Abstract

This paper addresses the problem of learning a Nash equilibrium in $\gamma$-discounted multiplayer general-sum Markov Games (MG). A key component of this model is the possibility for the players to either collaborate or team apart to increase their rewards. Building an artificial player for general-sum MGs implies to learn more complex strategies which are impossible to obtain by using techniques developed for two-player zero-sum MGs. In this paper, we introduce a new definition of $\epsilon$-Nash equilibrium in MGs which grasps the strategy’s quality for multiplayer games. We prove that minimizing the norm of two Bellman-like residuals implies the convergence to such an $\epsilon$-Nash equilibrium. Then, we show that minimizing an empirical estimate of the $L_p$ norm of these Bellman-like residuals allows learning for general-sum games within the batch setting. Finally, we introduce a neural network architecture named NashNetwork that successfully learns a Nash equilibrium in a generic multiplayer general-sum turn-based MG.

1 Introduction

A Markov Game (MG) is a model for Multi-Agent Reinforcement Learning (MARL) (Littman 1994). Chess, checkers, robotics or human-machine dialogue are a few examples of the myriad of applications that can be modeled by an MG. At each state of an MG, all agents have to pick an action simultaneously. As a result of this mutual action, the game moves to another state and each agent receives a reward. Here, we focus on general-sum MGs meaning the reward of every agent has no special structure as opposed to zero-sum two-player MGs (Shapley 1953; Perolat et al. 2015). It enables to model settings where agents may have either to cooperate or to confront in different states. For instance, usual models of reinforcement learning (Markov Decision Process (MDP), Partially Observable MDP (POMDP)) assume that agents do not adapt their behavior to others while following their own objective. One challenge is thus to develop algorithms to learn in collaborative or competitive environments. Another challenge is to compute a joint strategy for multiple agents. In this paper, agents are not allowed to settle a common strategy. They have to choose their strategy secretly and independently to maximize their cumulative reward. In game theory, this concept is traditionally addressed by searching for the Nash equilibrium (Filar and Vrieze 2012). Computing a Nash equilibrium in MGs requires a perfect knowledge of the dynamics and the reward of the game. In most practicals situations the dynamics and the reward are unknown. To overcome this difficulty two approaches are possible: either learn the game by interacting with other agents, either extract information from the game through a finite record set of interactions between the agents. The former is known as the online scenario while the later is called the batch scenario. This paper focuses on the batch setting. This setting is often preferred by practitioners as they cannot afford to deploy a totally online learning algorithm. On the other side, it can limit the access to the dynamics because the records are taken from previously deployed systems.

A wide literature does address the batch scenario for MDPs which actually can be seen as a single agent MG. The wider family of algorithms applied to MDPs builds upon approximate versions of
value iteration (Riedmiller, 2005; Munos and Szepesvari, 2008) or approximate versions of policy iteration (Lagoudakis and Parr, 2003). Another family relies on the direct minimization of the optimal Bellman residual (Baird et al., 1995; Piot et al., 2014b). These techniques can handle value function approximation and have proven their efficacy in large MDPs especially when combined with neural networks (Riedmiller, 2005). The batch scenario is also well studied for the particular case of zero-sum two-player MGs. It can be solved with analogue techniques as the ones for MDPs (Lagoudakis and Parr, 2002; Perolat et al., 2015; 2016). Those algorithms cannot be applied when there are more than two players or when the game is cooperative. Zinkevich et al. demonstrate that no algorithms based on the value function or on the state-action value function can be applied to find a stationary-Nash equilibrium in general-sum MGs. All the previously referenced algorithms fall under this category. Recent approaches to find a Nash equilibrium consider an optimization problem on the strategy of each player (Prasad et al., 2015) in addition to the value functions. Finding a Nash equilibrium is well studied when the model of the MG is known (Prasad et al., 2015) or estimated (Akchurina, 2010) or in the online scenario (Littman, 2001; Prasad et al., 2015). To our knowledge, finding an approximate equilibrium with batch data has not been addressed. Moreover, none of the herein-before algorithms handle a value function approximation or approximations of the strategy. Such architectures recorded recently tremendous successes for one player video games (Mnih et al., 2013) but are limited to MDPs.

As a first contribution, we consider a novel approach to learning an \( \epsilon \)-Nash equilibrium from data. It relies on the minimization of two Bellman-like residuals and on the definition of the notion of \( \epsilon \)-Nash equilibrium. Then, as a second contribution, we develop a new neural network architecture, named NashNetwork, based on these Bellman residuals to approximate the strategy and the state-action value-function of each player. We build two networks per player: one approximates the \( Q \)-values and the second approximates the strategy of the agent. These contributions are structured as follows in the remaining: first, we recall the definition of an MG, of a Nash equilibrium and we define a weaker notion of an \( \epsilon \)-Nash equilibrium. Then, we show that controlling the sum of several Bellman residuals allows us to learn an \( \epsilon \)-Nash equilibrium. Later, we explain that controlling the empirical norm of those Bellman residuals allows addressing the batch scenario. We then provide a description of the NashNetwork. Finally, we empirically evaluate our method on randomly generated MGs.

### 2 Background

An \( N \)-player MG is described by a finite state space \( S \), a finite action set \( ((A^i(s))_{s \in S})_{i \in \{1,\ldots, N\}} \) depending on the player and on the state, a transition function \( p(s'|s,a^i \in A^i(s),\ldots, a^N \in A^N(s)) \) describing the stochastic behavior of the system when all players play the joint action \( \mathbf{a} = (a^1,\ldots, a^N) \) in state \( s \) and a reward function \( r^i(s,a^1,\ldots, a^N) \) for each player. The constant \( \gamma \in [0,1) \) is the discount factor. For the sake of simplicity we will note \( \mathbf{a} = (a^1,\ldots, a^N) = (a^i, a^{-i}) \), \( p(s'|s,a^1,\ldots, a^N) = p(s'|s,a^i,a^{-i}) \) and \( r^i(s,a^1,\ldots, a^N) = r^i(s,a) = r^i(s,a^i,a^{-i}) \). All actions indexed by \(-i\) are joint actions of all players except player \( i \) (i.e. \( a^{-i} = (a^1,\ldots, a^{i-1}, a^{i+1},\ldots, a^N) \)). In a MARL problem, the goal is typically to find a strategy for each player. A strategy \( \pi^i \) maps to each state a distribution over the action space \( A^i(s) \). The strategy \( \pi^i(.|s) \) is such a distribution. Alike previously, we will write \( \pi = (\pi^1,\ldots, \pi^N) = (\pi^i, \pi^{-i}) \) the product distribution of those strategies (i.e. \( \pi^{-i} = (\pi^1,\ldots, \pi^{i-1}, \pi^{i+1},\ldots, \pi^N) \)). The stochastic kernel defining the dynamics of the game when all players follow their strategy \( \pi \) is \( \mathcal{P}_\pi(s'|s) = E_{\mathbf{a} \sim \pi}[p(s'|s, \mathbf{a})] \) and the one when all players but player \( i \) follow their strategy \( \pi^{-i} \) is \( \mathcal{P}_{\pi^{-i}}(s'|s, a^i) = E_{\mathbf{a}^{-i} \sim \pi^{-i}}[p(s'|s, \mathbf{a})] \). In the same manner, we can define the reward averaged over all strategies \( r^i(s) = E_{\mathbf{a} \sim \pi}[r^i(s, \mathbf{a})] \) and the one averaged over all players but player \( i \), \( r^i_{-i}(s,a^i) = E_{\mathbf{a}^{-i} \sim \pi^{-i}}[r^i(s, \mathbf{a})] \).

In an MG, when each player plays his part of the joint strategy \( \pi \), it is usual to define the value for player \( i \) as the \( \gamma \)-discounted sum of its rewards (Prasad et al., 2015; Filar and Vrieze, 2012):

\[
v^i_\pi(s) = E \left( \sum_{t=0}^{+\infty} \gamma^t r^i_\pi(s_t) | s_0 = s, s_{t+1} \sim \mathcal{P}_\pi(.|s_t) \right) \quad \text{and} \quad v^i_\pi = (I - \gamma \mathcal{P}_\pi)^{-1} r^i_\pi.
\]

The joint value of each player is the following: \( v_\pi = (v^1_\pi,\ldots, v^N_\pi) = (v^i_\pi, v^{-i}_\pi) \). In the MG theory, one can define three Bellman operators on the value functions with respect to player \( i \). The first one is \( T^i_\pi v^i_\pi(s) = r^i(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) v^i_\pi(s') \) and represents the expected value player \( i \) will get in state \( s \) if all players play the joint action \( \mathbf{a} \) and if the value for player \( i \) after one transition is...
\[v^i\]. If we consider that instead of playing action \(a\) every player plays according to a joint strategy \(\pi\) the Bellman operator to consider is \(T^i_{\pi}v^i = E_{a \sim \pi}[T^i_{a}v^i]\). This Bellman operator can also be interpreted in term of normal form game. Indeed, for all state \(s\), \(T^i_{\pi}v^i(s)\) is the reward player \(i\) would receive if all players played the strategy \(\pi(.)|s\) in the normal form game \((M^1_s, \ldots, M^N_s)\) where \(M^j_s(a) = r^j(s, a) + \gamma \sum_{s' \in S} p(s'|s, a)v^j(s') = T^j_{a}v^j(s)\). If we consider that player \(i\) plays his best response against the strategy of the opponents’ \(\pi_{\neq i}\), we would have the following operator \(T^i_{\pi_{\neq i}}v^i = \max_{a \sim \pi_{\neq i}}[T^i_{a}v^i]\). This third Bellman operator can be interpreted, for all \(s\) in \(S\), as the value of a best response player \(i\) would get in the normal form game \((M^1_s, \ldots, M^N_s)\) if all other players played \(\pi_{\neq i}(.|s)\). The value of the joint strategy \(v^\pi\) is the fixed point of the operator \(T^i\). The fixed point of operator \(T^i_{\pi_{\neq i}}\) is the value of a best response of player \(i\) to the opponents’ strategy \(\pi_{\neq i}\) meaning this fixed point is equal to \(\max v^i_{\pi_{\neq i}}, \pi_{\neq i}\).

### 3. Nash, \(\epsilon\)-Nash and Weak \(\epsilon\)-Nash Equilibrium

A Nash equilibrium is a solution concept well defined in game theory. It states that one player cannot improve his own value by switching his strategy if the other player do not vary their own one (Filar and Vrieze, 2012).

**Definition 1.** In an MG, a strategy \(\pi\) is a Nash equilibrium if: \(\forall i \in \{1, \ldots, N\}, v^i_{\pi_{\neq i}, \pi_{\neq i} = \max_{\pi_{\neq i}} v^i_{\pi_{\neq i}, \pi_{\neq i}}\) .

One should notice that if \(\gamma = 0\) and if \(|S| = 1\), definition 3 is the definition of a Nash equilibrium in a normal-form game. An equivalent definition of a Nash equilibrium is (Akchurin, 2010):

**Definition 2.** In an MG, a strategy \(\pi\) is a Nash equilibrium if \(\exists v\) such that \(\forall s \in S\) the joint strategy \(\pi(.|s) = (\pi^1(.|s), \ldots, \pi^N(.|s))\) is a Nash equilibrium of value \(v(s) = (v^1(s), \ldots, v^N(s))\) of the normal form game \((M^1_s, \ldots, M^N_s)\) defined by \(M^j_s(a) = r^j(s, a) + \gamma \sum_{s' \in S} p(s'|s, a)v^j(s') = T^j_{a}v^j(s)\).

This definition can be rewritten with Bellman operators as follows:

**Definition 3.** In an MG, a strategy \(\pi\) is a Nash equilibrium if \(\exists v\) such as \(\forall i \in \{1, \ldots, N\}, T^i_{\pi_{\neq i}}v^i = v^i\) and \(T^{\pi_{\neq i}}v^i = v^i\).

**Proof.** The proof of this equivalence is left in appendix A.

One can notice that, in the case of a single player MG (an MDP), a Nash equilibrium is simply the optimal strategy. An \(\epsilon\)-Nash equilibrium is a relaxed solution concept in game theory. When all player play an \(\epsilon\)-Nash equilibrium the value they will receive is \(\epsilon\) sub-optimal compared to a best response. Formally (Filar and Vrieze, 2012):

**Definition 4.** In a MG, an strategy \(\pi\) is an \(\epsilon\)-Nash equilibrium if:
\[\forall i \in \{1, \ldots, N\}, v^i_{\pi} + \epsilon \geq \max_{\pi_{\neq i}} v^i_{\pi_{\neq i}, \pi_{\neq i}}, \text{ or also } \forall i \in \{1, \ldots, N\}, \max_{\pi_{\neq i}} v^i_{\pi_{\neq i}, \pi_{\neq i}} - v^i_{\pi} \leq \epsilon,\]

which is equivalent to: \(\|\max_{\pi_{\neq i}} v^i_{\pi_{\neq i}, \pi_{\neq i}} - v^i_{\pi}\|_{L_\infty} \leq \epsilon\).

Interestingly, when considering an MDP, the definition of an \(\epsilon\)-Nash equilibrium reduces to controlling the \(L_\infty\)-norm between the value of the players’ strategy and the optimal value. However, it is known that approximate dynamic programming algorithms do not control a \(L_\infty\)-norm but rather an \(L_\infty\)-norm (Scherrer et al., 2012). Thus, we define a natural relaxation of the previous definition of the \(\epsilon\)-Nash equilibrium in \(L_\infty\)-norm which is consistent with the existing work on MDPs.

**Definition 5.** In a MG, \(\pi\) is a weak \(\epsilon\)-Nash equilibrium if: \(\|\max_{\pi_{\neq i}} v^i_{\pi_{\neq i}, \pi_{\neq i}} - v^i_{\pi}\|_{L_\infty} \leq \epsilon\).

One should notice that an \(\epsilon\)-Nash equilibrium is a weak \(\epsilon\)-Nash equilibrium. It is not true the other way around and thus, those \(\epsilon\) do not need to be equal. The notion of weak \(\epsilon\)-Nash equilibrium
defines a performance criterion to evaluate a strategy while learning a Nash equilibrium. Often, algorithms seeking to learn a Nash equilibrium in the literature do not provide any convincing ways to evaluate the final strategy. This definition reduces, in the case of an MDP, to controlling the difference in $L_p$-norm between the optimal value and the value of the learned strategy. In the following, we will consider that learning a Nash equilibrium will result in minimizing the loss $\left\| \max_{\pi} v^i_{\pi_0, \pi - i} - v^i_{\pi} \right\|_{\mu, p}$, meaning we want to minimize, for each player, the difference between the value of his strategy and a best response considering the strategy of the others is fixed. However, even for MDPs, a direct minimization of that norm is not possible in the batch setting. In the following, we consider the minimization of a surrogate loss to learn a Nash equilibrium with batch data.

4 Bellman Residual Minimization in Markov Games

From Definition 3, we know that a strategy $\pi$ is a Nash equilibrium if it exists $v$ such that, for any player $i$, $v^i$ is the value of the joint strategy for player $i$ (i.e. $T^i_{\pi} v^i = v^i$) and $v^i$ is the value of the best response player $i$ can achieve regarding the opponent’s strategy $\pi^{-i}$ (i.e. $T^i_{\pi^{-i}} v^i = v^i$). One can wonder if it exists $v$ such that $T^i_{\pi} v^i = v^i$ and $T^i_{\pi^{-i}} v^i = v^i$. In this section, we prove that, if we are able to control over $(v, \pi)$ a sum of the $L_p$-norm of the associated Bellman residuals $(\|T^i_{\pi^{-i}} v^i - v^i\|_{\mu, p} + \|T^i_{\pi} v^i - v^i\|_{\mu, p})$, then we are able to control $\left\| \max_{\pi} v^i_{\pi_0, \pi - i} - v^i_{\pi} \right\|_{\mu, p}$.

\[ \text{Theorem 1.} \quad \forall \mu, \nu, \rho, p \text{ positive reals such that } \frac{1}{p} + \frac{1}{p'} = 1:\]
\[ \left\| \max_{\pi} v^i_{\pi_0, \pi - i} - v^i_{\pi} \right\|_{\mu, p} \leq 2^\frac{1}{p} C_{\infty}(\mu, \nu) \left[ \sum_{i=1}^{N} \rho(i) \left( \|T^i_{\pi^{-i}} v^i - v^i\|_{\mu, p} + \|T^i_{\pi} v^i - v^i\|_{\mu, p} \right) \right]^{\frac{1}{p}}, \]

With the following concentrability coefficient $C_{\infty}(\mu, \nu, \pi^{-i}) = \left\| \frac{\partial^{(1-\gamma)T^i_{\pi^{-i}}} - \partial^{2}}{(1-\gamma)T^i_{\pi^{-i}} \pi_{\pi^{-i}} - \pi_{\pi^{-i}}} \right\|_{\nu, \infty}$ and

\[ C_{\infty}(\mu, \nu) = \left( \sup_{\pi_0, \pi - i} C_{\infty}(\mu, \nu, \pi_0, \pi - i) \right). \]

\[ \text{Proof:} \quad \text{The proof is left in appendix B.} \]

We will note $f_{\nu, p, p}(\pi, v) = \sum_{i=1}^{N} \rho(i) \left( \|T^i_{\pi^{-i}} v^i - v^i\|_{\mu, p} + \|T^i_{\pi} v^i - v^i\|_{\mu, p} \right)$. The function $f_{\nu, p, p}(\pi, v)$ is a weighted sum over players of two kinds of Bellman residuals.

Finding a Nash equilibrium is then reduced to a non-convex optimization problem. If we can find a $(\pi, v)$ such that $f_{\nu, p, p}(\pi, v) = 0$, then the joint strategy $\pi$ is a Nash equilibrium. But this procedure implies a search over the joint value function space and the joint strategy space. This might be intractable in the case of a large MG due to representation problems. This is usually addressed by making use of approximate $Q$-functions and strategies.

If one wants to find a joint strategy $\pi$ within an approximate strategy space $\Pi$ then Theorem 4 implies that $\pi$, which is such that $\pi, v \in \text{argmin}_{\pi \in \Pi, v \in F} f_{\nu, p, p}(\pi, v)$ (where $F$ is an approximate joint value space), is at least a weak $\epsilon$-Nash equilibrium (with $\epsilon = 2^\frac{1}{p} C_{\infty}(\mu, \nu) f_{\nu, p, p}(\pi, v)$). This is, to our knowledge, the first approach to solve MGs within an approximate strategy space and an approximate value function space. Theorem 5 also emphasizes the necessity of a weakened notion of an $\epsilon$-Nash equilibrium. In general, it is much easier to control a $L_p$-norm than a $L_\infty$-norm with samples.

5 The Batch Scenario

In the batch scenario, it is common to work on state-action value functions (also named $Q$-functions). One can think of the $Q$-function for a fixed strategy as the expected $\gamma$-discounted sum of rewards considering the first joint action is $\alpha$ and then every player follows the joint strategy $\pi$. Formally,
The $Q$-function is defined as $Q_{\pi}^i(s, a) = T^{i}_{\pi}v^i_{\pi}$. Moreover, one can define two analogue Bellman operators to the ones defined for the value function $B^x_{\pi}Q = r^i(s, a) + \sum_{s' \in S} p(s'|s, a)E_{b \sim p}[Q(s', b)]$ and $B^x_{\pi}Q = r^i(s, a) + \sum_{s' \in S} p(s'|s, a) \max_{b} \left[ E_{b \sim \pi} - |Q(s', b', b^{-i})] \right]$. A similar expression as the one in Theorem 6 will hold. Using the $Q$-function, we will have to minimize the following function depending on strategies and values:

$$f(Q, \pi) = \sum_{i=1}^{N} \rho(i) \left( \|B^{x^i}_{\pi}Q - Q^{i}\|_{\nu, p}^p + \|B^{x}_{\pi}Q - Q^{i}\|_{\nu, p}^p \right)$$

The batch scenario consists in having a set of $k$ samples $(s_j, (a_j^1, ..., a_j^N), (r_j^1, ..., r_j^N), s_j')_{j \in \{1, ..., k\}}$ where $r_j^i = r^i(s_j, a_j^i, ..., a_j^N)$ and where the next state is sampled according to $p(\cdot|s_j, a_j^1, ..., a_j^N)$ meaning we need to observe the actions and rewards of every player. This will result in minimizing an empirical-norm. Related works give an analysis of the minimization of the Bellman residual in MDPs (Piot et al. [2014b]). We will minimize the following empirical estimator of the criterion defined in Equation 5:

$$\hat{f}_k(Q, \pi) = \sum_{j=1}^{k} \sum_{i=1}^{N} \rho(i) \left( \|B^{x^i}_{\pi}Q(s_j, a_j) - Q^{i}(s_j, a_j)\|_{\nu, p}^p + \|B^{x}_{\pi}Q(s_j, a_j) - Q^{i}(s_j, a_j)\|_{\nu, p}^p \right)$$

In the following, we will discuss how to estimate $\|B^{x^i}_{\pi}Q(s_j, a_j) - Q^{i}(s_j, a_j)\|_{\nu, p}^p$ and $\|B^{x}_{\pi}Q(s_j, a_j) - Q^{i}(s_j, a_j)\|_{\nu, p}^p$ in different cases.

**Deterministic Dynamics:** With a deterministic dynamics, the estimation is straightforward. We estimate $B^{x}_{\pi}Q(s_j, a_j)$ with $r_j^i + \gamma E_{b \sim \pi}[Q(s_j, b)]$ and $B^{x^i}_{\pi}Q(s_j, a_j)$ with $r_j^i + \gamma \max_{b} \left[ E_{b \sim \pi} - |Q(s_j, b', b^{-i})] \right]$. 

**Stochastic Dynamics:** In case of a stochastic dynamic, the previous estimator is known to be biased. Even if this bias might be uncontrolled, it can be seen as an uncontrolled regularization of the $Q$-function (Piot et al. [2014a]). If the dynamics of the game is known, one can use the following unbiased estimator: $B^{x}_{\pi}Q(s_j, a_j)$ is estimated with $r_j^i + \gamma \sum_{s' \in S} p(s'|s_j, a_j)E_{b \sim \pi}[Q(s', b)]$ and $B^{x^i}_{\pi}Q(s_j, a_j)$ with $r_j^i + \gamma \sum_{s' \in S} p(s'|s_j, a_j) \max_{b} \left[ E_{b \sim \pi} - |Q(s', b', b^{-i})] \right]$. 

However, in the batch scenario, the dynamics of the game is not known and the second estimator cannot be used since it requires the kernel of the MG. However, as in the MDP setting, the kernel can be embedded in a Reproducing Kernel Hilbert Space (RKHS) (Grunewalder et al. [2012] Piot et al. [2014a,b]) or using kernel estimators (Taylor and Parr [2012] Piot et al. [2014b]). The same technique could be used to estimate the kernel. Then, in the estimator of our Bellman residuals for stochastic dynamics, we would replace $p(\cdot|s_j, a_j)$ by an estimator of the dynamics $\hat{p}(\cdot|s_j, a_j)$. Another technique to alleviate this common problem is to use double-sampling of the next state.

## 6 Neural Network architecture

Minimizing the Bellman residual is a challenging problem as the objective function is not convex. It is all the more difficult as both the $Q$-representation and the strategy of every players must be learned independently. Nevertheless, neural networks have been able to address complex non-convex problems such as image classification, speech recognition (LeCun et al. [2015]). Neural networks were successfully applied to reinforcement learning to approximate the $Q$-function (Mnih et al. [2013]) with eclectic state representation.

We introduce a novel neural architecture: the NashNetwork. For every players, a two-fold network is defined: a $Q$-Network that learns a representation of the game and a $\pi$-Network that learns the strategy of the players. The $Q$-network is a multilayer perceptron which takes the state representation as input. The outputs correspond to the predicted $Q$-values of the individual action such as the network used by [Mnih et al. [2013]]. Identically, the $\pi$-network is also a multilayer perceptron which takes the state representation as input. It outputs a probability distribution over the action space by using a softmax. In our experiments (Section 7), we focus on turn-based games. It entails a specific
neural network architecture that is fully described in Figure 1. During the training phase, all the Q-Networks and the \( \pi \)-Networks of the players are used to minimize the Bellman residual. Once the training is over, only the \( \pi \)-Network is kept to retrieve the strategy of each player.

Figure 1: NashNetwork. This scheme summarizes the neural architecture that learns a Nash Equilibrium by minimizing the Bellman residual for turn-based games. Each player has two networks: a Q-Network that learns a representation of the game and a \( \pi \)-Network that learns the strategy of the player. The final loss is an empirical estimate of the sum of Bellman residuals given a batch of data of the following shape \((s, (a^1, \ldots, a^N), (p^1, \ldots, p^N), s')\). The equation (1) can be divided into key operations that are described below.

\[
\begin{align*}
    &\text{(A.0): } s' = A \pi_i(s, a) \\
    &\text{(A.1): } Q_i(s, a) = E_{b_i} \pi_i[Q_i(s', b_i)] + \gamma \max_{b} [E_{b_i} \pi_{i-1} \pi_{i-2} \cdots \pi_{i-k} \pi_{i-k-1}[Q_i(s', b_i, b_{i-1}, \ldots, b_{i-k})]] \\
    &\text{(B): } \text{Select the strategy } \pi'_i(s') = \max_a [Q_i(s', a)] \\
    &\text{(C.0): } \text{Select the next state } s' = \max \pi'_i(s') \\
    &\text{(C.1): } \text{Select the strategy } \pi'_i(s') = \max_a [Q_i(s', a)] \\
    &\text{(C.2): } \text{Select the strategy } \pi'_i(s') = \max_a [Q_i(s', a)] \\
    &\text{(D): } \text{Select the action } a = \pi'_i(s') \\
    &\text{(E): } \text{Compute the dot product between } Q_i(s', a) \\
    &\text{(F): } \text{Compute the error between the reward } r^i \\
    &\text{(G): } \text{Compute the error between the reward } r^i \\
    &\text{(H): } \text{Compute the error between the reward } r^i
\end{align*}
\]

We focus on turn-based MGs for practical reasons. Indeed, in simultaneous games the complexity of the problem grows exponentially with the number of player whereas in turn-based MGs it only grows linearly. We focus on MGs with deterministic dynamics. In a turn-based MG, only one player can choose an action in each state. It is said that the player controls the state.

Dataset: Randomly generated MGs are flexible prototyping tools as they are a generic representation of games. More importantly, it is possible to compute the underlying model of the dynamics in order to investigate whether a Nash-Equilibrium would have been reached.

7 Experiments

In this section, we report an empirical evaluation of our method on randomly generated MGs. This class of problems has been first described by Archibald et al. (1995) for MDPs and has been studied for the minimization of the optimal Bellman residual (Piot et al., 2014) and in zero-sum two-player MGs (Perolat et al., 2016). First, we extend the class of randomly generated MDP to general-sum MGs, then we describe the training setting, finally we analyze the results. Without loss of generality we focus on turn-based MGs for practical reasons. Indeed, in simultaneous games the complexity of the problem grows exponentially with the number of player whereas in turn-based MGs it only grows linearly. We focus on MGs with deterministic dynamics. In a turn-based MG, only one player can choose an action in each state. It is said that the player controls the state.
Thus, for a given joint strategy $\pi$ of the players, we can exactly evaluate the value of the joint strategy $v^t_\pi$, for each player $i$, the value of the best response to the strategy of the others $v^{t+1}_i$, and the performance criterion $\|\max_{\tilde{s_i}} v^t_{\tilde{s_i},\pi^{-i}} - v^t_\pi \|_2$. The value $v^t_\pi = (I - \gamma P_\pi)^{-1}v^0_\pi$ is computed by inverting a linear system and the value $v^{t+1}_i$ with the policy iteration algorithm.

A $N$-player Garnets is described by a tuple $(N_S, N_A)$. The constant $N_S$ is the number of states and $N_A$ is the number of actions of the MG. For every state, a player is selected uniformly over $\{1, \ldots, N\}$. The resulting vector encodes which player plays at each state as it is a turn-based game. For each state and action $(s, a)$, we draw a random variable $\xi$ according to a normal law $\mathcal{N}(s, \xi)$ then we build the transition matrix such as $p([\xi] (\mod N_S)|s, a) = 1$.

In order to enforce a structure into the Garnets, the reward of the players linearly decreases by the distance from the current state to a randomly selected critical state $\hat{s}_i$ for $i$ in $\{1, \ldots, N\}$. A Gaussian noise of standard deviation $\hat{\sigma}$ and sparsity ratio $S$ are also added to harden the task. $r^i(s, a) = (r^i(s) + \mathcal{N}(0, \sigma)) \times \text{Bern}(1 - S)$ with $r^i(s) = \frac{2 \min(|s - \hat{s}_1|, N_S) - |s - \hat{s}_i|)}{N_S}$ One may note that the reward has a tore structure. More importantly, players with close critical states are pushed to collaborate while players with distant critical states would play against each other.

Finally, a sample $(s, (a_1, \ldots, a^N), (r_1, \ldots, r^N), s')$ is generated by randomly selecting a state and the action uniformly. The reward, the next state, and the next player are selected according to the model described above.

**Evaluation:** To compute the strategy of a player, we evaluate the $\pi$-network by iterating over every state. We keep track of two key metrics. The empirical norm of the Bellman residual on both the training dataset and the test dataset and the Error vs Best Response for every player defined as $\|v^t_{\pi_i} - v^t_{\pi_i}^*\|_2$. The latter is a normalized quantification of how sub-optimal the player’s strategy is compared to his best response. In other words, it indicates by which margin a player would have been able to increase his cumulative rewards by improving his strategy while other players keep playing the same strategies. If this metric is unbalanced among the players, it means that one of the player has a poor strategy in that sense. More importantly, if this metric is close to zero for all players, then they have little incentive to switch from their current strategy. Thus, all players reach a weak $\epsilon$-Nash equilibrium with $\epsilon$ close to zero. Thus, this metric is absolutely crucial to evaluate our system.

**Training parameters:** We use $N$-player Garnets with either 2 or 5 players. The state space and the action space are respectively of size 100 and 5. The state is encoded by a binary vector. The transition kernel is built with $\hat{\sigma} = 1$ and the state vector has a tore structure to fit the reward function. The reward sparsity is set to 0.5 and the reward white noise has a standard deviation of 0.05. The discounted factor is set to 0.9. The $Q$-Networks and $\pi$-Networks have one hidden layers of size 80 with elu activation functions. $Q$-Networks have no output transfer function while $\pi$-Networks have a softmax. The gradient descent is performed with AdamGrad with an initial learning rate of 1e-3 for the $Q$-Network and 5e-5 for $\pi$-Networks. We use a weight decay of 1e-6. The training set is composed of 5$N_S N_A$ samples split into random minibatch of size 20 while the testing dataset contains $N_S N_A$ samples.

**Results:** Results are reported in Figure 2. In both scenarios, the quality of the learned strategies converges in parallel with the empirical Bellman residual. One the training is over, the players can increase their cumulative reward by no more than 8% in average. Therefore, neural networks succeed in learning a weak $\epsilon$-Nash equilibrium. Moreover, the strategies’ quality are well-balanced among the players as the standard deviation is below 5 points.

Our neural architecture has good scaling properties. First, scaling from 2 players to 5 results in the same strategy quality. Increasing the state space by another order of magnitude has little interest for Garnets as they are represented as a binary vector with no inherent structure. However, this neural architecture is agnostic to the state representation. In other words, one is free to have another state representation or to change input layers with convolutional networks.

**Limitations:** The neural architecture faces some over-fitting issues. It requires a high number of samples to converge as described on Figure 3 in the appendix. Enhancing the regularization and the gradient descent step may greatly reduce the number of required samples. Yet, those limitations remain specific to the neural architecture and they do not reduce the scope of the residual approach.
8 Conclusion

In this paper, we present a novel approach to learn a Nash equilibrium in MGs. The contributions of this paper are both theoretical and empirical. First we define a new (weaker) concept of an $\epsilon$-Nash equilibrium. In the case of an MDP, this notion is summarized to the minimization of the difference between the value of the learned policy and the value of the optimal policy. Then, we prove that minimizing the sum of different Bellman residuals is sufficient to learn a weak $\epsilon$-Nash equilibrium. From this result, we provide empirical estimators of these Bellman residuals from batch data. Finally, we describe a novel neural network architecture, called NashNetwork, to learn a Nash equilibrium from batch data. This architecture is agnostic to the MG and to the representation of the state space. It also scales to a high number of players. Thus it can be applied to a trove of applications.

As future works, the NashNetwork could be extended to more eclectic games such simultaneous games (i.e. Alesia Perolat et al. (2015) or a stick together game such as in Prasad et al. (2015)) or Atari’s games with several players such as Pong or Bomber. Additional optimization methods can be studied to this specific class of neural networks to increase the quality of learning. For instance, one can alternate the back-propagation among the networks or use different learning rates for the $\pi$-Networks and $Q$-Network. Finally, the $Q$-function and the strategy could be parametrised with other classes of function approximation such as trees. Yet, it requires to study functional gradient descent on the loss function.

Acknowledgments

This work has received partial funding from CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020 and the Project CHIST-ERA IGLU
References

Akchurina, N. (2010). Multi-agent reinforcement learning algorithms. PhD thesis, Paderborn, Univ., Diss., 2010.

Archibald, T., McKinnon, K., and Thomas, L. (1995). On the generation of markov decision processes. Journal of the Operational Research Society, pages 354–361.

Baird, L. et al. (1995). Residual algorithms: Reinforcement learning with function approximation. In Proc. of ICML.

Filar, J. and Vrieze, K. (2012). Competitive Markov decision processes. Springer Science & Business Media.

Grunewalder, S., Lever, G., Baldassarre, L., Pontil, M., and Gretton, A. (2012). Modelling transition dynamics in mdps with rkhs embeddings. Proc. of ICML.

Lagoudakis, M. G. and Parr, R. (2002). Value function approximation in zero-sum markov games. In Proc. of UAI.

Lagoudakis, M. G. and Parr, R. (2003). Reinforcement Learning as Classification: Leveraging Modern Classifiers. In Proc. of ICML.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. Nature, 521(7553):436–444.

Littman, M. L. (1994). Markov Games as a Framework for Multi-Agent Reinforcement Learning. In Proc. of ICML.

Littman, M. L. (2001). Friend-or-foe q-learning in general-sum games. In Proc. of ICML.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

Munos, R. and Szepesvári, C. (2008). Finite-time bounds for fitted value iteration. The Journal of Machine Learning Research, pages 815–857.

Perolat, J., Piot, B., Scherrer, B., and Pietquin, O. (2016). On the use of non-stationary strategies for solving two-player zero-sum Markov games. In Proc. of AISTATS.

Perolat, J., Scherrer, B., Piot, B., and Pietquin, O. (2015). Approximate Dynamic Programming for Two-Player Zero-Sum Markov Games. In Proc. of ICML.

Piot, B., Geist, M., and Pietquin, O. (2014a). Boosted Bellman Residual Minimization Handling Expert Demonstrations. In Proc. of ECML PKDD.

Piot, B., Geist, M., and Pietquin, O. (2014b). Difference of convex functions programming for reinforcement learning. In Proc. of NIPS.

Prasad, H., LA, P., and Bhatnagar, S. (2015). Two-timescale algorithms for learning nash equilibria in general-sum stochastic games. In Proc. of AAMAS.

Riedmiller, M. (2005). Neural fitted q iteration—first experiences with a data efficient neural reinforcement learning method. In Machine Learning: ECML 2005, pages 317–328. Springer.

Scherrer, B., Ghavamzadeh, M., Gabillon, V., and Geist, M. (2012). Approximate Modified Policy Iteration. In Proc. of ICML.

Shapley, L. S. (1953). Stochastic Games. Proceedings of the National Academy of Sciences of the United States of America.

Taylor, G. and Parr, R. (2012). Value function approximation in noisy environments using locally smoothed regularized approximate linear programs. Proc. of UAI.

Zinkevich, M., Greenwald, A., and Littman, M. (2006). Cyclic equilibria in markov games. Proc. of NIPS.
A Proof of the equivalence of definition 3, 3 and 3

First, let’s prove the equivalence of definition 3 and definition 3 if the Nash equilibrium of value $v(s)$ of the normal form game $(M_1^*, ..., M_N^*)$ is equivalent to

$$\forall s \in S, \forall i \in \{1, ..., N\}, E_{a \sim \pi} [M_i^*(a)] = \max_{a_i} E_{a_i \sim \pi_i^{-1}} [M_i^*(a_i)] = v^i(s)$$

which is equivalent to:

$$\forall s \in S, \forall i \in \{1, ..., N\}, T^i_\pi v^i(s) = T^{\pi_i^{-1}}_\pi v^i(s) = v^i(s)$$ (since $M_i^*(a) = T^i_\pi v^i(s)$)

Let’s now prove the equivalence between definition 3 and 3.

$$(3) \Rightarrow (3):$$

If there exists such that $v^i \in \{1, ..., N\}, T^i_\pi v^i = v^i$ and $T^{\pi_i^{-1}}_\pi v^i = v^i$, then $v^i = v^i_{\pi_i, \lambda^{-1}}$ and $v^i = v^i_{\pi_i, \lambda^{-1}}$.

$$(3) \Rightarrow (3):$$

if $v^i \in \{1, ..., N\}, v^i_{\pi_i, \lambda^{-1}} = \max v^i_{\pi_i, \lambda^{-1}}$, then $v^i = v^i_{\pi_i, \lambda^{-1}}$ is such as $T^i_\pi v^i = v^i$ and $T^{\pi_i^{-1}}_\pi v^i = v^i$

B Proof of Theorem 4

First we will prove the following lemma. The proof is strongly inspired by previous work on the minimization of the Bellman residual for MDPs [Piot et al., 2014b].

**Lemma 1.** Let $\rho$ and $p'$ be a real numbers such that $\frac{1}{\rho} + \frac{1}{p'} = 1$, then $\forall \nu, \pi$ and $v_i \in \{1, ..., N\}$:

$$\|v_i^{\pi_i, \pi^{-1}} - v_i^{\pi_i, \pi^{-1}}\|_{\nu, \rho} \leq \frac{1}{1 - \gamma} \left( C_{\infty, \mu, \nu, \pi_i, \pi^{-1}} T^\nu \|v^i - v^i\|_{\mu, \rho} + \|\|T^{\pi_i^{-1}}_\pi v^i - v^i\|_{\mu, p'}\|^p \right)^{\frac{1}{p'}}$$

where $\pi_i^*$ is the best response to $\pi^{-1}$. Meaning $v^i_{\pi_i^*, \pi^{-1}}$ is the fixed point of $T^{\pi_i^{-1}}_\pi$. And with the following concentrability coefficient $C_{\infty, \mu, \nu, \pi, \pi^{-1}} = \|\mu T_1(1-\gamma)(I-\gamma P_{\pi_i, \pi^{-1}})^{-1}\|_{\mu, \infty}$.

**Proof.** The proof uses similar techniques as in [Piot et al., 2014b]. First we have:

$$v_i^{\pi_i, \pi^{-1}} - v^i = (I - \gamma P_{\pi_i, \pi^{-1}})^{-1}(v_i^{\pi_i, \pi^{-1}} - (I - \gamma P_{\pi_i, \pi^{-1}}) v^i),$$

$$= (I - \gamma P_{\pi_i, \pi^{-1}})^{-1}(T^{\pi_i^{-1}}_\pi v^i - v^i).$$

But we also have:

$$v_i^{\pi_i, \pi^{-1}} - v^i = (I - \gamma P_{\pi_i, \pi^{-1}})^{-1}(T^{\pi_i^{-1}}_\pi v^i - v^i),$$

then:

$$v_i^{\pi_i, \pi^{-1}} - v_i^{\pi_i, \pi^{-1}} = v_i^{\pi_i, \pi^{-1}} - v_i^{\pi_i, \pi^{-1}} + v_i^{\pi_i, \pi^{-1}} - v_i^{\pi_i, \pi^{-1}} = (I - \gamma P_{\pi_i, \pi^{-1}})^{-1}(T^{\pi_i^{-1}}_\pi v^i - v^i) - (I - \gamma P_{\pi_i, \pi^{-1}})^{-1}(T^{\pi_i^{-1}}_\pi v^i - v^i) \leq (I - \gamma P_{\pi_i, \pi^{-1}})^{-1} \|T^{\pi_i^{-1}}_\pi v^i - v^i\| + (I - \gamma P_{\pi_i, \pi^{-1}})^{-1} \|T^{\pi_i^{-1}}_\pi v^i - v^i\|.$$

Finally, using the same technique as the one in Piot et al. (2014b), we get:

\[
\|v_{\pi^i,\pi^{-i}}^i - v_{\pi^i,\pi^{-i}}^i\|_{\mu,p} \\
\leq \|(I - \gamma \mathcal{P}_{\pi^i,\pi^{-i}})^{-1} T_{\pi^i,\pi^{-i}}^i v^i - v^i\|_{\mu,p} + \|(I - \gamma \mathcal{P}_{\pi^i,\pi^{-i}})^{-1} T_{\pi^i,\pi^{-i}}^i v^i - v^i\|_{\mu,p},
\]

\[
\leq \frac{1}{1 - \gamma} \left[ C_{\infty}(\mu,\nu,\pi^i,\pi^{-i})^{\frac{1}{p}} \|\mathcal{T}_{\pi^i,\pi^{-i}} v^i - v^i\|_{\nu,p} + C_{\infty}(\mu,\nu,\pi^i,\pi^{-i})^{\frac{1}{p}} \|\mathcal{T}_{\pi^i,\pi^{-i}} v^i - v^i\|_{\nu,p} \right],
\]

\[
\leq \frac{1}{1 - \gamma} \left( C_{\infty}(\mu,\nu,\pi^i,\pi^{-i})^{\frac{1}{p}} + C_{\infty}(\mu,\nu,\pi^i,\pi^{-i})^{\frac{1}{p}} \right)^{\frac{1}{p}} \left[ \|\mathcal{T}_{\pi^i,\pi^{-i}} v^i - v^i\|_{\nu,p} + \|\mathcal{T}_{\pi^i,\pi^{-i}} v^i - v^i\|_{\nu,p} \right].
\]

Theorem 4 falls in two steps:

\[
\left\| \max_{\pi^i} v_{\pi^i,\pi^{-i}} - v_{\pi^i} \right\|_{\mu(\pi),p} \leq \frac{1}{1 - \gamma} \left[ \max_{\pi^i \in \{1,\ldots,N\}} \left( C_{\infty}(\mu,\nu,\pi^i,\pi^{-i})^{\frac{1}{p}} + C_{\infty}(\mu,\nu,\pi^i,\pi^{-i})^{\frac{1}{p}} \right)^{\frac{1}{p}} \right]
\]

\[
\times \left[ \sum_{i=1}^{N} \rho(i) \left( \|\mathcal{T}_{\pi^i,\pi^{-i}} v^i - v^i\|_{\nu,p} + \|\mathcal{T}_{\pi^i,\pi^{-i}} v^i - v^i\|_{\nu,p} \right) \right],
\]

\[
\leq \frac{2^{\frac{1}{p}} C_{\infty}(\mu,\nu)^{\frac{1}{p}}}{1 - \gamma} \left[ \sum_{i=1}^{N} \rho(i) \left( \|\mathcal{T}_{\pi^i,\pi^{-i}} v^i - v^i\|_{\nu,p} + \|\mathcal{T}_{\pi^i,\pi^{-i}} v^i - v^i\|_{\nu,p} \right) \right],
\]

with \( C_{\infty}(\mu,\nu) = \left( \sup_{\pi^i,\pi^{-i}} C_{\infty}(\mu,\nu,\pi^i,\pi^{-i}) \right) \)

The first inequality is proven using lemma 1 and Holder inequality. The second inequality falls noticing \( \forall \pi^i, \pi^{-i}, C_{\infty}(\mu,\nu,\pi^i,\pi^{-i}) \leq \sup_{\pi^i,\pi^{-i}} C_{\infty}(\mu,\nu,\pi^i,\pi^{-i}). \)
C Additional curves

This section provides additional curves regarding the training of the NashNetwork.

Figure 3: Impact of the number of sampling on the quality of the learned strategy. The number of samples per batch is computed by $N_{\text{samples}} = \alpha N_A N_S$. Experiments are run on 3 different Garnets which is re-sampled 3 times to average the metrics.

Figure 4: (left) Evolution of the Error vs Best Response during the training for a given Garnet. When the Garnet has a complex structure or the batch is badly distributed, one player sometimes fails to learn a good strategy. (right) Distribution of actions in the strategy among the players with the highest probability. This plots highlights that $\pi$-Networks do modify the strategy during the training.