Infinite population evolutionary dynamics match infinite memory reinforcement learning dynamics

Wolfram Barfuss
School of Mathematics, University of Leeds, UK
Max Planck Institute for Mathematics in the Sciences, Leipzig, GER
barfuss@mis.mpg.de

ABSTRACT

Reinforcement learning algorithms have been shown to converge to the classic replicator dynamics of evolutionary game theory, which describe the evolutionary process in the limit of an infinite population. However, it is not clear how to interpret these dynamics from the perspective of a learning agent. In this paper, we propose a data-inefficient batch-learning algorithm for temporal difference Q learning and show that it converges to a recently proposed deterministic limit of temporal difference reinforcement learning. In a second step, we state a data-efficient learning algorithm, that uses a form of experience replay, and show that it retains core features of the batch learning algorithm. Thus, we propose an agent-interpretation for the learning dynamics: What is the infinite population limit of evolutionary dynamics is the infinite memory limit of learning dynamics.

KEYWORDS

Multi-agent reinforcement learning; Evolutionary game theory; Batch learning; Experience replay; Stochastic games

1 INTRODUCTION

Collectives of autonomous learning agents are a widely accepted method for solving problems of distributed nature. Consequently, the field of multi-agent learning has developed various algorithms [8], yet, multiple challenges remain: nonstationarities due to other learning agents, the curse of dimensionality of the joint-state-action space, the increased number of hyperparameters, that need tuning, coordination needs between agents and the possibility of social dilemmas [31].

A dynamical systems approach, based on the link between evolutionary game theory and reinforcement learning, can help to overcome these challenges by providing improved, qualitative insights into the emerging collective learning dynamics [6]. The relationship between the two fields is as follows: one population with a frequency over phenotypes in the evolutionary setting corresponds to one agent with a frequency over actions in the learning setting [29]. For example, when studying the emergence of cooperation, one population composed of cooperators and defectors corresponds to one agent with a mixed strategy over a cooperative and a defective action. In their seminal work, Börgers and Sarin showed how one of the most basic reinforcement learning update schemes, Cross learning [10], converges to the deterministic replicator dynamics of evolutionary games theory [7]. Likewise, the convergence to the replicator dynamics has been shown for single-state Q learning [27, 30].

This deterministic - sometimes also called *evolutionary* - limit can be taken in multiple ways. In continuous time, the learning rate is sent to zero [27, 30]. In discrete-time, the batch size of a batch learning algorithm is sent to infinity [5, 12–14]. In essence, both ways assume that policy updates occur on much slower time scales than actual interactions with other agents and the environment.

However, it is still unclear how algorithmic implementations and analytical equations relate to each other, or in other words, how to interpret this deterministic limit in the context of reinforcement learning agents. Improved understanding of such learning equations is important to advance their practical use for overcoming critical challenges of multi-agent reinforcement learning.

The classic replicator equations have a clear interpretation. They model the dynamics of an infinite population evolving under the pressures of selection [18]. In this paper, we propose to interpret the deterministic, *evolutionary*, limit of reinforcement learning as learning in the infinite memory limit.

We focus on the discrete-time limit of temporal difference reinforcement learning dynamics with discounted rewards [3]. After introducing necessary background (Sec. 2) we first propose a novel temporal difference batch-learning algorithm for Q learning (Sec. 3). We show that this batch learning algorithm matches the deterministic learning equations for large batch sizes (Sec. 4). Yet, it requires many interactions with the environment and is therefore highly data-inefficient.

Second, we transform the data-inefficient batch learning algorithm into a data-efficient learning algorithm, that uses a form of experience replay. We shift the batch of actual interactions with the environment into the memory of the agent (Sec. 5). Thus, we can conclude (Sec. 6), what is the infinite population limit of evolutionary dynamics is the infinite joint-action memory limit of learning dynamics.

Related work. Previous research on the dynamics of learning found that discrepancies can arise between the predictions of continuous-time learning equations with the actual Q learning algorithm at a small learning rate. This policy-bias problem [1] occurs because, at any time step, an agent can execute and get feedback for only one action. If reward information were available for all actions and states at every time step, these discrepancies would disappear. To address this policy-bias problem, Frequency-Adjusted Q Learning (FAQL) [20] was proposed as a simple and effective modification to the Q learning update. The update value of an action is normalized by the probability of choosing that action. Practical concerns of FAQL were improved by performing updates repeatedly, proportional to the inverse of the frequency of choosing that action [1]. Thus, both variants [1, 20] extend the basic Q learning algorithm to improve the

link between the replicator dynamics and Q learning. They suggest that replicator dynamics learning matches a frequency-adjusted Q learning algorithm.

Alternatively, a deterministic limit of reinforcement learning can be taken in discrete time, resulting in a set of difference equations for the action probabilities [5, 12–14]. This approach begins with a simple batch reinforcement learning algorithm. Experience is collected inside the batch under keeping the current policy fixed. Then, the policy is updated, using the average batch reward. Deterministic dynamics emerge by sending the size of the batch to infinity. Note that a batch learning approach automatically addresses the policy-bias problem by repeated interactions with the same policy.

The majority of papers on learning dynamics consider only single-state repeated games. The first article which considers multi-state environments introduced the piecewise replicator dynamics [33]. They combine replicator dynamics with switching dynamics between cell partitions of the state space of the dynamical system. State-coupled replicator dynamics [16] improve on this idea by the direct coupling between states, yet, lack an exploration component. This limitation is overcome by the reverse engineering state-coupled replicator dynamics [15] which incorporate insights about Q learning with Boltzmann exploration. Yet, all of these dynamics consider an average reward setting, whereas in Q learning a discounted reward is commonly used.

Only recently, an analytical method to derive the deterministic, discrete-time limit of temporal difference reinforcement learning with discounted rewards was proposed [3]. We use them as a starting point for this work. They extend on the idea of batch learning with an infinite batch size, yet, an explicit comparison between the predictions of these learning dynamics with actual algorithmic implementations were still pending.

2 BACKGROUND

2.1 Stochastic games

Stochastic games are a formal model for multi-agent environment systems. They generalize both repeated normal form games and Markov decision processes (MDPs). MDPs are generalized by introducing multiple agents. Repeated games are generalized by introducing an environment with multiple states and transition probabilities between those states. All agents choose their actions simultaneously. The transition probabilities depend on the joint action and the current environmental state. So do the rewards, the agents receive. Formally, the game $G = \langle N, \mathcal{S}, \underline{\mathcal{A}}, T, \underline{R}, \underline{X} \rangle$ is a stochastic game with $N \in \mathbb{N}$ agents. The environment consists of $Z \in \mathbb{N}$ states $\mathcal{S} = (S_1, \ldots, S_Z)$. In each state s, each agent i has $M \in \mathbb{N}$ available actions $\mathcal{A}^i = (A_1^i, \ldots, A_M^i)$ to choose from. $\underline{\mathcal{A}} = \prod_i \mathcal{A}^i$ is the joint-action set and agents choose their actions simultaneously.

The transition function $T : \mathcal{S} \times \underline{\mathcal{A}} \times \mathcal{S} \to [0, 1]$ determines the probabilistic state change. $T(s, \underline{a}, s')$ is the transition probability from current state s to next state s' under joint action $\underline{a} = (a^1, \ldots, a^N) \in \underline{\mathcal{A}}$.

The reward function $\underline{R} : \mathcal{S} \times \underline{\mathcal{A}} \times \mathcal{S} \to \mathbb{R}^N$ maps the triple of current state s, joint action $\underline{a} = (a^1, \ldots, a^N)$ and next state s' to an immediate reward value for each agent. $R^i(s, \underline{a}, s')$ is the reward agent i receives.

Agents choose their actions probabilistically according to their policy $X^i : \mathcal{S} \times \mathcal{A}^i \to [0, 1]$. $X^i(s, a)$ is the probability that agent i chooses action a given the environment is in state s. $\underline{X}(s, \underline{a}) = \prod_i X^i(s, a^i)$ is the joint policy.

We chose an identical number of actions for all states and all agents out of notational convenience. With $\underline{a}^{-i} = (a^1, \ldots, a^{i-1}, a^{i+1}, \ldots, a^N)$ we denote the joint action except agent i's. Throughout this paper we restrict ourselves to ergodic environments without absorbing states (c.f. Ref. [15])

2.2 Temporal difference Q learning

Temporal difference Q learning is one of the most widely used reinforcement learning algorithms [28]. In essence, at time step t agent i estimates how valuable action a is when the environment is in state s by its state-action values $Q_t^i(s, a)$. Subsequently, agent i chooses one action from its action set with a probability derived from these state-action values and continues with time steps t + 1.

State-action values get iteratively updated according to:

$$Q_{t+1}^i(s, a) = Q_t^i(s, a) + \alpha \cdot TD_t^i(s, a), \tag{1}$$

with the temporal-difference error

$$TD_t^i(s, a) := (1 - \gamma) r_t^i + \gamma \max_b Q_t^i(s', b) - Q_t^i(s, a). \tag{2}$$

This update can be derived from the assumption that agent i aims to maximizes its expected discounted future reward $G_t^i = (1 - \gamma) \sum_k \gamma^k r_{t+k}^i$, where the *discount factor* parameter $\gamma \in [0, 1)$ regulates how much the agent cares for future rewards. The prefactor $(1 - \gamma)$ normalizes the state-action values to be in the same scale as the rewards [3]. The *learning rate* parameter α regulates how much new information is used for a state-action-value update. We assume identical parameters across agents throughout this paper and therefore do not equip parameters with agent indices. The variable r_t^i refers to the immediate reward at time step t, s' denotes the next state after executing action a at state s.

Based on these state-action values, agents choose their actions according to the Boltzmann policy

$$X_t^i(s, a) = \frac{e^{\beta Q_t^i(s,a)}}{\sum_b e^{\beta Q_t^i(s,b)}}, \tag{3}$$

where the *intensity of choice* parameter β controls the exploration-exploitation trade-off. In the baseline scenario, an agent can only execute one action at a time and thus, will only receive reward feedback for the action that was chosen.

Combining Eqs. 1, 2, and 3, one can derive the update equation for the joint policy,

$$X_{t+1}^i(s, a) = \frac{X_t^i(s, a) e^{\alpha \beta TD_t^i(s,a)}}{\sum_b X_t^i(s, b) e^{\alpha \beta TD_t^i(s,b)}}. \tag{4}$$

2.3 Testbeds

2.3.1 Temporal risk-reward dilemma. The first environment we use as a testbed is a one-agent stochastic game, i.e., a Markov decision process. It is a simple, two-state, two-action environment and models the intertemporal dilemma between a risky choice with possibly high immediate reward and a safe choice with a guaranteed, but low immediate reward [4]. The action set reads

$\mathcal{A} = \{safe, risky\}$, the environment can either be in a prosperous or a degraded state, $\mathcal{S} = \{prosp., deg.\}$. The transition function reads

$$T(prosp., a, deg.) = \begin{cases} 0 & a = safe \\ 0.2 & a = risky \end{cases},$$

$$T(deg., a, prosp.) = \begin{cases} 0.1 & a = safe \\ 0 & a = risky \end{cases},$$

and $T(prosp., a, prosp.) = 1 - T(prosp., a, deg.)$ and $T(deg., a, deg.) = 1 - T(deg., a, prosp.)$. The reward function is given by

$$R(prosp., a, s') = \begin{cases} 1 & a = risky \text{ and } s' = prosp. \\ 0.5 & a = safe \\ 0 & \text{elsewhere} \end{cases}.$$

By applying the safe action in the prosperous state, the agent is guaranteed to remain in the prosperous state and obtains a reward of 0.5. If it applies the risky action and remains in the prosperous state, it obtains a reward of 1. If, however, the environment collapses under the risky action, the agent obtains 0 reward until the environment is recovered again. Recovery is possible only under the safe action after waiting for some iterations in the degraded state. In the prosperous state, it depends on the agent's discount factor whether the risky or the safe action is more valuable to the agent.

2.3.2 Two-state matching pennies. The other environment we use as a testbed in this article is the two-agent ($N = 2$), two-state ($\mathcal{S} = \{1, 2\}$), two-action ($\mathcal{A} = \{1, 2\}$) matching pennies game [15], which presents a challenge of coordination. Its reward function is given by the two payoff bi-matrices, independent of the next state s',

$$\left(R^1(1, \underline{a}, s'), R^2(1, \underline{a}, s')\right) = \begin{pmatrix} 1,0 & 0,1 \\ 0,1 & 1,0 \end{pmatrix},$$

$$\left(R^1(2, \underline{a}, s'), R^2(2, \underline{a}, s')\right) = \begin{pmatrix} 0,1 & 1,0 \\ 1,0 & 0,1 \end{pmatrix}.$$

Its transition function reads

$$T(1, \underline{a}, 2) = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}, \quad T(2, \underline{a}, 1) = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix},$$

and $T(1, \underline{a}, 1) = 1 - T(1, \underline{a}, 2)$ and $T(2, \underline{a}, 2) = 1 - T(2, \underline{a}, 1)$. This game models the situation of penalty kicks between a kicker and a keeper. Players change roles under state transitions, which depend only on agent 1's actions. With symmetrical rewards but unsymmetrical state transitions, this game presents the challenge of coordinating both agents on playing a mixed strategy with equiprobable actions.

3 SAMPLE-BATCH LEARNING

Research activity on batch reinforcement learning has grown substantially in recent years, primarily due to the central merits of the batch approach: i) its efficient use of collected data and ii) the stability of the learning process when used with function approximation [21]. In this work, we exclusively use the tabular case (without function approximation) and thus, we focus on the issue of data efficiency [34].

As a first step, we propose a highly data-inefficient sample-batch-learning algorithm (SBATCH, Algorithm 1). Our focus lies on the structure of the Q-update. In the next section, we show that this specific structure of SBATCH approaches the deterministic limit of temporal difference reinforcement learning [3] under large batch sizes. This section theoretically compares the novel SBATCH algorithm and the known derivation of the deterministic learning equations.

Algorithm 1: Tabular Sample-Batch Q Learning

1 **begin**
2 Initialize ${}^{act}Q(s, a)$ and ${}^{val}Q(s, a)$ arbitrarily.
3 Initialize $count(s, a)$, $reward(s, a)$ and $nextQ(s, a)$ to zero.
4 Observe current state s
5 **repeat**
6 Compute policy $X(s, a)$ using ${}^{act}Q(s, a)$ according to Eq. 3
7 **for** $k \leftarrow 1$ **to** *batch size* K **do**
8 Choose action a according to $X(s, a)$
9 Execute a and observe reward r and next state s'
/* Interaction phase */
10 Set $count(s, a) \leftarrow count(s, a) + 1$
11 Set $reward(s, a) \leftarrow reward(s, a) + r$
12 Set $nextQ(s, a) \leftarrow nextQ(s, a) + \max_b {}^{val}Q(s', b)$
13 Set ${}^{val}Q(s, a) \leftarrow {}^{val}Q(s, a) + \alpha \cdot \left[(1-\gamma)r + \gamma \sum_b X(s', b) {}^{val}Q(s', b) - {}^{val}Q(s, a)\right]$
14 Set $s \leftarrow s'$
/* Adaptation phase */
15 **foreach** $\hat{s}, \hat{a}$ **do**
16 Set $c(\hat{s}, \hat{a}) \leftarrow count(\hat{s}, \hat{a})$ where $count(\hat{s}, \hat{a}) \neq 0$
17 Set $c(\hat{s}, \hat{a}) \leftarrow 1$ where $count(\hat{s}, \hat{a}) = 0$
18 Set $TD(\hat{s}, \hat{a}) \leftarrow (1-\gamma)\frac{reward(\hat{s}, \hat{a})}{c(\hat{s}, \hat{a})} + \gamma \frac{nextQ(\hat{s}, \hat{a})}{c(\hat{s}, \hat{a})} - {}^{act}Q(\hat{s}, \hat{a})$
19 Set ${}^{act}Q(\hat{s}, \hat{a}) \leftarrow {}^{act}Q(\hat{s}, \hat{a}) + \alpha \cdot TD(\hat{s}, \hat{a})$
20 Set ${}^{val}Q(\hat{s}, \hat{a}) \leftarrow {}^{act}Q(\hat{s}, \hat{a})$
21 Set $count(\hat{s}, \hat{a})$, $reward(\hat{s}, \hat{a})$ and $nextQ(\hat{s}, \hat{a})$ to zero.
22 **until** *done*;

The learning process of SBATCH is divided into two phases, an interaction phase (Algorithm 1 ll. 7 - 14) and an adaptation phase (Algorithm 1 ll. 15 - 20). During the interaction phase, the agent keeps its policy fixed while interacting with its environment for K time steps. It collects state, action and reward information. During the adaptation phase, the agent uses the collected information for an update of its policy. Key is the use of two state-action value tables, one for acting (${}^{act}Q$), the other for improved value estimation (${}^{val}Q$). While ${}^{act}Q$ is kept constant during the interaction phase, ${}^{val}Q$ is iteratively updated.

Mathematically, we can express this sample-batch learning by formulating the temporal difference error of batch size K:

$${}^{K}TD^{i}_{t}(s, a) := \frac{1}{C(s, a)} \sum_{k=0}^{K-1} \delta_{s s_{t+k}} \delta_{a a_{t+k}} \cdot \hat{TD}^{i}_{t,k}(s, a) \qquad (5)$$

where the number of times the state-action pair (s, a) was visited is denoted by $C(s, a) := \max(1, \sum_{k=0}^{K-1} \delta_{ss_{t+k}} \delta_{aa_{t+k}})$ (ll. 16 and 17). The Kronecker deltas $\delta_{ss_{t+k}} \delta_{aa_{t+k}}$ yield zero, except if state s is the state visited in time step $t + k$ and the action a is the action chosen in time step $t + k$. Then they yield 1. The notation $TD_t^i(s, a)$ denotes a temporal difference error of batch size 1: $TD_t^i(s, a) = {}^1TD_t^i(s, a)$ and matches indeed Eq. 2, as one can easily show.

$$\hat{TD}_{t,k}^i(s, a) := (1-\gamma) r_{t+k}^i + \gamma \max_b {}^{val}Q_{t,k}^i(s'', b) - {}^{act}Q_t^i(s_t, a_t) \quad (6)$$

is the auxiliary temporal difference error with $s'' = s_{t+k+1}$ being the next state inside the batch and the auxiliary state-action value update (l. 13)

$$\begin{aligned} {}^{val}Q_{t,k+1}^i(s, a) = {}^{val}Q_{t,k}^i(s, a) + \\ \alpha \Big[(1-\gamma) r_{t+k}^i + \gamma V_{t,k}^i(s'') - {}^{val}Q_{t,k}^i(s, a) \Big], \end{aligned} \quad (7)$$

where $V_{t,k}^i(s'') := \sum_b X_t^i(s'', b) {}^{val}Q_{t,k}^i(s'', b)$ is the state-value estimate of state s_{k+1} at time step k.

Algorithmically, the information is collected during the interaction phase as follows: the agent counts the number of visited state-action pairs (l. 10), it sums up immediate rewards for each state-action pair (l. 11), as well as the value estimation for the next state (l. 12). This is the specific Q-update.

Finally, during the adaptation phase, the agent uses the sample averages of the immediate rewards and next-state-value estimates to summarize the collected information in order to update the state-action values ${}^{act}Q$, used for acting (l. 19):

$$\forall s, a \quad {}^{act}Q_{t+K}^i(s, a) = {}^{act}Q_t^i(s, a) + \alpha \cdot {}^K TD_t^i(s, a). \quad (8)$$

Since the agent interacts physically with the environment during the interaction phase, SBATCH requires many interaction time steps and is therefore highly data-inefficient. However, crucial is the structure of the Q-update. The state-action value table of the basic Q-update (Eq. 1) is separated between two state-action value tables, one for acting (${}^{act}Q$), the other for improved value estimation (${}^{val}Q$).

4 DETERMINISTIC LIMIT

In this section, we present how the prescribed sample-batch learning algorithm approaches the recently proposed deterministic limit of temporal difference reinforcement learning dynamics (DetRL) [3] under an increasing batch size K. Equivalently, this can be regarded as a separation of time scales between the processes of interaction and adaptation. We assume that (infinitely) many interactions happen before one step of policy adaptation occurs.

We begin by briefly reviewing how this deterministic limit is constructed analytically by sending $K \to \infty$ [3]. Under this assumption and because of the assumed ergodicity, one can replace the sample average in Eq. 5, i.e., the sum over sequences of states and actions with the policy average according to

$$\frac{1}{C(s, a)} \sum_{k=0}^{K-1} \delta_{ss_{t+k}} \delta_{aa_{t+k}} \to \sum_{s'} \sum_{\underline{a}^{-i}} \underline{X}^{-i}(s, \underline{a}^{-i}) T(s, \underline{a}, s'). \quad (9)$$

We replace the sample average with a policy average, where we average over the policies of the other agents and the environmental state transitions. This is a general method, applicable to various kinds of temporal difference reinforcement learning algorithms (e.g., Q, SARSA, Actor-Critic learning [3]). Throughout this article, we solely focus on multi-state Q learning. The notation $\sum_{s'} \sum_{\underline{a}^{-i}} \underline{X}^{-i}(s, \underline{a}^{-i}) T(s, \underline{a}, s')$ is an abbreviation for $\sum_{s'} \sum_{a^1} \cdots \sum_{a^{i-1}} \sum_{a^{i+1}} \cdots \sum_{a^N} X^1(s, a^1) \cdots X^{i-1}(s, a^{i-1}) X^{i+1}(s, a^{i+1}) \cdots X^N(s, a^N) T(s, \underline{a}, s')$. The time t is rescaled accordingly.

We apply this conversion rule (Eq. 9) to the three terms of the temporal difference error (Eq. 6): the immediate reward, the value estimate of the next state and the current state-action value.

The first term, the immediate reward $r_k^i = R^i(s_k, \underline{a}_k, s_{k+1})$ in the temporal difference error becomes

$$\langle R \rangle_{\underline{X}}^i(s, a) := \sum_{s'} \sum_{\underline{a}^{-i}} \underline{X}^{-i}(s, \underline{a}^{-i}) T(s, \underline{a}, s') R^i(s, \underline{a}, s').$$

The second term, $\max_b {}^{val}Q_t^i(s_{k+1}, b)$, i.e., the value estimate of the next state becomes

$$\langle {}^{max}Q \rangle_{\underline{X}}^i(s, a) := \sum_{s'} \sum_{\underline{a}^{-i}} \underline{X}^{-i}(s, \underline{a}^{-i}) T(s, \underline{a}, s') \max_b Q_{\underline{X}}^i(s', b).$$

Since we assume an infinite number of interactions, we can replace the state-action value estimates Q_t^i, which evolve in time t (Eq. 7), with the converged true state-action values $Q_{\underline{X}}^i$, which depend on the joint policy $\underline{X}$. We compute $Q_{\underline{X}}^i(s, a) = (1-\gamma)\langle R \rangle_{\underline{X}}^i(s, a) + \gamma V_{\underline{X}}^i(s)$ via the matrix inversion according to $V_{\underline{X}}^i = (1-\gamma)[\mathbb{1}_Z - \gamma \overline{T}_{\underline{X}}]^{-1} \langle \overline{R} \rangle_{\underline{X}}^i$ with the effective Markov chain transition matrix $\overline{T}_{\underline{X}}(s, s') := \sum_{\underline{a}} \underline{X}(s, \underline{a}) T(s, \underline{a}, s')$ and the policy average reward $\langle \overline{R} \rangle_{\underline{X}}^i(s) := \sum_{s'} \sum_{\underline{a}} \underline{X}(s, \underline{a}) T(s, \underline{a}, s') R^i(s, \underline{a}, s')$. This matrix inversion, and therefore the whole dynamics, are only applicable to environments with relatively small state-action spaces [3]. Note that $\langle {}^{max}Q \rangle_{\underline{X}}^i(s, a)$ is the policy averaged maximum state-action value of the next state, viewed from the current state-action pair (s, a).

The third term, the current state-action value ${}^{act}Q_t^i(s_t, a_t)$ becomes $\beta^{-1} \log X^i(s, a)$. This can be shown by inverting Eq. 3 and realizing that the dynamics induced by Eq. 4 are invariant under additive transformations which are constant in actions [3].

All together, the temporal difference error for Q learning in the deterministic limit reads

$${}^{\infty}TD_{\underline{X}}^i(s, a) = (1-\gamma)\langle R \rangle_{\underline{X}}^i(s, a) + \gamma \langle {}^{max}Q \rangle_{\underline{X}}^i(s, a) - \frac{1}{\beta} \log X^i(s, a). \quad (10)$$

To obtain the DetRL dynamics, the temporal difference error of the current policy in the infinite batch limit ${}^{\infty}TD_{\underline{X}_t}^i(s, a)$ has to be inserted into Eq. 4.

4.1 Results and discussion

We compare the DetRL dynamics (Sec. 4) with the SBATCH algorithm (Sec. 3). The influence of the three parameters α, β, and γ on the deterministic learning dynamics can be summarized as follows: the intensity of choice β and the discount factor γ control *where* the learners adapt toward in policy space, weighting current reward, expected future reward and the level of forgetting. The learning rate α controls *how fast* the learners adapt along these directions [3]. To

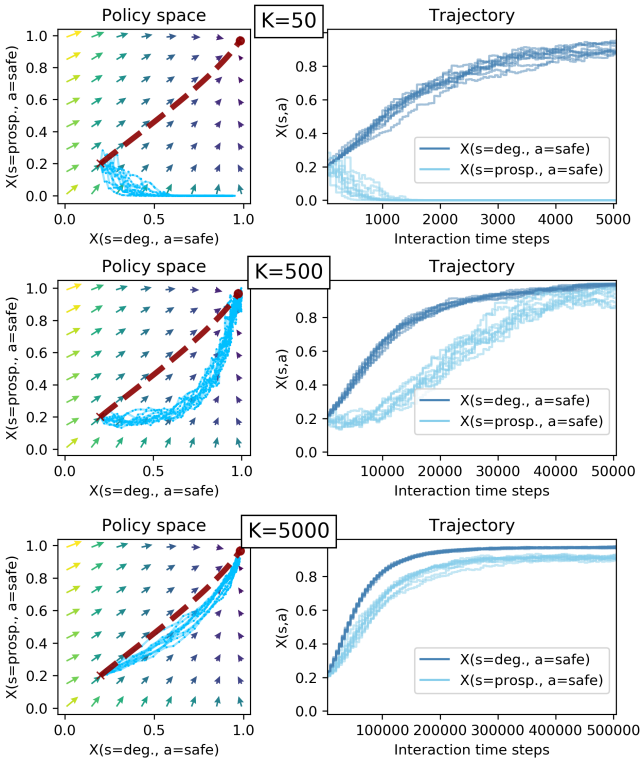

Figure 1: Comparison between DetRL dynamics (dark red dashed line) with 10 runs of SBATCH learning (light blue straight lines) in the temporal risk-reward dilemma for varying batch sizes K; $\alpha = 0.05, \beta = 150, \gamma = 0.9$. Note that the blue lines converge to the red lines as K in increased from 50 to 5000.

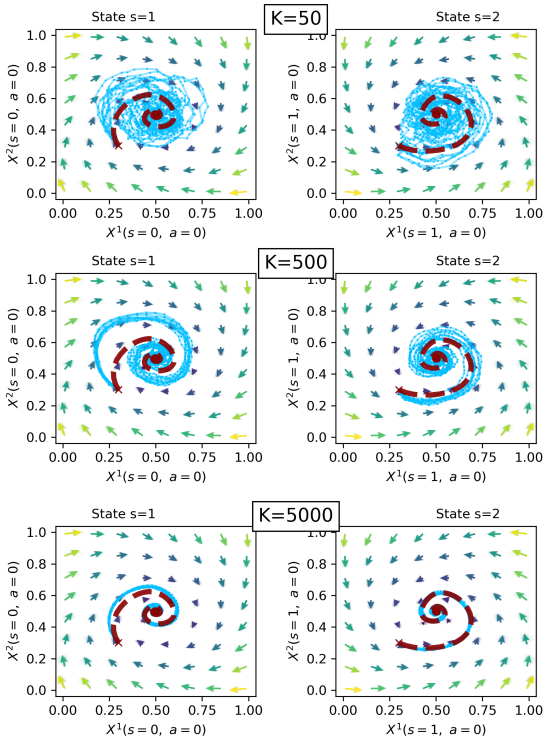

Figure 2: Comparison between DetRL dynamics (dark red dashed line) with 10 runs of SBATCH learning (light blue straight lines) in two-state matching pennies game for varying batch sizes K; $\alpha = 0.05, \beta = 25, \gamma = 0.75$. Note that the blue lines converge to the red lines as K in increased from 50 to 5000.

emphasize the analogy between DetRL and SBATCH we keep these three parameters constant and study the influence of the batch size K.

In the single-agent temporal risk-reward dilemma, it is always best to choose the safe action in the degraded state. For an agent with a discount factor of $\gamma = 0.9$, it is also optimal to choose the safe action in the prosperous state. The challenge presented in Fig. 1 is to learn to choose the safe actions in both states, starting from an initial policy, where in both states, the safe action is chosen only with 20%. The arrows in the policy space indicate the average direction the temporal difference errors (Eq. 10) drive the learner toward. All point to the optimal policy in the upper right.

DetRL dynamics (dark red dashed line) follow these temporal difference arrows and learn to play safe. SBATCH learners with batch size $K = 50$, however, do not. The agent does not manage to learn to choose the safe action in the prosperous state. For a batch size of $K = 500$, the agent learns to do so, yet, the trajectory through the policy space differs from the one of DetRL. The agent learns to play safe in the degraded state faster than in the prosperous state. For a batch size of $K = 5000$, SBATCH learning and DetRL dynamics match relatively well.

In the two-state matching pennies game, the two agents have to coordinate on a mixed policy profile. Due to the finite intensity of choice, DetRL dynamics manage to coordinate on the mixed Nash equilibrium in the centre of the policy space from a spiralling learning trajectory (Fig. 2). Here, a larger batch size makes the match between DetRL dynamics and SBATCH learning increasingly close.

In both environments, we gave the agents 100 opportunities to change their policy. Thus, the actual interactions with the environment ($100 \cdot K$) scale with the batch size K. The actual computing time in seconds scales also linearly with the batch size K (Tab. 1).

5 CORRELATED EXPERIENCE REPLAY

Our sample batch algorithm is highly inefficient concerning the interaction steps it requires with the environment. Experience replay [22] intends to speed up convergence not only by using observed state transitions (the experience) once but by replaying them repeatedly to the agent as if they were new observations. Doing so is useful because in the basic Q-Update (Eqs. 1 and 2), information is not able to back-propagate from updated states to preceding states without further interaction. Replaying experience enables this back-propagation of information, and, ideally, speeds up the learning process [21]. However, some important transitions may

Algorithm 2: Correlated Experience Replay Q Learning

1 **begin**
2 Initialize ${}^{act}Q(s,a)$ and ${}^{val}Q(s,a)$ arbitrarily.
3 Initialize $count(s,a)$, $reward(s,a)$ and $nextQ(s,a)$ to zero.
4 Initialize replay buffer $\mathcal{B}$ of size K' with initial policy.
5 Observe current state s
6 **repeat**
7 Compute policy $X(s,a)$ using ${}^{act}Q(s,a)$ according to Eq. 3
8 Choose action a according to $X(s,a)$
9 Execute a and observe reward r and next state s'
10 Store experience (s,a,r,s') in $\mathcal{B}$ and remove the oldest experience
11 Sample *batch* of size K of subsequent experiences $((s,a,r,s'),(s',a',r',s''),\ldots)$ from $\mathcal{B}$, where actions are choosen according to X.
12 Use *batch* to update $count(s,a)$, $reward(s,a)$, $nextQ(s,a)$ and ${}^{val}Q(s,a)$ according to Algorithm 1, ll. 7 - 14.
13 Update ${}^{act}Q$ according to Algorithm 1, ll. 20 - 15.
14 Set $s \leftarrow s'$
15 **until** *done*;

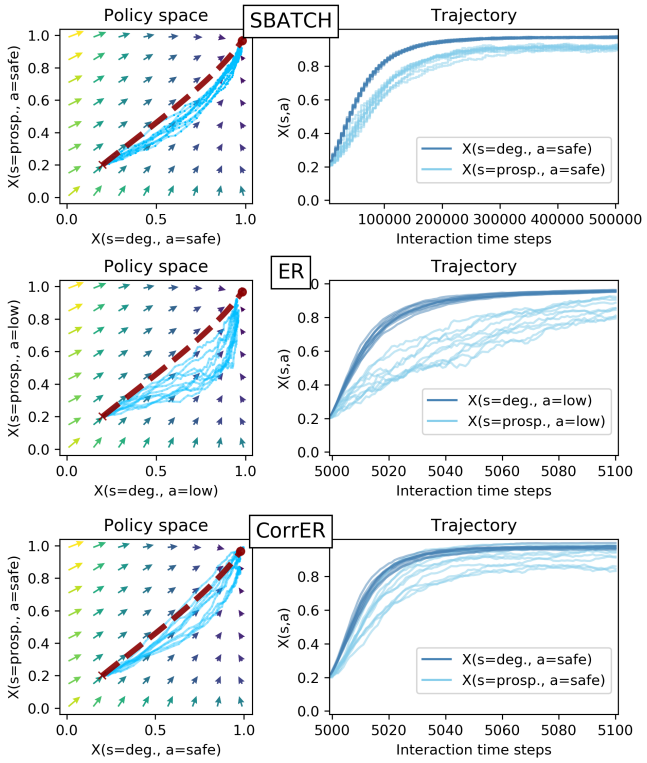

Figure 3: Comparison between SBATCH, ER, and CorrER learning (light blue straight lines) in the temporal risk-reward dilemma for $K = K' = 5000, \alpha = 0.05, \beta = 150, \gamma = 0.9$ and 10 sample runs. DetRL dynamics (dark red dashed line) are shown for reference. Note that trajectories of CorrER are closer to the ones of SBATCH and DetRL, compared to the ones of ER, with distinctly fewer interaction time steps than SBATCH.

come delayed to make efficient use of experience replay. This negative effect is partially controlled by the size of the replay buffer [34].

With Algorithm 2 we shift the batch update from actual experience into memorized experience. To retain similar learning behaviour in policy space as with the SBATCH algorithm, correlated experience replay is crucial. Therefore we term it CorrER. The imaginary batch is constructed as follows (l. 11): Starting from a random initial state s, the agent draws an action a according to its current policy. A randomly selected experience (s,a,r,s') is drawn from the agent's memory buffer, which matches the current state-action pair s,a. The agent continues with the next state s' accordingly for K steps. These K experiences are used to perform the batch update (l. 12) which is then used to update the policy (l. 13). These processes happen all between two-time steps of interaction with the environment. CorrER may be computationally not advantageous. Yet, it is constructed to be data-efficient (concerning the number of environmental interactions required for successful learning) through the imaginary batch.

5.1 Results and discussion

In this section, we compare the correlated experience replay learning (CorrER) with SBATCH learning (Sec. 3). Additionally, we compare both algorithms with an equal probability experience replay learning (ER), where samples are drawn from the memory buffer with equal probabilities. Apart from l. 11 it is identical to Algorithm 2.

For the temporal risk-reward dilemma, Fig. 3 shows that correlated experience replay (CorrER) learning matches the SBATCH baseline noticeably closer than the equal probability experience replay (ER) learning. As in Section 4 we gave each learner the possibility to adapt its policy for 100 learning steps. As a consequence, both experience replay variants take only 5100 time steps of physical interactions with the environment, from which 5000 are the initialization of the memory buffer. In contrast, the SBATCH learner requires 500.000 time steps of interaction with the environment. This is a factor of approximately 100, equaling the number of learning steps. Interestingly, all three algorithms require approximately the same computing time (Tab. 1). Taken together, the learning process of CorrER is data-efficient (in contrast to SBATCH) and similar to DetRL (in comparison to ER).

However, when applied to the two-state matching pennies environment, CorrER fails drastically to retain similarity to SBATCH learning and the DetRL dynamics concerning the learning trajectory (Fig. 4). Here, both, in terms of the learning trajectory and data-efficiency, CorrER and ER appear alike. This result is expected since CorrER correlates its memorized experience only with respect to its own policy. The policy and actions of other agents are not considered. This suggests that extending CorrER to observe and memorize not only own but the joint action, is a way to regain

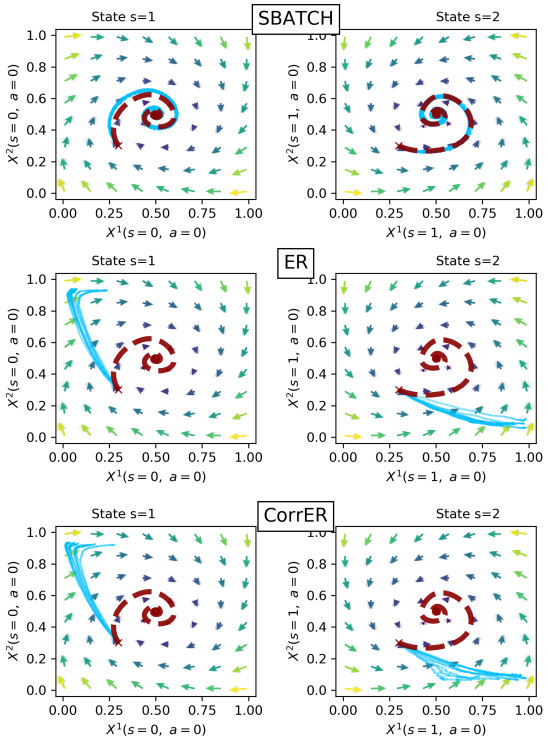

Figure 4: Comparison between SBATCH, ER, and CorrER learning (light blue straight lines) in the two-state matching pennies game for $K = K' = 5000, \alpha = 0.05, \beta = 25, \gamma = 0.75$ and 10 sample runs. DetRL dynamics (dark red dashed line) are shown for reference. Note that here, in the multi-agent environment, CorrER trajectories are not closer to SBATCH trajectories than the ones of ER.

similarity to the learning trajectory of DetRL. Additionally, in the multi-agent case, CorrER and ER require approximately 60% more computing time than SBATCH for a memory batch size of $K = 5000$ (Tab. 1). This suggests that the algorithmic complexity of CorrER scales super-linearly with the dimension of the joint-action space.

Importantly, Fig. 4 shows that the DetRL dynamics - interpreted in the infinite memory limit - represent a form of joint-action learning. Agents in multi-agent reinforcement learning settings can be categorized into two categories [6, 31]: independent learners, which learn Q-values only for their own actions and joint-action learners, where the learning takes place over the joint-action space [9]. This is interesting because the DetRL Q learning dynamics result from the infinite batch limit of independent Q learners. However, when interpreted as a form of memory replay, the DetRL dynamics behave as if experience is replayed from the joint-action space.

Additionally, one can interpret the DetRL dynamics as a form of model-based learning. Reinforcement learning algorithms can be divided into two categories: model-free and model-based. Model-free methods learn a value function directly from samples, whereas model-based methods first learn a model of the environment. Experience replay can be interpreted as a form of model-based reinforcement learning [32]. This is interesting because the DetRL Q learning dynamics result from the infinite batch limit of model-free Q learners.

6 CONCLUSION

In this article we made the following contributions:

First, we provided a novel, data-inefficient, sample-batch reinforcement learning algorithm (SBATCH) and showed theoretically and experimentally that it approaches the deterministic limit of temporal difference reinforcement learning (DetRL) under large batch sizes.

Second, we introduced a novel, data-efficient learning algorithm that uses correlated experiences replay (CorrER). We showed experimentally that the learning process of CorrER is data-efficient (in contrast to SBATCH) and similar to DetRL (in comparison to standard, uncorrelated experience replay) when used within a single-agent environment. However, when used in a multi-agent environment, CorrER trajectories were not closer to the one of DetRL than the ones of uncorrelated experience replay.

Taken together, we provide an individual agent interpretation for the dynamics of learning. The deterministic limit of reinforcement learning, which results from a time-scale separation of interaction and adaptation, is like learning under infinite joint-action memory. Although derived from independent, model-free learners, this suggests that these dynamics represent a form of model-based joint-action reinforcement learning.

Especially when the evolutionary process is understood as a form of social, cultural learning [17] we can state this equivalence as follows. Evolutionary imitation learning from others' experience in an infinite population of equals resembles individual learning from own experience under infinite memory of joint-action observations. What is the infinite population limit of evolutionary dynamics is an infinite memory limit of learning dynamics.

This result is of potential use for broadening the scope of previous research in ecology and economics, where an infinite population approximation is often used to study the convergence to equilibria [11, 19, 24–26].

Furthermore, we hope that our results contribute to a better understanding of the dynamics of collective reinforcement learning. Such an understanding is crucial to advance the practical use of the study of collective learning dynamics to overcome critical challenges of multi-agent reinforcement learning [2].

Outlook. Of course, much like an infinite population, infinite memory of the joint actions is unrealistic. While both limits can be interpreted as relative frequencies of either phenotypes in a population or state-action transition probabilities in an agent, future work needs to develop dynamical systems methods to analyze finite as well as selective memory recalls.

In this work, we focused on the DetRL dynamics in discrete time. Future work needs to develop their continuous-time equivalents and compare them with the discrete DetRL dynamics.

The presented, deterministic limit of temporal difference reinforcement learning retains all three essential parameters: the learning rate α, the intensity of choice β and the discount factor γ. This parameter equivalence is important when learning dynamics are used to aid the parameter tuning of learning algorithms in high-dimensional environments. Future work needs to address

the conditions of how well batch or experience replay algorithms converge to the DetRL dynamics.

Essentially, the structure of the reinforcement learning algorithm, which leads to the DetRL dynamics, consists of two additions to the basic Q-update, experience replay and two data structures to estimate the state-action values. This structure is similar to the one of the influential DQN algorithm [23], although significant differences remain. It is interesting to explore these connections in more detail in future work.

Acknowledgements

W.B. thanks Richard P. Mann for comments on the manuscript.

REFERENCES

[1] Sherief Abdallah and Michael Kaisers. 2013. Addressing the policy-bias of q-learning by repeating updates. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1045–1052.

[2] Wolfram Barfuss. 2020. Towards a unified treatment of the dynamics of collective learning. *Accepted for AAAI Spring Symposium: Challenges and Opportunities for Multi-Agent Reinforcement Learning* (2020).

[3] Wolfram Barfuss, Jonathan F Donges, and Jürgen Kurths. 2019. Deterministic limit of temporal difference reinforcement learning for stochastic games. *Physical Review E* 99, 4 (2019), 043305.

[4] Wolfram Barfuss, Jonathan F Donges, Steven J Lade, and Jürgen Kurths. 2018. When optimization for governing human-environment tipping elements is neither sustainable nor safe. *Nature communications* 9, 1 (2018), 2354. https://doi.org/10.1038/s41467-018-04738-z

[5] Alex J Bladon and Tobias Galla. 2011. Learning dynamics in public goods games. *Physical Review E* 84, 4 (oct 2011). https://doi.org/10.1103/physreve.84.041132

[6] Daan Bloembergen, Karl Tuyls, Daniel Hennes, and Michael Kaisers. 2015. Evolutionary Dynamics of Multi-Agent Learning: A Survey. *Journal of Artificial Intelligence Research* 53 (aug 2015), 659–697. https://doi.org/10.1613/jair.4818

[7] Tilman Börgers and Rajiv Sarin. 1997. Learning Through Reinforcement and Replicator Dynamics. *Journal of Economic Theory* 77, 1 (nov 1997), 1–14. https://doi.org/10.1006/jeth.1997.2319

[8] L. Busoniu, R. Babuska, and B. De Schutter. 2008. A Comprehensive Survey of Multiagent Reinforcement Learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38, 2 (mar 2008), 156–172. https://doi.org/10.1109/tsmcc.2007.913919

[9] Caroline Claus and Craig Boutilier. 1998. The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI* 1998, 746-752 (1998), 2.

[10] John G. Cross. 1973. A Stochastic Learning Model of Economic Behavior. *The Quarterly Journal of Economics* 87, 2 (may 1973), 239. https://doi.org/10.2307/1882186

[11] Michael Doebeli and Christoph Hauert. 2005. Models of cooperation based on the Prisoner's Dilemma and the Snowdrift game. *Ecology letters* 8, 7 (2005), 748–766.

[12] Tobias Galla. 2009. Intrinsic Noise in Game Dynamical Learning. *Physical Review Letters* 103, 19 (nov 2009). https://doi.org/10.1103/physrevlett.103.198702

[13] Tobias Galla. 2011. Cycles of cooperation and defection in imperfect learning. *Journal of Statistical Mechanics: Theory and Experiment* 2011, 08 (aug 2011), P08007. https://doi.org/10.1088/1742-5468/2011/08/p08007

[14] Tobias Galla and J. Doyne Farmer. 2013. Complex dynamics in learning complicated games. *Proceedings of the National Academy of Sciences* 110, 4 (jan 2013), 1232–1236. https://doi.org/10.1073/pnas.1109672110

[15] Daniel Hennes, Michael Kaisers, and Karl Tuyls. 2010. RESQ-learning in stochastic games. In *Adaptive and Learning Agents Workshop at AAMAS*. 8.

[16] Daniel Hennes, Karl Tuyls, and Matthias Rauterberg. 2009. State-coupled replicator dynamics. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009)*. 789–796.

[17] Joseph Henrich and Robert Boyd. 2002. On modeling cognition and culture. *Journal of Cognition and Culture* 2, 2 (2002), 87–112.

[18] Josef Hofbauer and Karl Sigmund. 1998. *Evolutionary games and population dynamics*. Cambridge university press.

[19] Josef Hofbauer and Karl Sigmund. 2003. Evolutionary game dynamics. *Bulletin of the American mathematical society* 40, 4 (2003), 479–519.

[20] Michael Kaisers and Karl Tuyls. 2010. Frequency adjusted multi-agent Q-learning. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 309–316.

[21] Sascha Lange, Thomas Gabel, and Martin Riedmiller. 2012. Batch reinforcement learning. In *Reinforcement learning*. Springer, 45–73.

[22] Long-Ji Lin. 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning* 8, 3-4 (1992), 293–321.

[23] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (feb 2015), 529–533. https://doi.org/10.1038/nature14236

[24] Martin A Nowak and Karl Sigmund. 2004. Evolutionary dynamics of biological games. *science* 303, 5659 (2004), 793–799.

[25] Jorge M Pacheco, Francisco C Santos, Max O Souza, and Brian Skyrms. 2009. Evolutionary dynamics of collective action in N-person stag hunt dilemmas. *Proceedings of the Royal Society B: Biological Sciences* 276, 1655 (2009), 315–321.

[26] Fernando P Santos, Francisco C Santos, Ana Paiva, and Jorge M Pacheco. 2015. Evolutionary dynamics of group fairness. *Journal of theoretical biology* 378 (2015), 96–102.

[27] Yuzuru Sato and James P Crutchfield. 2003. Coupled replicator equations for the dynamics of learning in multiagent systems. *Physical Review E* 67, 1 (jan 2003). https://doi.org/10.1103/physreve.67.015206

[28] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[29] Karl Tuyls and Ann Nowé. 2005. Evolutionary game theory and multi-agent reinforcement learning. *The Knowledge Engineering Review* 20, 1 (2005), 63–90.

[30] Karl Tuyls, Katja Verbeeck, and Tom Lenaerts. 2003. 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*. ACM, 693–700.

[31] Karl Tuyls and Gerhard Weiss. 2012. Multiagent learning: Basics, challenges, and prospects. *Ai Magazine* 33, 3 (2012), 41–41.

[32] Harm Vanseijen and Richard Sutton. 2015. A deeper look at planning as learning from replay. In *International conference on machine learning*. 2314–2322.

[33] Peter Vrancx, Karl Tuyls, and Ronald Westra. 2008. Switching dynamics of multi-agent learning. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent systems (AAMAS 2008)*. 307–313.

[34] Shangtong Zhang and Richard Sutton. 2018. A Deeper Look at Experience Replay. *arXiv preprint arXiv:1712.01275* (2018).

A APPENDIX

Table 1: Average computing times in seconds (computed on a standard personal computer).

Temporal risk reward dilemma			
Batch size K	SBATCH	CorrER	ER
50	0.66	0.60	0.81
500	5.97	4.43	4.22
5000	58.6	60.7	65.6

Two state matching pennies			
Batch size K	SBATCH	CorrER	ER
50	1.16	1.02	1.28
500	8.59	7.85	8.10
5000	85.3	135	147