Reinforcement Learning in Auction Games

Niall Larkin and Rory Hurley

1. Introduction, Domain and Task

Auctions have been utilised as a means in society for purchasing and selling goods since time immemorial. With the age of the internet, naturally auctions have come into existence online with companies such as E-Bay in the western world and the now booming online auction space in Alibaba. Real Time Bidding (RTB) Auctions systems have been the main trading mechanism for online advertising since online advertising started in the 2000's [1]. This project aimed to leverage an artificially intelligent software agent that would be able to compete in an auction scenario. There were two performance objectives for our agent at the beginning of this project:

1. A single agent will learn the optimal bidding strategy when taking part in an auction over a set period of time whereby it will purchase an item at the lowest possible bidding price it can achieve based on the agents own initial valuation of that said item.
2. 2 agents will try to beat each other as they bid for the same item using the same approach as above with either the same or different valuations on said bidding item.

Participants in this approach start with no bid present in the auction and for each time iteration bid in turn where they can either hold at their present bid or increase the set limit on their bid in order to purchase the item on offer in the auction. This is done for a set number of auction rounds T where the individual with the highest bid wins the item. The level of reward associated with acquiring the item however is dictated by the initial valuation put on that item by the bidder versus how much the final bid that placed on that item. All participants can withdraw from the auction at any point in time however when this occurs there final bid is void and they cannot re-enter the auction to rebid in further auction rounds.

2. State Transition function

The agent's state (s) in this Q-learning scenario at any time point (t) represents the value of the agent's bid at that time point. At each turn then the agent can either remain at its current bid or may move from state to another taking action $a \in A_s$ (all the actions available for state s), up to a maximum bidding value that is allowed by the auction system which is denoted as P. As a result of this the following state transition function illustrates this state transition:

$$s_{t+1}=\delta(s_t,a_t)$$

The Markov Decision Process flow for the state transitions can be seen in Figure 1 below.

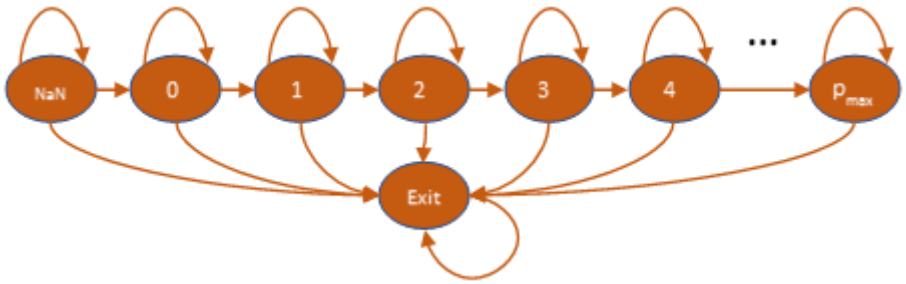

Figure 1 Markov Decision Process for Auction Game

3. Reward matrix

In a typical reward matrix for Q-learning rewards are paid following each state transition based on the state it was present in and the action it decided to take [2]. With such an approach it is logical that this can be constructed into a 2 dimensional matrix R with actions acting as a row index and states acting as column index. This ensures that reward r=(s,a) is obtained when moving from state s utilising action a. This typical approach will not work however with the Auction task as stated above due to the time limit dependence of T auction rounds stated in the problem statement. For the general Q-learning method there is no limit to number of time periods an agent can take in order to achieve its final goal state. Hence the agent can achieve its reward at any time point t. Whereas, in this auction problem the agent will only achieve its reward at the predetermined final time point T at the end of the auction.

As a result an adaptation to the rewards matrix is required whereby instead of the matrix being the typical 2 dimensional it instead must be 3 dimensional where r=(s,a,t) i.e. the reward r_t for taking action a_t from state s_t at time t. As the auction consists of sequential rounds the choice of the agents reward will always move from time t to t+1 which ensures that the state transition function remains valid as stated above.

There is one other departure from the typical reward matrix with regards to the fact that each agent has its own estimation V of what the value of the item being auctioned is worth. As a result this means that the reward is an attribute of the agent as opposed to the environment. This therefore means that when multiple agents are competing in the same auction they have the potential for different reward matrices and as a result develop different bidding strategies as a result.

4. Choosing a Policy

"A policy defines the learning agent's way of behaving at a given time" [2]. Essentially, the policy is a mapping of the current state of an agent, to the action it should take when in that particular state.

The reinforcement learning (RL) policy (π) defines how to map states to actions, with the aim of learning a policy ($\pi : S \rightarrow A$) that maximises expected returns ($E[R_t]$) in the long term. For the agent to learn which sequence of actions maximize the reward in the long run, a mechanism to make the decision of an agent to explore or exploit its knowledge, is dictated by the exploration policy.

There is a trade-off between exploration and exploitation with regards to the policy. For example, an agent could follow a policy that is more explorative, which aims to find the optimal route to its goal via finding numerous potential routes over time, however this may mean that the agent sacrifices exploiting the best option possible at each state. On the other hand, a highly exploitative policy will aim to maximise immediate rewards at each state, which could compromise the likelihood of finding the optimal route through exploration in the long term.

Finding the balance between exploitation and exploitation is therefore a key factor in determining the success and speed at which the agent finds the optimal path. In order to define an approach to this problem, an exploration policy, π, is defined using a threshold ε, as:

$$
\begin{aligned}
&IF\ random\ number\ [0,1] > \mathcal{E}: \\
&\pi^*(s) = argmax\{Q(s_t, a \in A_s)\} \\
&ELSE: \\
&\pi(s) = random\{Q(s_t, a \in A_s)\}
\end{aligned}
$$

As time goes on and the number of trials increases, it is likely that the agent will have explored its environment sufficiently and will have good experience of mapping actions to states and their expected reward. At this stage, the aim will be for the agent to begin to exploit this knowledge. One way of accounting for this trade-off is to use a decay parameter. This ensures that the policy begins as explorative, encouraging the agent to explore multiple options at each state at the beginning, but over time becomes more exploitative (greedy) over time. As a result, at the end of each auction episode (*ep*), ε is updated by a decay factor (*df*), which is dependent on the current value of ε:

$$IF\ \varepsilon_{ep} > d:$$
$$\varepsilon_{ep+1} = \varepsilon_{ep} * df_1$$
$$ELSE:$$
$$\varepsilon_{ep+1} = \varepsilon_{ep} * df_2$$

It must be noted however that this approach does not work entirely for this problem set due to the fact there is the potential for multiple zero and negative values to select from when utilising the argmax policy above. As a result additional Boolean logic need to be integrated into this policy selection process to mitigate this. This logic is utilised when for a given state only negative and zero valued Q values exists for the given action set. As a result the action that is selected for a given state then is a random action from the set of actions that have a Q –value of 0.

5. Setting the Parameters for Q-Learning

The structure of Q and R matrices are determined by pre-defined parameters. This project will use an auction scenario with a maximum bid of 5. Therefore, the possible states are from 1 to 5, as well as the initial state of no bid (*nan*) and the exit state (*Exit*). This results in a 7 x 7 grid. However, as this is auction scenario, we also have a time element to consider, determined by the number of timesteps allowed. This adds a third dimension to our grid from the start time (t = 0) to the auction end time (t = 5). This results in a 7 x 7 x 5 grid for our Q and R matrices.
We have defined our agent auction value as 3. This will be the basis for allocating rewards. If at the end of the auction the agent is below the value, it receives a positive reward. If over, it receives a negative reward. If the agents bid ends on the agent's valuation, a reward of 0 is received.
We will also experiment with a range of parameters for our alpha, gamma, epsilon and decay factor 1 and 2 (df_1 &df_2). The second decay factor (df_2) was kept at 0.01 to ensure when testing epsilon and df_1 had decreased past the threshold *d,* fixed at 0.2, the agent would change to an exploitative policy. Additional analysis was performed using a fixed epsilon value of 0.99 when testing the decay factors 1 and 2. The range for the decay factors is as below. The reason for this test was to establish what is an effective range of values but also to determine whether or not if there is dramatic shift in the epsilon value due to df_2 being close to 0 will performance of the model be effected significantly if significant enough exploration occurred during the exploration phase while df_1 was in effect for the policy. The full list of parameters are as follows:

Number of epochs(same meaning as RL Trials) = 10000
Alpha = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
Gamma = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
Epsilon ε= [0.9, 0.99, 0.999, 0.9999]
Epsilon df_1 = [0.9, 0.99, 0.999, 0.9999]
Epsilon df_2 = [0.01, 0.9, 0.99, 0.999, 0.9999]
Timesteps = 5
Agent Value = 5
Maximum Bid = 3

6. An example learning episode

As discussed, the Q-matrix, which stores the agent's knowledge of its environment, is updated on each learning episode based on the agents experience. This section runs through a typical learning episode, illustrating exactly how the Q-matrix is updated. For reference, the Q-learning algorithm [3] is shown below:

$$Q_{new}(s_t,a_t) = Q_{old}(s_t,a_t)+\alpha[(r(s_t,a_t)+\gamma.argmax\{Q(s_{t+1},a_{t+1})\})-Q_{old}(s_t,a_t)]$$

For the purposes of this demonstration, arbitrary parameters were selected as follows:

- Auction Duration (T) = 2
- Learning Rate (α) = 0.3
- Discount Factor (γ) = 0.8
- Item Value (V) = 4
- Maximum Bid Price = 5

Episode 1

The first timestamp of an episode always starts at 'nan' as no bids have been made yet. In accordance with the exploration policy, a random number is selected for comparison with the pre-defined epsilon value. This random number happens to be less than epsilon, so a random action is taken – to bid at any value up to the maximum bid price, or to exit the auction. In this case, the agent elects the action to bid 2. Using the R-matrix, the reward for this action is 0 and the Q-matrix is updated accordingly.

t = 0

$Q_{new}(s_t,a_t) = Q_{old}(s_t,a_t)+\alpha[(r(s_t,a_t)+\gamma.argmax\{Q(s_{t+1},a_{t+1})\})-Q_{old}(s_t,a_t)]$

$Q_{new}(\text{nan},2,t_0) = Q_{old}(\text{nan},2,t_0)+\alpha[(r(\text{nan},2,t_0)+\gamma.argmax\{Q(s_1,a_1,t_1)\})-Q_{old}(\text{nan},2,t_0)]$

$Q(\text{nan},2,t_0) = 0 + 0.3\ [\ (\ 0 + 0.8 * 0) - 0\ ]\ ]$

$Q(\text{nan},2,t_0) = 0$

The Q-value is therefore updated with the new Q-value for the action of bidding 2 from state nan at time 0.

t = 1

Another random number is selected for comparison with epsilon. Again, it is lower than epsilon, and so a random action is selected again. This time, the random action is to stay at bid 2.

$Q_{new}(s_t,a_t) = Q_{old}(s_t,a_t)+\alpha[(r(s_t,a_t)+\gamma.argmax\{Q(s_{t+1},a_{t+1})\})-Q_{old}(s_t,a_t)]$

$Q_{new}(2,2,t_1) = Q_{old}(2,2,t_1)+\alpha[(r(2,2,t_1)+\gamma.argmax\{Q(s_2,a_2,t_2)\})-Q_{old}(2,2,t_1)]$

$Q(2,2,t_1) = 0 + 0.3\ [\ (\ 2 + 0.8 * 0) - 0\ ]\ ]$

$Q(2,2,t_1) = 0.6$

Since the auction has reached the time limit, the Q-matrix us updated accordingly with a reward of 0.6 for the action of staying at a bid of 2 at state 2 at the time where t=1.

Episode 2

t = 0

The episode restarts at t=0 with the agent at state 'nan'. A random number is selected which is less than epsilon, so the agent choses a random action to bid 2 again. The reward is 0 again, but this time $arg\{Q(s_1,a_1,t_1)\}$ is 0.6, due to the knowledge gained in the previous episode. The Q-matrix is updated as per the first episode.

$Q_{new}(s_t,a_t) = Q_{old}(s_t,a_t)+\alpha[(r(s_t,a_t)+\gamma.argmax\{Q(s_{t+1},a_{t+1})\})-Q_{old}(s_t,a_t)]$
$Q_{new}(\text{nan},2,t_0) = Q_{old}(\text{nan},2,t_0)+\alpha[(r(\text{nan},2,t_0)+\gamma.argmax\{Q(s_1,a_1,t_1)\})-Q_{old}(\text{nan},2,t_0)]$
$Q(\text{nan},2,t_0) = 0 + 0.3\ [\ (\ 0 + 0.8 * 0.6) - 0\ ]\ ]$
$Q(\text{nan},2,t_0) = 0.144$

The Q-value is updated for the action of bidding 2 from state nan at time 0.
t = 1
A random number is generated again. This time, the number happens to be larger than epsilon, so instead of choosing a random action. The agent choses to exploit its knowledge gained thus far, and chose the action with the highest immediate reward. The possible actions at state 1 are A ∈ {2, 3, 4, 5, exit}. Using its knowledge, the agent choses to move to bid 2, as it knows that the reward for this is 0.6, which is the argmax of all the other options.

$Q_{new}(s_t,a_t) = Q_{old}(s_t,a_t)+\alpha[(r(s_t,a_t)+\gamma.argmax\{Q(s_{t+1},a_{t+1})\})-Q_{old}(s_t,a_t)]$
$Q_{new}(2,2,t_1) = Q_{old}(2,2,t_1)+\alpha[(r(2,2,t_1)+\gamma.argmax\{Q(s_2,a_2,t_2)\})-Q_{old}(2,2,t_1)]$
$Q(2,2,t_1) = 0.144 + 0.3\ [\ (\ 2 + 0.8 * 0) - 0.144\ ]$
$Q_n(2,2,t_1) = 0.7008$

7. Q-learning Results

7.1. Grid search for Epsilon ε and Epsilon decay df_1

For this grid search, the following parameters were alpha and gamma values where held fixed at 0.9 and 0.9 respectively for maximum exploration. Overall the lowest rewards where directly tied to the fastest converging systems (See Figure *2* and Figure 3). This is as anticipated as with convergence occurring so rapidly correct exploration to achieve an optimal policy prior to exploitation is not achieved. Due to the fact that the agent can only place bids higher than all previous bids, an exploratory policy tends to force the probability of selecting an overbidding or exit action more likely which results in significant numbers of zero or negative rewards. Hence when an agent switches to the optimal policy quite rapidly due to quick convergence there is a high probability of its policy resulting in either negative or zero results. By contrast utilising a large decay constant produced optimal performance as convergence occurred after significantly longer period of exploration had been performed in the state action matrix.

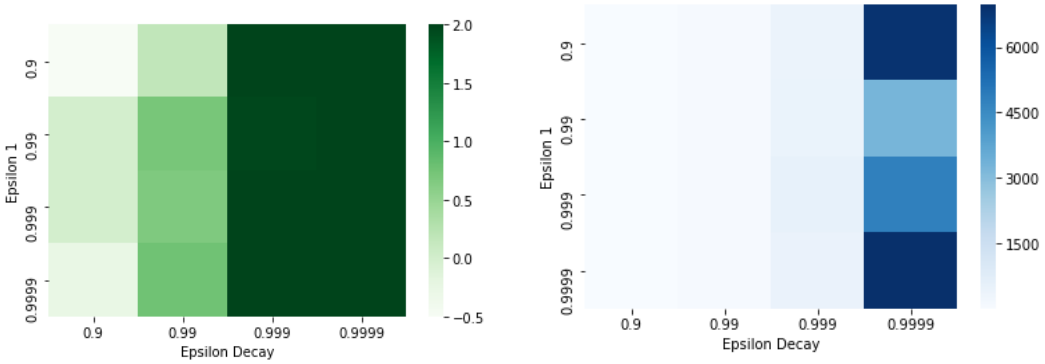

Figure 2 (left) *Average rewards over the last 100 epochs w.r.t to Epsilon and Epsilon decay values.*
Figure 3 (right) *Numbers of steps until convergence of Q matrix w.r.t to Epsilon and Epsilon decay values*

Grid search for the Epsilon decay factors df_1 and df_2

For the gridsearch for the decay factors the fixed parameters utilised where kept fixed with $\alpha = 0.8,\ \gamma = 0.9\ and\ \varepsilon_{T=0} = 0.99$. The results can be seen in Figure 4 and Figure 5 respectively. High alpha and gamma values where utilised to ensure quick convergence. What was found was that when low decay factors where utilised in combination average rewards where typically lowest here. This was due to the policy method switching to quickly to exploitation policy and hence not exploring the state action space to determine an effective strategy. It was found however that when df_2 was set low at 0.01 and df_1 was at its highest setting optimum convergence was still achieved. This denotes that when enough exploration is achieved with df_1 minimal exploration during the exploitation phase is required due to the optimal strategy being determined at this point. However it must be noted also that when df_1 factors are <0.999 and df_2 is set at 0.01 the average per step visitation is <5 occurrences for the specified state action matrix. As a result the optimum parameters to utilise are 0.9999 for df_1 and 0.01 for df_2 as this ensures that approximately >20visitations per state action in the Q-matrix ensuring a more stable and dense Q-matrix as a result.

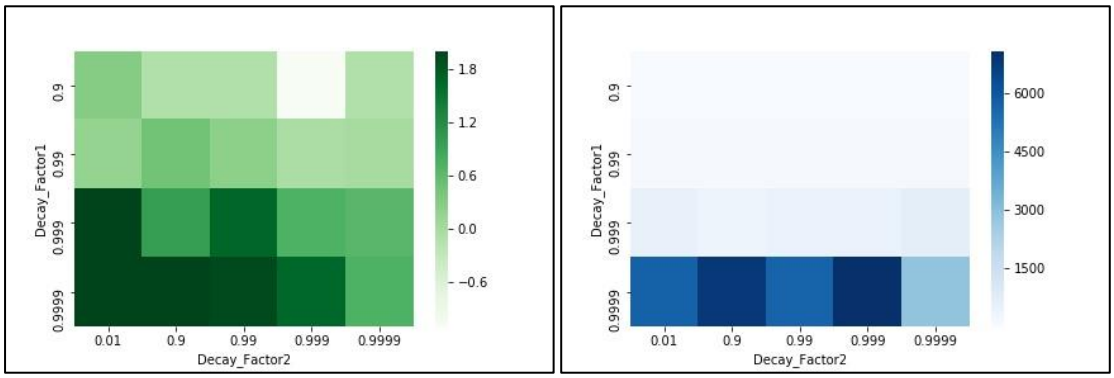

Figure 4 Left Average final 100 reward w.r.t to varying decay factors df_1 and df_2
Figure 5 Right Convergence of Q values w.r.t to varying decay factors df_1 and df_2

7.2. Grid search for learning rate α and Discount factor λ

For this grid search from the previous Epsilon ε and Epsilon decay df_1 and df_2 value grid search where kept constant at 0.99, 0.9999 and 0.01 respectively. The number of time steps, actions and states where kept constant also as before.

An overarching trend was observed, illustrated in Figure 6 with regards to overall average rewards achieved with low learning rate alpha and discount factor gamma value on average producing lower rewards with higher alpha values producing on average the highest rewards. This appears to be the strongest factor in dictating final reward, as even when gamma even is at its highest with high alpha values optimum performance is still achieved. The reason for why this is case is due to the fact that with high alpha values ensure satisfactory exploration of the entire policy space can be achieved and as a result the optimal policy and strategy for implementing achieving this this result can be acquired more efficiently. The fact that the discount factor does not have impact the final result though does not take into account how it stabilises the overall exploration policy however. With decreasing gamma values the overall exploration policy is more conservative due to the forward looking nature of the algorithm and as can be seen in Figure 9 the mean and variance of the Q table is less negative and lower respectively when lower discount factors are utilised ensuring that the all polices explored are more conservative in nature. Lower discount factors can ensure that this is adhered to also.

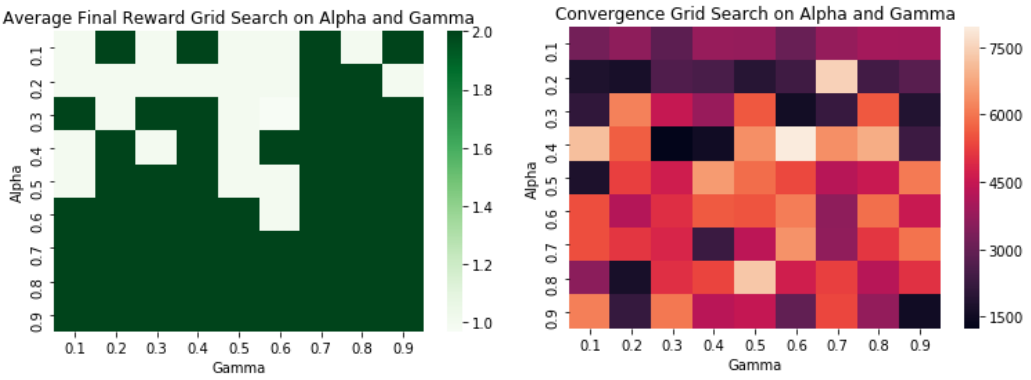

Figure 6 (left) Q-learning gridsearch heatmaps of Average final reward wrt to learning rate alpha and discount factor Gamma
Figure 7 (right) convergence epoch wrt to learning rate alpha and discount factor Gamma

Figure 7 illustrates the effect alpha and gamma configuration on the convergence speed of learning. Convergence was determined by first finding the cumulative absolute sum value of the configurations over 10,000 epochs. The convergence point was then calculated at the point where the cumulative lines 'plateaued' - where the cumulative value at an epoch was within 7% of the configurations final cumulative value (see Figure 8). A few trends are apparent. With low alpha values, the solution is faster to converge. This is expected as lower learning rates mean that he agent acts more conservatively, less inclined to explore its environment, and thus settle at an optimal solution more quickly. Convergence occurs noticeably quickly when both alpha and gamma are 0.9 - this is likely due to the agent being able to utilise its knowledge of future rewards effectively in this auction scenario. Above an alpha threshold of ~0.3, convergence appears to take longer to converge as the agent is more inclined to explore its environment for a longer time. This is with the exception of configurations with high alpha and low gamma values seen in the bottom left corner of Figure 7. In these instances, convergence appears to happen quickly, but the high learning rate instigates some intermitted jumps at points the rest of the epochs. Generally, the overarching trend is that configurations with low alpha values, and configurations with high alpha values and low *or* high gamma values resulted in faster convergence. When both alpha and gamma are initialised in the mid-range of 0 and 1, convergence generally is slowest, which is likely due to the fact that neither of the inherent behaviours controlled by alpha or gamma takes precedence, resulting in an agent that behaves in a way that takes longer to reach an optimal solution.

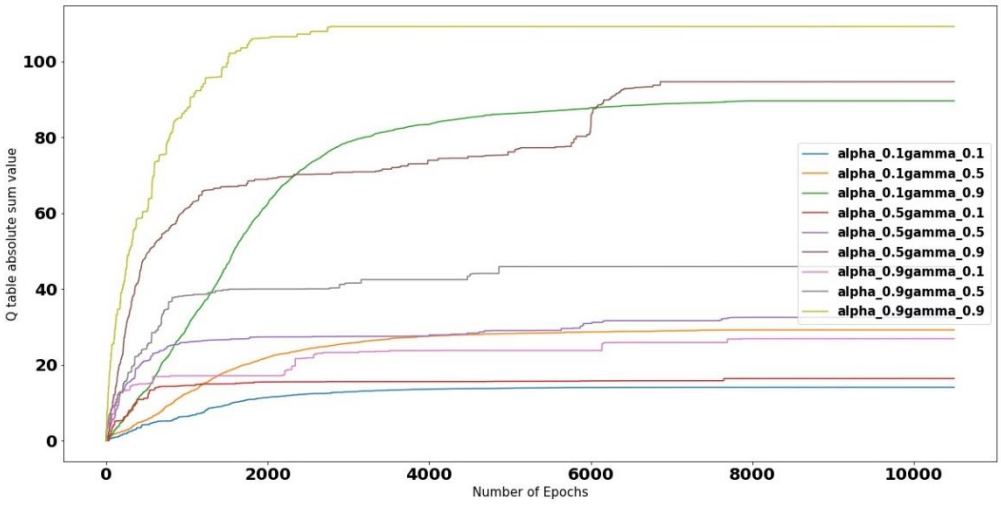

Figure 8 Q table absolute sum value per epoch vs No of epochs

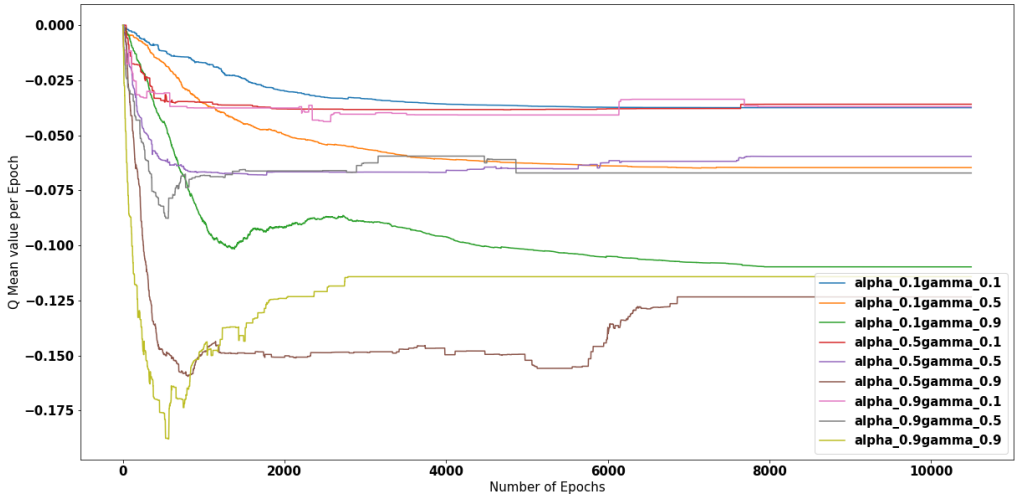

Figure 9 Q table mean value per epoch vs No of epochs

8. Advanced Reinforcement Learning

8.1. General Sum

As part of the experiments with multi-agent systems, we model a two-player scenario where the number of timesteps was 5, agent auction value of 2 and max bidding price of 3. The learning rate was kept high at 0.9and gamma value of 0.9 to ensure optimal convergence. Both agents have the same valuation of the auction item and therefore identically structured payoff matrices. In each time period, each player is allowed to make one bid, with one player always taking the first bid. Due to the consistent order of the bids, the agent that takes the second bid is always in a position to outbid the first player at the final timestep.

The multi-agent scenario means that a different Q-matrix structure is needed, as the actions of two agents are considered rather than just one. Therefore, the structure is S^2xA^2xT in a two player situation.

8.2. Multi Agent Experiments and SARSA modelling experiments

Multi agents reinforcement learning (MARL) and SARSA [(State Action Reward (Next)State (Next) Action) where done additionally to the initial Q learning experiments. The goals of these experiment where to assess the following null hypothesis:

1. Convergence is not possible with regards to 2 player stochastic general sum Auction games when using Nash Q-learning [4] and Friend or Foe Q learning (FFQ) [5]algorithms.
2. SARSA models [2] as an on-policy method will converge in a more conservative fashion than Q learning resulting in higher per epoch rewards during exploration/exploitation.

Overall it was found that Null hypothesis 1 was accepted and 2 was rejected.

Nash-Q learning assumes that all agents have knowledge of the other agents Q functions and update their own Q function using the following update rule as a result:

$$Q_i[s, a_1, \dots, a_n] \coloneqq Q_i[s, a_1, \dots, a_n] + \alpha(r_i + \gamma \cdot \mathrm{Nash}_i(s, Q_1, \dots, Q_n) - Q_i[s, a_1, \dots, a_n]) \quad \textit{Eqn(1)}$$

$Nash_i$ is denotes as the Nash equilibria found in the set of values of each players respective Q functions.

For FFQ learning takes a different approach where it makes no assumption of any knowledge of the other players Q matrices. As a result under the foe setting the expectation($Nash_i$ element of equation [*1*]) is that each agent is working to maximise its own payoff while subject to the optimisation constraint that it assume all other agents are attempting to minimise its reward(See equation [*2*]).

$$\mathrm{Nash}_1(\mathrm{s}, \mathrm{Q}_1, \mathrm{Q}_2) = \max_{\pi \in \Pi(\mathrm{A}_1)} \min_{a_2 \in A_2} \sum_{a_1 \in A_1} \pi(a_1) Q_1[s, a_1, a_2] \qquad \textit{Eqn(2)}$$

Null hypothesis 1 was accepted with convergence not occurring in either MARL model. There are 2 causes that pertain to the limitations in the assumptions of both Nash-Q and FFQ learnings. In Nash-Q learning as stated above it requires to find that there must be adversarial equilibria present in order for FFQ convergence to occur. Reviewing the final reward matrix at the terminal time state T_{end} (See Figure 10) from which all equilibria positions propagate from it can be seen that no true adversarial exits in FFQ for this game. It must be noted also that this is present in all auction games irrespective of the player's valuation of the auction item as in all auctions the losing player never gets punished for not acquiring the item in this game only if the player overspends past its budget.

		P1 Player Actions						
		NaN	1	2	3	4	5	Exit
P2 Player Actions	NaN	0,0	2,0	1,0	0,0	-1,0	-2,0	0,0
	1	0,2	0,0	1,0	0,0	-1,0	-2,0	0,2
	2	0,1	0,1	0,0	0,0	-1,0	-2,0	0,1
	3	0,0	0,0	0,0	0,0	-1,0	-2,0	0,0
	4	0,-1	0,-1	0,-1	0,-1	0,0	-2,0	0,-1
	5	0,-2	0,-2	0,-2	0,-2	0,-2	0,0	0,-2
	Exit	0,0	2,0	1,0	0,0	-1,0	-2,0	0,0

Figure 10 Nash equilibrium payoff matrix for 2 player Auction game where both players have the auction evaluated at the same price.

For Nash-Q learning convergence could not be obtained due to the high number of degenerate games present resulting in all actions for the agent having equivalent likelihood. NashPy python library was used to determine this across the initial reward state where there was a total of on average 14 different degenerate equilibria points to choose from for each player when utilising this algorithm. This in essence makes each time point stage selection in the Markov Decision process a near random decision at each time point. Therefore this highlights the issues already known with regards to both of these models that when applied to more complex non binary state matrix games the instability of these algorithms becomes apparent making convergence unlikely due to the restrictive natures of the constraints of these models [6].

SARSA model is a type of a different temporal difference model which while similar to Q-learning algorithm differs in one significant fashion in that it is an on-policy method. This means that SARSA learns the Q-value based on the action performed by the current policy instead of the greedy policy in Q-learning (See Equation [3]). This generally means that SARSA produces a more conservative strategy than Q learning as it

$$\mathrm{Q}[s_t, \mathrm{a}_t] := \mathrm{Q}[s_t, \mathrm{a}_t] + \alpha(\mathrm{r}_{t+1} + \gamma \cdot \mathrm{Q}[\mathrm{s}_{t+1}, \mathrm{a}_{t+1}] - \mathrm{Q}[\mathrm{s}_t, \mathrm{a}_t]) \qquad \text{Eqn(3)}$$

Overall it was found that the null hypothesis was rejected for the SARSA model compared to Q learning. It was found that on the whole SARSA explored significantly more negative region of the exploration space during exploration than Q learning (See figure Figure 11 L and R for example). The specific reason as to why this occurs it is postulated is down to the fact that with 60% of the initial rewards from this game being either negative or zero values and due to the fact that the SARSA model propogates the Q values made by the agent as opposed to the one with the maximum value the rewards are more negative than the Q-learning algorithm during exploration by comparison as a result.

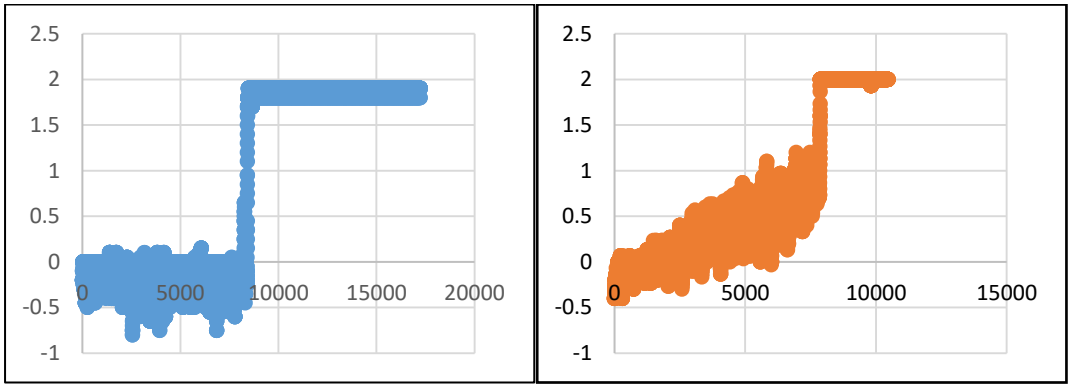

Figure 11 Moving Average rewards per epoch wrt to number of epochs for SARSA model (Left) and Q –learning model (right) with gamma and alpha are stated at 0.1 and 0.1 respectively

9. Conclusion

Experiments in this project proved that parameterisation of the Q-learning algorithm is extremely important in this auction scenario. Epsilon and Epsilon decay parameters had a profound impact on performance in terms of reward and speed. Epsilon decay was singled out as being extremely important in these respects, and experiments found that although high epsilon decay values would result in the model taking longer than low values to converge, these would be necessary to ensure that highest average rewards would be received by the end of the trials. Alpha and gamma parameters were also important factors in the Q-learning models. To maximise final reward, it was found that avoiding a combination of both low alpha *and* low gamma values should be avoided. An optimal strategy would need to incorporate high alpha values, with gamma values less important if the alpha value was high. Again, this was found to result in maximising rewards, albeit at the expense of convergence speed, which increased with higher alpha values.

For MARL Nash and FFQ Q-learning convergence could not be achieved. This was due to the fact of in both cases either the lack of specific adversarial equilibria being present for FFQ or there being too many degenerate Nash matrix options to select from when utilizing Nash-Q meant that neither method could reach convergence. This again denotes the limitations of utilizing these methods for non-bimatrix games where more than two options are available to each agent. Suggested future work with regards to evaluating this approach would be to analyze the potential of a random walk methodology to overcome the degenerate Nash game issue for Nash-Q learning and look to utilize Bayesian Nash Q learning which has been found to work in real life automated auction systems [7].

For SARSA while on average it did provide equivalent performance to Q-learning in terms of rewards it did not provide a more conservative behavior. It is postulated that this could be due to the higher frequency of negative or zero values being the cause for this however further research into the cause of this is proposed in order to better understand this phenomena.

Essay 1: Parallelisms between the Q-learning algorithm and error correction models in psychology

Reinforcement learning is a branch of machine learning whereby an artificial agent learns by interacting with its environment and makes decisions based on its experiences from these interactions over time. Q-learning, developed by Chris Watkins in 1989 [3], is a popular algorithm for reinforcement learning. Its main aim is to iteratively update its knowledge as it gains more experience, to the point where it understands enough about its environment so that it can exploit its knowledge and take actions that maximise long term reward.

In recent years similar computational models that follow similar premises have been developed in psychology, such as the Rescorla-Wagner (RW) model [8] and Temporal Difference (TD) model [9]. These are error correction models of associative learning which aims to calculate how much a stimulus predicts a given outcome. These have become two of the most successful models of associative learning in psychology.

There are many parallels that can be drawn between the Q-learning algorithms and error correction models. Perhaps the most obvious comparison between the two is how the models are initialised. Both models start with no knowledge of their environment. For the Q-learning algorithm, the Q-matrix (where cumulative knowledge of the agents environment is stored) in unpopulated prior to any runs being carried out. Likewise, error correction models start with no knowledge prior to any trials occurring.

Both models therefore also have similarities in how they gain knowledge. A Q-learning agent will populate its Q-matrix based on interactions with the environment around it, and similarly, popular error correction models will update their knowledge after a learning experience occurs. Both models therefore follow very similar knowledge update mechanisms, whereby the general prediction equation is the previous prediction, plus any learning experienced.

There are also parallelisms to be drawn between how both models approach convergence. Q-learning will continue to learn from its environment for as long as it explicitly programmed to. There are mechanisms to alter this, but the overarching idea is that the algorithm will continue to learn until it is populated its Q-matrix and has a sufficient view of the environment around it and will converge appropriately [10]. Similarly, the learning curve of the RW model indicates that the associative strength increases as the number of trials increases, as the more you know, the less you need to learn resulting in a curve learning curve that plateaus due to the fact there is a limit to knowledge and there is nothing to be learned after asymptote.

Both the Q-learning model and error correction models contain an important parameter that controls how fast the algorithm modifies its estimates. One expects to start with a high learning rate, which allows fast changes, and lowers the learning rate as time progresses [11]. Likewise, this parameter is important in psychological error correction models. Further, the TD model employs a discount factor (gamma) that represents the fact that predictions similar to final outcome are in principle more accurate than previous ones, and more important when calculating errors. The discount factor is important in Q-learning for determining the importance of future rewards.

In conclusion, there are inherent fundamental differences between the Q-learning algorithm and various error correction models in psychology, which can be expected from algorithms designed for different applications. Despite these, there are also many similarities between them, including their overall aims, mechanisms, solution convergence similarities and parameter considerations.

Essay 2: How error correction models could be implemented as (advanced) reinforcement learning architectures

Reinforcement algorithms as a whole are typically divided into two broad categories algorithms for prediction or control [2]. Equivalent categories exist with regards to how learning occurs in psychology specifically classical/Pavlovian and instrument /operant conditioning [2]. The main Error correction models used in these fields is the Rescorla Wagner model (See Equation [4]) where:

1. V_i^n is the increase in associative strength between an unconditioned (US) and conditioned stimuli(CS).
2. Specific learning rates $\alpha_i * \beta_i$ which represent the conditioned stimulus(CS or outcome) and unconditioned stimulus respectively
3. γ^n presence of CS in trial being analysed.

$$V_i^n = V_i^{n-1} + \alpha_i * \beta_i * (\gamma^n - \sum_{i=A}^{Z} V_i^{n-1} X_i^n) \qquad \text{Eqn(4)}$$

There are a number of specific properties within this model which are of interest and could be utilised in reinforcement learning specifically Q-Learning algorithms:

1. Specific learning rates $\alpha_i * \beta_i$
2. The blocking of effect that is captured by this model for similar salient stimuli

Specifically where these properties of the error correction (EC) models can provide value to Reinforcement learning algorithms is with regards to overcoming some of the limitations that are present with present reinforcement learning approaches specifically being:

1. The overall learning rate is static for all rewards denoting that the concept of contingency or the saliency of different types of reward that the agent receives for its action is not captured other than in its absolute value.
2. There is no association between the impact of different consecutive types of similar stimuli captured in RL i.e. the blocking effect.

This is where the RW ECM model parameters could provide additional nuance to RL models specifically with regards to models that pertain to different categories of rewards as with regards to physical stimuli i.e. temperature, air pressure, light etc. Therefore there are two specific proposals that are put forward to add to RL Q-learning algorithms(Reference equation [5]) in order to effectively capture these elements of the associative learning EC models that being:

1. Introduce different alpha and beta learning rate values $\alpha_z \beta_z$ for different categories of actions and rewards as per the RW model in order to better differentiate between them as opposed to stimulus only reward.
2. Utilise a per epoch categorical count of each type of stimuli z_i in the environment equivalent to the lambda model captures how repeated stimuli degrade in learning over time.

$$\mathrm{Q}[s_t, \mathrm{a}_t] := \mathrm{Q}[s_t, \mathrm{a}_t] + \alpha_z \beta_z \left(\mathrm{r}_{t+1} + \gamma \cdot \arg\max_a Q(\mathrm{s}_{t+1}, \mathrm{a}) - \sum_Z \mathrm{Q}[\mathrm{s}_t, \mathrm{a}_t] \right) \qquad \text{Eqn(5)}$$

With these parameterisations in effect RL algorithms have the potential to more effectively capture stimulus to stimulus associated and the impact of blocking between equivalent stimuli. This would in turn mean that the exploratory behaviour of the model could improve its default exploratory nature which would inherently make it better at generalising to dynamic environments as a result.

References

[1] C. Han, K. Ren, W. Zhang, K. Malialis, J. Wang, Y. Yu and D. Guo, "Real-time bidding by reinforcement learning in display advertising," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, Cambridge, 2017.

[2] R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction, Cambridge: MIT press, 2018.

[3] C. J. C. H. Watkins, Learning from Delayed Rewards, Kings College, London, 1989.

[4] H. Junling and M. P. Wellman, "Nash Q-learning for general-sum stochastic games," *Journal of machine learning research,* vol. 4, no. Nov, pp. 1039-1069, 2003.

[5] M. L. Littman, "Friend-or-foe Q-learning in general-sum games," *ICML,* vol. 1, pp. 322-328, 2001.

[6] X. Wang and T. Sandholm, "Reinforcement learning to play an optimal Nash equilibrium in team Markov games," in *NIPS*, Cambridge, 2003.

[7] R. Gomes and K. Sweeney, "Bayes–nash equilibria of the generalized second-price auction.," *Games and Economic Behavior,* vol. 86, pp. 421-437, 2014.

[8] R. Rescorla and A. Wagner, "A Theory of Pavlovian Conditioning: Variations in the Effectiveness of Reinforcement and Nonreinforcement," in *Classical Conditioning II: Current Research and Theory*, New York, Appleton-Century-Crofts, 1972, pp. 64-99.

[9] R. Sutton and A. Barto, "A temporal-difference model of classical conditioning," in *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, 1987.

[10] F. Melo, "Convergence of Q-learning: a simple proof," Institute for Systems and Robotics, Instituto Superior Técnico, Lisbon.

[11] E. Even-Dar and Y. Mansour, "Learning Rates for Q-learning," *Journal of Machine Learning Research,* vol. 5, pp. 1-25, 2003.

[12] S. P. Singh and R. S. Sutton, "Reinforcement learning with replacing eligibility traces," *Machine learning,* vol. 22, no. 1-3, pp. 123-158, 1996.