Nash Equilibria in Infinitely Repeated Prisoner's Dilemma

Nishant Rai

Contents

1 Introduction

The rough structure and aim of this note is to highlight the use of Q Learning in learning efficient strategies for Normal Form Games (The specific case of Infinitely Repeated Prisoner's Dilemma is considered here). It might be interesting (And highly possible) to make a similar claim for all Infinitely Repeated Normal Form Games in general. So, we first start off by discussing Q-Learning and how it can be used to learn adaptive (Self Learning) strategies for a game (say, IPD). Then we show that this strategy is a Nash Equilibrium for the infinitely repeated case. Then a few perks of this strategy compared to the already known and famous 'Always Defect' and 'Trigger' Nash Equilibriums.

Note that this method involves learning the best strategy, so the initial behavior of the strategies is not discussed. We assume (It can be shown that it will) that the learning of the Q-Values (Discussed later) has converged and we are considering the future behavior of the players. It can also be thought of as already giving the learned Q-table to the players and making them play accordingly.

2 Q-Learning

Q-Learning is an algorithm which can be said to belong to the area of Reinforcement Learning. Much of this section is inspired from a blog.

2.1 Discounted Future Reward

It's obvious that to perform well in the long-term, we need to take into account not only the immediate rewards but also the future rewards we are going to get. An easy way to get around this issue is using discounted rewards (i.e. the rewards in the future are worth less than the ones we get immediately).

Given one run of the process, we can calculate the total reward for it by the following equation,

$$R = r_0 + r_1 + \cdots + r_n \tag{1}$$

Generally, we like a large n. The total future reward from time point t onward can be expressed as:

$$R_t = r_t + r_{t+2} + \cdots + r_{t+n} \tag{2}$$

Discounted future reward are formulated as,

$$R_t = r_t + \gamma r_{t+2} + \gamma^2 \cdots + \gamma^n r_{t+n} \tag{3}$$

Here γ is the discount factor, generally between 0 and 1; the more into the future the reward is, the less we take it into consideration. It is easy to see, that discounted future reward at time step t can be expressed in terms of the same thing at time step t+1,

$$R_t = r_t + \gamma R_{t+1} \tag{4}$$

If we set the discount factor $\gamma = 0$, then our strategy will be short-sighted and we rely only on the immediate rewards. If we want to balance between immediate and future rewards, we should set discount factor to something like $\gamma = 0.9$. If our environment is deterministic and the same actions always result in same rewards, then we can set discount factor $\gamma = 1.0$.

A good strategy for an agent would be to always choose an action that maximizes the (discounted) future reward. This is the strategy we'll be using and discussing.

2.2 Details of Q-Learning

In Q-learning we define a function $Q(s, a)$ representing the maximum discounted future reward when we perform action a in state s, and continue optimally from that point on.

$$Q(s_t, a_t) = max R_{t+1} \tag{5}$$

A way to think about $Q(s, a)$ is that it is 'the **best** (Please note it, extremely useful for further discussions) possible score at the end of the game after performing action a in state s'. It is called Q-function, because it represents the 'quality' of a certain action in a given state.

This may sound like quite a puzzling definition. How can we estimate the score at the end of game, if we know just the current state and action and not the actions and rewards coming after that? Actually, we really can't. But as a theoretical construct we can assume existence of such a function. Consider the implications of having such a function. Suppose we are in a state and pondering about the action we should take. We just want to select the action that results in the highest score at the end of game. Since we already have the Q-table, the answer becomes really simple; we pick the action with the highest Q-value.

Just like with discounted future rewards in the previous section, we can express the Q-value of state s and action a in terms of the Q-value of the next state s'.

$$Q(s, a) = r + max_{a'} Q(s', a') \tag{6}$$

This is called the Bellman equation. The main idea in Q-learning is that we can iteratively approximate the Q-function using the Bellman equation. In the simplest case the Q-function is implemented as a table, with states as rows and actions as columns.

3 Defining the strategies

The strategies defined are the same as in the original report. The states discussed in the previous section are refer to the k-histories (My case study considered the special case of 3 plays) of the game and the actions are C or D (Cooperate or Defect).

4 Is it a Nash Equilibrium?

Let's say we have two players who follow the Q strategy (Defined above). Then, after sufficient time (i.e. when they've learned their Q-Tables) we consider their behaviors. Note that the learned Q-tables would be the same for the both. So, will this strategy be a Nash Equilibrium?

Yes, it will be! The proof is pretty simple and just involved using the 'best' term discussed earlier. A contradiction arises and we get our desired result i.e. this is a Nash Equilibrium.

We can also show using a simple Pigeon Hole principle based argument that the behavior of the players would be periodic (i.e. they'll play similar moves in a periodic manner).

5 Is it the optimal one?

Well, it depends on what do we mean by optimal. Do we refer to the case when both players are playing the same strategy or is it optimal while facing multiple strategies.

5.1 Optimality in case of symmetric plays

It's not a very good question. The best in such a case would be to always cooperate. But, theoretically even the Q-strategy should manage to learn that.

5.2 Optimality in case of going against various strategies

It's much better than other hard coded strategies as it is adaptive and is the optimal one by definition. The pros are discussed in my original report.

6 Proof of Convergence

Same as that of the original Q-learning algorithm. It can be shown that it converges to the optimal solution.

7 Cooperation in case of Noisy Environments

Well, I thought that this might be interesting too. Since, it is relatively more mathematically intensive and shows that the strategies promote cooperation in case of noisy games (i.e. in case of mis-communications). For example, consider Tit for Tat, let's say both players are cooperating but due to some noisy signals, both the players feel that the other had defected in the last turn. This would lead to both the players starting to play defect for all the rest of the plays. Or in case we get another noisy signal, both of them would get 'stuck' in the low scoring 'CD DC CD..' cycle. Sorry for the rushed introduction to it, but I'm not sure if this'll be interesting or not.

8 Extension to General Normal Form Games

Note that all the discussions above can be extended to any deterministic Normal Form Game (Not the imperfect information games) (I think). This might also be interesting, since this can be theoretically and practically extended to other games.