Multi-Agents Reinforcement Learning In Iterative Voting

Loujain Liekah, Jean Monnet University, Machine Learning and Data Mining

Supervisor. Dr. Umberto Grandi, IRIT, University Toulouse 1 Capitole

Abstract

In this paper we tackle the challenge of Multi Agents Reinforcement Learning (MARL) in a situation of collective social choice. We evaluate the learning performance of multiple independent learning agents interacting in an iterative plurality voting game, which is a competitive game where each agent has her private ordered preferences over a number of questions, the players submit their voting action, and the winner has the highest number of votes calculated with plurality rule. Agents receive a reward signal depending on the winner of the previous iteration. Each agent's goal is to maximize her individual long term reward. We show in our simulations that the population of agents learn to make a better collective decision. We model each voting agent as a multi-armed bandit machine and discuss the parameters that affect the outcome of the elections conducted by agents exploring under ϵ-greedy and Upper Confidence Bound (UCB) learning strategies with respect to a variant of Borda score, namely Aggregated Score Index (ASI) that reflects the overall satisfaction and the quality of the winner according to social choice criteria. We illustrate the effect of two different reward functions on the exploration strategies and discuss the impact of the size of action space and number of agents on the learning process.

Keywords: Reinforcement learning, Multi-agents, Iterative voting, Game theory

1 Introduction

Reinforcement learning (RL) has a growing value because of its potential to solve a considerable number of sequential decision making tasks with feedback like games, control and recommender systems; it is proposed to solve many challenges in game theory and multiagent systems. Repeated games have been studied for a long time because of their easy rules and they provide a very good setting for multiple artificial intelligence tasks. RL challenges planning approaches like tree search by approximating a value function, proposed by (Shannon, 1950) for chess, these value function approximators were represented

as a feature vector for states, and a weight vector that is learned by deep neural networks to approximate the value function (DQN)(Mnih et al., 2015), also employed in Alpha-go by (Silver et al., 2016) where the game was learned using two deep neural networks to approximate the value function and the policy and achieved super human results.
Social choice theory is concerned with methods used in decision making to choose an alternative among different candidates, in referendum elections voters submit their ballots on multiple proposals simultaneously, a paradoxical outcome is the separability problem, it happens when the winner is the last preference of every voter. Iterative voting is the setting where election is repeated so that voters can revise their policy depending on the winner of the previous iteration which indicates the current state of election. It is proposed to solve the separability problem since voters can change their sincere view and vote strategically in order to obtain a better collective result. (Bowman et al., 2014). The iterative voting is a repeated games and can be modeled as a mutiarmed bandit (MAB) to assess the learning capabilities of voting agents to make a good collective decision(Airiau et al., 2017).

In this work we propose a multi agent reinforcement learning (MARL) setting with independent learning agents playing an iterative voting game. We study the learning performance of the agents following $\epsilon - greedy$ and Upper Confident Bound (UCB) exploration policies, with two proposed reward functions. We show that our agents learn to make a better collective decision under social quality measurements. We discuss the effect of the reward signal, the size of action space and the number of agents on the quality of the voting result with these approaches.

Outline: The rest of the report is organized as follows. Section 2 presents principles of reinforcement learning, multi-armed bandits, multi-agents reinforcement learning, repeated games and iterative voting. Section 3 introduces the problem we are tackling and how we formalize it, Section 4 explains the simulations environment and presents the results of the experiments. Finally we conclude in section 5.

2 Preliminaries

2.1 Classical Reinforcement Learning

"All goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal (called reward)" (Sutton and Barto, 2018). The concept of reinforcement learning is that an agent learns how to map states and actions by sufficient trial-and-error contact with its environment. To formalize a reinforcement learning task: we represent the environment as a Markov Decision Process (MDP), then formulate the learning objective by assigning an intermediate reward for each step which specify the policy to be learned, and finally solve an optimization problem to find the optimal policy. At each discrete time step t the agent receives an observation that represents the environment's state s_t; the agent then takes an action a_t that transits the environment to a new state and receives a scalar reward signal r_t that indicates the quality of this transition. However the signal might be delayed or assigned to a sequence of actions. The environment is modeled as a Markov Decision Process (MDP).

Definition 2.1. A finite Markov decision process is a tuple $< S, A, P, R, \gamma >$
S is a finite set of environment states.
A is a finite set of agent's actions.
P is state transition probability matrix,$P^a_{ss'} = P[S_{t+1} = s'|S_t = s, A_t = a]$.
R is a reward function, $R^a_s = E[R_{t+1}|S_t = s, A_t = a]$, the expected reward of taking action a in state s
γ is a discount factor, $\gamma \in [0, 1]$ which avoids infinite sum.

Figure 1 shows a representation of this interaction.

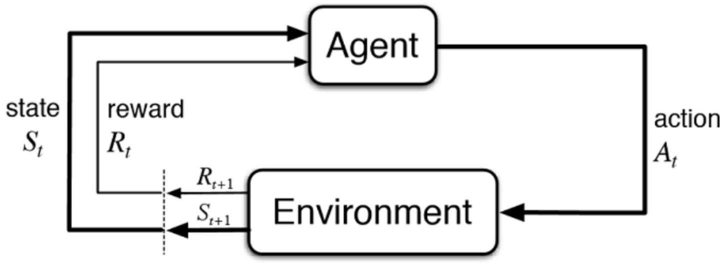


Figure 1: Agent-Environment interaction in Markov decision process

An information state (Markovian) contains all useful information from the history; "The future is independent of the past given the present".

Definition 2.2. Markov property: A state S_t is Markovian if and only if:

$$P[S_{t+1}|S_t] = P[S_{t+1}|S_1 \cdots S_t]$$

The trajectory or the episode is a sequence of states, actions, rewards like: $s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \cdots$

Definition 2.3. The return G_t is the sum of the discounted reward from timestep t onward
$G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \Sigma_{k=0}^{\infty} \gamma^k R_{t+k+1}$

A reinforcement Learning agent may include one or more of these components: Policy, Value Function, Model.
The behaviour of the agent is identified by its **policy** π. The policy indicates how the agent chooses its action depending on the current environment state.

Definition 2.4. A policy π is a distribution over actions, given states $\pi(a|s) = P[A_t = a|S_t = s]$

The goal of the agent is to maximize her long term reward. The traditional techniques allow a single agent to learn the optimal policy by trial-and-error interaction with the environment, assuming that if the agent has enough interactions with the environment or a sufficient load of trajectories, it ensures convergence to the optimal policy π^* (Nowe et al., 2012).
Most reinforcement learning algorithms aim to learn the optimal value function q^* for control tasks, or to estimate the value function of an action, a state or a pair (state,action) under a given policy q_π; which indicates how good it is to take this action in the current state; that is an expectation of the discounted future rewards or the return.

Definition 2.5. The state action value function for policy π denoted $q_\pi(s, a)$ is defined using the bellman equation by (Bellman, 1957) as the expected return for taking action a in state s following policy π :
$q_\pi(s, a) = E_\pi[G_t|S_t = s, A_t = a] = E_\pi[\Sigma_{k=0}^{\infty}\gamma^k R_{t+k+1}|S_t = s, A_t = a]$

The value function is usually estimated from trajectories using *Monte Carlo* methods that learns from complete episodes and estimate the value as the mean return, or *Temporal-Difference* TD learning that learns online from incomplete trajectories by bootstrapping and updating an estimated return at each time step. As the number of trajectories approach infinity these estimation converges to the true value function.
Solving the reinforcement learning problem means for the optimal policy π^* that corresponds to the optimal value function, we achieve that by the principle of optimality (Bellman, 1957) which is the basis for value iteration algorithms to compute q^*

$$q^*_\pi(s, a) = \max_a q_\pi(s, a) = \max E_\pi[\Sigma_{k=0}^{\infty}\gamma^k R_{t+k+1}|S_t = s, A_t = a]$$

If we have the MDP model $< S, Q, P, R, \gamma >$ and we are following a policy π that we want to evaluate, our task is to find the value function $q_\pi(s, a)$ corresponding to this policy. Improving a policy is attained by acting greedily with respect to the value function.$\pi'(s) = argmax_{a\in A} q_\pi(s, a)$. An optimal policy π^* can be divided into an optimal first action a^* and an optimal policy behaviour from the new state s'.

2.2 K-armed Bandit Problem

A famous reinforcement learning problem is the multi-armed bandit, where we need to learn the best action (bandit).

Definition 2.6. A multi-armed bandit is a tuple $< A, R >$
A is a set of k different actions.
$R^a(r)$= $P[r|a]$ is unknown probability distribution of rewards over actions.

At every time step t the agent selects an action arm $a_t \in A$, receives a reward from the environment $r_t \in R^{a_t}$. The objective of the agent is to maximize the cumulative reward over a period of time $\Sigma_{\tau=1}^{t} r_\tau$

In the multi-armed bandits problem each arm has an expected mean reward that represents the estimated value of this action $q_t(a)$ and we would like this estimation to be close to the optimal value $q^*(a)$. The simplest action selection is the greedy which picks the action with the highest estimated value function; we can write it formally as $A_t = \operatorname{argmax}_a q_t(a)$. Greedy policy always exploits and maximize the immediate reward. A critical choice in online decision making between k different actions is between exploitation and exploration. **Exploitation** means taking the best available known action given the current data. **Exploration** is to take an action with less certainty value to collect new information in order to find the best overall behaviour that enhances the final result. Finding the right balance is generally difficult as the returns from exploration actions are less certain and might be more preferable after a certain history or in specific runs (March, 1991)
There exists different exploration principles, a simple but efficient one is the $\epsilon - greedy$ strategy, which usually acts greedy but with a small probability it chooses an action randomly independently from its estimated value, and with similar probabilities for all actions. This policy guarantees with time testing all possible actions and will converge to the optimum value function $q_*(a)$ (Sutton and Barto, 2018)

$$A_t = \begin{cases} \operatorname{argmax}_a q_t(a), & \text{with probability } 1-\epsilon \\ \text{a random action}, & \text{with probability } \epsilon \end{cases}$$

Upper Confidence Bound (UCB) is an exploration strategy that selects the actions which are more likely to be optimal by taking into consideration both their estimation and the the uncertainty of their values.

$$A_t = \operatorname*{argmax}_a \left[q_t(a) + \sqrt{\frac{\ln t}{N_t(a)}} \right]$$

Where $\ln t$ is the natural logarithm of t the total number of iterations, $N_t(a)$ is the number of times action a has been selected. The uncertainty of an action decreases as $N_t(a)$ increases.

2.3 Multi-Agents Reinforcement Learning

A multiagent system is defined as a collection of autonomous entities interacting in a shared environment. In this case applying the traditional RL methods usually do not succeed, because the convergence requirements are not satisfied. There are many challenges in the simplest case where agents are learning in a shared environment with a single state because the environment becomes non-stationary and the markov property does not hold, therefore each agent is

pursuing a moving target (Buşoniu et al., 2010).
The interaction between multiple agents within an environment is shown in figure 2 from (Nowe et al., 2012), each agent selects its action and they are all submitted as a joint action to the environment. The environment responses with a reward and a new state for each agent. In multiagent system the state transitions and the rewards at time step t depend on the joint action of all agents, $a_t = [a_{1,t}, \cdots, a_{i,t}]$ where i denotes the number of active agents in the environment.

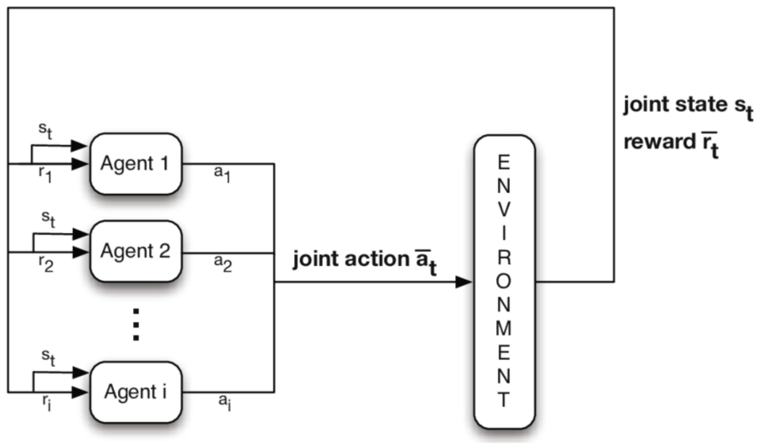


Figure 2: Multiple Agents Interacting in the Same Environment

We distinguish two main settings of multi agent learning,**independent learning** which reduces the multi agent learning problem into one agent learning by ignoring other agents and considering them as a part of the environment; in this approach we can still apply classical RL algorithms like Q-Learning by (Watkins and Dayan, 1992). The convergence features are not guaranteed; nevertheless independent learners result in good performance under many multiagent settings. A general multi-agent Q learning algorithm for stateless game is given below.

Algorithm 1 Multi-Agent Q Learning Algorithms in Stateless Games

```
$t = 0$ $q_k(a) = 0 \forall k, a$
repeat
for all agents $k$ do:
 select action $a_k(t)$
execute joint action $a = (a_1, \cdots, a_n)$
observe rewards $r_k$
for all agents $k$ do:
 $q_k(a) = q_k(a) + \frac{1}{N}[r_k(a) - q_k(a)]$
until termination
```

The second setting for multiagent learning is referred to as **joint-action learning**; here the MDP model is usually not sufficient when multiple agents interact, other approaches include improving a policy or approximating the value function of a joint action that is a combination of each agent's action. At every time step each agent updates its approximation with respect to the reward it got for this joint action.
The complexity of the problem grows when the environment is dynamic with more than a single state or requires a series of decision making; here the agents need to coordinate and identify the state of the environment, moreover it becomes even more difficult when the agents don't have sufficient information about the scheme because they usually can not recognize other agents' actions and rewards; although these actions take part of their own reward. In those cases either the agent is not informed of the existence of other agents, or the agent has all the information over other agent.
The key challenge in the multi-agent reinforcement learning (MARL) is defining a good learning goal; with two main identified objectives; stability of the agents' dynamics and adjustment to other agents' behaviour (Busoniu et al., 2008).

2.4 Repeated Games

Multi-agent reinforcement learning has a strong connection with game theory which studies strategic interactions between multiple players trying to maximize their individual payoffs, most of the results are concerned with stateless environments or repeated games, a static stateless game has empty state set $S = \Phi$, and normal form games where agents select their actions at the same time(Busoniu et al., 2008).
Stateless games usually provide a basic setting to experiment with MARL, specifically for independent learning agents and to question coordination frameworks. The learning question in this setting also rises from interacting with other agents. The difference in the reward of an action is determined only by the changes of other agents' policies, therefore the game is iterated to improve the policy over time. A fundamental difference from reinforcement learning, is that in RL the game rules are unknown and the learners have no prior knowledge of the reward function. Many RL methods suppose that the learner can model its opponents by observing their actions or rewards, while other approaches assume that these information could be unavailable.
Usually there are two kinds of games in **cooperative games** the agents share a common goal and they all try to maximize a shared reward signal, coordinating between agents is critical to reach optimum. The second case is **competitive games,** the agents have conflicting goals and a clear optimum may no longer reside. In this case,the optimal policy for a player is called *best response* if other players fix their policies. An *equilibrium* between agents is usually looked for, which is the case when no player benefits from playing any different policy, and it is the best response for all players.

2.5 Iterative Voting

Computational social choice is concerned with combining information from individuals into one collective perspective. In voting theory, agents have their personal preferences over a set of alternatives, these preferences are usually modelled as utility functions. Social choice offers techniques of aggregating these preferences for collective decision making. In collective choice, we differentiate between two voting behaviours, **sincere voting** is when the agent submits her truthful preference order, and **strategic voting** is when the voter manipulates and submits a ballot that is not sincere in hope for a better collective outcome. There are different voting rules that serve this objective, the most common voting rule is the **plurality voting** which assigns the winner as the alternative who is ranked at top position by the largest number of voters. This role is perfect for two candidates and is easy to implement with good convergence features, paradoxes are probable for more than two alternatives. It is however practical in an **iterative voting** setting in which the election has sequential phases, at each iteration voters are allowed to change their ballots after being exposed to the winner of the previous iteration. This setting is useful and guaranteed to converge at equilibrium (Meir et al., 2010).

In the paper Learning agents for Iterative Voting by (Airiau et al., 2017), authors formulate iterative voting as a repeated game and estimate the learning abilities of agents in a collective decision making setting, where agents engage in an iterated plurality election, every agent has their individual preferences regarding a number of alternatives. The winner of the ballot defines the reward signal for the learning algorithm. The authors restrict the information available to the agents in the environment to the winner of the previous iteration and allow agents to change their votes simultaneously. The authors evaluate the scalability of decisions made by the learning agents modelled as multi-armed bandit case and show that the agents learn to make better collective choice by assessing the winner under standard quality measures like Borda score and Condorcet efficiency. They estimate utility (reward) function for each candidate and control exploration using simple "optimism in face of uncertainty".

3 Problem definition

In referendum elections voters usually submit their votes on multiple proposals simultaneously. This type of multiple elections can lead to paradoxical outcomes, namely the **separability problem** which occurs when the favoured outcome on one proposal depends on the result of other proposals (Brams et al., 1997), the problem happens when the winner turns to be the last preference of every voter which fails to represent the will of the electorate.
An example of the separability problem is shown below with three voters submitting their vote as a **sincere** preference indicated by the **first row** of each matrix. The vote is binary where 1 represents yes and 0 represent no. For example having the top row as 110 means that the voter's sincere preference is

for the first and second proposal to pass, and the third to fail. in this example the winner of this iteration for sincere voters is 111 calculated by the majority vote, which is the last preference of every voter (Bowman et al., 2014)

$$\begin{bmatrix}1&1&0\\1&0&1\\0&1&1\\0&0&1\\1&0&0\\0&1&0\\0&0&0\\1&1&1\end{bmatrix}\begin{bmatrix}1&0&1\\0&1&1\\1&1&0\\0&1&0\\1&0&0\\0&0&1\\0&0&0\\1&1&1\end{bmatrix}\begin{bmatrix}0&1&1\\1&1&0\\1&0&1\\1&0&0\\0&1&0\\0&0&1\\0&0&0\\1&1&1\end{bmatrix}$$

Let *num_quest* denote the number of binary questions in the voting game, the number of possible answers is 2^{num_quest}, an ordered preference profile is presented by a matrix $2^{num\_quest} \times num\_quest$ where the ith row represents the voters ith preferred outcome.
Borda score is assigned to the winner in each profile preference matrix, the scores range between 0 and $2^{num\_quest} - 1$. A winner in row i for a profile p will receive a score

$$B_p(i) = 2^{num\_quest} - i$$

The Borda's score for our example with num_quest=3 is between 0 and 7, the score of the winner is shown below with respect to the profile preference of the first voter in our example.

$$\begin{array}{c}7\\6\\5\\4\\3\\2\\1\\0\end{array}\begin{pmatrix}1&1&0\\1&0&1\\0&1&1\\0&0&1\\1&0&0\\0&1&0\\0&0&0\\1&1&1\end{pmatrix}$$

To assess the global satisfaction of the society in the result of the election, we use the **Aggregate Score Index, ASI** defined by (Hodge and Schwallier, 2006), which is an average of all agents' Borda scores and was used by (Airiau et al., 2017). The aggregated score index of a winner w is denoted $ASI(w)$ is the averaged scores for all voters. Submitting the sincere preference here that is the first row in each of the preference matrices for the voters will end up with the winner being 111 that has a score of 0 for each profile and therefore ASI(111) = 0. An outcome 101 will have ASI(101) $= \frac{6+7+5}{3} = 6$.

Problem Formalisation

We want to study and assess the learning capabilities of agents playing a strategic iterative voting game and acting as independent learners, each agent ignores other agents and they all follow similar learning policies.
In our simulations we create an iterative voting game with *num_quest* is the number of questions in the election, *num_agents* is the number of voting agents. Every agent has a **preference profile** matrix that represents their choices ordered by the desired outcome. Agents submit their votes simultaneously and the result is an aggregation of these votes by majority vote on each question. We use an odd number of voters throughout the work to avoid tie situations.
To map the iterative voting game with the reinforcement learning setting, we need to define the actions and rewards for the voting agents. Each row of the preference matrix is a possible answer of the ballot and defines a single action for the respective agent; each of these actions is assigned a unique reward that is distributed after aggregating the actions and finding the winner by majority vote, each reward is proportional to the position of the winner in the respective preference matrix.
More precisely, we represent each agent by a multi-arm bandit machine following previous research (Airiau et al., 2017). We associate the bandits of each machine with the actions of the corresponding agent, each action being a bandit. The action space of an agent is mapped with its profile of preferences. At each iteration, agents cast their votes by submitting their actions simultaneously. The environment collects the joint action and calculates the winner of the current iteration using majority voting rule. The rank of the winner in one agent's profile affects her reward in two proposed reward functions. The counter and the q-value of the selected action-bandit is updated at each iteration for every agent depending on the reward signal.
We measure the **ASI** of the winner in each iteration, as the average of all agents' Borda scores, and observe the development of the score as the agents learn their best response in the non stationary environment.

$$ASI = \frac{1}{num\_agents} \Sigma_{a=1}^{num\_agents} B_a$$

Different rewards showed different exploration effect in single agent learning by (Tijsma et al., 2016). To study this effect in a MARL setting we define two reward functions that reflect the goodness of a winner in the agents' preference profiles.

Linear Reward

We define the linear reward for an agent as the Borda score of the winner in the profile matrix of the respective agent as explained earlier, that means the ASI is the average of the linear rewards for the learning agents. Precisely: Let the winner be on the ith row of the profile matrix for agent a, then the linear reward is defined:

$$R_a^{linear}(i) = B_a(i) = 2^{num\_quest} - i$$

The linear reward ranges between 0 for the case when the winner is rated last in the profile matrix and $2^{num\_quest} - 1$ for a winner in the first position.

Exponentially Decaying Reward (EDR)

We define the reward for agent a for a winner in the row i as

$$R_a^{exp}(i) = \frac{1}{2^i} = 2^{-i}$$

This reward decreases exponentially with respect to i the position of the winner in the profile matrix for agent a.

4 Simulations Settings

To evaluate the performance of our learning agents in iterative voting game, we explore the toolkits that implement reinforcement learning environments like OpenAI Gym by (Brockman et al., 2016) which provides an abstract more focusedd on deep reinforcement learning environment's observations and rewards and implements a collection of environments classes like Atari games, algorithmic and classic control reinforcement learning tasks. We also examine RLib by (Liang et al., 2017) which is a library more entitled with distributed reinforcement learning that proposes methods to enhance performance of RL algorithms. Finally we develop our own scheme *MARL-IV*[1] that implements the specifications of an iterative voting game. We develop the needed methods to create preference profiles for the agents and the dynamics of the environment to calculate the plurality winner of an iteration and distribute the rewards to the agents. We implement the agent as a MAB entity, and the exploration policies: greedy, ϵ-greedy and upper confidence bound (UCB).
We run simulations of an iterative voting game on elections where agents submit their votes on multiple binary proposals simultaneously. We calculate the winner and distribute the rewards, we execute the election iteratively to monitor the behaviour of ASI over time steps. The sole information available to each agent is the reward of the previous iterations. This information is used to repeatedly update the q-value of the bandits.

4.1 Experiments

We will measure the ASI score of each iteration and average 500 voting games per setting. We tune number of iterations for the voting game, $num\_iterations =$

[1]Code available on github `https://github.com/Loujainl/iterative_vote`

2000 and expect the ASI score to increase and stabilize.
To analyze the performance of our MARL model for iterative voting game, we run the simulations under different parameters and observe the behaviour of ASI obtained by the learning agents following ϵ-greedy and UCB, we tune $\epsilon = 0.1$ to be a sufficient noise for exploration in the greedy approach. We plot the ASI over the iterations and examine the main parameters affecting the learning process, we run our simulations to average 500 voting games.

4.2 Results

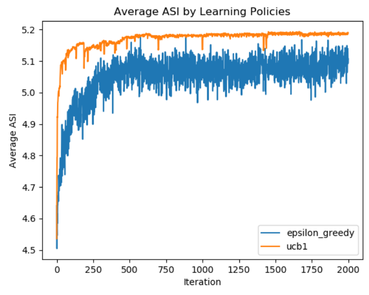
The number of questions in the ballot determines the complexity of the learning process because the action space is exponential with respect to the number of questions: $num\_actions = 2^{num\_quest}$ and the maximum attainable ASI score in this election is $ASI\_max = 2^{num\_quest} - 1$. We reason about the size of action space and the type of reward on the collective behaviour of the voting agents.

4.2.1 Number of Questions = 3

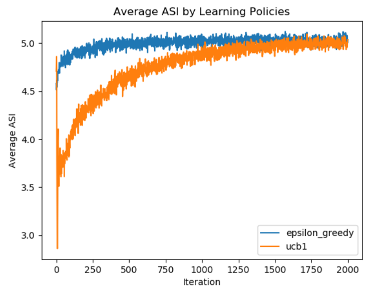
We detail here the influence of the number of agents and the type of the reward signal on the learning process for $\epsilon - greedy$ and UCB exploration policies of 3 agents voting on 3 proposals, here $num\_actions = 2^3 = 8$ and $ASI\_max = 7$

Number of Agents = 3

By observing the averaged ASI obtained by 3 learning agents voting on a ballot with 3 questions, we notice that these agents learn to take decisions with improved results. In 3a the blue curve is obtained by the agents following the $\epsilon - greedy$ strategy with $\epsilon = 0.1$, and the orange curve is for the agents exploring under UCB's optimism in the face of uncertainty principle.
We detect that in this action space UCB learns faster than $\epsilon - greedy$ under the linear reward signal and produces a higher and more stable collective outcome. To the contrary, with the EDR signal in 3b we recognize that $\epsilon - greedy$ learning agents achieve an optimum in a small number of iterations, while UCB exploration is slower and takes a longer time to learn.



(a) Linear Reward

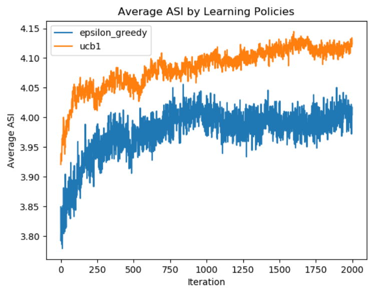


(b) EDR

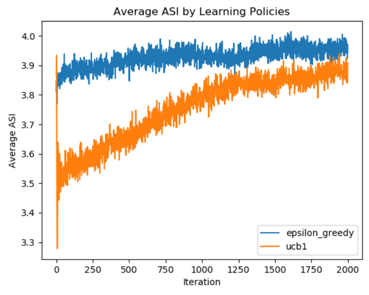
Figure 3: Average ASI Obtained by 3 Learning Agents Voting on 3 Questions Following $\epsilon - greedy$ and UCB Exploration Policies and Receiving Different Reward Signals

Number of Agents = 15

With a larger number of agents, it is a more difficult task to learn, we run our iterative voting game with 15 agents and notice with the linear reward in 4a the improvement in the ASI score takes longer time for the community of 15 agents, agents exploring under UCB still result in a better collective decision making than $\epsilon - greedy$ agents . With the EDR in 4b $\epsilon - greedy$ does not make significant improvement over time, the UCB agents take more iterations to improve the quality of the collective decision over time and still does not outperforms $\epsilon - greedy$.



(a) Linear Reward



(b) EDR

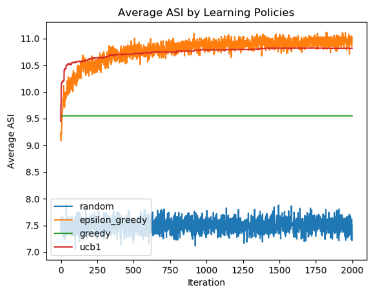
Figure 4: Average ASI Obtained by 15 Learning Agents Voting on 3 Questions Following $\epsilon - greedy$ and UCB Exploration Policies and Receiving Different Reward Signals

4.2.2 Number of Questions = 4

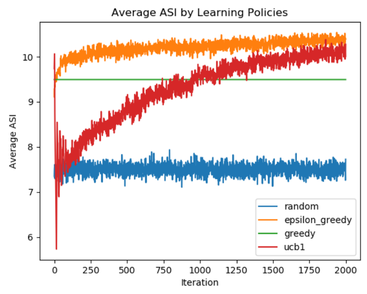
With 4 Questions, the action space size per agent is $2^4 = 16$ action, and $ASI\_max = 15$. We clone the previous setting for the new ballot and observe the effect for different number of voting agents and reward signals.

Number of Agents = 3

We add in this setting a comparison for the performance of classical greedy algorithm and random action selection with $\epsilon - greedy$ and UCB. In figure 5 we show the averaged ASI score obtained by three agents submitting their votes. In blue is the score of agents selecting their actions randomly which results in a low arbitrary ASI; the greedy algorithm in green gets stuck in a local optimum, under the linear reward in 5a $\epsilon - greedy$ policy behaves similar to the setting with 3 Questions. Also UCB agents still manage to explore efficiently in this action space and stabilize with a good collective score in few epochs. With the exponentially decreasing reward, UCB suffers for a long time again before reaching the result obtained rapidly by $\epsilon - greedy$ agents.



(a) Linear Reward

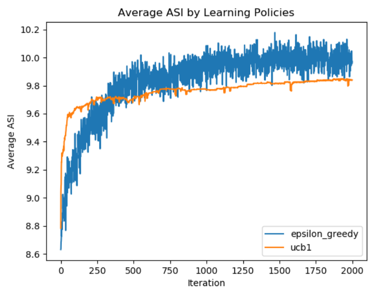


(b) EDR

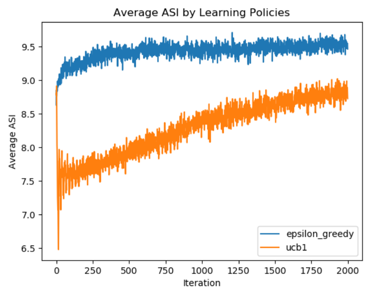
Figure 5: Average ASI Obtained by 3 Learning Agents Voting on 4 Questions Following $\epsilon - greedy$ and UCB Exploration Policies and Receiving Different Reward Signals

Number of Agents = 5

In this setting we have 5 agents voting iteratively, each with 16 action options. We observe in 7a that the ASI for agents exploring with linear reward using $\epsilon - greedy$ policy slightly surpass the performance of UCB for the first setting after sufficient number of interactions between the agents, UCB converges to a stabilized outcome and less exploration. In 6b a clearly slow but steady learning behaviour is shown for UCB agents, and a repetitive reasoning with the group of $\epsilon - greedy$ agents that reach a considerably satisfying collective result in a short run, illustrating that the performance of UCB is notably less significant for multiple agents learning with low rewards similarly to the single agent learning discussed by (Garivier and Cappé, 2011).



(a) Linear Reward

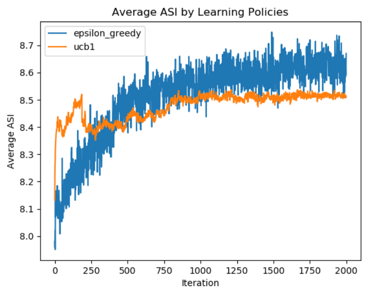


(b) EDR

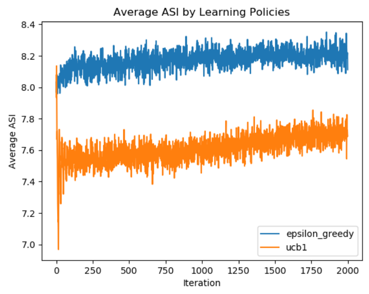
Figure 6: Average ASI Obtained by 5 Learning Agents Voting on 4 Questions Following $\epsilon - greedy$ and UCB Exploration Policies and Receiving Different Reward Signals

Number of Agents = 15

Each agent adds her own input to the joint action and it is evidently more challenging to learn. But we observe that both UCB and $\epsilon - greedy$ agents receiving the linear reward still learn to increase the ASI with time. No important improvement in the ASI for the EDR in 7b.



(a) Linear Reward



(b) EDR

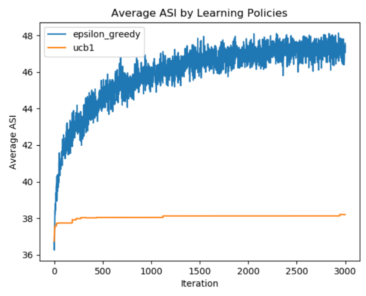
Figure 7: Average ASI Obtained by 15 Learning Agents Voting on 4 Questions Following $\epsilon - greedy$ and UCB Exploration Policies and Receiving Different Reward Signals

4.2.3 Number of Questions = 6

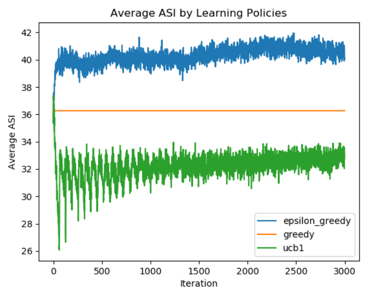
The size of action space for 6 questions is $2^6 = 64$ actions, and $ASI\_max = 63$. This is already a big number of alternatives for a stateless game with a single agent learning problem. We run the iterative voting game with multi agents learning using $\epsilon - greedy$ and UCB policies and monitor the features of the outcome.

Number of Agents = 3

In the figure 8 we show the average ASI score obtained by three agents submitting their votes on 6 Questions. The EDR does not show worthy improvement in ASI score. While it's remarkable for the linear reward in 8a that the agents following $\epsilon - greedy$ learn to make an important increase in the ASI after an adequate number of iterations.



(a) Linear Reward

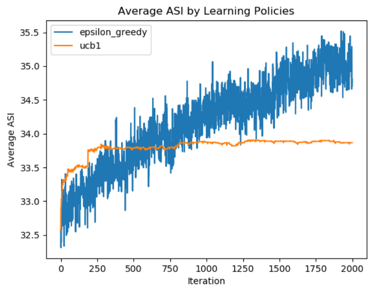


(b) EDR

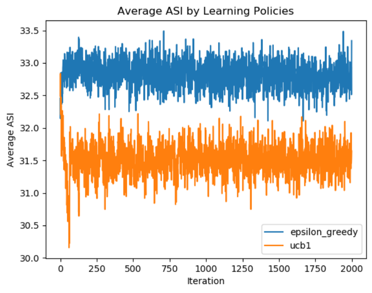
Figure 8: Average ASI Obtained by 3 Learning Agents Voting on 6 Questions Following $\epsilon - greedy$ and UCB Exploration Policies and Receiving Different Reward Signals

Number of Agents = 15

Interestingly in this setting with 15 agents playing iterative voting game on 6 Questions, the $\epsilon - greedy$ agents receiving linear reward in 9a consistently improve the ASI score with time as they are exploring the action space and adapting to the environment change caused by other agents.



(a) Linear Reward



(b) EDR

Figure 9: Average ASI Obtained by 15 Learning Agents Voting on 6 Questions Following $\epsilon - greedy$ and UCB Exploration Policies and Receiving Different Reward Signals

5 Conclusion

Our experiments show that multiple independent learning agents in the setting of iterative plurality voting learn to produce in this environment a good global result measured according to an aggregated score that indicates the level of social satisfaction in the winner of the election. We contribute a discussion of the parameters that influence the learning progress of two famous exploration policies, ϵ-greedy and UCB in a multi-armed bandit model. We remark the difference in the collective ASI obtained and illustrate the variance in performance with respect to two different reward functions. We show that in a small action space and with a linear reward, UCB produced a better collective outcome than $\epsilon - greedy$. While in large action spaces with small reward signal, UCB is slow and does not produce important improvement. On the other hand we show that agents following $\epsilon - greedy$ and a linear reward learn to improve in large action space and obtain good collective decision and that this result is scalable and robust with respect to the number of actions for a small number of agents. The number of agents influence the number of sufficient iterations needed to reach an optimum.

References

S. Airiau, U. Grandi, and F. S. Perotto. Learning agents for iterative voting. In J. Rothe, editor, *Algorithmic Decision Theory*, pages 139–152, Cham, 2017. Springer International Publishing. ISBN 978-3-319-67504-6.

R. Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957.

C. Bowman, J. K. Hodge, and A. H. yan Yu. The potential of iterative voting to solve the separability problem in referendum elections. *Theory and Decision*, 77:111–124, 2014.

S. J. Brams, D. M. Kilgour, and W. Zwicker. Voting on Referenda: The Separability Problem and Possible Solutions. Working Papers 97-15, C.V. Starr Center for Applied Economics, New York University, 1997. URL `https://ideas.repec.org/p/cvs/starer/97-15.html`.

G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

L. Busoniu, R. Babuska, and B. De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, March 2008. ISSN 1094-6977. doi: 10.1109/TSMCC.2007.913919.

L. Buşoniu, R. Babuška, and B. De Schutter. Multi-agent reinforcement learning: An overview. In *Innovations in multi-agent systems and applications-1*, pages 183–221. Springer, 2010.

A. Garivier and O. Cappé. The kl-ucb algorithm for bounded stochastic bandits and beyond. In *Proceedings of the 24th annual conference on learning theory*, pages 359–376, 2011.

J. K. Hodge and P. Schwallier. How does separability affect the desirability of referendum election outcomes? *Theory and Decision*, 61(3):251–276, 2006.

E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, K. Y. Goldberg, J. Gonzalez, M. I. Jordan, and I. Stoica. Rllib: Abstractions for distributed reinforcement learning. In *ICML*, 2017.

J. G. March. Exploration and exploitation in organizational learning. *Organization science*, 2(1):71–87, 1991.

R. Meir, M. Polukarov, J. S. Rosenschein, and N. R. Jennings. Convergence to equilibria in plurality voting. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

A. Nowe, P. Vrancx, and Y.-M. De Hauwere. *Game Theory and Multi-agent Reinforcement Learning*, pages 441–470. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-27645-3. doi: 10.1007/978-3-642-27645-3_14. URL `https://doi.org/10.1007/978-3-642-27645-3_14`.

C. E. Shannon. Xxii. programming a computer for playing chess. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 41 (314):256–275, 1950.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529 (7587):484, 2016.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

A. D. Tijsma, M. M. Drugan, and M. A. Wiering. Comparing exploration strategies for q-learning in random stochastic mazes. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8. IEEE, 2016.

C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.