Project 3: Correlated Q-Learning

Latest hash commit: 6c6bd11210032e0ea189cf1545457891aa0a4be3

Author: Poujon Adrien

I. Introduction

In the following analysis, I rely on the study of an example simple enough for conclusions to be drawn upon the efficiency of different Q-learning methods.I start by describing the implementation and coding of the soccer game as describe in Greenwald ans Hall paper's [1][2] (Section II). This game then leads to discussion about equilibrium in game theory (Section III) I then present the different learning procedures used to produce the graphs (Section IV). This report ends with a discussion on the implementation of the algorithms and with comparison of the obtained graphs with Greenwald's figures (Section V).

II. Soccer Game

The article by Greenwald and Hall develops several examples on different games. However, in what follows we focus on the Soccer Game which is used to generate the graphs that we aim at reproducing. This Soccer Game was first introduced by Littman in *Markov games as a framework for multi-agent reinforcement learning* [3]. For our purpose we consider the game played on a $2 * 4$ grid.

The grid is composed of a field, here in green, of size $2 * 2$ on which the players can move and for additional cells on each side representing the goals. On Fig.1, player B as to bring the ball in the blue cells and symmetrically player A has to goal in the orange cells. We name *goal* the event of any player being on a goal cell while possessing the ball. That means that a player can possibly goals against himself. For instance, on Fig.1, where A possesses the ball, if player A realizes action *move to left* then, they will goal against their own camp. Consequently rewards are of two kind: -100 points for the player in whose goal the ball ended, and +100 for the other player.

There are five possible actions depicted by the wind rose on Fig.1 which are: going East, going North, going West, going South and sTicking to current location. In the way the game is implemented, this actions are mapped to integers 0 to 4 respectively. Also, the board is stocked in a list rather than in a matrix. Consequently, is a player is in the bottom orange goal we refers to its position as being in position 7 instead of position (1,3). The current state of the game is thus represented by a tuple of length 3 containing the position of player A, the position of player B and the owner of the ball, either 1 for player A, or 2 for player B (see Fig.2):

$$state = \begin{bmatrix} position(A) \\ position(B) \\ ball \end{bmatrix}$$

The rules of the games allow a player to steal the ball from another one when a collision happens. The stealing only takes place when the player with the ball tries to enter a cell where

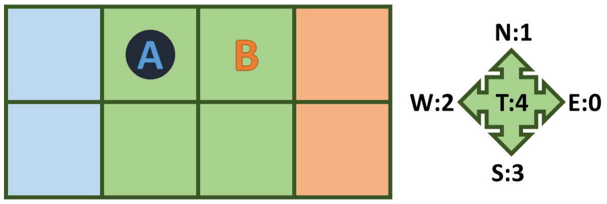
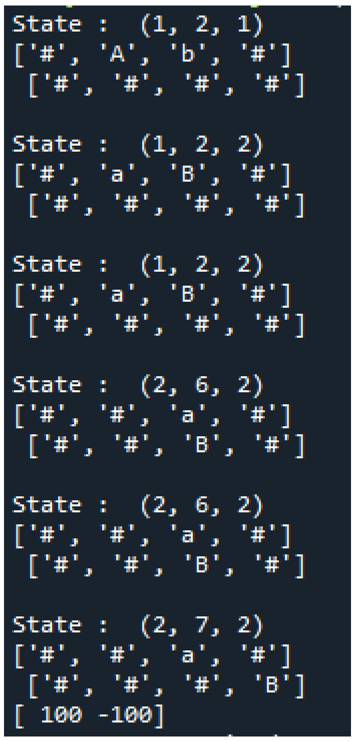


Fig. 1. Soccer Game in state [1,2,1]



```
State :  (1, 2, 1)
['#', 'A', 'b', '#']
 ['#', '#', '#', '#']

State :  (1, 2, 2)
['#', 'a', 'B', '#']
 ['#', '#', '#', '#']

State :  (1, 2, 2)
['#', 'a', 'B', '#']
 ['#', '#', '#', '#']

State :  (2, 6, 2)
['#', '#', 'a', '#']
 ['#', '#', 'B', '#']

State :  (2, 6, 2)
['#', '#', 'a', '#']
 ['#', '#', 'B', '#']

State :  (2, 7, 2)
['#', '#', 'a', '#']
 ['#', '#', '#', 'B']
[ 100 -100]
```

Fig. 2. Soccer Game at random

the player without ball stands. In any case, when a collision happens, the current sequence of actions is ended an we go to the next sequence-step of actions.

Several points were not very clear for implementing this game. Firstly, one might wonder what happens when a player decides to go toward a border a the field. Three solutions are possible here: either we do not authorize this move and the player are left we only possible moves for their choice of move, or we do authorize it and we can choose to make the move toward the border impossible and thus equivalent to a staying on cell move, or we can consider the board as being a torus and make the player reappear at the opposite end of it. I decided to consider the moves toward the border as impossible and equivalent to sticking to position. An example of a run of the game at random is showed Fig.2. The convention is the same as on Fig.1 concerning the goals, and the possession of the ball is denoted by the player being in capital letter.

III. Equilibria in n-player games

Stochastic games are generalizations of repeated games and of Markov decision processes. In that sense, optimal state- and action-values for multiple agents games follow equations that are similar to Bellman's equations. What actually differs is the way values are anticipated, or in other words, one can define different learning strategies depending on the computing of the values, as we shall see in next section.

From Hu and Wellman onward [4], the definition of the value function that prevailed was the Nash equilibrium. This sort of equilibrium denotes situations in which no player can achieve better rewards if it chooses other actions all other things remaining equal -especially the strategy of other player(s).

In the paper by Greenwald and Hall, a new kind of equilibrium are presented, which are correlated equilibrium. This new kind of equilibrium intertwine the agent's actions so that an agent policy depends of the other agents policies, and conversely. As with Nash equilibrium, there is a set of possible equilibria that are accessible for the agents, and different methods of implementing correlated-Q learning will lead to the agents ending up in different equilibrium. But the great advantage of correlated equilibrium is that it can be computed by using linear optimisation for which a lot of libraries and methods already exist.

In the paper, four types of correlated-Q learning are introduced, but only one is of interest for us here. In the following of the paper, correlated-Q learning is intended in the sense of utilitarian correlated-Q learning which is called that way for it maximizes the sum of players' rewards:

$$\sigma \in arg \max_{\sigma \in CE} \sum_{i \in I} \sum_{a \in A} \sigma(a) Q_i(s,a)$$

IV. Four learning procedures

Greenwald proposes four strategies for implementing learning in her paper. All these strategies relies on a general Q-learning procedure. That is presented in the algorithm 1. What makes the difference between the different procedures is thus the choice of f. In other words, what we seek to do is to see how different computations of V will affect the results.

Algorithm 1 General Q-Learning procedure

```
 1: select $f$                              ▷ correspond to a learning procedure
 2: initialize $\alpha, \delta$                        ▷ learning rate and decay
 3: initialize $\gamma$                                      ▷ discount factor
 4: initialize number of sequence-steps $T$
 5: for $t = 1$ to $T$ do
 6:     We are now in state $S$
 7:     Choose some actions $A \leftarrow a1, a2$ to take
 8:     Take actions $A$
 9:     Observe rewards $R \leftarrow r1, r2$ and new state $S'$
10:     for $i = 1$ to $n$ do              ▷ n is the number of players
11:         $V_i(S') \leftarrow f_i(Q_1(S'), Q_2(S'), ...)$
12:         $Q_i(S,A) \leftarrow (1-\alpha)Q_i(S,A) + \alpha[(1-\gamma) * R_i + \gamma * V_i(S')]$
13:         $\alpha \leftarrow decay(\alpha, \delta)$
14:     end for
15: end for
```

We see that the Q-update procedure is largely independent of the learning method and will in fact stay unchanged in all the algorithms. As one can see, it is an update based on Bellmann's equations. The only differences between algorithm will be the structure and the number of Q-tables depending on how many agents we consider and whether of not we place ourselves in a mixed-strategies framework.

A. Classical Q-learning

Classically, such learning procedure is implemented via Q-learning. The Q-learning procedure is described by the following update equation for V:

$$V^*(s) = \max_{a \in A(s)} Q^*(s,a)$$

In that case the expected value is an optimistic one: the estimated value from a state s is the maximum of the Q-values for this state, considering all possible actions. Following what was described in Section II, A(s) is always a set of actions $\{0,1,2,3,4\}$ corresponding to moving East, North, West, South or Sticking to the position respectively.

Note that there is here only one Q-table that only contain the Q-values for the actions of one player, meaning that we consider potential other players as part of the environment instead of actual agents themselves. Philosophically, I would call this approach the "solipsism Q-learning". The Q-learning is thus learning from one agent actions only, and does not rely on learning from mixed strategies. Also, the resulting policy is deterministic and based on a maximization of the Q-values:

$$a = \arg \max_{a \in A(s)} Q(s,a)$$

B. Friend Q-learning

The Friend-Q learning procedure relies on the idea that the actions other agents will take will grant you advantages and rewards. It is a collaborative framework where cooperation and coordination are inherent to the game structure, or naively assumed. For instance, in the prisoner's dilemma, if both agents chooses to stay silent then they will globally get a better reward which is good if one consider that these two agents know each other and are friends.

In that framework, the expectation value is computed such that it considers the maximum Q-value being possibly attained for the tuple of actions for each agents. As there are only two players in the soccer game, thence there are only couples of actions which I denote (a_1, a_2) and are taken in A^2.The choice of V is thus:

$$V^*(s) = \max_{(a1,a2)\in A(s)^2} Q_i^*(s, a1, a2)$$

Note that it is a formula very similar to the one used in classical Q-learning except for the use of couple of actions and of two Q-tables. This is indeed a generalization of the Q-learning for one agent into multi-agents games. The resulting policy will also be deterministic and will seek to maximize the Q-value over the couple of actions:

$$a_1 = \arg \max_{(a1,a2)\in A(s)^2} Q(s, a_1, a_2)$$

The main issue with this approach, is this hypothesis that the agents will actually collaborate with each other. This might be the case in certain games, but even in the case of the prisoner's dilemma, it is better for an agent B to betray the other agent A if it knows that A follows a friendly policy. Furthermore, in plenty of game, a collaborative behavior might not even be possible. In our soccer game for instance, it appears that the Friend-Q learning will lead to the agent B waiting for agent A to goal on his behalf (and conversely), which is probably what one actually wants to do in this game.

C. *Foe Q-learning*

The Foe-Q learning is opposite to the Friend-Q for it considers the other agent as being an opponent rather than an ally. In other word, we will consider that the other agent always plays in order for us to benefit only the smallest possible reward. It is a pessimistic learning strategy. As in Friend-Q we must consider couple of actions. The expectation for the value is as follows:

$$V_i(s) = \max_{\pi\in\Pi_A(s))} \min_{a_2\in A(s)} \sum \pi(a_1) Q_1^*(s, a1, a2)$$

What we see here is that we actually do a minimax operation aiming at maximizing the value of our action while considering the opponent will to make us lose. This is done by using a Π_A probabilistic action space for player A, providing us with a mixed strategy.

From there, using linear programming can help to compute π from a given Q(s)-table associated with specifications. Obviously, we will want the π vector to be a vector of probability, thence its elements should be comprised between 0 and 1 and should also sum up to 1. Thus we can try to maximize the sum of products of π by Q as shown in the equation by using a linear-programming solver. The π vector will then be the policy for action distribution, and actions will be chosen accordingly.

D. *utilitarian-Correlated Q-learning*

As we have seen in Section III about equilibria in n-player games, the correlated-equilibrium in another possible approach to the solution of the game. it assumes that there is a sharing of information between the two agents. Concretely it consists of a randomized assignment of action recommendations to agents such that neither of the agents would want to deviate from these actions. This is typically the case in the case of the chicken game when played with a traffic light for instance. It that sense it constitute a generalization of the Nash equilibrium.

Here, we use again a π vector for the probability distribution of actions, but this time it will have to be a probability on the couple of actions. Concretely, where in the Foe-learning the π vector only took a_1 as an argument, it now takes (a_1, a_2). As we have said, four different kind of correlated-Q learning are envisioned by Greenwald, but we only focus on utilitarian correlated-Q learning (uCE-Q). For uCE-Q, the conditions on π are as follow:

$$\pi \in \arg \max_{\pi\in CE} \sum_{i\in I} \sum_{(a_1,a_2)\in A^2} \pi(a_1, a_2) Q_1(s, a_1, a_2)$$

$$\sum_{a_{-i}\in A_{-i}} \pi_s(a_{-i}, a_i) Q_i(s, a_{-i}, a_i) >= \sum_{a\in A_{-i}} \pi_s(a_{-i}, a_i) Q_i(s, a_{-i}, a_i')$$

$$\sum_{(a_1,a_2)\in A^2} \pi(a_1, a_2) = 1$$

$$\pi(a_1, a_2) >= 0, \forall(a_1, a_2) \in A^2$$

And once π has been determined by a linear solver, than we can easily get the expectation values by computing the product of π time Q.

V. Results and Analysis

A. *Considerations on implementation*

For these simulations I tried to follow the paper's method as close as possible. The four methods described above are use along with a choice of action based on pure randomness for Foe-Q and uCE-Q, corresponding to the off-policy learning specified in the paper. The Friend-Q is also done off-policy in the paper but I decided to implement it with an epsilon greedy policy in my work, as it provided better results. The classical Q-Learning is done on policy with an epsilon-greedy policy too.

The parameters are identical for all experiments. Especially, α goes from 1 to 0.001 as indicated by Greenwald, with an exponential decay of 0.000005. This way I was sure that this alpha-decay was not a prime cause of the convergence of the Q-differences as the graph for classical Q-learning demonstrate. The discount factor used is identical to the paper, as $\gamma = 0.9$. As described by Greenwald, the graphs plot the difference between the values of Q for the state of Fig.1 at successive steps, in absolute value. Finally, I truncated the graphs y-axis at 0.5 as was done in the paper.

The initial values within the Q-tables are not specified within the paper. I decided to implement it so that the initial values are distributed at random according to a normal distribution centered on 0.

A note must be done on the graphs for CE-Q and Foe-Q in the original paper. In fact, these two graphs are perfectly identical which seems kind of strange and very unexpected.

B. Classical-Q Learning

The first figure Fig.3 represents the results obtained via the Classical-Q learning procedure. On this procedure, I use a low decay rate for the epsilon and we see that the convergence of the difference is mostly due to the decay of the learning rate alpha. As the graph shows, we obtain a figure quite close from the one in the article, with oscillations of the differences going up to the envelop by the learning rate. Although this procedure prove to be of very low interest in our game, it has the advantage to illustrate the influence of the learning rate which will allow to conclude that the other graphs do not converge because of the learning rate effect.

Concretely, what is seen on the graph is that the learning procedure can not reach a stable strategy. For each step in the training, the agent learns a new strategy and modifies greatly its previous strategy. This is due to the fact that in the classical-Q learning there is not taking into account of the second player as such. As a consequence, no optimal strategy can be reached and this procedure is doomed to oscillate as long as the learning rate is sufficiently great.

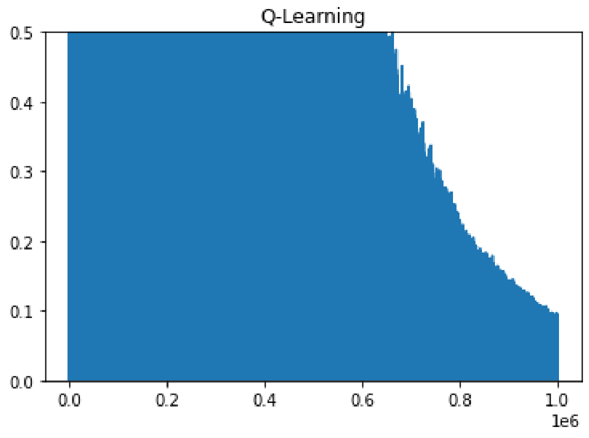


Fig. 3. Classical-Q Learning

C. Friend-Q Learning

When first seeing the graph for the Friend-Q learning procedure, one might naively think that this a very great procedure. However, one should not forget about remarks on this procedure from the previous section. In particular, if the learning is particularly fast, it is only because the players do not exhibit specific preference about the actions they have to take. In fact, considering that the adverse player will score for them, the player without ball has nothing special to do. Even if this strategy offers very fast convergence, it does not offer good strategy in our case, and furthermore for any game in which cooperation is not embedded.

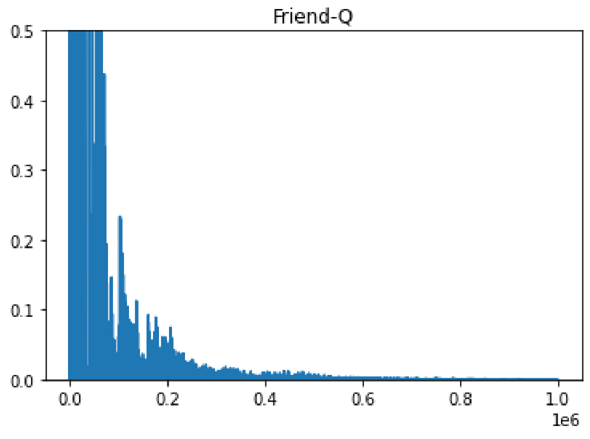


Fig. 4. Friend-Q Learning

D. Foe-Q Learning

The Foe-Learning is interesting because it is not as naive as the Friend learning and better fit the actual aim of the soccer game. In that case, players aims at blocking each other and steal the ball from the other for example.

Here the convergence is gradual. It is slower than in the Friend-Q learning, but still way better than in the case of the classical Q-learning. From the Fig.3 of the classical-Q learning, we can say that the convergence in the case of the Foe-Q learning, although slower than in the case of Friend-Q learning, is not due to the decay of the learning rate -or if yes, it is only very partially.

We see that the envelop of the convergence follows a nice exponential curvature, which reflects a stable convergence, unlike what is happening in the classical Q-learning.

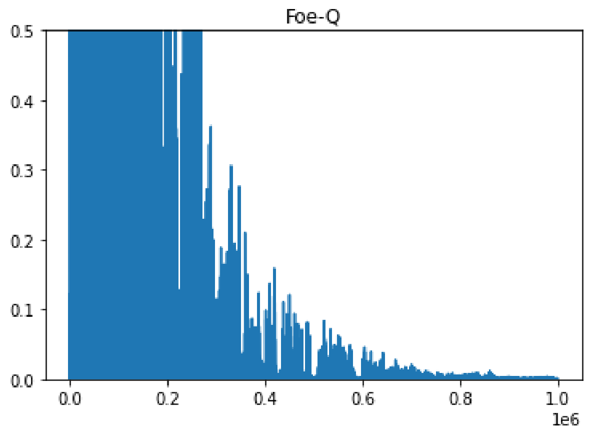


Fig. 5. Foe-Q Learning

E. Utilitarian Correlated-Q Learning

In this last example, we obtain similar results as in the case of Foe-Q learning. Indeed, the uCE-Q learning procedure seems very well fitted for the agents to learn strategy in the case of the soccer game by taking into account the actions of the opponents and making non-deterministic decisions based on the correlated analysis of the actions' effects.

We see however, that the curvature is a bit more jerky than the converging curve obtained for the Foe-Q learning algorithm. Also, the computational time was at least twice of the order of the one for Foe-Q learning. In fact, there are much more conditions to fit in this linear optimization problem than in the case of the Foe-Q learning problem.

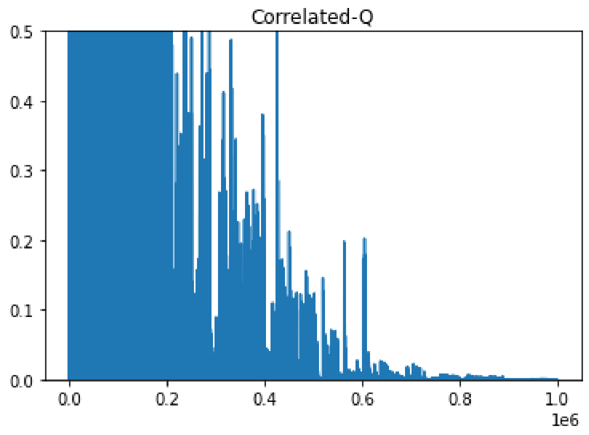


Fig. 6. uCE-Q Learning

VI. Remarks and Conclusions

A. Remarks

I am not absolutely certain about the way I implemented the game. I did some tests by hand and using a random generator for the moves of the two players, and I did not find any mistakes, but it is still possible that I misunderstood some specifications for the game. I am also a bit skeptical about my understanding of the linear solver and the way I implemented the conditions for the optimizations of π in the procedures which required it. As a consequence, even if my graphs roughly resemble those obtained by Greenwald and match the overall tendency for the kinetic of the convergence, they do not perfectly imitate the ones in the paper, and a perfect match might requires some more advanced code verification.

Playing with variation of the learning rate and other parameters might also lead to better graphs, but I was not able to obtain proper results in doing so, as I assumed each plot was based on the same set of values for the parameter. Indeed, a variation on the learning rate or its decays affects some graphs to be closer to the paper's, whereas other graphs goes farther in their differences. Consequently I was not able to find a unique shared set of parameters that perfectly fit all the graph of the paper, but was forced to use a medium set which is good overall.

B. Conclusion

In this report I have been able to present the four procedures proposed in Greenwald's paper for implementing Q-learning in a multi-agents game. The soccer game has been chosen as an illustration where two agents compete in a given environment and their actions can impact the other player's state. Through the graphs, I was able to show that in such an environment, the taking into account of the opponent is really important to allow the development of a stable strategy. However, different procedures might lead to different results and all are not equal on the behavior of the agents. On the one hand, Friend-Q learning will, for instance, lead to an equilibrium in which the agent is irrational regarding the goal fixed by the game. On the other hand, Foe-Q and uCE-Q learning should learn good strategy that indeed correspond to the expected behavior of a player of the soccer game.

As a conclusion, I shall thus say that in finding a strategy for a multi-agents game, one should take into account the actions of the other. However, the way of taking into account these other players' actions depends on the goal you propose to fulfil and what you assume that the other player will do.

References

[1] Amy Greenwald, Keith Hall, Correlated-Q Learning, 2003
[2] Amy Greenwald, Keith Hall, Martin Zinkevich, Correlated-Q Learning, 2007
[3] Michael L.Littman, Markov games as a framework for multi-agent reinforcement learning.
[4] J.Hu, M.P.Wellman, Nash Q-Learning for General-Sum Stochastic Games, 2003.