Learning Nash Equilibrium for General-Sum Markov Games from Batch Data

Julien Pérolat$^{(1)}$ julien.perolat@ed.univ-lille1.fr
Florian Strub$^{(1)}$ florian.strub@ed.univ-lille1.fr
Bilal Piot$^{(1,3)}$ bilal.piot@univ-lille1.fr
Olivier Pietquin$^{(1,2,3)}$ olivier.pietquin@univ-lille1.fr

$^{(1)}$Univ. Lille, CNRS, Centrale Lille, Inria, UMR 9189 - CRIStAL, F-59000 Lille, France
$^{(2)}$Institut Universitaire de France (IUF), France
$^{(3)}$DeepMind, UK

Abstract

This paper addresses the problem of learning a Nash equilibrium in γ-discounted multiplayer general-sum Markov Games (MGs) in a batch setting. As the number of players increases in MG, the agents may either collaborate or team apart to increase their final rewards. One solution to address this problem is to look for a Nash equilibrium. Although, several techniques were found for the subcase of two-player zero-sum MGs, those techniques fail to find a Nash equilibrium in general-sum Markov Games. In this paper, we introduce a new definition of ϵ-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 to learn such an ϵ-Nash equilibrium. Then, we show that minimizing an empirical estimate of the $\mathcal{L}_p$ norm of these Bellman-like residuals allows learning for general-sum games within the batch setting. Finally, we introduce a neural network architecture that successfully learns a Nash equilibrium in generic multiplayer general-sum turn-based MGs.

1 Introduction

A Markov Game (MG) is a model for Multi-Agent Reinforcement Learning (MARL) [Littman, 1994]. Chess, robotics or human-machine dialogues 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. In this paper, we focus on presently the most generic MG framework with perfect information: general-sum multiplayer MGs. This framework includes MDPs (which are single agent MGs), zero-sum two-player games, normal form games, extensive form games and other derivatives [Nisan et al., 2007]. For instance, general-sum multiplayer MGs have neither a specific reward structure as opposed to zero-sum two-player MGs [Shapley, 1953, Perolat et al., 2015] nor state space constraints such as the tree structure in extensive form games [Nisan et al., 2007]. More importantly, when it comes to MARL, using classic models of reinforcement learning (MDP or Partially Observable MDP) assumes that the other players are part of the environment. Therefore, other players have a stationary strategy and do not update it according to the current player. General-sum multiplayer MGs remove this assumption by allowing the agents to dynamically re-adapt their strategy according the other players. Therefore, they may either cooperate or to confront in different states which entails joint-strategies between the players.

In this paper, agents are not allowed to directly settle a common strategy beforehand. 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 a Nash equilibrium [Filar and Vrieze, 2012]. Note that different Nash equilibria may co-exist in a single MG. Computing a Nash equilibrium in MGs requires a perfect knowledge of the dynamics and the rewards of the game. In most practical situations, the dynamics and the rewards are unknown. To overcome this difficulty two approaches are possible: Either learn the game by interacting with other agents or extract information from the game through a finite record of interactions between the agents. The former is known as the online

Appearing in Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS) 2017, Fort Lauderdale, Florida, USA. JMLR: W&CP volume 54. Copyright 2017 by the authors.

scenario while the latter is called the batch scenario. This paper focuses on the batch scenario. Furthermore, when a player learns a strategy, he faces a representation problem. In every state of the game (or configuration), he has to store a strategy corresponding to his goal. However, this approach does not scale with the number of states and it prevents from generalizing a strategy from one state to another. Thus, function approximation is introduced to estimate and generalize the strategy.

First, the wider family of batch algorithms applied to MDPs builds upon approximate value iteration [Riedmiller, 2005, Munos and Szepesvári, 2008] or approximate 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, Perolat et al., 2016]. All the previously referenced algorithms use the value function or the state-action value function to compute the optimal strategy. However, [Zinkevich et al., 2006] demonstrates that no algorithm 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. Therefore, 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]. However, none of the herein-before algorithms approximates the value functions or the strategies. To our knowledge, using function approximation to find a Nash-equilibrium in MGs has not been attempted before this paper.

The key contribution of this paper is to introduce a novel approach to learn an ϵ-Nash equilibrium in a general-sum multiplayer MG in the batch setting. This contribution relies on two ideas: the minimization of two Bellman-like residuals and the definition of the notion of weak ϵ-Nash equilibrium. Again, our approach is the first work that considers function approximation to find an ϵ-Nash equilibrium. As a second contribution, we empirically evaluate our idea on randomly generated deterministic MGs by designing a neural network architecture. This network minimizes the Bellman-like residuals by approximating the strategy and the state-action value-function of each player.

These contributions are structured as follows: first, we recall the definitions of an MG, of a Nash equilibrium and we define a weaker notion of an ϵ-Nash equilibrium. Then, we show that controlling the sum of several Bellman residuals allows us to learn an ϵ-Nash equilibrium. Later, we explain that controlling the empirical norm of those Bellman residuals addresses the batch scenario problem. At this point, we provide a description of the NashNetwork. Finally, we empirically evaluate our method on randomly generated MGs (Garnets). Using classic games would have limited the scope of the evaluation. Garnets enable to set up a myriad of relevant configuration and they avoid drawing conclusion on our approach from specific games.

2 Background

In an N-player MG with a finite state space S all players take actions in a finite set $((A^i(s))_{s\in S})_{i\in\{1,\dots,N\}}$ depending on the player and on the state. In every state s, each player has to take an action within his/her action space $a^i \in A^i(s)$ and receives a reward signal $r^i(s, a^1, ..., a^N)$ which depends on the joint action of all players in state s. Then, we assume that the state changes according to a transition kernel $p(s'|s, a^1, ..., a^N)$. All actions indexed by $-i$ are joint actions of all players except player i (i.e. $\boldsymbol{a}^{\text{-}\boldsymbol{i}} = (a^1, \dots, a^{i-1}, a^{i+1}, \dots, a^N)$). For the sake of simplicity we will note $\mathbf{a} = (a^1, \dots, a^N) = (a^i, \boldsymbol{a}^{\text{-}\boldsymbol{i}})$, $p(s'|s, a^1, ..., a^N) = p(s'|s, \mathbf{a}) = p(s'|s, a^i, \boldsymbol{a}^{\text{-}\boldsymbol{i}})$ and $r^i(s, a^1, ..., a^N) = r^i(s, \mathbf{a}) = r^i(s, a^i, \boldsymbol{a}^{\text{-}\boldsymbol{i}})$. The constant $\gamma \in [0, 1)$ is the discount factor.

In a MARL problem, the goal is typically to find a strategy for each player (a policy in the MDP literature). A stationary strategy π^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 $\boldsymbol{\pi} = (\pi^1, ..., \pi^N) = (\pi^i, \boldsymbol{\pi}^{\text{-}\boldsymbol{i}})$ the product distribution of those strategies (i.e. $\boldsymbol{\pi}^{\text{-}\boldsymbol{i}} = (\pi^1, \dots, \pi^{i-1}, \pi^{i+1}, \dots, \pi^N)$). The stochastic kernel defining the dynamics of the Markov Chain when all players follow their strategy $\boldsymbol{\pi}$ is $\mathcal{P}_{\boldsymbol{\pi}}(s'|s) = E_{\mathbf{a}\sim\boldsymbol{\pi}}[p(s'|s, \mathbf{a})]$ and the one defining the MDP when all players but player i follow their strategy $\boldsymbol{\pi}^{\text{-}\boldsymbol{i}}$ is $\mathcal{P}_{\boldsymbol{\pi}^{\text{-}i}}(s'|s, a^i) = E_{\boldsymbol{a}^{\text{-}i}\sim\boldsymbol{\pi}^{\text{-}i}}[p(s'|s, \mathbf{a})]$. The kernel $\mathcal{P}_{\boldsymbol{\pi}}$ can be seen as a squared matrix of size the number of states$|S|$. Identically, we can define the averaged reward over all strategies $r^i_{\boldsymbol{\pi}}(s) = E_{\mathbf{a}\sim\boldsymbol{\pi}}[r^i(s, \mathbf{a})]$ and the averaged reward over all players but player i, $r^i_{\boldsymbol{\pi}^{\text{-}i}}(s, a^i) = E_{\boldsymbol{a}^{\text{-}i}\sim\boldsymbol{\pi}^{\text{-}i}}[r^i(s, \mathbf{a})]$. Again, the reward $r^i_{\boldsymbol{\pi}}$ can be seen as a vector of size $|S|$. Finally, the identity matrix is noted $\mathcal{I}$.

Value of the joint strategy : For each state s, the expected return considering each player plays his part of the joint strategy $\boldsymbol{\pi}$ is the γ-discounted sum of his rewards [Prasad et al., 2015, Filar and Vrieze, 2012]:

$$\begin{aligned} v^i_{\boldsymbol{\pi}}(s) &= E[\sum_{t=0}^{+\infty} \gamma^t r^i_{\boldsymbol{\pi}}(s_t)|s_0 = s, s_{t+1} \sim \mathcal{P}_{\boldsymbol{\pi}}(.|s_t)], \\ &= (\mathcal{I} - \gamma \mathcal{P}_{\boldsymbol{\pi}})^{-1} r^i_{\boldsymbol{\pi}}. \end{aligned}$$

The joint value of each player is the following: $\boldsymbol{v_\pi} = (v^1_{\boldsymbol{\pi}}, ..., v^N_{\boldsymbol{\pi}})$. In the MG theory, two Bellman operators can be defined on the value function with respect to player i. The first one is $\mathcal{T}^i_{\mathbf{a}} v^i\ (s) = r^i(s, \mathbf{a}) + \gamma \sum_{s' \in S} p(s'|s, \mathbf{a}) v^i(s')$. It represents the expected value player i will get in state s if all players play the joint action $\boldsymbol{a}$ and if the value for player i after one transition is v^i. If we consider that instead of playing action $\boldsymbol{a}$ every player plays according to a joint strategy $\boldsymbol{\pi}$ the Bellman operator to consider is $\mathcal{T}^i_{\boldsymbol{\pi}} v^i\ (s) = E_{\mathbf{a} \sim \boldsymbol{\pi}}[\mathcal{T}^i_{\mathbf{a}} v^i\ (s)] = r^i_{\boldsymbol{\pi}}(s) + \gamma \sum_{s' \in S} \mathcal{P}_{\boldsymbol{\pi}}(s'|s) v^i(s')$. As in MDPs theory, this operator is a γ-contraction in $\mathcal{L}_{+\infty}$-norm [Puterman, 1994]. Thus, it admits a unique fixed point which is the value for player i of the joint strategy $\boldsymbol{\pi}$: $v^i_{\boldsymbol{\pi}}$.

Value of the best response : When the strategy $\boldsymbol{\pi^{\text{-}i}}$ of all players but player i is fixed, the problem is reduced to an MDP of kernel $\mathcal{P}_{\boldsymbol{\pi}^{\text{-}i}}$ and reward function $r^i_{\boldsymbol{\pi}^{\text{-}i}}$. In this underlying MDP, the optimal Bellman operator would be $\mathcal{T}^{*i}_{\boldsymbol{\pi}^{\text{-}i}} v^i\ (s) = \max_{a^i} E_{\boldsymbol{a}^{\text{-}i} \sim \boldsymbol{\pi}^{\text{-}i}}[\mathcal{T}^i_{a^i, \boldsymbol{a}^{\text{-}i}} v^i\ (s)] = \max_{a^i}[r^i_{\boldsymbol{\pi}^{\text{-}i}}(s, a^i) + \gamma \sum_{s' \in S} \mathcal{P}_{\boldsymbol{\pi}^{\text{-}i}}(s'|s, a^i) v^i(s')]$. The fixed point of this operator is the value of a best response of player i when all other players play $\boldsymbol{\pi^{\text{-}i}}$ and will be written $v^{*i}_{\boldsymbol{\pi}^{\text{-}i}} = \max_{\tilde{\pi}^i} v^i_{\tilde{\pi}^i, \boldsymbol{\pi}^{\text{-}i}}$.

3 Nash, ϵ-Nash and Weak ϵ-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 players do not vary their own one [Filar and Vrieze, 2012]. The goal of this paper is to find one strategy for players which is as close as possible to a Nash equilibrium.

Definition 1. *In an MG, a strategy $\boldsymbol{\pi}$ is a Nash equilibrium if:* $\forall i \in \{1, ..., N\}$, $v^i_{\boldsymbol{\pi}} = v^{*i}_{\boldsymbol{\pi}^{\text{-}i}}$.

This definition can be rewritten with Bellman operators:

Definition 2. *In an MG, a strategy $\boldsymbol{\pi}$ is a Nash equilibrium if $\exists \boldsymbol{v}$ such as $\forall i \in \{1, ..., N\}, \mathcal{T}^i_{\boldsymbol{\pi}} v^i = v^i$ and $\mathcal{T}^{*i}_{\boldsymbol{\pi}^{\text{-}i}} v^i = v^i$.*

Proof. The proof is left in appendix A. □

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

Definition 3. *In an MG, a strategy $\boldsymbol{\pi}$ is an ϵ-Nash equilibrium if:*
$\forall i \in \{1, ..., N\}$, $v^i_{\boldsymbol{\pi}} + \epsilon \geq v^{*i}_{\boldsymbol{\pi}^{\text{-}i}}$
or $\forall i \in \{1, ..., N\}$, $v^{*i}_{\boldsymbol{\pi}^{\text{-}i}} - v^i_{\boldsymbol{\pi}} \leq \epsilon$,
which is equivalent to: $\left\| \left\| v^{*i}_{\boldsymbol{\pi}^{\text{-}i}} - v^i_{\boldsymbol{\pi}} \right\|_{s,\infty} \right\|_{i,\infty} \leq \epsilon$.

Interestingly, when considering an MDP, the definition of an ϵ-Nash equilibrium is reduced to control the $\mathcal{L}_{+\infty}$-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 $\mathcal{L}_{+\infty}$-norm but rather an $\mathcal{L}_p$-norm [Munos and Szepesvári, 2008] (we take the definition of the $\mathcal{L}_p$-norm of [Piot et al., 2014b]). Using $\mathcal{L}_p$ -norm is necessary for approximate dynamic programming algorithms to use complexity bounds from learning theory [Piot et al., 2014b]. The convergence of these algorithms was analyzed using supervised learning bounds in $\mathcal{L}_p$-norm and thus guaranties are given in $\mathcal{L}_p$-norm [Scherrer et al., 2012]. In addition, Bellman residual approaches on MDPs also give guaranties in $\mathcal{L}_p$-norm [Maillard et al., 2010, Piot et al., 2014b]. Thus, we define a natural relaxation of the previous definition of the ϵ-Nash equilibrium in $\mathcal{L}_p$-norm which is consistent with the existing work on MDPs.

Definition 4. *In a MG, $\boldsymbol{\pi}$ is a weak ϵ-Nash equilibrium if:* $\left\| \left\| v^{*i}_{\boldsymbol{\pi}^{\text{-}i}} - v^i_{\boldsymbol{\pi}} \right\|_{\mu(s),p} \right\|_{\rho(i),p} \leq \epsilon$.

One should notice that an ϵ-Nash equilibrium is a weak ϵ-Nash equilibrium. Conversely, a weak ϵ-Nash equilibrium is not always an ϵ-Nash equilibrium. Furthermore, both ϵ do not need to be equal. The notion of weak ϵ-Nash equilibrium defines a performance criterion to evaluate a strategy while seeking for a Nash equilibrium. Thus, this definition has the great advantage to provide a convincing way to evaluate the final strategy. In the case of an MDP, it states that a weak ϵ-Nash equilibrium only consists in controlling the difference in $\mathcal{L}_p$-norm between the optimal value and the value of the learned strategy. For an MG, this criterion is an $\mathcal{L}_p$-norm over players ($i \in \{1, \ldots, N\}$)

of an $\mathcal{L}_p$-norm over states ($s \in S$) of the difference between the value of the joint strategy $\boldsymbol{\pi}$ for player i in state s and of his best response against the joint strategy $\boldsymbol{\pi}^{\text{-}\boldsymbol{i}}$. In the following, we consider that learning a Nash equilibrium should result in minimizing the loss $\left\|\left\|v^{*i}_{\boldsymbol{\pi}^{\text{-}i}} - v^i_{\boldsymbol{\pi}}\right\|_{\mu(s),p}\right\|_{\rho(i),p}$. For each player, we want to minimize the difference between the value of his strategy and a best response considering the strategy of the other players is fixed. However, a direct minimization of that norm is not possible in the batch setting even for MDPs. Indeed, $v^{*i}_{\boldsymbol{\pi}^{\text{-}i}}$ cannot be directly observed and be used as a target. A common strategy to alleviate this problem is to minimize a surrogate loss. In MDPs, a possible surrogate loss is the optimal Bellman residual $\|v - T^* v\|_{\mu,p}$ (where μ is a distribution over states and T^* is the optimal Bellman operator for MDPs) [Piot et al., 2014b, Baird et al., 1995]. The optimal policy is then extracted from the learnt optimal value (or Q-value in general). In the following section, we extend this Bellman residual approach to MGs.

4 Bellman Residual Minimization in MGs

Optimal Bellman residual minimization is not straightforwardly extensible to MGs because multiplayer strategies cannot be directly extracted from value functions as shown in [Zinkevich et al., 2006]. Yet, from Definition 2, we know that a joint strategy $\boldsymbol{\pi}$ is a Nash equilibrium if there exists $\boldsymbol{v}$ such that, for any player i, v^i is the value of the joint strategy $\boldsymbol{\pi}$ for player i (i.e. $\mathcal{T}^i_{\boldsymbol{\pi}} v^i = v^i$) and v^i is the value of the best response player i can achieve regarding the opponent's strategy $\boldsymbol{\pi}^{\text{-}\boldsymbol{i}}$ (i.e. $\mathcal{T}^{*i}_{\boldsymbol{\pi}^{\text{-}i}} v^i = v^i$). We thus propose to build a second Bellman-like residual optimizing over the set of strategies so as to directly learn an ϵ-Nash equilibrium. The first (traditional) residual (i.e. $\left\|\mathcal{T}^{*i}_{\boldsymbol{\pi}^{\text{-}i}} v^i - v^i\right\|_{\rho(i),p}$) forces the value of each player to be close to their respective best response to every other player while the second residual (i.e. $\left\|\mathcal{T}^i_{\boldsymbol{\pi}} v^i - v^i\right\|_{\rho(i),p}$) will force every player to play the strategy corresponding to that value.

One can thus wonder how close from a Nash-Equilibrium $\boldsymbol{\pi}$ would be if there existed $\boldsymbol{v}$ such that $\mathcal{T}^{*i}_{\boldsymbol{\pi}^{\text{-}i}} v^i \approx v^i$ and $\mathcal{T}^i_{\boldsymbol{\pi}} v^i \approx v^i$. In this section, we prove that, if we are able to control over $(\boldsymbol{v}, \boldsymbol{\pi})$ a sum of the $\mathcal{L}_p$-norm of the associated Bellman residuals ($\left\|\mathcal{T}^{*i}_{\boldsymbol{\pi}^{\text{-}i}} v^i - v^i\right\|_{\mu,p}$and $\left\|\mathcal{T}^i_{\boldsymbol{\pi}} v^i - v^i\right\|_{\mu,p}$), then we are able to control $\left\|\left\|v^{*i}_{\boldsymbol{\pi}^{\text{-}i}} - v^i_{\boldsymbol{\pi}}\right\|_{\mu(s),p}\right\|_{\rho(i),p}$.

Theorem 1. *$\forall p, p'$ positive reals such that $\frac{1}{p} + \frac{1}{p'} = 1$ and $\forall (v^1, \dots, v^N)$:*

$$\left\|\left\|v^{*i}_{\boldsymbol{\pi}^{\text{-}i}} - v^i_{\boldsymbol{\pi}}\right\|_{\mu(s),p}\right\|_{\rho(i),p} \leq \frac{2^{\frac{1}{p'}} C_\infty(\mu,\nu)^{\frac{1}{p}}}{1-\gamma}$$
$$\times \left[\sum_{i=1}^{N} \rho(i) \left(\left\|\mathcal{T}^{*i}_{\boldsymbol{\pi}^{\text{-}i}} v^i - v^i\right\|^p_{\nu,p} + \left\|\mathcal{T}^i_{\boldsymbol{\pi}} v^i - v^i\right\|^p_{\nu,p}\right)\right]^{\frac{1}{p}},$$

with the following concentrability coefficient (the norm of a Radon-Nikodym derivative) $C_\infty(\mu, \nu, \pi^i, \boldsymbol{\pi}^{\text{-}\boldsymbol{i}}) = \left\|\frac{\partial \mu^T (1-\gamma)(\mathcal{I} - \gamma \mathcal{P}_{\pi^i, \boldsymbol{\pi}^{\text{-}i}})^{-1}}{\partial \nu^T}\right\|_{\nu,\infty}$ *and* $C_\infty(\mu,\nu) = \sup_{\boldsymbol{\pi}} C_\infty(\mu, \nu, \pi^i, \boldsymbol{\pi}^{\text{-}\boldsymbol{i}})$.

Proof. The proof is left in appendix B. $\square$

This theorem shows that an ϵ-Nash equilibrium can be controlled by the sum over the players of the sum of the norm of two Bellman-like residuals: the Bellman Residual of the best response of each player and the Bellman residual of the joint strategy. If the residual of the best response of player i ($\left\|\mathcal{T}^{*i}_{\boldsymbol{\pi}^{\text{-}i}} v^i - v^i\right\|^p_{\nu,p}$) is small, then the value v^i is close to the value of the best response $v^{*i}_{\boldsymbol{\pi}^{\text{-}i}}$ and if the Bellman residual of the joint strategy $\left\|\mathcal{T}^i_{\boldsymbol{\pi}} v^i - v^i\right\|^p_{\nu,p}$ for player i is small, then v^i is close to $v^i_{\boldsymbol{\pi}}$. In the end, if all those residuals are small, the joint strategy is an ϵ-Nash equilibrium with ϵ small since $v^{*i}_{\boldsymbol{\pi}^{\text{-}i}} \simeq v^i \simeq v^i_{\boldsymbol{\pi}}$.

Theorem 1 also emphasizes the necessity of a weakened notion of an ϵ-Nash equilibrium. It is much easier to control a $\mathcal{L}_p$-norm than a $\mathcal{L}_\infty$-norm with samples. In the following, the weighted sum of the norms of the two Bellman residuals will be noted as:

$$f_{\nu,\rho,p}(\boldsymbol{\pi}, \boldsymbol{v}) = \sum_{i=1}^{N} \rho(i) \left(\left\|\mathcal{T}^{*i}_{\boldsymbol{\pi}^{\text{-}i}} v^i - v^i\right\|^p_{\nu,p} + \left\|\mathcal{T}^i_{\boldsymbol{\pi}} v^i - v^i\right\|^p_{\nu,p}\right).$$

Finding a Nash equilibrium is then reduced to a non-convex optimization problem. If we can find a $(\boldsymbol{\pi}, \boldsymbol{v})$ such that $f_{\nu,\rho,p}(\boldsymbol{\pi}, \boldsymbol{v}) = 0$, then the joint strategy $\boldsymbol{\pi}$ is a Nash equilibrium. This procedure relies on a search over the joint value function space and the joint strategy space.Besides, if the state space or the number of joint actions is too large, the search over values and strategies might be intractable. We addressed the issue by making use of approximate value functions and strategies. Actually, Theorem 1 can be extended with function approximation. A good joint strategy $\boldsymbol{\pi}_\theta$ within an approximate strategy space Π can still be found by computing $\boldsymbol{\pi}_{\boldsymbol{\theta}}, \boldsymbol{v}_{\boldsymbol{\eta}} \in \underset{\boldsymbol{\theta}, \boldsymbol{\eta}}{\operatorname{argmin}} f_{\nu,\rho,p}(\boldsymbol{\pi}_{\boldsymbol{\theta}}, \boldsymbol{v}_{\boldsymbol{\eta}})$ (where $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$ respectively parameterize $\boldsymbol{\pi}_{\boldsymbol{\theta}}, \boldsymbol{v}_{\boldsymbol{\eta}}$). Even with function approximation, the learned joint strategy $\boldsymbol{\pi}_\theta$ would be at least a weak ϵ-Nash equilibrium (with $\epsilon \leq \frac{2^{\frac{1}{p'}} C_\infty(\mu,\nu)^{\frac{1}{p}}}{1-\gamma} f_{\nu,\rho,p}(\boldsymbol{\pi}_{\boldsymbol{\theta}}, \boldsymbol{v}_{\boldsymbol{\eta}})$).This is, to our knowledge, the first approach to solve MGs within an approximate strategy space and an approximate value function space.

5 The Batch Scenario

In this section, we explain how to learn a weak ϵ-Nash equilibrium from Theorem 1 with approximate strategies and an approximate value functions. As said previously, we will focus on the batch setting where only a finite number of historical data sampled from an MG are available.

In the batch scenario, it is common to work on state-action value functions (also named Q-functions). In MDPs, the Q-function is defined on the state and the action of one agent. In MGs, the Q-function has to be extended over the joint action of the agents [Perolat et al., 2015, Hu and Wellman, 2003]. Thus, the Q-function in MGs for a fixed joint strategy $\boldsymbol{\pi}$ is the expected γ-discounted sum of rewards considering that the players first pick the joint action $\boldsymbol{a}$ and follow the joint strategy $\boldsymbol{\pi}$. Formally, the Q-function is defined as $Q^i_{\boldsymbol{\pi}}(s, \mathbf{a}) = \mathcal{T}^i_{\mathbf{a}} v^i_{\boldsymbol{\pi}}$. Moreover, one can define two analogue Bellman operators to the ones defined for the value function:

$$\mathcal{B}^i_{\boldsymbol{\pi}} Q\ (s, \mathbf{a}) = r^i(s, \mathbf{a}) + \sum_{s' \in S} p(s'|s, \mathbf{a}) E_{\mathbf{b} \sim \boldsymbol{\pi}}[Q(s', \mathbf{b})]$$
$$\mathcal{B}^{*i}_{\boldsymbol{\pi}} Q\ (s, \mathbf{a}) = r^i(s, \mathbf{a}) + \sum_{s' \in S} p(s'|s, \mathbf{a}) \max_{b^i} \left[E_{\boldsymbol{b}^{\text{-}i} \sim \boldsymbol{\pi}^{\text{-}i}}[Q(s', b^i, \boldsymbol{b}^{\text{-}\boldsymbol{i}})] \right].$$

As for the value function, $Q^i_{\boldsymbol{\pi}}$ is the fixed point of $\mathcal{B}^i_{\boldsymbol{\pi}}$ and $Q^{*i}_{\boldsymbol{\pi}^{\text{-}i}}$ is the fixed point of $\mathcal{B}^{*i}_{\boldsymbol{\pi}}$ (where $Q^{*i}_{\boldsymbol{\pi}^{\text{-}i}} = \max_{\pi^i} Q^i_{\boldsymbol{\pi}}$). The extension of Theorem 1 to Q-functions instead of value functions is straightforward. Thus, we will have to minimize the following function depending on strategies and Q-functions:

$$f(\mathbf{Q}, \boldsymbol{\pi}) = \sum_{i=1}^{N} \rho(i) \left(\left\| \mathcal{B}^{*i}_{\boldsymbol{\pi}^{\text{-}i}} Q^i - Q^i \right\|^p_{\nu,p} + \left\| \mathcal{B}^i_{\boldsymbol{\pi}} Q^i - Q^i \right\|^p_{\nu,p} \right) \tag{1}$$

The batch scenario consists in having a set of k samples $(s_j, (a^1_j, ..., a^N_j), (r^1_j, ..., r^N_j), s'_j)_{j \in \{1,...,k\}}$ where $r^i_j = r^i(s_j, a^1_j, ..., a^N_j)$ and where the next state is sampled according to $p(.|s_j, a^1_j, ..., a^N_j)$. From Equation (1), we can minimize the empirical-norm by using the k samples to obtain the empirical estimator of the Bellman residual error.

$$\tilde{f}_k(\mathbf{Q}, \boldsymbol{\pi}) = \sum_{j=1}^{k} \sum_{i=1}^{N} \rho(i) \left[\left| \mathcal{B}^{*i}_{\boldsymbol{\pi}^{\text{-}i}} Q^i(s_j, \mathbf{a}_j) - Q^i(s_j, \mathbf{a}_j) \right|^p + \left| \mathcal{B}^i_{\boldsymbol{\pi}} Q^i(s_j, \mathbf{a}_j) - Q^i(s_j, \mathbf{a}_j) \right|^p \right], \tag{2}$$

For more details, an extensive analysis beyond the minimization of the Bellman residual in MDPs can be found in [Piot et al., 2014b]. In the following we discuss the estimation of the two Bellman residuals $|\mathcal{B}^{*i}_{\boldsymbol{\pi}^{\text{-}i}} Q^i(s_j, \mathbf{a}_j) - Q^i(s_j, \mathbf{a}_j)|^p$ and $|\mathcal{B}^i_{\boldsymbol{\pi}} Q^i(s_j, \mathbf{a}_j) - Q^i(s_j, \mathbf{a}_j)|^p$ in different cases.

Deterministic Dynamics: With deterministic dynamics, the estimation is straightforward. We estimate $\mathcal{B}^i_{\boldsymbol{\pi}} Q^i(s_j, \mathbf{a}_j)$ with $r^i_j + \gamma E_{\mathbf{b} \sim \boldsymbol{\pi}}[Q^i(s'_j, \mathbf{b})]$ and $\mathcal{B}^{*i}_{\boldsymbol{\pi}^{\text{-}i}} Q^i(s_j, \mathbf{a}_j)$ with $r^i_j + \gamma \max_{b^i} \left[E_{\boldsymbol{b}^{\text{-}i} \sim \boldsymbol{\pi}^{\text{-}i}}[Q^i(s'_j, b^i, \boldsymbol{b}^{\text{-}\boldsymbol{i}})] \right]$, where the expectation are:

$$E_{\mathbf{b} \sim \boldsymbol{\pi}}[Q^i(s', \mathbf{b})] = \sum_{b^1 \in A^1} \cdots \sum_{b^N \in A^N} \pi^1(b^1|s') \dots \pi^N(b^N|s') Q^i(s', b^1, \dots, b^N)$$

and where

$$E_{\boldsymbol{b}^{\text{-}i} \sim \boldsymbol{\pi}^{\text{-}i}}[Q^i(s', b^i, \boldsymbol{b}^{\text{-}\boldsymbol{i}})] = \sum_{b^1 \in A^1} \cdots \sum_{b^{i-1} \in A^{i-1}} \sum_{b^{i+1} \in A^{i+1}} \cdots \sum_{b^N \in A^N} \pi^1(b^1|s') \dots \pi^{i-1}(b^{i-1}|s') \pi^{i+1}(b^{i+1}|s') \dots \times \pi^N(b^N|s') Q^i(s', b^1, \dots, b^N)$$

Please note that this computation can be turned into tensor operations as described in Appendix-C.

Stochastic Dynamics: In the case of stochastic dynamics, the previous estimator (1) is known to be biased [Maillard et al., 2010, Piot et al., 2014b]. If the dynamic is known, one can use the following unbiased estimator: $\mathcal{B}^i_{\boldsymbol{\pi}} Q^i(s_j, \mathbf{a}_j)$ is estimated with $r^i_j + \gamma \sum_{s' \in S} p(s'|s_j, \mathbf{a}_j) E_{\mathbf{b} \sim \boldsymbol{\pi}}[Q^i(s', \mathbf{b})]$ and $\mathcal{B}^{*i}_{\boldsymbol{\pi}^{\text{-}i}} Q^i(s_j, \mathbf{a}_j)$ with $r^i_j + \gamma \sum_{s' \in S} p(s'|s_j, \mathbf{a}_j) \max_{b^i} \left[E_{\boldsymbol{b}^{\text{-}i} \sim \boldsymbol{\pi}^{\text{-}i}}[Q^i(s', b^i, \boldsymbol{b}^{\text{-}\boldsymbol{i}})] \right]$.

If the dynamic of the game is not known (e.g. batch scenario), the unbiased estimator cannot be used since the kernel of the MG is required and other techniques must be applied. One idea would be to first learn an approximation of this kernel. For instance, one could extend the MDP techniques to MGs such as embedding a kernel in a Reproducing Kernel Hilbert Space (RKHS) [Grunewalder et al., 2012, Piot et al., 2014a, Piot et al., 2014b] or using kernel estimators [Taylor and Parr, 2012, Piot et al., 2014b]. Therefore, $p(s'|s_j, \boldsymbol{a}_j)$ would be replaced by an estimator of the dynamics $\tilde{p}(s'|s_j, \boldsymbol{a}_j)$ in the previous equation. If a generative model is available, the issue can be addressed with double sampling as discussed in [Maillard et al., 2010, Piot et al., 2014b].

6 Neural Network architecture

Minimizing the sum of Bellman residuals is a challenging problem as the objective function is not convex. It is all the more difficult as both the Q-value function and the strategy of every player must be learned independently. Nevertheless, neural networks have been able to provide good solutions to problems that require minimizing non-convex objective such as image classification or speech recognition [LeCun et al., 2015]. Furthermore, neural networks were successfully applied to reinforcement learning to approximate the Q-function [Mnih et al., 2015] in one agent MGs with eclectic state representation.

Here, we introduce a novel neural architecture: the NashNetwork[1]. For every player, a two-fold network is defined: the Q-network that learns a Q-value function and a π-network that learns the stochastic strategy of the players. The Q-network is a multilayer perceptron which takes the state representation as input. It outputs the predicted Q-values of the individual action such as the network used by [Mnih et al., 2015]. Identically, the π-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. We then compute the two Bellman residuals for every player following Equation 5. Finally, we back-propagate the error by using classic gradient descent operations.

In our experiments, we focus on deterministic turn-based games. It entails a specific neural architecture that is fully described in Figure 6. During the training phase, all the Q-networks and the π-networks of the players are used to minimize the Bellman residual. Once the training is over, only the π-network is kept to retrieve the strategy of each player. Note that this architecture differs from classic actor-critic networks [Lillicrap et al., 2016] for several reasons. Although Q-network is an intermediate support to the computation of π-network, neither policy gradient nor advantage functions are used in our model. Besides, the Q-network is simply discarded at the end of the training. The NashNetwork directly searches in the strategy space by minimizing the Bellman residual.

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., 2014b] and in zero-sum two-player MGs [Perolat et al., 2016]. First, we extend the class of randomly generated MDPs to general-sum MGs, then we describe the training setting, finally we analyze the results. Without loss of generality we focus on deterministic turn-based MGs for practical reasons. Indeed, in simultaneous games the complexity of the state actions space grows exponentially with the number of player whereas in turn-based MGs it only grows linearly. Besides, as in the case of simultaneous actions, a Nash equilibrium in a turn-based MG (even deterministic) might be a stochastic strategy [Zinkevich et al., 2006]. In a turn-based MG, only one player can choose an action in each state. Finally, we run our experiments on deterministic MGs to avoid bias in the estimator as discussed in Sec.5.

To sum up, we use turn-based games to avoid the exponential growth of the number of actions and deterministic games to avoid bias in the estimator. The techniques described in Sec.5 could be implemented with a slight modification of the architecture described in Sec.6.

Dataset: We chose to use artificially generated MGs (named Garnet) to evaluate our method as they are not problem dependent. They have the great advantage to allow to easily change the state space (and its underlying structure), the action space and the number of players. Furthermore, Garnets have the property to present all the characteristics of a complete MG. Standard games such as "rock-paper-scissors" (no states and zero-sum), or other games such as chess or checkers (deterministic and zero-sum), always rely on a limited number of MG properties. Using classic games would thus have hidden the generality of our approach even though it would have been more appealing. Finally, the underlying model of the dynamics of the Garnet is fully known. Thus, it is possible to investigate whether an ϵ-Nash equilibrium have been reached during the training and to quantify the ϵ. It would have been impossible to do so with more complex games.

An N-player Garnet 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 each state and action (s, a), we randomly pick the next state s' in the neighborhood of indices of s where s' follow a rounded normal distribution $\mathcal{N}(s, \hat{\sigma}_{s'})$. We then build the transition matrix such as $p(s'|s, a) = 1$. Next, we enforce an arbitrary reward structure into the Garnet to enable the emergence of an Nash equilibrium. Each player i has a random critical state $\hat{s}^i$ and the reward of this player linearly decreases by the distance (encoded by indices) to his critical state. Therefore, the goal of the player is to get as close as possible from his critical state. Some players may have close critical states and may act together while other players have to follow opposite directions. A Gaussian noise of standard deviation $\hat{\sigma}_{noise}$ is added to the reward, then we sparsify it with a ratio m_{noise} to harden the task. Finally,

[1]`https://github.com/fstrub95/nash_network`

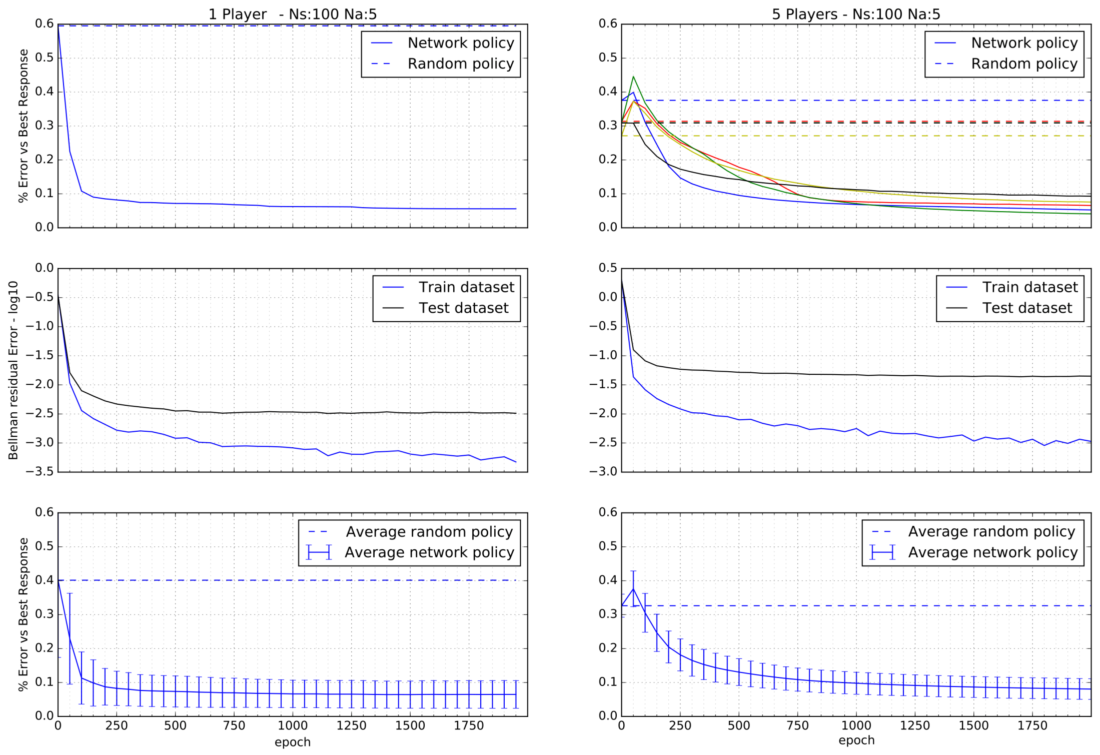


Figure 1: (top) Evolution of the Error vs Best Response during the training for a given Garnet. In both cases, players manage to iteratively improve their strategies. A strong drop may occur when one of the player finds a important move. (middle) Empirical Bellman residual for the training dataset and the testing dataset. It highlights the link between minimizing the Bellman residual and the quality of the learned strategy. (bottom) Average Error vs Best Response averaged over every players, Garnets and every batch. It highlights the robustness of minimizing the Bellman residual over several games. Experiments are run on 5 Garnets which is re-sampled 5 times to average the metrics.

a player is selected uniformly over $\{1, \dots, N\}$ for every state once for all. The resulting vector v_c encodes which player plays at each state as it is a turn-based game. Concretely, a Garnet works as follows: Given a state s, a player is first selected according the vector $v_c(s)$. This player then chooses an action according its strategy π^i. Once the action is picked, every player i receives its individual reward $r^i(s, a)$. Finally, the game moves to a new state according $p(s'|s, a)$ and a new state's player $(v_c(s'))$. Again, the goal of each player is to move as close as possible to its critical state. Thus, players benefit from choosing the action that maximizes its cumulative reward and leads to a state controlled by a non-adversarial player.

Finally, samples $(s, (a^1, \dots, a^N), (r^1, \dots, r^N), s')$ are generated by randomly selecting a state and an action uniformly. The reward, the next state, and the next player are selected according to the model described above.

Evaluation: Our goal is to verify whether the-joint strategy of the players is an ϵ-Nash equilibrium. To do so, we first retrieve the strategy of every player by evaluating the π^i-networks over the state space. Then, given $\boldsymbol{\pi}$, we can exactly evaluate the value of the joint strategy $v^i_{\boldsymbol{\pi}}$ for each player i and the value of the best response to the strategy of the others $v^{*i}_{\boldsymbol{\pi}^{-i}}$. To do so, the value $v^i_{\boldsymbol{\pi}}$ is computed by inverting the linear system $v^i_{\boldsymbol{\pi}} = (\mathcal{I} - \gamma \mathcal{P}_{\boldsymbol{\pi}})^{-1} r^i_{\boldsymbol{\pi}}$. The value $v^{*i}_{\boldsymbol{\pi}^{-i}}$ is computed with the policy iteration algorithm. Finally, we compute the *Error vs Best Response* for every player defined as $\frac{\left\| v^i_{\boldsymbol{\pi}} - v^{*i}_{\boldsymbol{\pi}^{-i}} \right\|_2}{\left\| v^{*i}_{\boldsymbol{\pi}^{-i}} \right\|_2}$. If this metric is close to zero for all players, the players reach a weak ϵ-Nash equilibrium with ϵ close to zero. Actually, *Error vs Best Response* is a normalized quantification of how sub-optimal the player's strategy is compared to his best response. 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 close to zero for all players, then they have little incentive to switch from their current strategy. It is an ϵ-Nash equilibrium. In addition, we keep track of the empirical norm of the Bellman residual on both the training dataset and the test dataset as it is our training objective.

Training parameters: We use N-player Garnet with 1, 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 a standard deviation $\hat{\sigma}_{s'}$ of 1. The reward function is $r^i(s) = \frac{2\min(|s-\hat{s}^i|, N_S) - |s-\hat{s}^i|)}{N_S}$ (it is a circular reward function). The reward sparsity m_{noise} is set to 0.5 and the reward white noise $\hat{\sigma}_{noise}$ has a standard deviation of 0.05. The discount factor γ is set to 0.9. The Q-networks and π-networks have one hidden layers of size 80 with RELUs [Goodfellow et al., 2016]. Q-networks have no output transfer function while π-networks have a softmax. The gradient descent is performed with AdamGrad [Goodfellow et al., 2016] with an initial learning rate of 1e-3 for the Q-network and 5e-5 for the π-networks. We use a weight decay of 1e-6. The training set is composed of $5N_SN_A$ samples split into random minibatch of size 20 while the testing dataset contains N_SN_A samples. The neural network is implemented by using the python framework Tensorflow [Abadi et al., 2015]. The source code is available on Github (Hidden for blind review) to run the experiments.

Results: Results for 1 player (MDP) and 5 players are reported in Figure 1. Additional settings are reported in the Appendix-C such as the two-player case and the tabular cases. In those scenarios, the quality of the learned strategies converges in parallel with the empirical Bellman residual. Once the training is over, the players can increase their cumulative reward by no more than 8% on average over the state space. Therefore, neural networks succeed in learning a weak ϵ-Nash equilibrium. Note that it is impossible to reach a zero error as (i) we are in the batch setting, (ii) we use function approximation and (iii) we only control a weak ϵ-Nash equilibrium. Moreover, the quality of the strategies 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. Furthermore, it can be adapted to a wide variety of problems by only changing the bottom of each network to fit with the state representation of the problem.

Discussions: The neural architecture faces some over-fitting issues. It requires a high number of samples to converge as described on Figure 2. [Lillicrap et al., 2016] introduces several tricks that may improve the training. Furthermore, we run additional experiments on non-deterministic Garnets but the result remains less conclusive. Indeed, the estimators of the Bellman residuals are biased for stochastic dynamics. As discussed, embedding the kernel or using kernel estimators may help to estimate properly the cost function (Equation (2)). Finally, our algorithm only seeks for a single Nash-Equilibrium when several equilibria might exist. Finding a specific equilibrium among others is out of the scope of this paper.

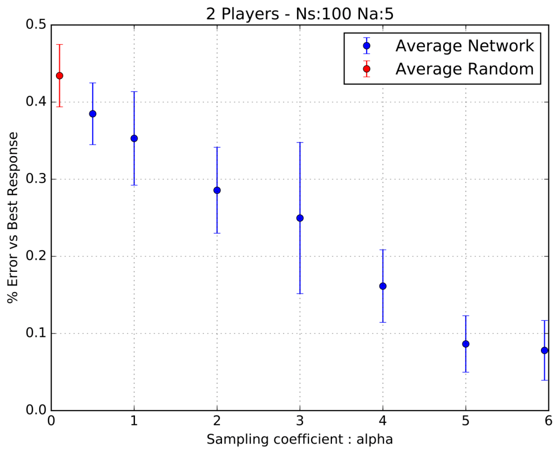


Figure 2: Impact of the number of samples on the quality of the learned strategy. The number of samples per batch is computed by $N_{samples} = \alpha N_A N_S$. Experiments are run on 3 different Garnets which are re-sampled 3 times to average the metrics.

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 ϵ-Nash equilibrium. We prove that minimizing the sum of different Bellman residuals is sufficient to learn a weak ϵ-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 does not rely on a specific MG. 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 as 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. Moreover, 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. And finally, future work could explore the use of projected Bellman residual as studied in MDPs and two-player zero-sum MGs [Pérolat et al., 2016]

Acknowledgments

This work has received partial funding from CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020, the European Commission H2020 framework program under the research Grant number 687831 (BabyRobot) and the Project CHIST-ERA IGLU.

References

[Abadi et al., 2015] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[Akchurina, 2010] Akchurina, N. (2010). *Multi-Agent Reinforcement Learning Algorithms*. PhD thesis, Paderborn, Univ., Diss., 2010.

[Archibald et al., 1995] Archibald, T., McKinnon, K., and Thomas, L. (1995). On the Generation of Markov Decision Processes. *Journal of the Operational Research Society*, 46:354–361.

[Baird et al., 1995] Baird, L. et al. (1995). Residual Algorithms: Reinforcement Learning with Function Approximation. In *Proc. of ICML*.

[Filar and Vrieze, 2012] Filar, J. and Vrieze, K. (2012). *Competitive Markov Decision Processes*. Springer Science & Business Media.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep Learning. Book in preparation for MIT Press.

[Grunewalder et al., 2012] Grunewalder, S., Lever, G., Baldassarre, L., Pontil, M., and Gretton, A. (2012). Modelling Transition Dynamics in MDPs With RKHS Embeddings. In *Proc. of ICML*.

[Hu and Wellman, 2003] Hu, J. and Wellman, M. P. (2003). Nash Q-Learning for General-Sum Stochastic Games. *Journal of Machine Learning Research*, 4:1039–1069.

[Lagoudakis and Parr, 2002] Lagoudakis, M. G. and Parr, R. (2002). Value Function Approximation in Zero-Sum Markov Games. In *Proc. of UAI*.

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

[LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep Learning. *Nature*, 521:436–444.

[Lillicrap et al., 2016] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2016). Continuous Control with Deep Reinforcement Learning. In *Proc. of ICLR*.

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

[Littman, 2001] Littman, M. L. (2001). Friend-or-Foe Q-Learning in General-Sum Games. In *Proc. of ICML*.

[Maillard et al., 2010] Maillard, O.-A., Munos, R., Lazaric, A., and Ghavamzadeh, M. (2010). Finite-Sample Analysis of Bellman Residual Minimization. In *Proc. of ACML*.

[Mnih et al., 2015] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-Level Control Through Deep Reinforcement Learning. *Nature*, 518:529–533.

[Munos and Szepesvári, 2008] Munos, R. and Szepesvári, C. (2008). Finite-Time Bounds for Fitted Value Iteration. *The Journal of Machine Learning Research*, 9:815–857.

[Nisan et al., 2007] Nisan, N., Roughgarden, T., Tardos, E., and Vazirani, V. V. (2007). *Algorithmic Game Theory*, volume 1. Cambridge University Press Cambridge.

[Pérolat et al., 2016] Pérolat, J., Piot, B., Geist, M., Scherrer, B., and Pietquin, O. (2016). Softened Approximate Policy Iteration for Markov Games. In *Proc. of ICML*.

[Perolat et al., 2016] 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 et al., 2015] 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 et al., 2014a] Piot, B., Geist, M., and Pietquin, O. (2014a). Boosted Bellman Residual Minimization Handling Expert Demonstrations. In *Proc. of ECML*.

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

[Prasad et al., 2015] Prasad, H., LA, P., and Bhatnagar, S. (2015). Two-Timescale Algorithms for Learning Nash Equilibria in General-Sum Stochastic Games. In *Proc. of AAMAS*.

[Puterman, 1994] Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons.

[Riedmiller, 2005] Riedmiller, M. (2005). Neural Fitted Q Iteration–First Experiences with a Data Efficient Neural Reinforcement Learning Method. In *Proc. of ECML*.

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

[Shapley, 1953] Shapley, L. S. (1953). Stochastic Games. *In Proc. of the National Academy of Sciences of the United States of America*.

[Taylor and Parr, 2012] Taylor, G. and Parr, R. (2012). Value Function Approximation in Noisy Environments Using Locally Smoothed Regularized Approximate Linear Programs. In *Proc. of UAI*.

[Zinkevich et al., 2006] Zinkevich, M., Greenwald, A., and Littman, M. (2006). Cyclic Equilibria in Markov Games. In *Proc. of NIPS*.

A Proof of the equivalence of definition 1 and 2

Proof of the equivalence between definition 2 and 1.

(2) $\Rightarrow$ (1):

If $\exists \boldsymbol{v}$ such as $\forall i \in \{1, ..., N\}, \mathcal{T}^i_{\boldsymbol{\pi}} v^i = v^i$ and $\mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}} v^i = v^i$, then $\forall i \in \{1, ..., N\}$, $v^i = v^i_{\pi^i, \boldsymbol{\pi}^{-i}}$ and $v^i = \max\limits_{\tilde{\pi}^i} v^i_{\tilde{\pi}^i, \boldsymbol{\pi}^{-i}}$

(1) $\Rightarrow$ (2):

if $\forall i \in \{1, ..., N\}$, $v^i_{\pi^i, \boldsymbol{\pi}^{-i}} = \max\limits_{\tilde{\pi}^i} v^i_{\tilde{\pi}^i, \boldsymbol{\pi}^{-i}}$., then $\forall i \in \{1, ..., N\}$, the value $v^i = v^i_{\pi^i, \boldsymbol{\pi}^{-i}}$ is such as $\mathcal{T}^i_{\boldsymbol{\pi}} v^i = v^i$ and $\mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}} v^i = v^i$

B Proof of Theorem 1

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 p and p' be a real numbers such that $\frac{1}{p} + \frac{1}{p'} = 1$, then $\forall \mathbf{v}, \boldsymbol{\pi}$ and $\forall i \in \{1, ..., N\}$:*

$$\begin{aligned}
&\left\| v^i_{\pi^i_*, \boldsymbol{\pi}^{-i}} - v^i_{\pi^i, \boldsymbol{\pi}^{-i}} \right\|_{\mu,p} \\
&\leq \frac{1}{1-\gamma} \left(C_\infty(\mu, \nu, \pi^i_*, \boldsymbol{\pi}^{-i})^{\frac{p'}{p}} + C_\infty(\mu, \nu, \pi^i, \boldsymbol{\pi}^{-i})^{\frac{p'}{p}} \right)^{\frac{1}{p'}} \left[\left\| \mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}} v^i - v^i \right\|^p_{\mu,p} + \left\| \mathcal{T}^i_{\boldsymbol{\pi}} v^i - v^i \right\|^p_{\mu,p} \right]^{\frac{1}{p}},
\end{aligned}$$

where π^i_* is the best response to $\boldsymbol{\pi}^{-i}$. Meaning $v^i_{\pi^i_*, \boldsymbol{\pi}^{-i}}$ is the fixed point of $\mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}}$. And with the following concentrability coefficient $C_\infty(\mu, \nu, \pi^i, \boldsymbol{\pi}^{-i}) = \left\| \frac{\partial \mu^T (1-\gamma)(\mathcal{I} - \gamma \mathcal{P}_{\pi^i, \boldsymbol{\pi}^{-i}})^{-1}}{\partial \nu^T} \right\|_{\nu,\infty}$.

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

$$\begin{aligned}
v^i_{\pi^i, \boldsymbol{\pi}^{-i}} - v^i &= (\mathcal{I} - \gamma \mathcal{P}_{\pi^i, \boldsymbol{\pi}^{-i}})^{-1} (r^i_{\pi^i, \boldsymbol{\pi}^{-i}} - (\mathcal{I} - \gamma \mathcal{P}_{\pi^i, \boldsymbol{\pi}^{-i}}) v^i), \\
&= (\mathcal{I} - \gamma \mathcal{P}_{\pi^i, \boldsymbol{\pi}^{-i}})^{-1} (\mathcal{T}^i_{\pi^i, \boldsymbol{\pi}^{-i}} v^i - v^i).
\end{aligned}$$

But we also have:

$$v^i_{\pi^i_*, \boldsymbol{\pi}^{-i}} - v^i = (\mathcal{I} - \gamma \mathcal{P}_{\pi^i_*, \boldsymbol{\pi}^{-i}})^{-1} (\mathcal{T}^i_{\pi^i_*, \boldsymbol{\pi}^{-i}} v^i - v^i),$$

then:

$$\begin{aligned}
v^i_{\pi^i_*, \boldsymbol{\pi}^{-i}} - v^i_{\pi^i, \boldsymbol{\pi}^{-i}} &= v^i_{\pi^i_*, \boldsymbol{\pi}^{-i}} - v^i + v^i - v^i_{\pi^i, \boldsymbol{\pi}^{-i}}, \\
&= (\mathcal{I} - \gamma \mathcal{P}_{\pi^i_*, \boldsymbol{\pi}^{-i}})^{-1} (\mathcal{T}^i_{\pi^i_*, \boldsymbol{\pi}^{-i}} v^i - v^i) - (\mathcal{I} - \gamma \mathcal{P}_{\pi^i, \boldsymbol{\pi}^{-i}})^{-1} (\mathcal{T}^i_{\pi^i, \boldsymbol{\pi}^{-i}} v^i - v^i), \\
&\leq (\mathcal{I} - \gamma \mathcal{P}_{\pi^i_*, \boldsymbol{\pi}^{-i}})^{-1} (\mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}} v^i - v^i) - (\mathcal{I} - \gamma \mathcal{P}_{\pi^i, \boldsymbol{\pi}^{-i}})^{-1} (\mathcal{T}^i_{\pi^i, \boldsymbol{\pi}^{-i}} v^i - v^i), \\
&\leq (\mathcal{I} - \gamma \mathcal{P}_{\pi^i_*, \boldsymbol{\pi}^{-i}})^{-1} \left| \mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}} v^i - v^i \right| + (\mathcal{I} - \gamma \mathcal{P}_{\pi^i, \boldsymbol{\pi}^{-i}})^{-1} \left| \mathcal{T}^i_{\pi^i, \boldsymbol{\pi}^{-i}} v^i - v^i \right|.
\end{aligned}$$

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

$$\begin{aligned}
&\left\| v^i_{\pi^i_*, \boldsymbol{\pi}^{-i}} - v^i_{\pi^i, \boldsymbol{\pi}^{-i}} \right\|_{\mu,p} \\
&\leq \left\| (\mathcal{I} - \gamma \mathcal{P}_{\pi^i_*, \boldsymbol{\pi}^{-i}})^{-1} \left| \mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}} v^i - v^i \right| \right\|_{\mu,p} + \left\| (\mathcal{I} - \gamma \mathcal{P}_{\pi^i, \boldsymbol{\pi}^{-i}})^{-1} \left| \mathcal{T}^i_{\pi^i, \boldsymbol{\pi}^{-i}} v^i - v^i \right| \right\|_{\mu,p}, \\
&\leq \frac{1}{1-\gamma} \left[C_\infty(\mu, \nu, \pi^i_*, \boldsymbol{\pi}^{-i})^{\frac{1}{p}} \left\| \mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}} v^i - v^i \right\|_{\nu,p} + C_\infty(\mu, \nu, \pi^i, \boldsymbol{\pi}^{-i})^{\frac{1}{p}} \left\| \mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}} v^i - v^i \right\|_{\nu,p} \right], \\
&\leq \frac{1}{1-\gamma} \left(C_\infty(\mu, \nu, \pi^i_*, \boldsymbol{\pi}^{-i})^{\frac{p'}{p}} + C_\infty(\mu, \nu, \pi^i, \boldsymbol{\pi}^{-i})^{\frac{p'}{p}} \right)^{\frac{1}{p'}} \left[\left\| \mathcal{T}^{*i}_{\boldsymbol{\pi}^{-i}} v^i - v^i \right\|^p_{\nu,p} + \left\| \mathcal{T}^i_{\boldsymbol{\pi}} v^i - v^i \right\|^p_{\nu,p} \right]^{\frac{1}{p}}.
\end{aligned}$$

□

Theorem 1 falls in two steps:

$$\left\| \left\| \max_{\tilde{\pi}^i} v_{\tilde{\pi}^i, \boldsymbol{\pi}^{\text{-}\boldsymbol{i}}} - v^i_{\pi} \right\|_{\mu(s),p} \right\|_{\rho(i),p} \leq \frac{1}{1-\gamma} \left[\max_{i \in \{1,\dots,N\}} \left(C_\infty(\mu,\nu,\pi^i_*, \boldsymbol{\pi}^{\text{-}\boldsymbol{i}})^{\frac{p'}{p}} + C_\infty(\mu,\nu,\pi^i, \boldsymbol{\pi}^{\text{-}\boldsymbol{i}})^{\frac{p'}{p}} \right)^{\frac{1}{p'}} \right]$$

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

$$\leq \frac{2^{\frac{1}{p'}} C_\infty(\mu,\nu)^{\frac{1}{p}}}{1-\gamma} \left[\sum_{i=1}^N \rho(i) \left(\left\| \mathcal{T}^{*i}_{\boldsymbol{\pi}^{\text{-}\boldsymbol{i}}} v^i - v^i \right\|^p_{\nu,p} + \left\| \mathcal{T}^i_{\boldsymbol{\pi}} v^i - v^i \right\|^p_{\nu,p} \right) \right]^{\frac{1}{p}},$$

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

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

C Additional curves

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

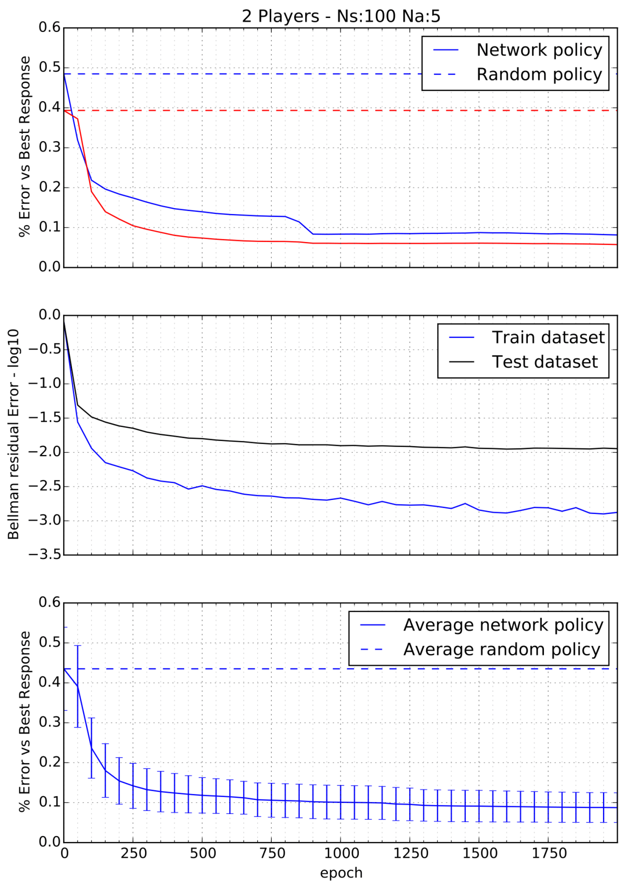


Figure 3: (top) Evolution of the Error vs Best Response during the training for a given Garnet. (middle) Empirical Bellman residual for the training dataset and the testing dataset. (bottom) Average Error vs Best Response averaged over every players, Garnets and every batch. Experiments are run on 5 Garnets which is re-sampled 5 times to average the metrics.

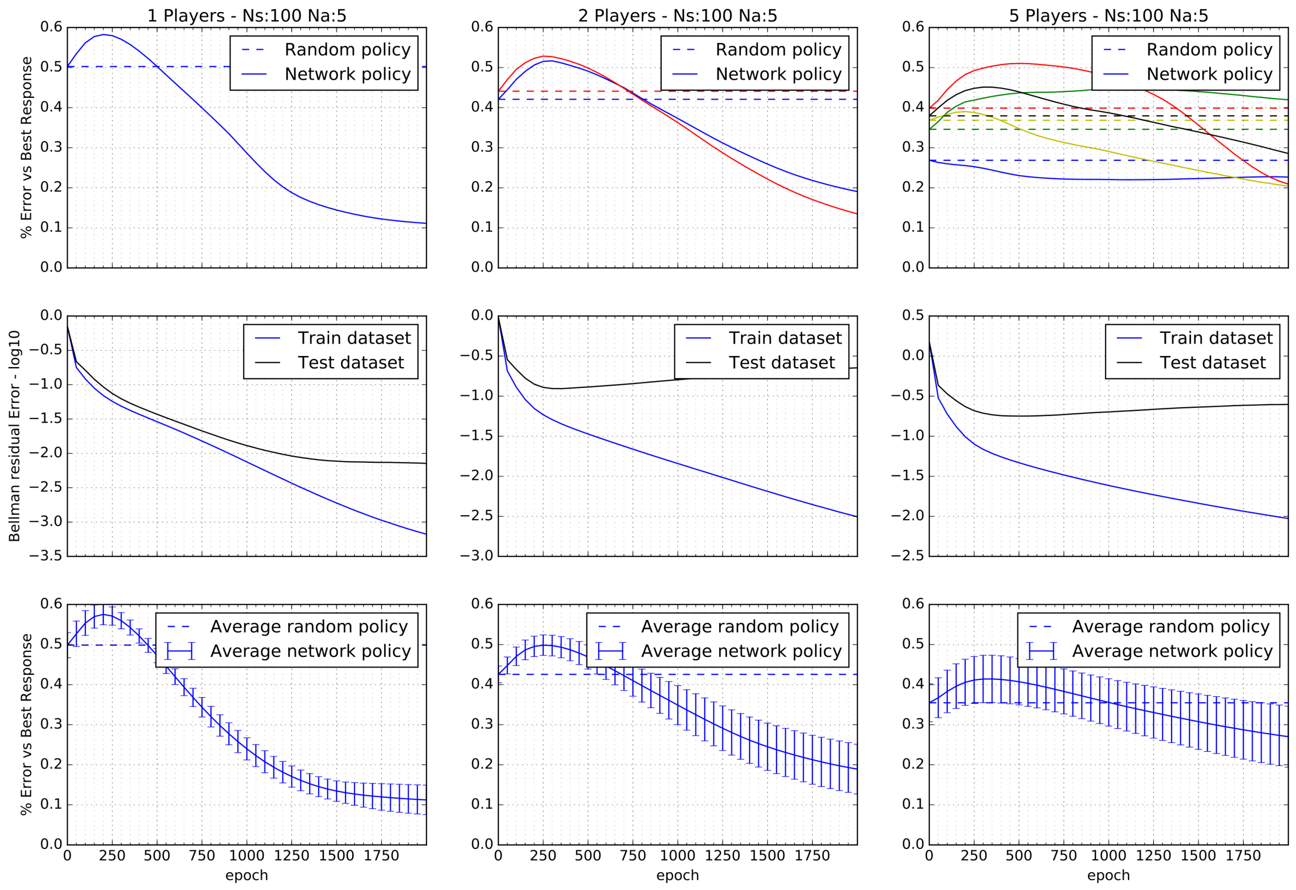


Figure 4: Tabular basis. One may notice that the tabular case works well for a 1 player game (MDP). Yet, the more players there are, the worth it performs. Experiments are run on 5 Garnets which is re-sampled 5 times to average the metrics.

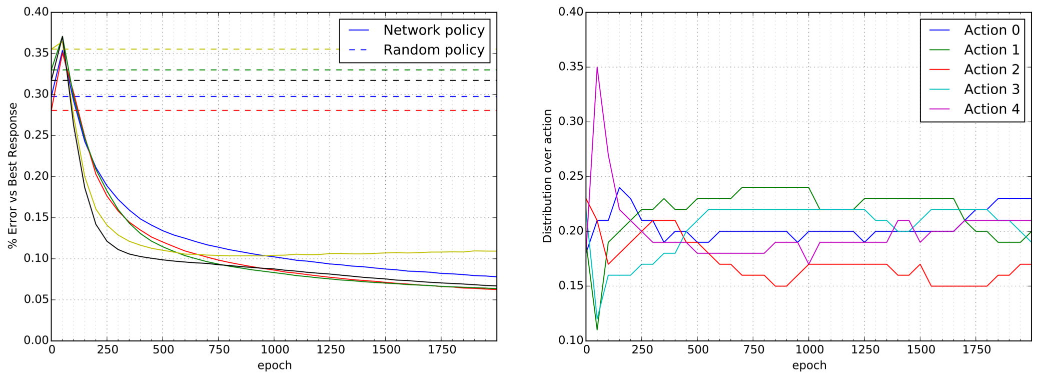
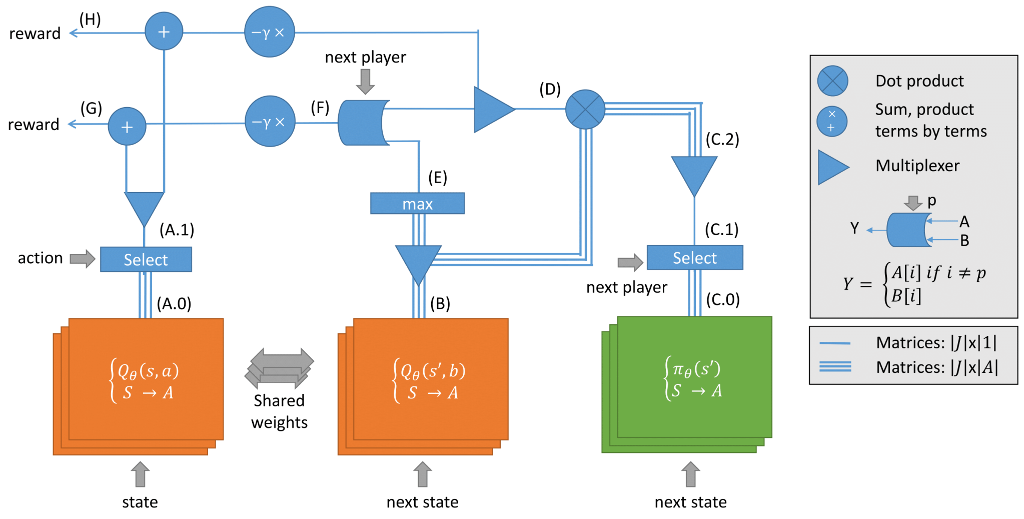


Figure 5: (left)Evolution of the Error vs Best Response during the training for a given Garnet. When the Garnet has a complex structure or the batch is badly distributed, one player sometimes fails to learn a good strategy. (right) Distribution of actions in the strategy among the players with the highest probability. This plots highlights that π-networks do modify the strategy during the training



reward
(H)
reward
(G)
$-\gamma\times$
$-\gamma\times$
next player
(F)
(D)
(E)
max
(C.2)
(A.1)
(C.1)
action
Select
Select
(A.0)
(B)
next player
(C.0)
$\{Q_\theta(s,a)$ $S \to A$
Shared weights
$\{Q_\theta(s',b)$ $S \to A$
$\{\pi_\theta(s')$ $S \to A$
state
next state
next state
Dot product
Sum, product terms by terms
Multiplexer
p
Y
A
B
$Y = \begin{cases} A[i] \text{ if } i \neq p \\ B[i] \end{cases}$
Matrices: $|J|\text{x}|1|$
Matrices: $|J|\text{x}|A|$

Figure 6: 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 state-action value function and a π-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 $(s,(a^1,\dots,a^N),(r^1,\dots,r^N),s')$. The equation (2) can be divided into key operations that are described below. For each player i : first of all, in **(A.0)** we compute $Q^i(s,a^1,\dots,a^N)$ by computing the tensor $Q^i(s,\boldsymbol{a})$ and then by selecting the value of $Q^i(s,\boldsymbol{a})$ **(A.1)** given the action of the batch. Step **(B)** computes the tensor $Q^i(s',\boldsymbol{b})$ and step **(C.0)** computes the strategy $\pi^i(.|s')$. In all Bellman residuals we need to average over the strategy of players $Q^i(s',\boldsymbol{b})$. Since we focus on turn-based MGs, we will only average over the strategy of the player controlling next state s' (in the following, this player is called the next player). In **(C.1)** we select the strategy $\pi(.|s')$ of the next player given the batch. In **(C.2)** we duplicate the strategy of the next player for all other players. In **(D)** we compute the dot product between the $Q^i(s',\boldsymbol{b})$ and the strategy of the next player to obtain $E_{\mathbf{b}\sim\boldsymbol{\pi}}[Q^i(s',\mathbf{b})]$ and in **(E)** we pick the highest expected rewards and obtain $\max_{\boldsymbol{b}} Q^i(s',\boldsymbol{b})$. Step **(F)** aims at computing $\max_{b^i}\left[E_{\boldsymbol{b}^{\text{-}i}\sim\boldsymbol{\pi}^{\text{-}i}}[Q^i(s',b^i,\boldsymbol{b}^{\text{-}\boldsymbol{i}})]\right]$ and, since we deal with a turn-based MG, we need to select between the output of **(D)** or **(E)** according to the next player. For all i, we either select the one coming from **(E)** if the next player is i or the one from **(D)** otherwise. In **(G)** we compute the error between the reward r^i to $Q^i(s,\mathbf{a})-\gamma\max_{b^i}\left[E_{\boldsymbol{b}^{\text{-}i}\sim\boldsymbol{\pi}^{\text{-}i}}[Q^i(s',b^i,\boldsymbol{b}^{\text{-}\boldsymbol{i}})]\right]$ and in **(H)** between r^i and $\gamma E_{\mathbf{b}\sim\boldsymbol{\pi}}[Q^i(s',\mathbf{b})]-Q^i(s,\mathbf{a})$. The final loss is the sum of all the residuals.