Desenvolvimento de Estratégias e Fenômenos em Dinâmicas de Jogos de Múltiplos Agentes

Learning Dynamics from Evolution and Reinforcement

Giovanni Amorim

Flávio Codeço Coelho

Fundação Getúlio Vargas - Escola de Matemática Aplicada

November 23, 2020

GIOVANNI ALMEIDA ARGENTO DE AMORIM

"DESENVOLVIMENTO DE ESTRATÉGIAS E FENÔMENOS EM DINÂMICAS DE JOGOS DE MÚLTIPLOS AGENTES"

Trabalho de Conclusão de Curso - TCC apresentado (a) ao Curso de Graduação em Matemática Aplicada da Escola de Matemática Aplicada para obtenção do grau de Bacharel (a) em Matemática Aplicada.

Data da Defesa: 07/12/2020

ASSINATURA DOS MEMBROS DA BANCA EXAMINADORA

Flávio Codeço Coelho
Orientador

Moacyr Alvim Horta Barbosa da Silva
Membro

Elizabeth Wegner Karas
Membro

Nos termos da Lei nº 13.979 de 06/02/20 - DOU nº 27 de 07/02/20 e Portaria MEC nº 544 de 16/06/20 - DOU nº 114 de 17/06/20 que dispõem sobre a suspensão temporária das atividades acadêmicas presenciais e a utilização de recursos tecnológicos face ao COVID-19, as apresentações dos Trabalhos de Conclusão de Curso, de forma excepcional, serão realizadas de forma remota e síncrona, incluindo-se nessa modalidade membros da banca e discente.

Abstract

Recent developments in Reinforcement Learning (RL) methods are focused on models that can learn good policies in non stationary environments, such as multi-agent games, where agents must learn how to react to changes in other agent's strategies or in the environment. Some development has been made by studying not only how one agent can develop it's policy, but how a population of agents can evolve from initial distributions to stable states of strategies. Evolutionary Game Theory (EGT) is the theoretical framework that applies mathematical and economical knowledge from game theory and biological evolution inspiration to study how individuals from a population dynamically interact in an environment. In this paper, we first introduce EGT concepts and show how they can be applied to understanding a population's learning dynamics in the context of RL. Then we link those concepts with learning algorithms and study how one can infer the behaviour of those methods from links with evolutionary dynamics. Finally, we study and evaluate a recently proposed algorithm derived from policy gradient model and EGT dynamics and discuss next steps.

Keywords

evolutionary game theory, reinforcement learning, multi agent, learning dynamics, replicator dynamics, neural replicator dynamics

Contents

1 Introduction

RL methods have been vastly used for solving complex real-world relevant problems like self-driving cars [15], stock prices prediction [8] and plain text classification [24] or mastering games, such as atari games [13], DOTA2 [3], variants of poker [4], chess and go [20] beyond human-level. Even with many good results, these algorithms often rely on the stationary nature of the environment [1], the usage of expert knowledge or tabular representations of the action-state space, constraints that may not be available in all applications, such as some multi agent games.

In such settings, there may be no complete knowledge about the environment, the reward $R(s,a)$ received for playing an action a in the state s can change depending on time and other agents actions, and at each decision an agent only knows the reward value from the chosen action.

To address the case where all the information about agents actions is not available, development has been made in studying how known RL methods can be modified in order to achieve learning dynamics that are more resilient to non stationary settings and present regret bounds [2, 10].

Evolutionary Game Theory presents a framework for studying a population of agents interacting on a game. Each agent behaves accordingly to a strategy and a fitness value is assigned based on the success of each agent compared to the rest of population. The concepts of selection, mutation and evolution are then derived from fitness and strategies dynamics can be evaluated as systems of differential equations [14].

As the optimization of an action-choice RL model based on received rewards is analogous to EGT strategies evolution as a function of it's frequency and fitness, RL theory can use concepts and results from EGT in order to better understand models optimization dynamics and possibly derive new algorithms that can enjoy well known convergence behaviours even in non stationary settings.

In order to be able to infer information about RL algorithms dynamics, we have to understand the EGT concepts that are closely related to these dynamics. In this paper, we present necessary definitions from EGT, as strategy fitness evaluation and evolutionary dynamics, the links between evolutionary dynamics and RL methods dynamics, like Q-learning and policy gradient and evaluate a recent method developed from these links called Neural Replicator Dynamics [6].

2 Evolutionary Game Theory

2.1 Basic Concepts

In the EGT framework, a game can be described by it's *payoff function*. Agents with fixed *strategies* will compete in encounters and the resulting payoffs are interpreted as agent's fitness and translated into reproductive success that results in evolution to the next generation *state*.

Let's consider a population playing a two-strategy game. Agent's encounters can be described by a *payoff matrix* F:

$$\begin{pmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{pmatrix}$$

When agent 1, following strategy $i \in \{1,2\}$, plays against agent 2, following strategy $j \in \{1,2\}$, the payoffs can be respectively calculated as:

$$f_1 = F_{ij}$$

$$f_2 = F_{ji}$$

allowing us to evaluate agents fitness and determine what are the survival/reproductive chances.

When studying how this population will evolve into next generations, we use the *population state* $x : \sum_{s \in S} x_s = 1$ that represents the fraction of total agents that follows each possible strategy. Using these concepts, one can calculate a strategy expected payoff:

$$f_s = \sum_{s' \in S} f_{ss'} x_{s'} = F_s x$$

In evolution, strategies with best average payoffs will increase in frequency as agents with these strategies will be more likely to survive and reproduce, following the *Survival of the Fittest* logic. But how strategies presence in population dynamically change is not yet determined. The way this reproductive selection is defined is with *revision protocols*, that we will cover in the next section.

2.2 Revision Protocols

After evaluating fitness of individuals in the population, we must define how much each strategy increase in frequency. In the same way there is not only one way of optimizing a RL model parameters, there are multiple ways of determining how evolution will select

the survivors in a population.

A revision protocol $\rho_{ij}(F,x)$ estimates the probability of an agent following strategy i to be effectively replaced by an agent with strategy j based on expected payoffs and population state. In evolutionary terms, this can represent the following mechanisms, depending on the chosen dynamic:

- an agent dies and does not reproduce because of low fitness value, decreasing it's genome presence in population;
- more fit organisms reproduce and conquer fractions of the population;
- randomly agents with different strategies are born and evaluated by the environment

This definition leads to the differential equations that represents the growth of strategies in presence called *evolutionary dynamics*:

$$\dot{x}_s = \sum_{s' \in S} \rho_{s's}(F,x)x_{s'} - x_s \sum_{s' \in S} \rho_{ss'}(F,x) \tag{1}$$

The two terms of this equation represents the proportion of population occupied by new agents with strategy s and the proportion of population replaced by other options. This will happen dynamically increasing presence of strategies with better performance, resulting in populations with higher fitness.

There is a clear analogy between evolving the fitness of a population with best strategies and optimizing an RL agent's reward by updating it's parameters. Our goal in section 3 is to derive differential equations from RL models that determine the growth of each action's probability for an agent in an equivalent way of evolutionary dynamics, with the interpretation of EGT's state as the actions probability distribution of an agent in RL.

Now we illustrate an evolutionary dynamic with an example in the *Hawks and Doves* game.

In a population of agents, there are two possible phenotypes: hawks and doves, that differ in behaviour when contesting for food. When a hawk competes with another hawk, both will fight for the food, potentially getting very hurt, and one of them will have the food. We can represent this result with rewards $-1-2=-3$ for the hurt and hungry hawk, and $1-2=-1$ for the possibly hurt but fed one. This results in an expected reward of -2 for a hawk when facing another hawk.

On the other hand, when facing a dove, the hawk will be feared and easily satisfied with the food. This represents a reward of 1 and -1 for the hawk and the dove, respectively.

	Dove	*Hawk*
Dove	$(0,0)$	$(-1,1)$
Hawk	$(1,-1)$	$(-2,-2)$

Finally, when a dove encounters a dove, there is a 50% chance for each getting the food, representing the expected outcome of 0 for a dove in this case.

The table above displays all possibles outcomes when playing hawks and doves. In a population where h represents hawks frequency and d represents doves ($h+w=1$), we can calculate the expected reward for both:

$$f_h = d - 2h$$

$$f_d = -h$$

From the reward equations we can conclude that hawks will dominate the population growth when doves are twice as frequent:

$$f_h > 0 > f_d \rightarrow d > 2h$$

but when the hawk population grows too much, they start to fight among them for food and the doves, even with negative payoffs, are more *fit* to the environment than hawks and will reproduce:

$$f_d > f_h \rightarrow h > d$$

and this population will find an equilibrium when the population is equally splitted among hawks and doves:

$$f_d = f_h \rightarrow h = d$$

Here we could define the basic rules extracted from the simple game definition and how this environment works, but how the population state will effectively evolve depends on the chosen revision protocol.

Examples of Revision Protocols and Evolutionary Dynamics

2.2.1 Pairwise Proportional Imitation and the Replicator Dynamics

The pairwise proportional imitation revision protocol simulates an evolutionary process where an agent following a strategy is substituted by a randomly chosen agent with probability proportional to the fitness difference (if positive). This is equivalent to:

$$\rho_{ij}(F,X) = x_i[F_j - F_i]_+$$

Evaluating this process as an expected evolutionary dynamics (eq.1) gives us the Replicator Dynamic (RD):

$$\dot{x}_i = \sum_{j \in S} x_i[F_i - F_j]_+ x_j - x_j[F_j - F_i]_+ x_i$$

$$= \sum_{j \in S: F_j \leq F_i} x_i x_j [F_i - F_j] - \sum_{j \in S: F_j > F_i} x_i x_j [F_j - F_i]$$

$$= \sum_{j \in S} x_i x_j [F_i - F_j]$$

$$= x_i F_i \sum_{j \in S} x_j - x_i \sum_{j \in S} x_j F_j$$

$$\dot{x}_i = x_i[F_i - \hat{F}] \tag{2}$$

Replicator dynamic is the most popular evolutionary dynamic in EGT. It has applications on biological [19], economical [17] and social [5] simulations and variant models. Under this dynamic, the marginal increase of a strategy is proportional to it's current state. This means that unknown or extinct strategies will not (re)appear and existing but unpopular strategies can grow, but not as fast as dominant ones.

RD also states that the growth of a frequent strategy i will slow down as it becomes more present and the average population payoff will approach F_i. In our dynamics comparison at the end of section 3, we show that this behaviour prevents the population of becoming too uniform in the Rock Papers Scissors (RPS) game.

Also, this dynamic is considered *no-regret*, as it guarantees that the regret metric evaluated from this method doesn't increase in time [6]. Regret is defined as the difference of an agent's strategy fitness value and the best possible fixed action value: $r(s_i) = f_i - max_{a \in A} f(a)$. As will be illustrated, this dynamic presents an evolution of states in the population more resistant to environment changes than other learning methods.

2.2.2 Logit Choice Rule and the Logit Dynamic

In the logit model, next generation strategy distribution will be selected proportionally to the fitness value of each strategy. The resulting equation is the following:

$$\rho_{ij}(F,x) = \frac{e^{\eta F_j}}{\sum_k e^{\eta F_k}}$$

where $\eta > 0$ is called the noise level and determines how optimal the choices will be made [16]. The logit evolutionary dynamic is

$$\dot{x}_i = \frac{e^{\eta^{-1} \hat{F}_i}}{\sum_{j \in S} e^{\eta^{-1} F_j}} - x_i \tag{3}$$

It is important to notice that this model revision protocol doesn't depend on the origin strategy: $\rho_{ij} = \rho_{kj}\ \forall i,k$. This will allow more 'drastic' immediate changes in the population distribution.

2.2.3 Smith model

Similar to RD, Smith model's revision protocol simulates a evolutionary process where an agent following a strategy is substituted by a randomly chosen strategy with probability proportional to the fitness difference (if positive):

$$\rho_{ij}(F,X) = [F_j - F_i]_+$$

and it's dynamics are represented by the following differential equation:

$$\hat{x_i} = \sum_{j \in S} x_j [F_i - F_j]_+ - x_i \sum_{j \in S} [F_j - F_i]_+$$

This model decreases (but not completely removes) the dependency of evolution on current state, allowing changes to be more directed into what is fit at each iteration.

2.3 Nash Equilibrium

A common goal when evolving a population is getting to a state that achieves the best possible fitness for all agents. Nash equilibrium is a concept from Game Theory that defines a group of states where no agent is rationally willing to change strategy. State x is a Nash Equilibrium if no agent would increase it's expected reward by unilaterally changing strategy:

$$F_i(x_1, x_2, ..., x_{i-1}, x_i, ..., x_N) \geq F_i(x_1, x_2, ..., x_{i-1}, x', ..., x_N), \forall x' \in X, \forall i \in \{1, ..., N\}$$

Nash equilibrium is not necessarily the joint strategy that outputs the most total reward. There is a group of symmetric two-player games called The Prisoner Dilemma where Nash equilibrium of pure strategies gives a total reward objectively worse than other configurations, but there is no 'one-side' action change that would alone increase the agent's outcome. Thus, in the context of learning, seeking for Nash equilibrium might result in sub-optimal policies.

2.4 Evolutionary Stable Strategies

In the context of EGT, it is more natural to characterize stable states that are resistant to the evolution selection process and not necessarily to rational agent actions. In a population

of agents, new strategies can appear and grow among agents if it presents a better fitness value. Evolutionary Stable Strategies (ESS) define a strategy that, once adopted by a whole population, is resistant to small invasions by other strategies, in a way that it extincts the invader.

Let's define $E(x_{s_1}, x_{s_2})$ the fitness evaluated for an agent following strategy s_1 in a state $x = (x_{s_1}, x_{s_2})$ where a fraction x_{s_1} of agents follow strategy 1 and x_{s_2} adopts 2. Strategy s^* is an ESS if, $\forall s \neq s^*$:

$$E(x_{s^*}, x_{s^*}) > E(x_s, x_{s^*})$$

or

$$E(x_{s^*}, x_{s^*}) = E(x_s, x_{s^*}) \text{ and } E(x_{s^*}, x_{s^*}) > E(x_s, x_s)$$

for a sufficiently small x_s.

3 Policy Optimization Algorithms Dynamics

A parametric policy $\pi_\theta : A \times S \to [0,1]$ is a function that depends on parameters θ and outputs the probability of choosing an action given an observed state in the environment. A natural way of evaluating a chosen action is to compute the future cumulative discounted rewards $u_t = (\sum_{t_i=t}^{\infty} \eta^{(t-t_i)} r_{t_i}) - v_t$ where r_t is the immediate reward on t and $\eta \in [0,1]$ is the discount factor. This evaluates actions not only by immediate utility but also with future results it possibly helped to accomplish.

This section introduces existing policy optimization methods and presents analysis about the dynamics of resulting policies.

3.1 Follow the Leader

A natural way of optimizing a simple (not parametric) policy model would be choosing the policy that maximizes the future rewards. Follow the Leader is a family of models that updates parameters minimizing a loss function:

$$\pi_t = argmin_{\pi \in \Delta^{|A|}} f(\pi)$$

Considering the loss function equal to minus expected cumulative rewards and our policy represented simply by the action probabilities:

$$\pi_t = argmax_{\pi \in \Delta^{|A|}} [\pi \cdot u_{t-1}] \tag{4}$$

where $u_t = \sum_{i=1}^{t} \eta^{i-1} r_i$.

Clearly (4) results in the policy $\pi(a) = \mathbb{1}_{[a=argmax_a u(a)]}$ and that is not plausible even for simple tasks. We must add a regularization factor in the Follow the Leader optimization method for this model.

3.2 Follow the Regularized Leader

A common regularization function for policy optimization is the negative entropy function $h(x) = xlog(x)$ that penalizes policies excessively concentrated on few actions:

$$\pi_t = argmax_{\pi \in \Delta^{|A|}} f(\pi) = argmax_{\pi \in \Delta^{|A|}} [\pi \cdot u_{t-1} - k\pi \cdot log(\pi)] \tag{5}$$

The solution of (5) gives us the softmax function over cumulative rewards [11]:

$$\pi_t(a) = \frac{e^{u_{t-1}(a)}}{\sum_{a'} e^{u_{t-1}(a')}} \tag{6}$$

Now we derive the dynamics of this model. First, calculate $d\pi(a)/du(a')$:

$$\frac{d\pi_t(a)}{du_t(a')} = (\sum_{a_i} e^{u_{t-1}(a_i)})^{-2}(\frac{d(e^{u_{t-1}(a)})}{d(u_{t-1}(a'))}\sum_{a'} e^{u_{t-1}(a)} - e^{u_{t-1}(a)}\frac{d(\sum_{a_i} e^{u_{t-1}(a_i)})}{d(u_{t-1}(a'))})$$

So, in the case $a = a'$:

$$\frac{d\pi_t(a)}{du_t(a')} = \frac{e^{u_{t-1}(a)}}{\sum_{a_i} e^{u_{t-1}(a_i)}} - \frac{e^{2u_{t-1}(a)}}{(\sum_{a_i} e^{u_{t-1}(a_i)})^2} = \pi_t(a)[1-\pi_t(a)]$$

and when $a \neq a'$:

$$\frac{d\pi_t(a)}{du_t(a')} = -\frac{e^{u_{t-1}(a)} \cdot e^{u_{t-1}(a')}}{(\sum_{a_i} e^{u_{t-1}(a_i)})^2} = -\pi_t(a) \cdot \pi_t(a')$$

Knowing $du(a)/dt$ allow us to calculate the policy actions probability growth:

$$\frac{du_t}{dt} = r_t$$

$$\frac{d\pi_t(a)}{dt} = \sum_{a' \in A} \frac{d\pi_t(a)}{du_t(a')} \cdot \frac{du_t(a')}{dt} = \sum_{a' \in A} \pi_t(a')[\mathbb{1}_{a=a'} - \pi_t(a)] \cdot r_t(a')$$

$$\frac{d\pi_t(a)}{dt} = \pi_t(a) \cdot r_t(a) - \pi_t(a) \sum_{a' \in A} \pi_t(a') \cdot r_t(a') = \pi_t(a)[r_t(a) - \hat{r}_t] \tag{7}$$

where $\hat{r}_t$ is the expected reward on iteration t. (7) is equivalent to the replicator dynamics.

The result shows that a model trained by this optimization method would dynamically update action probabilities following the replicator equation. This dynamic can perform good in lots of settings, but the Follow the Leader model is restricted, as it doesn't directly allows parametric models, like neural networks.

3.3 Q-Learning

Q-learning is one of the most popular RL methods and has multiple variants for many use cases. It optimizes a value-function to determine, for each state of environment, which is the best possible action by a value $Q(a,s)$ and update this values using the immediate rewards and the estimates of future accumulated rewards. Considering a simple model mapping states to action-weights and applying a softmax function to output action probabilities updated using Q-learning [23]:

$$\pi_t(a,s) = softmax(Q(.,s))(a)$$

$$\frac{dQ(a,s)_{t+1}}{dt} = \alpha(r_t(a) + \gamma max_{a'}[Q_t(a', s_{t+1}(a))] - Q_t(a,s))$$

Following the same logic as before, we can derive the policy dynamics as in [22]:

$$\frac{\pi(a,.)}{dt} = \alpha\pi(a,.)[r_t(a) - \hat{r}_t + \sum_{a' \in A} \pi_t(a',.) log(\frac{\pi_t(a',.)}{\pi_t(a,.)})] \tag{8}$$

Equation (6) show a replicator dynamics alike term in Q-learning dynamics, combined with a positive entropy factor. This shows Q-learning methods tend to increase entropy, influencing the model to explore more strategies and move towards a more uniform strategy distribution.

3.4 Policy Gradient

Like Q-learning, policy gradient (PG) methods and it's derivatives are very popular in the RL community and have been applied to multiple domains. In this example, the policy will be updated directly using the calculated gradient from the rewards. Using actor-critic algorithms [7], while one model outputs the parametric policy π_θ, another one is trained to inform an expected future reward value $q(a,s)$. While this second model is trained in the values outputted by the environment using the advantage values, the first uses the chosen action probability gradient and the values $q(a,s), v(s) = \sum_{a' \in A} \pi(a',s) q(a',s)$. We can calculate the dynamics in a similar way of Q-learning case, with the $softmax$ function applied to weights y. Considering the simple case where $dy_t/d\theta_t = 1$ [21]:

$$\pi_t(a,s) = softmax(y)(a)$$

$$\frac{dy(a)_{t+1}}{dt} = \gamma \sum_{a' \in A} \nabla_{y(a)} \pi(a')[q(s,a') - v(s)]$$

$$\frac{\pi_{t+1}(a)}{dt} = \pi_t(a)[\sum_{a' \in A} \pi_t(a')(\pi_t(a)A(a) - \pi_t(a')A(a'))] \tag{9}$$

Here we can observe a square nature in the action probability value. This indicates that policy gradient algorithm will have more tendency (than RD) to concentrate probability in actions with high immediate rewards. This means more exploitation of actions that are working instead of exploration for alternate actions that might perform better.

Figure 1 compares the previous shown EGT and RL dynamics in the standard RPS game. This comparison gives good intuition on how these algorithms explore the state/action space when trying to learn a good policy, the learning dynamics.

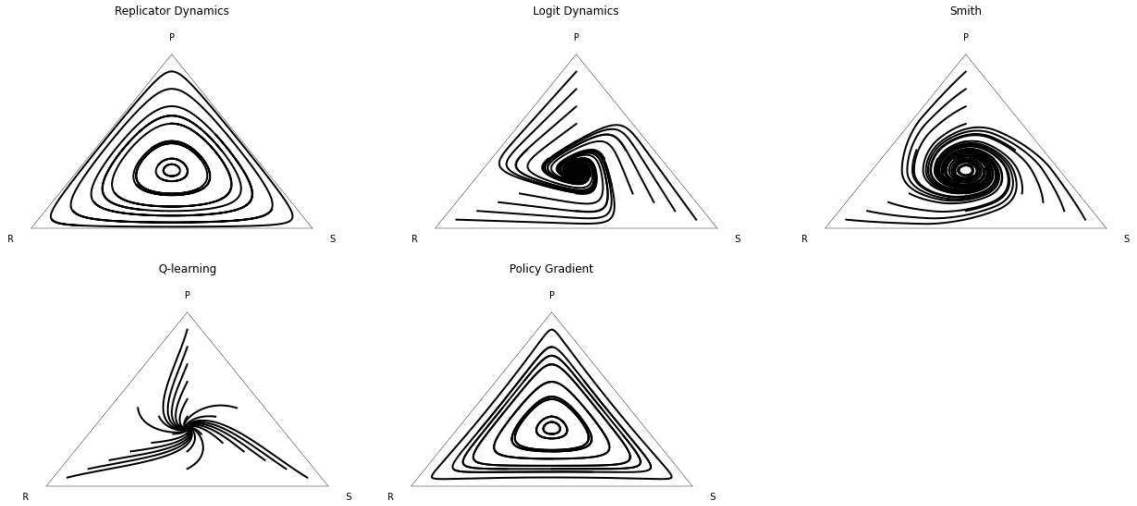

Figure 1: Models dynamics represented as ternary plots. The three possible pure states (rock, papers or scissors) are defined by the triangle vertex, meaning 100% of population adopting the pure strategy. The black lines represent simulations of how a population evolves from initial states to nash equilibrium $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ in some cases and repeatedly cycle around this center in others.

4 Neural Replicator Dynamics

As replicator dynamics has proved to work under various environment settings and for different problems, but is restrained to non parametric models in tabular games, it is valuable to develop a RL function approximation approach that follows the same dynamics.

Neural Replicator Dynamics [6] presents an 'one line fix' from the Actor-Critic Softmax Policy Gradient algorithm that relies on minimizing the distance between state of the logits y_t from an optimal moved logit state based on future accumulated discounted rewards from the critic model:

$$\frac{dy_t(a)}{dt} = q^{\pi}(a) - v^{\pi}$$

Using Euler Method and then updating parameters in order to approximate the logits to estimated direction of solution $y(a)$ using Euclidean distance:

$$y(a) = y_t(a) + (q^{\pi}(a) - v^{\pi}) \tag{10}$$

$$\theta_{t+1} = \theta_t - \sum_{a \in A} \frac{1}{2} \nabla_{\theta} \| y(a) - y_t(a) \|_2^2$$

using (10):

$$\theta_{t+1} = \theta_t + \sum_{a \in A} \nabla_{\theta} y_t(a)(q^{\pi}(a) - v^{\pi})$$

This model dynamics can be derived in the same way of the previous models and proved to be the same as the replicator dynamics (under the actor-critic approximation). Different of the *more exploration* Q-learning algorithm and the *more exploitation* Policy Gradient dynamics, Neural Replicator Dynamics can be a great alternative to learning policies in non stationary environments.

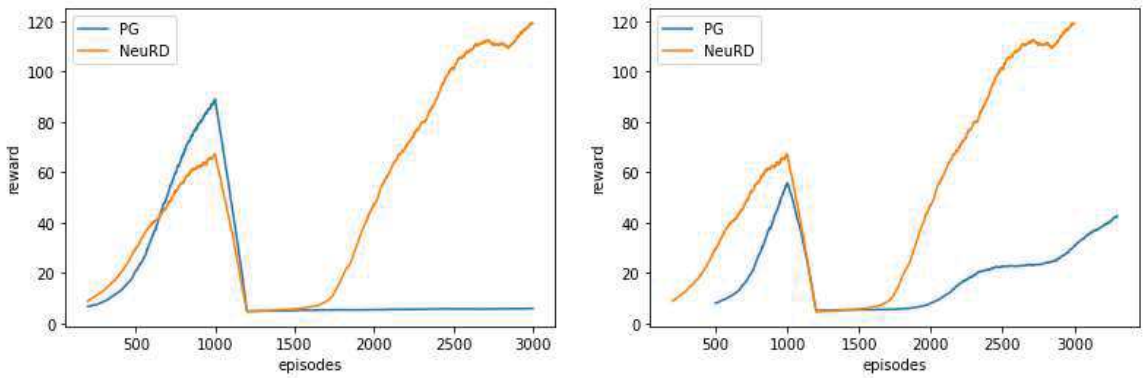

Figure 2: (a) shows performance direct comparison in the non stationary game setting between actor-critic Softmax PG and NeuRD. (b) shows the same comparison, but the PG model was trained in the first configuration of environment 30% less.

5 Evaluation

We evaluated how both Policy Gradient and Neural Replicator Dynamics models dynamics perform in a non stationary created environment. It is a simple 7x7 grid two agent game: the first agent, controlled by the model, can move in four directions in the grid or stay in the same place. His objective is to not be caught by the second agent, which is a hard encoded script that simply follows agent 1.

The non-stationary nature is in how we encode agent 2 strategy. When following its 'prey', it calculates the vector $V = Agent1Position - Agent2Position$ and from there has two options:

1) moves preferably in the x axis, if $V_x > 0$, else moves in y axis

2) moves preferably in the y axis, if $V_y > 0$, else moves in x axis

First we fix agent 2 strategy to option 1 and after 10^3 episodes we change it by option 2. Our goal is to evaluate how fast and well an agent can learn a new strategy to react from a change in the environment reward function, as agent 2 will change how it follows him. The results in figure 2a shows that both models suffer from the reward change, but PG didn't show results of getting any good *re-fit* after 2000 more episodes, while NeuRD rapidly recovered and learned how to react to the new configuration.

It is also visible from the results that after 1000 episodes PG got a better average score and one could argue that this better fit causes the model to perform poorly in sequence. We corrected that factor by training the PG network 30% less (700 episodes) than the NeuRD and comparing the new score, this result is in figure 2b.

On both comparisons, NeuRD performs significantly better in the 'second turn' of the game, reacting more quickly to the change in the it's predator moves. For each run, we initialized a one-layer neural network with 64 hidden neurons, for separated actor and

critic models. The learning rates were 2e-4 (actor) and 0.01 (critic) with Adam optimizer. The plots shows average results for 10 complete runs.

6 Conclusion

To summarize our contribution, we detailed important concepts from evolutionary game theory and analyzed multiple EGT models dynamics. This dynamics were then compared to RL models and a discussion about the behaviour of these methods in non stationary environment was presented. Finally, we provided an explanation of how a new method (NeuRD) was created from inspiration in EGT concepts to attack RL problems and illustrated how this recently presented model can potentially overcome other RL popular methods.

Interesting possible future work involves studying how other popular well performing PG-based algorithms, such as A3C [12], PPO [18], DDPG [9] and others can have it's optimization dynamics characterized and even modified to increase performance in non stationary environments.

References

[1] Lucas N Alegre, Ana LC Bazzan, and Bruno C da Silva. “Quantifying the Impact of Non-Stationarity in Reinforcement Learning-Based Traffic Signal Control”. In: *arXiv preprint arXiv:2004.04778* (2020).

[2] Peter Auer, Thomas Jaksch, and Ronald Ortner. “Near-optimal regret bounds for reinforcement learning”. In: *Advances in neural information processing systems*. 2009, pp. 89–96.

[3] Christopher Berner et al. “Dota 2 with large scale deep reinforcement learning”. In: *arXiv preprint arXiv:1912.06680* (2019).

[4] Fredrik A Dahl. “A reinforcement learning algorithm applied to simplified two-player Texas Hold’em poker”. In: *European Conference on Machine Learning*. Springer. 2001, pp. 85–96.

[5] Christoph Hauert et al. “Replicator dynamics for optional public good games”. In: *Journal of Theoretical Biology* 218.2 (2002), pp. 187–194.

[6] Daniel Hennes et al. “Neural Replicator Dynamics: Multiagent Learning via Hedging Policy Gradients”. In: *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*. 2020, pp. 492–501.

[7] Vijay R Konda and John N Tsitsiklis. “Actor-critic algorithms”. In: *Advances in neural information processing systems*. 2000, pp. 1008–1014.

[8] H. Li, C. H. Dagli, and D. Enke. “Short-term Stock Market Timing Prediction under Reinforcement Learning Schemes”. In: *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*. 2007, pp. 233–240.

[9] Timothy P Lillicrap et al. “Continuous control with deep reinforcement learning”. In: *arXiv preprint arXiv:1509.02971* (2015).

[10] Ryan Lowe et al. “Multi-agent actor-critic for mixed cooperative-competitive environments”. In: *Advances in neural information processing systems*. 2017, pp. 6379–6390.

[11] Panayotis Mertikopoulos and William H Sandholm. “Learning in games via reinforcement and regularization”. In: *Mathematics of Operations Research* 41.4 (2016), pp. 1297–1324.

[12] Volodymyr Mnih et al. “Asynchronous methods for deep reinforcement learning”. In: *International conference on machine learning*. 2016, pp. 1928–1937.

[13] Volodymyr Mnih et al. “Playing atari with deep reinforcement learning”. In: *arXiv preprint arXiv:1312.5602* (2013).

[14] Martin A Nowak. *Evolutionary dynamics: exploring the equations of life*. Harvard university press, 2006.

[15] Ahmad EL Sallab et al. "Deep Reinforcement Learning framework for Autonomous Driving". In: *Electronic Imaging* 2017.19 (2017), pp. 70–76. ISSN: 2470-1173. DOI: doi : 10 . 2352 / ISSN . 2470 - 1173 . 2017 . 19 . AVM - 023. URL: https : //www.ingentaconnect.com/content/ist/ei/2017/00002017/00000019/art00012.

[16] William H Sandholm. "Evolutionary game theory". In: *Complex Social and Behavioral Systems: Game Theory and Agent-Based Models* (2020), pp. 573–608.

[17] Pier Paolo Saviotti and GS Mani. "Competition, variety and technological evolution: a replicator dynamics model". In: *Journal of Evolutionary Economics* 5.4 (1995), pp. 369–392.

[18] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[19] Peter Schuster and Karl Sigmund. "Replicator dynamics". In: *Journal of Theoretical Biology* 100.3 (1983), pp. 533–538. ISSN: 0022-5193. DOI: https://doi.org/10.1016/0022-5193(83)90445-9. URL: http://www.sciencedirect.com/science/article/pii/0022519383904459.

[20] David Silver et al. "Mastering chess and shogi by self-play with a general reinforcement learning algorithm". In: *arXiv preprint arXiv:1712.01815* (2017).

[21] Sriram Srinivasan et al. "Actor-critic policy optimization in partially observable multiagent environments". In: *Advances in neural information processing systems*. 2018, pp. 3422–3435.

[22] Karl Tuyls, Katja Verbeeck, and Tom Lenaerts. "A selection-mutation model for q-learning in multi-agent systems". In: *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*. 2003, pp. 693–700.

[23] Christopher JCH Watkins and Peter Dayan. "Q-learning". In: *Machine learning* 8.3-4 (1992), pp. 279–292.

[24] Tianyang Zhang, Minlie Huang, and Li Zhao. "Learning structured representation for text classification via reinforcement learning". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.