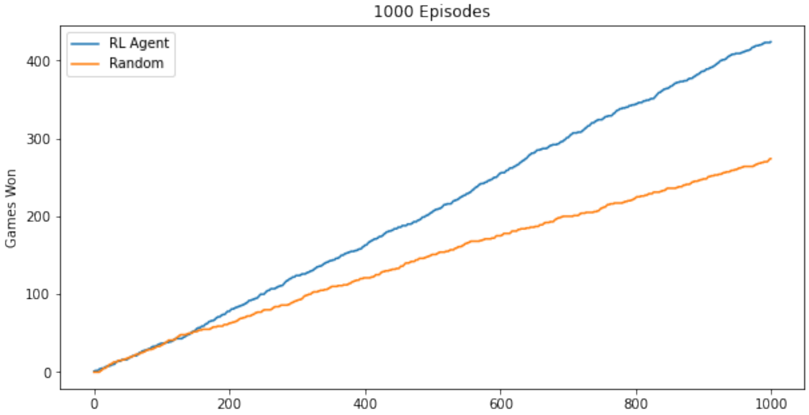

Figure 1: Performance of RL Agent vs Random Agent

4.3 Further Analysis

5 Conclusion

References

[1] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

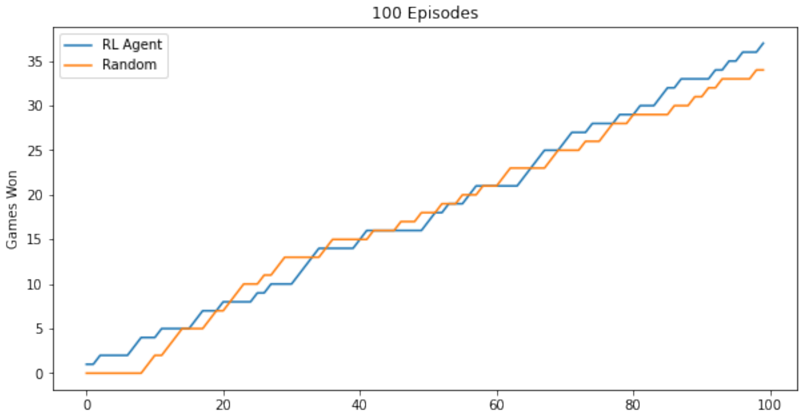

Figure 2: First 100 games of RL Agent vs Random Agent