Econ 791: Computational Economics

Daniel Ortega

December 1st, 2020

Prisoner’s Dilemma with Q-Learning

1. Introduction

Prisoner’s Dilemma is a 2-player simultaneous game in which the policy of agents following their own self-interest, leads to a suboptimal outcome. This is due to imperfect information since players would cooperate more often if they knew the action of the other player. Games like this can often be modeled using a Markov Decision Process in which certain methods can find an optimal policy such as Q learning. Q learning is a popular reinforcement learning technique in which an agent learns an optimal policy by exploring the state action space and finding the state action pairs that give the highest long-term reward. Simulation of the Prisoner’s Dilemma been researched extensively in game theory and is a very popular model in reinforcement learning. In this paper I will be creating a Prisoner’s Dilemma simulation while using the Q learning algorithm presented in Littman, Cassandra, and Kaelbling (1995) to find an optimal policy.

2. Literature Review

Research that has simulated the Prisoner’s Dilemma typically have done it in a multi agent reinforcement learning (MARL) setting. This means that the agents are constantly updating their strategies, therefore there might not be global convergence, and if there is global convergence, the question becomes whether those solutions are optimal (Tesauro & Kephart, 2002). Additionally, when simulating problems such as the Prisoner’s Dilemma, since players do not share the same information, finding pure strategy equilibrium is extremely difficult (Iwasaki, Joe, Kandori, Obara ,& Yokoo, 2011). Therefore, one of the problems is figuring out what algorithm is best for computing optimal policies in such dynamic systems. Littman, Cassandra, and Kaelbling (1995), describe several algorithms including truncated exact value iteration, Qmdp value method, and replicated Q-learning method.

3. POMDP Framework

In order to simulate the Prisoner's Dilemma, it must be first constructed into a Partially Observable Markov Decision Process (POMDP) framework. A POMDP's has the same properties as a Markov Decision Process, the difference being that the agent does not directly observe the state. This idea of a MDP with imperfect information was first described by Astrom (1965) and framework was more formally established by Kaelbling, Littman and Cassandra (1998). I will be describing the framework established by Kaelbling, Littman, and Cassandra (1998) in this section. The POMDP is defined as a 7-Tuple $(S, A, T, R, \Omega, O, \gamma)$, where:

- S is a set of states
- A is a set of actions
- T is a set of conditional transition probabilities between states
- R: SXA is the reward function
- Ω is a set of observations
- O is a set of conditional observation probabilities
- γ is the discount factor

The goal of the agent is to choose actions at each step that will maximize its expected future discounted reward. When simulated the environment is in a state s, the agent takes an action based on a given belief of the state. After choosing an action, the agent transitions to a different state using the conditional transition probability T. It receives a reward after transitioning to the new state as well as an observation o from Ω. This observation gives information on the probability of the observation occurring given the new state and action, $O(o|s',a)$.

As mentioned previously, in POMDPS actions depend on a belief state. The belief state is described by a probability distribution, in which the agent believes specific states have a probability of occurring. Since this is a Markovian problem, maintaining a belief over the states depends only on the previous belief of the previous state, previous action, and the current observation, therefore future beliefs can be described as $b' = \tau(\mathrm{b}, \mathrm{a}, \mathrm{o})$. Let b be defined as a probability distribution over the state space. Let b(s) be an agent's belief that is in state s when the current belief state is b. The new belief in a state s' can be defined as:

$$b'(s') = \eta \mathrm{O}(\mathrm{o}|\mathrm{s}', \mathrm{a}) \sum_{\mathrm{s}} T(s'|s,a) b(s)$$

Where $\eta = 1/\mathrm{P}(\mathrm{o}|\mathrm{s}', \mathrm{a})$ (normalizing constant)

$$P(o|b,a) = \sum_{s'} O(o|s',a) \sum_{s} T(s'|s,a) b(s)$$

To find an optimal policy within this framework, it requires that the agent use his current belief state to determine an action. Fortunately, a Markovian belief state allows for this POMDP to be defined as a MDP where every belief is the state. The belief MDP is the tuple ($B, A, \tau, \Upsilon, \gamma$) where:

- B is the set of belief states over the POMDP states
- A is the same finite set of action
- τ is the belief state transition function
- r: B X A is the reward function on belief states
- γ is the same discount factor

τ and r are derived from the original POMDP, where τ is defined as:

$$\tau(\mathrm{b}, \mathrm{a}, \mathrm{b}') = \sum_{\mathrm{o}} P(b' |\, b, a, o) P(o|a, b),$$

Where $P(b'|b, a, o)$ is equal to one if the belief updates with arguments b,a,o and zero otherwise. The reward function is then defined as:

$$r(b, a) = \sum_{s} b(s) R(s, a)$$

The goal is to find an optimal policy (or Q-function) that optimizes this bellman optimality equation below. An optimal policy for a POMDP chooses an action that is optimal given a belief state. The optimal Q-function is like an optimal value function from dynamic programming since it inputs a state and gives out the best action which maximizes the reward. However, the key difference is that the agent in Q learning does not know the transition probabilities and must explore the state space in order to find rewards. Q values will later be used in this paper to describe the output of the Q function.

Bellman Optimality Equation:

$$V^*(b) = \max_{a \in A} \left[r(b, a) + \gamma \sum_{o \in \Omega} \Pr(o \mid b, a) V^*(\tau(b, a, o)) \right]$$

4. Model

Due to the difficulties of a MARL setting, I will be training one agent using Q learning, whilst the opponent will be following a fixed strategy. Since the opponent or environment has a fixed strategy, Q-learning is guaranteed to find the optimal policy (Tesauro & Kephart, 2002). The structure of my POMDP will be created using the POMDP.jl packages in Julia. The action space for the agent is defined as two discrete actions: confess or keep quiet. The state space is defined as two discrete states, which represent whether the other prisoner kept quiet or confessed. Transition probability is described as for any action, there is a 60% probability the state is confessed and a 40% probability the state is kept quiet. The initial belief state is a uniform probability (default for QuickPOMDPs.jl).

The reward function depends on the current state and action and can be defined by the reward matrix (See Figure 1). There are many ways to described the reward matrix for the Prisoner's Dilemma, I chose to give a large reward for keeping quiet when the state is kept quiet, a small penalty for confessing when the state is confess, and a large penalty otherwise. This will affect the agent's strategy later, since there will be more of an incentive keep quiet when the agent believes state is kept quiet, rather than choosing the action of confess.

As for the conditional observation probability, one common way to describe the observations is as two discrete values of good or bad (Iwasaki, Joe, Kandori, Obara, Yokoo, 2011). In my simulation, I define the possible observations as four discrete observations: Short Prison sentence, medium prison sentence, long prison sentence, and no prison sentence. I believe this is a realistic assumption since those are the real observations a player might face when in the Prisoner's Dilemma. Different combinations of actions and future state gives more weight or probability that a prisoner will see a certain observation. My conditional observation function is as follows:

- If action is confess, and future state is confessed then there is a 70% probability that observation will be a Medium Prison sentence, and 10% probability for all other observations.
- If action is confess and the future state is kept silent, then there is a 70% probability that observation will be a short prison sentence, and 10% probability for others.
- If action is keep silent and the future state is confessed, then there is a 70% probability for long prison sentence, and 10% for all other observations.

- If action is keep silent and the future state is keep silent, then there is a 70% chance of no prison sentence, and 10% probability for all other observations.

Since the conditional probability is the probability that the agent will receive the observation given the action and future state, I chose these because it gives a high probability that the agent will know what state it is based on the observation while also keeping a decent level of uncertainty.

After creating the POMDP environment, a policy must be created, I chose to use the QMDP value method presented by Littman, Cassandra, and Kaelbling (1995), which is conveniently integrated into QuickPOMDP.jl. The algorithm goal is to find a Q function that will compute the optimal policy. In order to find the Q function, this method temporarily ignores the observations and finds the Q values of the underlying MDP. All actions for these Q values will be treated as a single linear function and the estimate Q value for a certain belief state b is:

$$Q_a(b) = \sum_s b(s)\ Q_{\mathrm{MDP}}(s, a)$$

This estimate is assuming that any current belief in the current state will be irrelevant after the next action is taken. Therefore, the action whose long-term reward from all states is the largest will be chosen at each step while considering the weighted probabilities of occupying certain states.

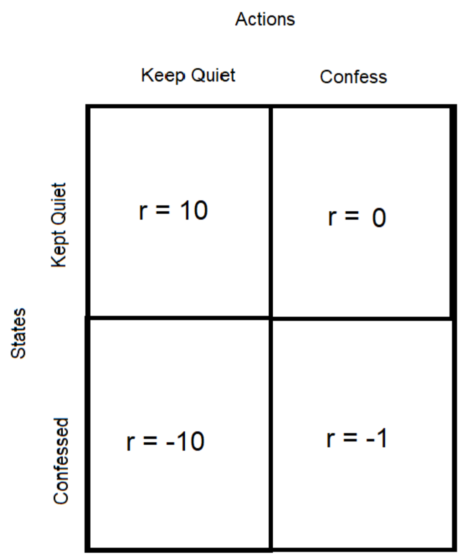
5. Results

After running the simulation twice, in which there are 10 max steps or iterations, through the environment with the QMDP policy, the 4-tuple is generated which gives the state, belief state, action, and observation (See Figure 2). The undiscounted reward is also generated, which was 119 for one simulation and 116 for the other. When examining this tuple, it can be seen how the belief state is updated using the QMDP algorithm. Recall, that the belief state is not updated by the current state but rather the observation. For example, when looking at the first tuple generated in the first simulation, the initial belief state was 50% probability of confessed and 50% probability of kept silent. With that belief state, the agent decided to keep silent and received an observation of Long Prison Sentence. Given that the observation long prison sentence has a high probability of occurring when the action is keep silent and the future state is confessed, the agent updates its belief so that it gives a higher probability of the state being confessed in the next step, hence why in the next step the probability of confessed and kept quiet is .913 and .086 respectively.

It is also important to note due to the stochastic nature of the conditional observation probabilities, an agent has a 30% probability of receiving the wrong signal, so therefore it will sometimes wrongly update its belief state. An example of this can be seen in Figure 3, where the agent receives a short prison sentence for choosing action confess when in state confessed. This makes the agent believe that the actual state was most likely kept silent, so the agent updates its belief state to give more probability that the state is kept silent for the next step. Given that a higher reward is given for when the action is keep silent and the state is kept silent, the player decides to keep silent the next step in order to receive this higher reward. However, since it received the wrong signal it receives an observation of a long prison sentence.

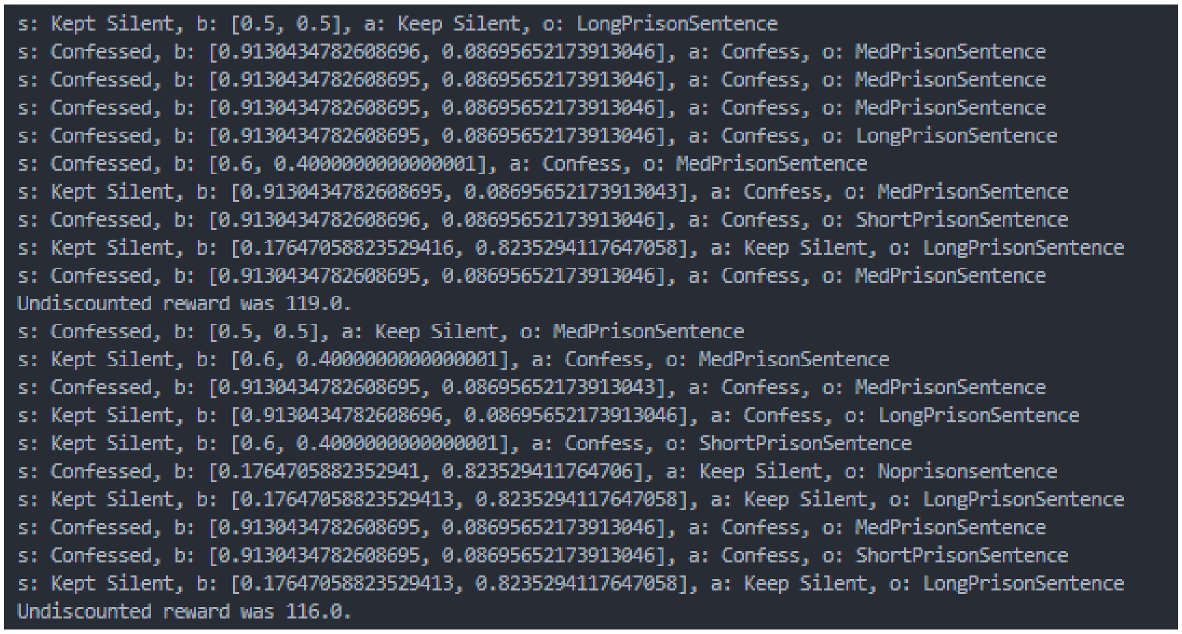
Despite sometimes receiving the wrong signal, the policy generated by the QMDP algorithm is an optimal policy because it chooses the optimal action based on the belief state. Meaning whenever the belief state gave a high probability of a certain state, the policy always chose the action that would maximize the reward.

Figure 1



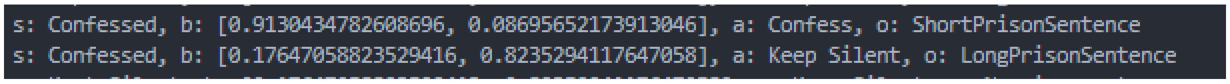
		Actions	
		Keep Quiet	Confess
States	Kept Quiet	r = 10	r = 0
	Confessed	r = -10	r = -1

Figure 2



```
s: Kept Silent, b: [0.5, 0.5], a: Keep Silent, o: LongPrisonSentence
s: Confessed, b: [0.9130434782608696, 0.08695652173913046], a: Confess, o: MedPrisonSentence
s: Confessed, b: [0.9130434782608695, 0.08695652173913046], a: Confess, o: MedPrisonSentence
s: Confessed, b: [0.9130434782608695, 0.08695652173913046], a: Confess, o: MedPrisonSentence
s: Confessed, b: [0.9130434782608695, 0.08695652173913046], a: Confess, o: LongPrisonSentence
s: Confessed, b: [0.6, 0.4000000000000001], a: Confess, o: MedPrisonSentence
s: Kept Silent, b: [0.9130434782608695, 0.08695652173913043], a: Confess, o: MedPrisonSentence
s: Confessed, b: [0.9130434782608696, 0.08695652173913046], a: Confess, o: ShortPrisonSentence
s: Kept Silent, b: [0.17647058823529416, 0.8235294117647058], a: Keep Silent, o: LongPrisonSentence
s: Confessed, b: [0.9130434782608695, 0.08695652173913046], a: Confess, o: MedPrisonSentence
Undiscounted reward was 119.0.
s: Confessed, b: [0.5, 0.5], a: Keep Silent, o: MedPrisonSentence
s: Kept Silent, b: [0.6, 0.4000000000000001], a: Confess, o: MedPrisonSentence
s: Confessed, b: [0.9130434782608695, 0.08695652173913043], a: Confess, o: MedPrisonSentence
s: Kept Silent, b: [0.9130434782608696, 0.08695652173913046], a: Confess, o: LongPrisonSentence
s: Kept Silent, b: [0.6, 0.4000000000000001], a: Confess, o: ShortPrisonSentence
s: Confessed, b: [0.1764705882352941, 0.823529411764706], a: Keep Silent, o: Noprisonsentence
s: Kept Silent, b: [0.17647058823529413, 0.8235294117647058], a: Keep Silent, o: LongPrisonSentence
s: Confessed, b: [0.9130434782608695, 0.08695652173913046], a: Confess, o: MedPrisonSentence
s: Confessed, b: [0.9130434782608695, 0.08695652173913046], a: Confess, o: ShortPrisonSentence
s: Kept Silent, b: [0.17647058823529413, 0.8235294117647058], a: Keep Silent, o: LongPrisonSentence
Undiscounted reward was 116.0.
```

Figure 3



```
s: Confessed, b: [0.9130434782608696, 0.08695652173913046], a: Confess, o: ShortPrisonSentence
s: Confessed, b: [0.17647058823529416, 0.8235294117647058], a: Keep Silent, o: LongPrisonSentence
```

References

Åström, K.J. (1965). "Optimal control of Markov processes with incomplete state information". *Journal of Mathematical Analysis and Applications*.

lbling, L.P., Littman, M.L., Cassandra, A.R. (1995). Learning Policies for partially observable environments: Scaling up. *Artificial Intelligence*.

lbling, L.P., Littman, M.L., Cassandra, A.R. (1998). "Planning and acting in partially observable stochastic domains". *Artificial Intelligence*.

Iwasaki, A., Joe, Y., Kandori, M., Obara, I., Yokoo, M. (2011). Coordination via Mutual Punishment POMDP Approach to Repeated Prisoner's Dilemma with Private Monitoring. *Psychology.*

Tesauro, G., & Kephart, J. O. (2002). Pricing in Agent Economies Using Multi-Agent Q-Learning. *Autonomous Agents and Multi-Agent Systems.*