Into the Wild: Approximating Learning Dynamics in 2×2 Games

Mees van Dartel
s5896215

Jiri Derks
s4003039

Abstract

In this paper we explore approximate methods to characterize the convergence properties of learning dynamics in 2×2 games under Experience-Weighted Attraction (EWA). Building on analytical work by Pangallo et al. (2021), we propose a computational approach for qualitative insights into learning dynamics in analytically intractable settings. Our method involves a novel convergence classification algorithm and an Artificial Neural Network (ANN) trained on data generated from simulations of EWA dynamics. We validate our approach by reproducing key analytical results and demonstrate efficient approximation. While our results show promise, they also reveal limitations related to data balance and sensitivity to classification parameters. This work proposes a method for gaining theoretical insight into complex learning dynamics.

1 Introduction

The experience of departing from rationality assumptions when modeling human behaviour was famously described by Sims (1980) as an expedition into the "wilderness of bounded rationality" [12]. Sargent [10] mentions that the difficulties ultimately arise from the fact that there is only one way to be correct, but there are infinite ways to be wrong, resulting in an infinite space of alternative models to explore (a wilderness of bounded rationality). This issue becomes particularly apparent in the context of the game-theoretic concept of the Nash equilibrium (NE). Hart and Mas-Colell [6] discuss how it is notoriously difficult to formulate any sensible adaptive dynamics that convergence to a NE. Naturally, the predictive power of the NE in the context of strategic interaction becomes questionable if learning agents are unable to find this point. However, rejecting the NE means rejecting "correctness", forcing us into the wilderness to find the needle in the haystack: the model that provides the correct prediction.

It is possible to categorize some of the "animals" in this wilderness into several classes. Belief-based models have agents keep track of the history of the other players' actions, create some model of what those players will do in the future, and maximize their expected payoff based on that belief. This category includes fictitious play [2] and best response dynamics [7]. Reinforcement-based models assume that agents update the quality of their possible actions based on feedback from experience. These approaches appear distinct due to their separate origins, belief-based being preferred by game- and decision theorists, and reinforcement-based by psychologists [3]. Yet another class of models stems from evolutionary biology, such as replicator dynamics [11]. For a more extensive overview of learning models, we recommend Fudenberg and Levine (1998)[5]. Such a diversity of alternative approaches might seem like a daunting prospect, however, it is surprisingly possible to include many of the afore mentioned models as limiting cases of a single, more general model: Experience-Weighted Attraction (EWA) [3]. Moreover, it is possible to estimate EWA parameters on experimental data, often resulting in some interior solution, or a mix of the more well-understood limit cases. This creates new problems, however, as interior parameterizations of EWA result in nontrivial dynamics that can make the model difficult to understand.

Pangallo et al. (2021) [9] provide a "taxonomy" of EWA's parameter space for 2×2 games, and showcase several interesting phenomena. For example, for the Matching Pennies game, longer memory makes convergence to equilibia more likely for fictitious play, while the opposite holds true for replicator dynamics. They map out the exact regions of the parameter space where the sign of the memory effect flips. Another interesting example regards the Pareto-dominated NE in cooperation games. For simple learning rules, agents that start close to this point remain there forever. However, for certain EWA values, this is no longer true. They map out the 13-dimensional parameter space of EWA that includes the eight payoffs that characterize 2×2 games, the three EWA parameters, and the thermodynamic beta that determines the stochasticity of the policy. Characterizing this space is quite a challenge, and due to the nonlinearity of EWA they are unable to provide closed-form solutions for the parameter combinations. Instead, they provide qualitative results for the convergence properties of the learning dynamics. They determine for each parameterization whether learning converges to a pure stable NE, a pure stable fixed point, a mixed stable fixed point, or instead produces limit cycles or chaos. Their approach is a modular strategy where they split up the parameter space, using analytical methods for deterministic policies and simulations for stochastic policies. The analytical results require keeping some parameters fixed, so they discuss four extensions they deem interesting, mentioning that, in principle, there is no limit to the cases that could be explored.

In this paper, we take a less sophisticated, "lazy" approach to the characterization of the convergence properties of the parameter space of EWA. We completely circumvent the analytical approach by constructing an algorithm that can quickly classify EWA parameterizations into one of the aforementioned convergence categories. We use this algorithm to train an Artificial Neural Network (ANN) to predict the convergence category for a given parametrization, allowing us to approximate the entire parameter space with relatively little data. A natural question that arises is why this is useful, since the upper bound on the quality of our results are the analytical results, which are already published. Our response is that we can use the analytical results as a benchmark. Our approach can in principle be extended to arbitrary games, and even other applications with learning dynamics, including those that are not as analytically tractable as 2×2 games. We present an efficient approximate method for qualitative insights into the "wilderness of bounded rationality".

In the second section, give an overview of the experience weighted attraction model by [3]. To provide some intuition, we also derive the special cases it reduces to. In the third section we will discuss the previous by Pangallo et al. [9], and describe their methodology. We then introduce our contribution in the context of their work. In the fourth section we present our methodology, consistent of a convergence classification algorithm and a neural network we train to predict the convergence properties for a given parametrization. Section 5 is our results section, where we first validate our models by reproducing the results from Pangallo et al. (2021), and then discuss an experiment where we train on arbitrary games. Section six concludes our paper.

2 Experience-Weighted Attraction

In this section we shortly present the EWA model as introduced by Camerer and Ho (1999) [3]. We study 2-player normal form games. Consider a two-player, two-action game, where the players are denoted by $i \in \{\text{Row}, \text{Column}\}$, and s_i^a for the two possible actions of i by $a = 1, 2$. When players choose their actions, player i receives payoff $\Pi_i(s_i^a, s_{-i}^a)$. Players can play a mixed strategy $\sigma_i = (\mu_i, 1 - \mu_i)$, where μ_i is the probability that player i plays s_i^1. As mentioned, EWA generalises reinforcement- and belief-based learning dynamics. Reinforcement has players select actions based on past performance, whereas belief-based players factor in forgone payoffs. EWA allows for a mix between the two approaches. We conciser a repeated game played at discrete times $t = 1, 2, 3, \ldots$. At each time step, players update two state variables $\mathcal{N}(t)$ and $Q_i^a(t)$. $\mathcal{N}(t)$ represents experience, and is updated as follows:

$$\mathcal{N}(t) = \rho \mathcal{N}(t-1) + 1. \tag{1}$$

Notice how we can interpret ρ as a "discount rate" for experience. For a $\rho \in (1,0)$, $\mathcal{N}(\infty) = 1/(1-\rho)$. For $\rho = 0$, experience is always 1, and for $\rho = 1$ it grows unbounded. $Q_i^a(t)$ is referred to as the attraction of action a. It is updates as follows:

$$Q_i^a(t) = \frac{(1-\alpha)\mathcal{N}(t-1)Q_i^a(t-1)}{\mathcal{N}(t)} + \frac{[\delta + (1-\delta)\mathbb{I}(s_i^a, s_{-i}(t))]\,\Pi_i(s_i^a, s_{-i}(t))}{\mathcal{N}(t)}. \tag{2}$$

In the first term, previous attractions are discounted by experience. We can interpret the α parameter as controlling "memory loss". When $\alpha = 1$, previous attractions are not carried over to the next period. The second term updates the action's attraction based on the payoff Π_i it would have obtained against the other player's action s_{-i}. The indicator function $\mathbb{I}(s_i^a, s_{-i}(t)$ takes value 1 if player i played action s^a at t, and 0 otherwise. The parameter δ controls the degree to which foregone payoffs enter into the update. If $\delta = 1$, then all actions are updated based on their hypothetical payoff, whereas if $\delta = 0$, only the payoff to the action that was played is considered, and EWA reduces to reinforcement learning. Note that when $\mathcal{N}(t) \to \infty$, the second term goes to 0, such that new experiences have a small influence on the attration values for players with large experience. Based off these attractions, players determine their mixed strategies. The probability of playing s_i^a is given by:

$$\mu_i(t) = \frac{e^{\beta Q_1(t)}}{e^{\beta Q_1^a(t)} + e^{\beta Q_2^b(t)}}, \tag{3}$$

Which is the normalized exponential function, or "softmax", with an added β parameter which controls the intensity of choice. When $\beta \to \infty$, the choice becomes deterministic as the probability of playing the action with the highest attraction is 1. When $\beta = 0$, players randomly select an action with uniform probability.

2.1 Special cases

As mentioned, EWA reduces to the learning rules it generalizes in specific limits of its parameters, which we now shortly discuss. We follows Pangallo et al. (2021) [9] and redefine $\rho = (1-\alpha)(1-\kappa)$, since once α is fixed, κ then fully determines ρ, which makes it easier to analyze separately the effects of memory loss and experience discounting. Substituting yields:

$$Q_i^a(t) = \frac{(1-\alpha)\mathcal{N}(t-1)Q_i^a(t-1)}{(1-\alpha)(1-\kappa)\mathcal{N}(t-1)+1} + \frac{[\delta + (1-\delta)\mathbb{I}(s_i^a, s_{-i}(t))]\,\Pi_i(s_i^a, s_{-i}(t))}{(1-\alpha)(1-\kappa)\mathcal{N}(t-1)+1}. \tag{4}$$

Reinforcement learning

Every parameterization where $\delta = 0$ is equivalent to some form of reinforcement learning, as the received payoff is the only one that enters the updating process as the second term of (4) reduces to $\frac{1}{\mathcal{N}(t)}\left(\mathbb{I}(s_i^a, s_{-i}(t))\Pi_i(s_i^a, s_{-i}(t))\right)$, which equals zero whenever a non-played action is considered through the indicator function. The value of κ now determines the type of reinforcement learning, where $\kappa = 1$ means that learning is cumulative, and new experiences have an increasingly small impact:

$$Q_i^a(t) = (1-\alpha)\mathcal{N}(t-1)Q_i^a(t-1) + \mathbb{I}(s_i^a, s_{-i}(t))\Pi_i(s_i^a, s_{-i}(t)), \tag{5}$$

As an illustration, observe that when we subsequently set $\alpha = 0$ we obtain:

$$Q_i^a(t) = \mathbb{I}(s_i^a, s_{-i}(t))\Pi_i(s_i^a, s_{-i}(t)), \tag{6}$$

which is equivalent to the simple "win-stay, lose-shift" strategy (also known as Pavlov), which emerged as an evolutionary outcome from the repeated Prisoner's Dilemma in Nowak and Sigmund (1993) [8].

Best response dynamics

When we set $\alpha = 1, \beta = +\infty$ and $\delta = 1$, EWA reduces to best response dynamics [7], where the player always plays the best response to the other's last action, which requires full consideration of foregone payoffs. Since attractions are scale invariant, we can set $\kappa = 1$ to obtain:

$$Q_i^a(t) = \Pi_i(s_i^a, s_{-i}(t)). \tag{7}$$

Recall that best response dynamics is the simplest belief-based model. This result highlights how EWA generalizes between reinforcement and beliefs, as (7) is simply (6) without the indicator function, controlled setting the degree of consideration of foregone payoffs δ.

Fictitious play

Under the parameterization $\alpha = 0, \kappa = 0, \delta = 1$ and $\beta = +\infty$, EWA reduces to fictitious play [2], which was created based on the idea that players construct a belief on the other players' actions based on their history of play. Players play a best response to the mathematical expectation of the other players' action. Plugging in, we obtain:

$$Q_i^a(t) = \frac{\mathcal{N}(t-1)Q_i^a(t-1)}{\mathcal{N}(t-1)+1} + \frac{\Pi_i(s_i^a, s_{-i}(t))}{\mathcal{N}(t-1)+1}. \tag{8}$$

Since the equivalence to fictitious play is not immediately apparent, we provide a formal proof in Appendix A, following Cramer and Ho (1999) [3]. If we set $\alpha \in (0,1)$ we obtain weighted fictitious play, where recent actions carry more weight. For $\beta < +\infty$, players are no longer deterministic and we obtain stochastic fictitious play.

Two-population replicator dynamics

Replicator dynamics comes out of the field of evolutionary biology, and is based on the idea that populations of players play a repeated game, where the population share of a strategy grows in proportion to its success. In the case where $\beta \to 0, /alpha = 0, \delta = 1 and \kappa \in (0, 1]$, EWA reduces to two-population replicator dynamics. The continuous time limit of the resulting equation can be shown to correspond to that of generalised replicator dynamics. The proof for this equivalence is beyond the scope of this paper and we refer the reader to [9].

Logit dynamics

The final special case of EWA is $\alpha = 1, \delta = 1$ and $\kappa = 1$, which corresponds to a discrete-time version of logit dynamics. This model was introduced by Blume (1993), based on the idea that players update their strategies based on a logit update rule, where a parameter γ determines the degree to which players take optimal versus random actions [1].
Figure 1 from Pangallo et al. (2021) [9] illustrates the location of the special cases of the the EWA parameter space.

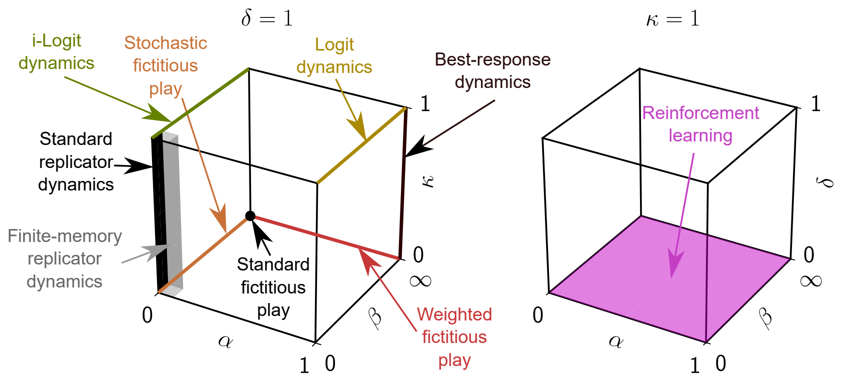

Figure 1: Special cases of EWA, from Pangallo et al. (2021)

3 Previous work

In this section, we shortly summarize the approach of Pangallo et al. (2021) [9], since we build directly on this work. Since the authors' approach to characterizing the parameter space is largely analytical, they start with several simplifications. Since they analyse the asymptotic properties of EWA, they assume that the state variable for experience $\mathcal{N}(t)$ takes on its fixed point value $\mathcal{N}* = 1/(1-(1-\alpha)(1-\kappa))$, which is equivalent to the infinite sum of a geometric series as long as $(1-\alpha)(1-\kappa) < 1$, which they assume to be the case. They then substitute to obtain the following update rule:

$$Q_i^a(t) = (1-\alpha)Q_i^a(t-1) + [1-(1-\alpha)(1-\kappa)]\,[\delta + (1-\delta)\mathbb{I}(s_i^a, s_{-i}(t))]\,\Pi_i(s_i^a, s_{-i}(t)). \tag{9}$$

They subsequently make a "deterministic approximation" of the limit of the dynamics. They argue that this asymptotically valid, even for stochastic dynamics, since in the limit the relative frequency of actions is equal to the mixed strategy they are drawn from. These assumptions allow them to formulate a closed map for both players' action probabilities. [1] Next, they specify additional assumptions that are valid only for specific regions of the parameter space. At this point, it is possible to analytically derive the properties of the resulting dynamical system of coupled equations.

Next, the authors propose a strategy where they subdivide the EWA parameter space into sections, on which they apply methods designed for the specific properties of that section. For each section, they fix specific parameters, and derive the convergence properties of the remaining varying parameters, either analytically or through simulations when no fixed points are stable. They then examine four extensions the authors deemed conceptually interesting. They argue that every possible parametrization can be analyzed in this manner. The remainder of the paper details the specific methods applied for each region of the parameter space. They also provide qualitative results that illustrate the convergence properties of different regions of the parameter space. Pangallo et al. (2021) provide a systematic characterization of EWA learning dynamics, and highlight some particular interesting phenomena. In their conclusion, they mention that the main purpose of the paper is to provide theoretical guidance on learning dynamics to be expected in experiments, as EWA is a popular model of learning behaviour in experimental settings.

3.1 Our contribution

As mentioned above, there is utility in providing qualitative insight into learning dynamics. Pangallo et al. (2021) [9] study 2×2 games not only because they encompass many of the strategic tensions studied by game theorists, but also for analytical tractability. Even then, it is not possible to provide closed-form solutions, and simulation is required at several points of the analysis. We argue that many relevant game theoretic applications include more complex games, including those with larger action spaces, higher player counts, asymmetries, those that are not described in normal form. Specifically, we focus on situations where simulation is the only available method. Due to the large parameter space, computational complexity can explode. With the purpose of providing qualitative insight, we develop a method efficiently approximate the convergence properties of the learning dynamics, which we discuss in the remaining sections of this paper.

4 Methodology

The 13-dimensional parameter space of EWA can make dynamics computationally expensive to evaluate. In terms of computational complexity, descritization of this space has a time complexity of $O(n^{13})$, where n is the number of discrete points along each dimension. For each point, EWA has to be updated until convergence. Since different parametrization for EWA have different convergence properties the number of iterations until convergence can vary. Let T be the maximum number of iterations until convergence for a single EWA evaluation. The upper bound on the time complexity of the descritization is then $O(n^{13} \times T)$, which is enormous even at very coarse resolutions. Our goal is therefore to

[1] We do not provide these equations as they are quite uninformative.

sufficiently approximate the true parameter space while minimizing amount of samples required. Our method consists of two main components: a convergence classification algorithm that classifies a given point in the parameter space, and an Artificial Neural Network (ANN) that approximates the space between the sampled points.

4.1 Convergence Classification Algorithm

Recall that the time complexity of approximating EWA has an upper bound of $O(n^{13} \times T)$. It is directly proportional to T, which is the maximum iterations per parameter combination until convergence. For some EWA can converge slowly or does not converge at all. Our first challenge is then to find convergence criteria that balance accuracy and speed. Based on Pangallo et al. (2021) [9], we want to classify specific EWA parametrisations into one of the following categories: a pure stable NE, a pure stable fixed point, a mixed stable fixed point, or limit cycles or chaos. We do not consider the possibility of multiple fixed points.
The parameters of the convergence classification algorithm are described in table 1. The algorithm itself is composed of two sub-algorithms: the EWA update step and the convergence classification, presented below in pseudo-code, and in Julia code in the appendix. Algorithm 1 is our implementation of equations (1), (2) and (3). Algorithm 2 requires some further explanation. A single EWA update requires constant time, such that the time complexity of a single run is $O(T)$, or linear in the maximum number of iterations. Since our goal is efficient approximation, our first simplification is to set an upper bound on T by specifying the number of iterations within the dynamics should converge before we classify as "limit cycles or chaos". If dynamics converge before T, we immediately break the loop. To classify convergence, we first require a minimum number of iterations γ (line 6) before we start checking, since our convergence criterion requires at least γ iterations. Next, line 7 specifies our convergence criterion. We slice off the last γ time steps, and check whether the maximum distance between every of the entries is below our tolerance level ϵ_1, which determines the strictness of the convergence criterion. If this condition is satisfied, we next check whether the converged strategy σ_t is an NE. Note that, depending on the strictness of the convergence criterion, σ_t might be close, but not completely converged to an NE. Therefore, we specify a tolerance level ϵ_2 within which we classify as NE. If σ_t is too far from an NE, it is classified as a fixed point. Next, we check if this fixed point is pure. Note, again, that σ_t might not have fully converged, we set another tolerance level ϵ_3 to classify pure fixed points. If none of the above conditions are met, then the fixed point must be mixed, and it is classified as such. For the details of our Julia implementation, see Appendix B.

Parameter	**Description**
Q_0	Prior attractions
N_0	Prior experience
α	Attraction decay rate (memory loss)
κ	Experience decay rate (discount rate)
δ	Weight on foregone payoffs (degree of belief learning)
β	Intensity of choice / stochasticity (thermodynamic beta)
$\Pi^{\mathcal{G}}$	Payoff matrix for game $\mathcal{G}$
$\text{NE}^{\mathcal{G}}$	Precomputed list of all pure and mixed Nash Equilibria of $\mathcal{G}$.
T	Maximum number of iterations (default = 10.000)
γ	Length of window for convergence check
ϵ_1	Tolerance level for convergence classification
ϵ_2	Tolerance level for NE classification
ϵ_3	Tolerance level for pure FP classification

Table 1: Convergence classification parameters.

Algorithm 1 EWA (update step)

```
Require: Input: $Q_{t-1}$, $\mathcal{N}_{t-1}$, $\alpha$, $\kappa$, $\delta$, $\beta$, $\Pi$
Ensure: Output: $s_t$, $\sigma_t$, $Q_t$, $\mathcal{N}_t$
 1: Step 1: Select actions
 2: for $i \in \{1,2\}$ do                                        ▷ Iterate over players
 3:     $Q_i \leftarrow Q_{t-1}[i]$                                ▷ Get player attractions
 4:     $\sigma_i = \text{softmax}(Q_{t-1}[i])$                    ▷ Compute mixed strategy
 5:     $s_i \sim \sigma_i$                                       ▷ Draw action from mixed strategy
 6: end for
 7: $s = (s_1, s_2)$                                             ▷ Return selected actions
 8: $\sigma = (\sigma_1, \sigma_2)$                              ▷ Return mixed strategies
 9:
10: Step 2: Update attractions
11: $\mathcal{N}_t = (1-\alpha)(1-\kappa)\mathcal{N}_{t-1} + 1$
12: for $i \in \{1,2\}$ do
13:     for $a \in \{a_1, a_2\}$ do                              ▷ Iterate over action space
14:         if $a = s_i$ then $\mathbb{I} = 1$ else $\mathbb{I} = 0$      ▷ Indicator if action was played
15:         $Q_i^a = (1-\alpha)\mathcal{N}_t Q_i^a + (\delta + (1-\delta)\mathbb{I})\Pi(a, s_{-i})$   ▷ Compute new attractions
16:     end for
17:     $Q_i = (Q_i^1, Q_i^2)$                                    ▷ Player $i$'s updated attractions
18: end for
19: $Q_t = (Q_1, Q_2)$                                            ▷ Updated attractions
20: return $\sigma_t$, $Q_t$, $\mathcal{N}_t$
```

Algorithm 2 Categorizing convergence

```
Require: Input: $Q_0$, $\mathcal{N}_0$, NE, $\alpha$, $\kappa$, $\delta$, $\beta$, $\Pi$, $T$, $\gamma$, $\epsilon_1, \epsilon_2, \epsilon_3$
Ensure: Output: $c \in \{1,2,3,4\}$                                ▷ Convergence categories
 1: Initialize: $c \leftarrow 1$          ▷ "Limit cycles or chaos" in case of no convergence within $T$
 2: $H_\sigma = [\sigma_0]$                                     ▷ Initialize vector to keep track of history
 3: for $t = 1$ to $T$ do
 4:     $s_t$, $\sigma_t$, $Q_t$, $\mathcal{N}_t \leftarrow \text{EWA}(Q_t, \mathcal{N}_t)$      ▷ Do EWA update step
 5:     Append $\sigma_t$ to $H_\sigma$
 6:     if $t > \gamma$ then                                     ▷ If minimum required iterations are completed
 7:         $\Delta_{\max} \leftarrow \max_{i,j\in(t-\gamma,t)} |\sigma_i - \sigma_j|$   ▷ Find max distance between strategies for last $\gamma$ time steps
 8:         if $\Delta_{\max} < \epsilon_1$ then                   ▷ Convergence criterion
 9:             if $|\sigma_t - \text{NE}| < \epsilon_2$ then
10:                 $c \leftarrow 4$                               ▷ NE criterion
11:             else if $\exists i, \exists a : |\sigma_i^a - 1| < \epsilon_3$ then   ▷ Check if fixed point is pure
12:                 $c \leftarrow 3$
13:             else
14:                 $c \leftarrow 2$                               ▷ Fixed point is mixed
15:             end if
16:             break
17:         end if
18:     end if
19: end for
20: return $c$
```

4.2 Data generation

Using the convergence classification algorithm we can generate labeled data for EWA parameterizations over arbitrary 2×2 games. To precompute the true NE for the input games, we utilize the *GameTheory.jl* package. We compute all NE using support enumeration, an algorithm with a time complexity that grows in the size of the action space A_i as $\binom{2A_i}{A_i - 1}$, which is fast for small games. The rest of our data generation process is as follows; For the parameters α, κ and δ, that are defined over $[0, 1]$, we discretize their ranges with a resolution of 0.0001. Since $\beta \in [0, +\infty)$, we first generate a discrete range between -0.5 and 1.5 with 0.0001 resolution, and then apply the exponential function on the points to generate a log-normally distributed range of values that is large enough to capture all of the interesting regions of the parameter space. From these four ranges, we randomly sample n points with uniform probability. We use the convergence classification algorithm to generate labels for these points. We set the following criteria: $T = 1000,\ \gamma = 6,\ \epsilon_1 = 0.003, \epsilon_2 = 0.001,\ \epsilon_3 = 0.003$. We then create a train/test split, which we feed into our neural network. We chose these values by trading off sufficiently accurate classifications against computation time. For our Julia implementation of the data generation, see Appendix C. We are able to generate 1.000.000 data points in about 10 minutes.

4.3 Neural Network Architecture

Our neural network employs a simple feedforward architecture comprising four fully connected (dense) layers. Each hidden layer contains 32 neurons and utilizes the Rectified Linear Unit (ReLU) activation function. The output layer is a dense layer with four units, corresponding to the four convergence categories, and applies a softmax activation to produce class probabilities. The network is trained for 2500 epochs using a batch size of 64. Cross-entropy loss is adopted as the objective function, which is standard for classification tasks. Our implementation in Julia can be found in Appendix D.

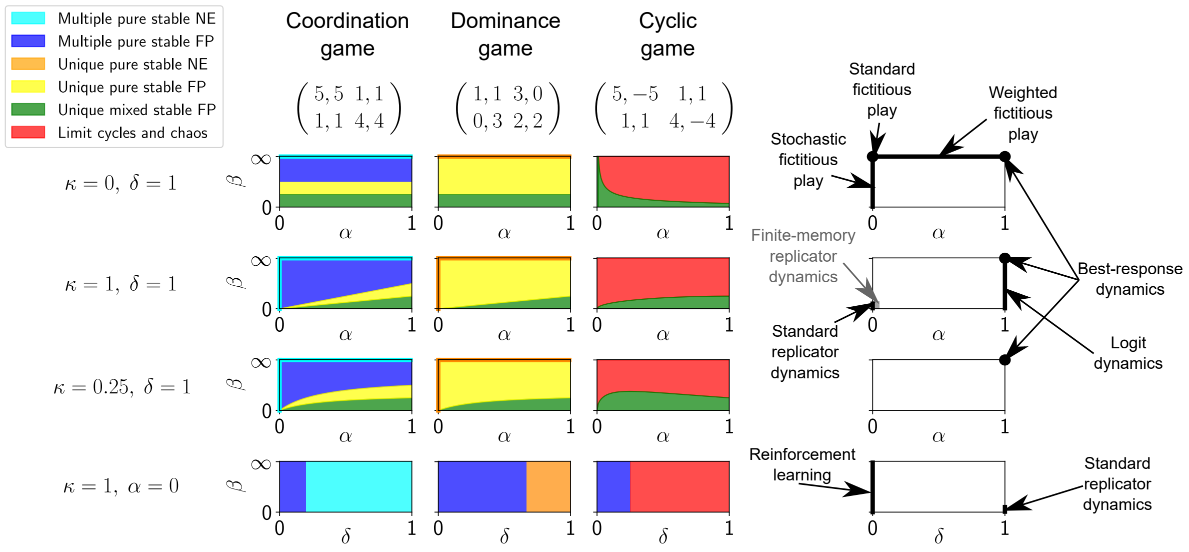

Figure 2: Qualitative characterisation of EWA, from Pangallo et al. (2021). They consider four cuts through the parameter space defined by the restrictions $\kappa = 0,\ \delta = 1;\ \kappa = 1,\ \delta = 1;\ \kappa = 0.25,\ \delta = 1;$ and $\kappa = 1,\ \alpha = 0$. Presented are the asymptotic outcome of learning for three games. (i) In cyan areas, learning converges to one of multiple pure NE; (ii) in blue zones, it converges to one of multiple fixed points that are located "close" to pure NE or at alternative pure strategy profiles; (iii) in orange areas, it converges to a unique pure strategy NE; (iv) in yellow zones, learning reaches a unique fixed point located close to a pure NE or at another pure strategy profile; (v) in green areas, it converges to a fixed point in the center of the strategy space; (vi) in red areas, it does not converge to any fixed point. On the right, they show the parameterizations for which EWA reduces to special cases.

5 Results

In this section we present the results for our two experiments. First, we try to reproduce the analytical results from [9], as validation for our methodology. Second, we experiment with training the network on arbitrary games.

5.1 Model validation

To validate our method, we first try to reproduce the analytical results from [9]. In particular, we focus on their qualitative characterization, which they conveniently summarize in a figure we have included in figure 2. If we are able to produce qualitatively similar plots from our generated data, we argue it suggests that our model is sufficiently accurate to generate the qualitative insight we aim for. We train three separate models for the three benchmark games we now describe.

5.1.1 Coordination

The payoff matrix for the coordination game we explore is given by table 2. All coordination games have three Nash Equilibria; those for our specific game we compute as:

$$\mathrm{NE}^c = \{(1.0, 0.0), (1.0, 0.0)\},\ \{(0.0, 1.0), (0.0, 1.0)\},\ \{(0.429, 0.571), (0.429, 0.571)\}.$$

Note that there are two pure and one mixed equilibria, where $\{\mathbf{T}, \mathbf{L}\}$ is Pareto dominant.

	L	**R**
T	$(5, 5)$	$(1, 1)$
B	$(1, 1)$	$(4, 4)$

Table 2: *Coordination* payoff matrix

For our analysis, we simply check if any of these equilibria is sufficiently close to the outcome of the learning dynamics. If so, we mark the point as NE. If not, we check whether the point is a pure or a mixed fixed point. If dynamics do not converge, then they either exhibit limit cycles or chaos. Note that we analyze the dynamics of the *mixed strategies* from which the players draw their actions, not those of the realized actions.

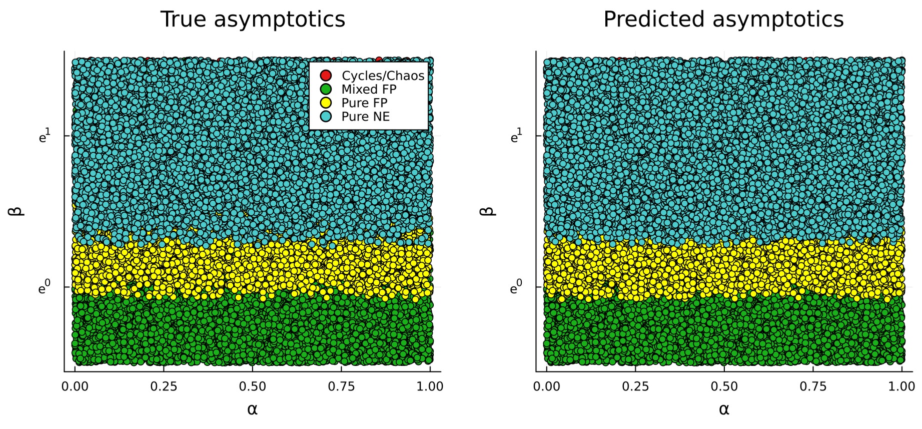

Figure 3: Qualitative characterization of the parameter space defined by the restrictions $\kappa = 0$, $\delta = 1$ (i) For cyan points, learning converges to one of multiple NE; (ii) for green points, it converges to a fixed points ; (iii) in yellow areas, it converges to a mixed fixed point; (iv) in red areas, it does not converge to any fixed point.

We trained our network on 25.000 randomly generated data points, and tested the accuracy of the subsequent predictions on 25.000 more randomly generated points. The network was trained for 2500 epochs with the AdaGrad optimizer [4]. This model trained on the coordination game attained an 88, 068% accuracy on the test data. In figure 3, we highlight an illustration of the restriction $\kappa = 0, \delta = 1$, which represents hybrids of stochastic and weighted fictitious play.
Observe that our results are highly similar to those seen in the top-left panel of figure 2, which represent the same setting. We make several observations; first of all, we obtain exactly the same insight that while the dynamics are invariant to α, they range from mixed FP, to pure FP, to Pure NE [2] as a function of the intensity of choice β. The only big qualitative difference between us and [9] is the size of the NE region. For higher values of β, they obtain fixed points "close" to the pure NE that only manifests at $\beta = +\infty$. The reason that our results show a larger region of NE is that our NE classification is approximate, based on tolerance level ϵ_3. For any $\beta <= \infty$, there remains some attraction towards the action that is part of the Pareto dominated NE, such that the probability of playing the Pareto dominant action is never completely 1. Our algorithm however classifies these points as "sufficiently close", which explains the upper region of our plot. Finally, note that the transition from mixed to pure FP happens around the point where $\beta = 1$. This result is intuitive, as this is exactly where the intensity of choice start to kick in.

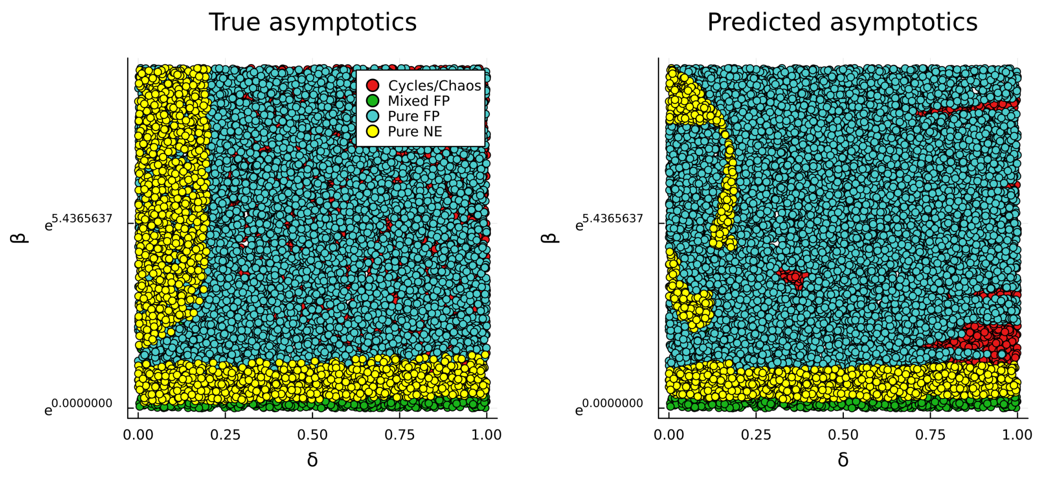

Figure 4: Qualitative characterization of the parameter space defined by the restrictions $\alpha = 0,\ \kappa = 1$.

We also replicate the restrictions $\alpha = 0,\ \kappa = 1$, which interpolates between reinforcement learning and best response dynamics, as illustrated in figure figure 4. For this model, the accuracy drops to 60.25%. Notice that we have increased the range for β to showcase the similarity with the benchmark in the bottom-left corner of figure 2. We see the same region of fixed points for low values of δ, that turns into NE's for higher values. This examples showcases the drawbacks from our approximate classification. Since parameterizations were here only given 1000 iterations to converge, we have quite some noise on the right are of the plot. We also see that our convergence criterion, which here was $\epsilon_1 = 0.01$, might have been too loose, as for the lower regions, the dynamics stop before full convergence to an NE. Alternatively, the NE tolerance level is was too tight ($\epsilon_2 = 0.001$), so that the bottom region is wrongly classified as a FP. This illustrates the sensitivity of the model to the tolerance levels, which in future work should be explored in more detail. However, still consider the fact we are able to obtain results that are qualitatively similar to the analytical benchmark, as evidence for the validity of our model.

[2]Note that the index has a mistake; the cyan points are NE and the yellow are pure FP.

5.1.2 Dominance

Dominance games (or dominance-solvable games) have a unique pure NE. The prisoners' dilemma is perhaps the most famous instance of this class. For our analysis, we use the payoff matrix specified in table 3. The unique pure NE is given by:

$$\mathrm{NE}^d = \{(0.0, 1.0), (0.0, 1.0)\}.$$

The results are illustrated in figure 5.

	L	**R**
T	$(1, 1)$	$(3, 0)$
B	$(0, 3)$	$(2, 2)$

Table 3: *Dominance* payoff matrix

Compared with the analytical benchmark (top-middle in fig. 2), we again observe a similar qualitative result: mixed FP in the lower region and pure FP for higher values of β. Our model accuracy here is 97.81%, which is very promising.

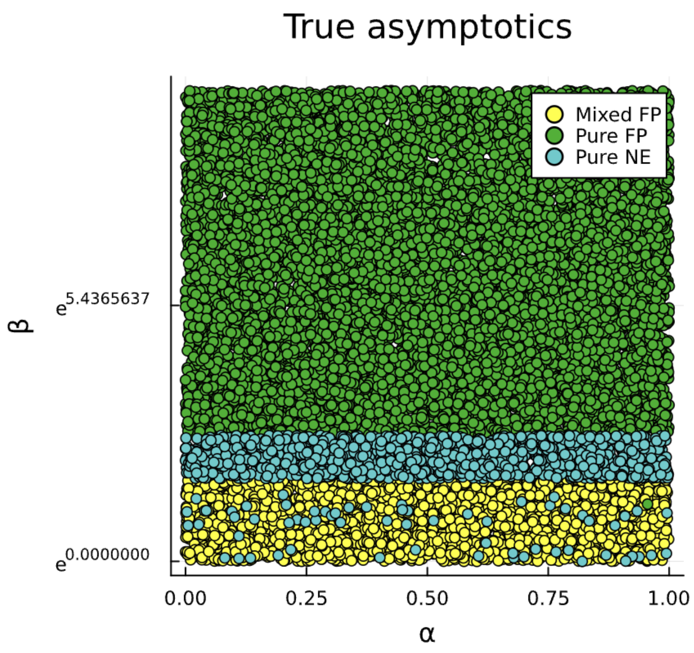

Figure 5: Dominance game asymptotics for the parameter space defined by the restrictions $\kappa = 0,\ \delta = 1$.

Notice that there is a region wrongly classified NE. Again, this highlights the drawbacks of the tolerance levels that allow us to compute our approximations efficiently. With stricter criteria, we more accurately approximate the truth, however we also increase the required computational load. This tradeoff is inherent to our method, and requires careful consideration.

5.1.3 Cyclical

Finally, we explore the cyclical game with the payoff matrix given by table 4. An important feature of cyclical games is that under best response dynamics, players would be stuck in a cycle of best replies. Cyclical games have no pure strategies and a single mixed strategy. Our unique mixed NE is given by:

$$\text{NE}^{cyc} = \{(0.45, 0.54), (0.42, 0.57)\}.$$

	L	**R**
T	$(5, -5)$	$(1, 1)$
B	$(1, 1)$	$(4, -4)$

Table 4: *Cyclical* payoff matrix

Here we obtain a high accuracy of 91.28%, and a high resemblance to the analytical benchmark (top right, fig. 2). For the cyclical game, we reduced the number of observations to 10.000, since the dynamics do not converge for a large region of the space, so that the length of the classification loop will often equal to the maximum number of iterations T. We find an interesting phenomenon in that for the predicted results, the boundaries are way more defined than in the sparse, noisy sample. This is a benefit of our approximation approach; even though the sampling is noisy due to the high tolerance levels, the neural net is able to provide a clearer picture of the underlying reality.

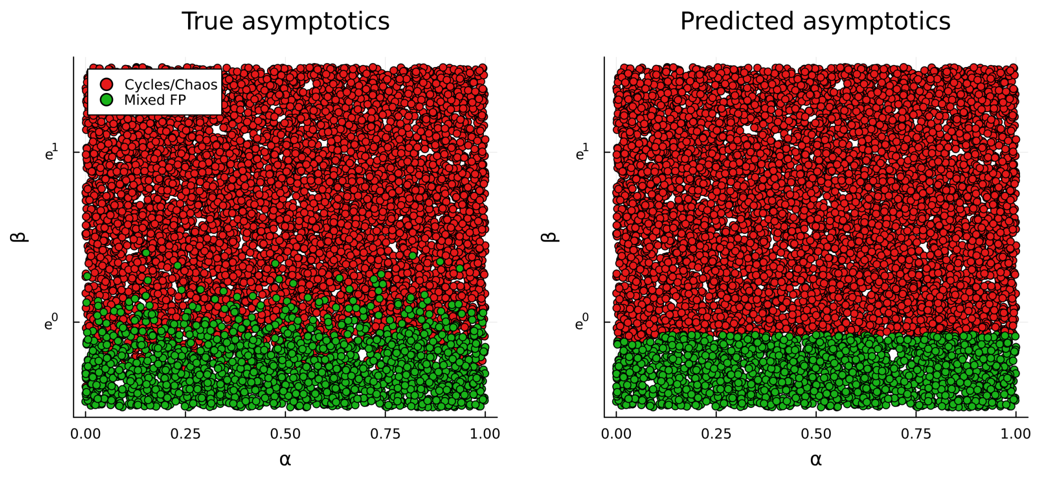

Figure 6: Cyclical game asymptotics for the parameter space defined by the restrictions $\kappa = 0,\ \delta = 1$.

5.1.4 Discussion of validation

We want to make some remarks regarding our validation results. We believe we have shown that 1. the classification algorithm works as intended and 2. the neural net can successfully approximate the entire space from the sampled points, given that the data is sufficiently precise. However, it is important to mention that the results are very sensitive to the strictness of the classification algorithm. More specifically, the three tolerance levels and the iteration maximum T. It should be further explored how to optimally trade off accuracy against efficiency. Sometimes the model can "see through" the noisy data and find the underlying pattern, such as with the cyclical game, which is when our method shines. Other times, such as with the second benchmark for the cooperation game, the noise results in poor accuracy. We do believe that further fine-tuning could significantly improve results.

6 Conclusion

In this paper we have have built upon the work of Pangallo et al. (2021) [9], who analytically the derive the asymptotic properties of learning dynamic for the Experience Weighted Attraction (EWA) model on 2×2 games, which provides theoretical guidance on the learning dynamics to be expected in experiments. We anticipate a need for a similar qualitative insight into learning dynamics on less analytically tractable games, and present a method to computationally approximate the asymptotic properties of learning dynamics. Using a new algorithm that classifies convergence classes for given tolerance levels, designed to reduce time complexity, we generate data on which we train an artificial neural network (ANN), that approximates the asymptotic results for of entire EWA parameter space. We show the validity of our model against the analytical benchmark from Pangallo et al. (2021), and experiment with training on randomized games. We demonstrate both the strengths and weaknesses of the method, and conclude that approximate classification could be a promising way of gaining qualitative insight into analytically intractable, high-dimensional models, while maintaining computational efficiency.

Source code is available at `https://github.com/mvdmvd/DeepEWA.git`.

References

[1] L. E. Blume. The statistical mechanics of strategic interaction. *Games and Economic Behavior*, 5:387–424, 1993.

[2] G. Brown. Iterative solutions of games by fictitious play. In T. Koopmans, editor, *Activity Analysis of Production and Allocation*. Wiley, New York, 1951.

[3] C. Camerer and T.-H. Ho. Experience-weighted attraction learning in normal form games. *Econometrica*, 67(4):827–874, 1999.

[4] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

[5] D. Fudenberg and D. K. Levine. *The Theory of Learning in Games*. MIT Press, 1998.

[6] S. Hart and A. Mas-Colell. Uncoupled dynamics do not lead to nash equilibrium. *The American Economic Review*, 93(5):1830–1836, 2003.

[7] A. Matsui. Best response dynamics and socially stable strategies. *Journal of Economic Theory*, 57:343–362, 1992.

[8] M. A. Nowak and K. Sigmund. The alternating prisoner's dilemma. *Journal of Theoretical Biology*, 168(2):219–226, 1994.

[9] M. Pangallo, J. B. Sanders, T. Galla, and J. D. Farmer. Towards a taxonomy of learning dynamics in 2×2 games. *Games and Economic Behavior*, 128:1–25, 2021.

[10] T. Sargent. Evolution and intelligent design. *American Economics Review*, 98(1):5–37, 2008.

[11] P. Schuster and K. Sigmund. Replicator dynamics. *Journal of Theoretical Biology*, 100:533–538, 1983.

[12] C. A. Sims. Macroeconomics and reality. *Econometrica*, 48(1):1–48, 1980.

Appendix

Appendix A: Proof that EWA generalizes fictitious play

Proof. In fictitious play, player i's beliefs about the choices of $-i$ are determined by the relative frequencies of $-i$'s previously chosen actions. Let the played frequency of action s^a_{-i} be denoted by $N^s_{-i}(t)$; then beliefs are given by the relative frequency of s^a_{-i}, given by:

$$B^s_{-i}(t) = \frac{N^s_{-i}(t)}{N(t)}, \quad \text{with } N(t) = \sum_{s=1}^{a_{-i}} N^s_{-i}(t), \tag{1}$$

where $N(t)$ represents the count of player $-i$'s total actions played. Beliefs are updated as:

$$B^s_{-i}(t) = \frac{N^s_{-i}(t-1) + \mathbb{I}(s^a_i, s_{-i}(t-1))}{\sum_{s=1}^{a_{-i}} N^s_{-i}(t-1) + \mathbb{I}(s^a_i, s_{-i}(t))}. \tag{2}$$

We can rewrite (2) recursively to express beliefs in terms of previous period beliefs:

$$B^s_{-i}(t) = \frac{B^s_{-i}(t-1) + \frac{\mathbb{I}(s^a_i, s_{-i}(t-1))}{N(t-1)}}{1 + \frac{1}{N(t-1)}} = \frac{N(t-1)B^s_{-i}(t-1)}{N(t-1)+1} + \frac{\mathbb{I}(s^a_i, s_{-i}(t-1))}{N(t-1)+1}. \tag{3}$$

Expected payoffs for period t are given by:

$$E^a_i(t) = \sum_{s=1}^{a_{-i}} \Pi_i(s^a_i, s_{-i}(t)) \cdot B^s_{-i}(t), \tag{4}$$

into which we can plug (3) and rearrange to obtain:

$$E^a_i(t) = \frac{N(t-1)E^a_i(t-1)}{N(t-1)+1} + \frac{\Pi_i(s^a_i, s_{-i}(t))}{N(t-1)+1}. \tag{5}$$

Observe that this is the functional form from (8), so that when priors are equally initialized such that $E^a_i(0) = Q^a_i(0)$, we have $E^a_i(t) = Q^a_i(t)$. Hence, fictitious play is a special case of EWA. □

Appendix B: Categorization algorithm in Julia

```julia
module pEWA
using Distributions, GameTheory

function find_NE_mixed(payoff::Vector{Matrix{Int64}})
    g = NormalFormGame([Player(payoff[1]), Player(payoff[2])]) # GameTheory.jl game formatting
    found = support_enumeration(g) # solve for NE with GameTheory.jl
    NE = [collect(ne) for ne in found] # convert to regular vector
    return NE
end

function init_pEWA(; # initialisation function
    s=[0, 0],                   # empty actions vector
    μ=[], # μ represents the probability of playing a
    Q=[[0.0001, 0.0001], [0.0001, 0.0001]], # prior attractions (technically a parameter)
    N=0.0,                      # prior history (also a parameter)
    α=1.0,                       # memory loss
    κ=0.0,                       # discount rate
    δ=1.0,                       # degree of foregone payoffs consideration
    β=Inf64,                     # thermodynamic beta, controls stochasticity (β→lim  = deterministic
    ↪  policy, β→lim  = fully random policy)
    game=dom_game)
    return s, μ, Q, N, α, κ, δ, β, game # default parameterisation: best response dynamics for a
    ↪  domination game (pure NE always found).
end

function pEWA_step!( # step function first selects actions based off attractions, then updates
↪  attractions based off actions
    s::Vector{Int64}, μ::Vector{Any}, Q::Vector{Vector{Float64}}, N::Float64,
    α::Float64, κ::Float64, δ::Float64, β::Float64, payoff::Vector{Matrix{Int64}})

    σ = [[0.0, 0.0], [0.0, 0.0]]
    # new actions
    @inbounds for i  1:2 # iterate over players
        Q, Q = Q[i...] # get current attractions
        local s1, s2 = β * Q, β * Q
        local m = s1 > s2 ? s1 : s2 # find max
        local exp1, exp2 = exp(s1 - m), exp(s2 - m)
        local p = exp1 / (exp1 + exp2) # softmax, subtracting the max for numerical stability
        s[i] = rand(Bernoulli(p)) == 1 ? 1 : 2 # draw action
        σ[i] = [p, 1 - p] # store mixed strategy
    end
    push!(μ, σ)

    # update attractions
    local N = (1 - α) * (1 - κ) * N + 1 # update history
    @inbounds for i  1:2
        j = 3 - i # j is other player
        for a  1:2 # iterate over action space
            I = (a == s[i]) ? 1.0 : 0.0 # indicator function for action was played
            Π = payoff[i][a, s[j]] # payoff for action
            local oldQ = Q[i][a]
            Q[i][a] = ((1 - α) * N * oldQ + (δ + (1 - δ) * I) * Π) / N # EWA updating
        end
    end
    return s, μ, Q, N
end

function multicat_pEWA(parameters::Tuple{Vector{Int64},Vector{Any},Vector{Vector{Float64}},
Float64,Float64,Float64,Float64,Float64,Vector{Vector}};
    T=5000)
    s, μ, Q, N, α, κ, δ, β, game = parameters
    payoff, NE = game

    # convergence criterion is whether the expectation of the histories of play are an NE
    local cat = 1 # 1 = cycles/chaos, 2 = mixed FP, 3 = pure FP, 4= pure NE
    s, μ, Q, N = probsEWA_step!(s, μ, Q, N, α, κ, δ, β, payoff) # the first EWA step is based on the
    ↪  priors
    @inbounds for t in 1:T # after T iterations, classifies as cycles/chaos
        s, μ, Q, N = pEWA_step!(s, μ, Q, N, α, κ, δ, β, payoff)
        if t > 6 && all(isapprox(μ[end-3:end-1], μ[end-2:end], atol=0.005)) # tolerance level
            if any(isapprox(μ[end], ne, atol=0.005) for ne in NE) # tolerance level
                cat = 4
            else
                any(x -> isapprox(x, 1.0, atol=0.005), μ[end][1]) ? cat = 3 : cat = 2 # tolerance level
                ↪
            end
            break
        end
    end
    return cat
end
end
```

Appendix C: Data generation in Julia

```
using Flux, .pEWA, Random, IterTools, ProgressMeter, DataFrames, CSV, Statistics

# precompute some benchmark games
coord = [[[5 1; 1 4], [5 1; 1 4]], pEWA.find_NE_mixed([[5 1; 1 4], [5 1; 1 4]])]
dom = [[[5 0; 20 1], [5 0; 20 1]], pEWA.find_NE_mixed([[5 0; 20 1], [5 0; 20 1]])]
cyclic = [[[5 1;1 4],[-5 1; 1 -4]], pEWA.find_NE_mixed([[5 1;1 4],[-5 1; 1 -4]])]

# set game and amount of observations
game = coord
points = 100_000

# generate data
α_grid, κ_grid, δ_grid = [rand(0.0:0.0001:1.0, points) for _ in 1:3]
β_grid = exp.(rand(-0.5:0.001:1.5, points)) # take exp to get the lim to infty effect since β is
↪  unbounded above
combs = [[α_grid[i], κ_grid[i], δ_grid[i], β_grid[i]] for i in 1:points]

# make empty 50/50 train test split vectors
subset=Int64(floor(0.5*length(combs)))
x_train, x_test = combs[1:subset], combs[(subset+1):end]
y_train, y_test = Vector{Int64}(undef, length(x_train)),Vector{Int64}(undef, length(x_test))

# populate vectors with convergence classifications
@showprogress for i in 1:length(x_train)
    comb=x_train[i]
    α, κ, δ, β = comb
    params = pEWA.init_pEWA(;α=α, κ=κ, δ=δ, β=β,game=game)
    cat = pEWA.multicat_pEWA(params)
    y_train[i] = cat
end

@showprogress for i in 1:length(x_test)
    comb = x_test[i]
    α, κ, δ, β = comb
    params = pEWA.init_pEWA(;α=α, κ=κ, δ=δ, β=β,game=game)
    cat = pEWA.multicat_pEWA(params)
    y_test[i] = cat
end

# formatting for Flux.jl NN
x_train = hcat([Float32.(x[1:4]) for x in x_train]...)
x_test = hcat([Float32.(x[1:4]) for x in x_test]...)
```

Appendix D: Neural network implementation in Julia

```julia
# 4 layer dense model
deepEWA= Chain(
    Dense(4 => 32, relu),
    Dense(32 => 32, relu),
    Dense(32 => 32, relu),
    Dense(32 => 4)
    )

# specify data loader and optimiser
target = Flux.onehotbatch(y_train, 1:4)
loader = Flux.DataLoader((x_train, target), batchsize=64, shuffle=true);
opt = Flux.setup(Flux.AdaGrad(), deepEWA)

# training loop
losses=[]
@showprogress for epoch in 1:2500
    for xy in loader
        x,y = xy
        loss, grads = Flux.withgradient(deepEWA) do m
            y_hat = m(x)
            # cross entropy for classification
            Flux.logitcrossentropy(y_hat, y)
        end
        Flux.update!(opt, deepEWA, grads[1])
        push!(losses, loss)
    end
end

# generate test data
out2 = deepEWA(x_test)
probs2 = softmax(out2)
predicted_labels = Flux.onecold(probs2, 1:4)
```