Is Nash Equilibrium Approximator Learnable?

Zhijian Duan
CFCS, School of Computer Science,
Peking University
Beijing, China
zjduan@pku.edu.cn

Wenhan Huang
Shanghai Jiao Tong University
Shanghai, China
rowdark@sjtu.edu.cn

Dinghuai Zhang
Mila - Quebec AI Institute
Montreal, Canada
dinghuai.zhang@mila.quebec

Yali Du
King's College London
London, United Kingdom
yali.du@kcl.ac.uk

Jun Wang
University College London
London, United Kingdom
jun.wang@cs.ucl.ac.uk

Yaodong Yang
CMAR, Institute for AI,
Peking University
Beijing, China
yaodong.yang@pku.edu.cn

Xiaotie Deng
CFCS, School of Computer Science,
CMAR, Institute for AI,
Peking University
Beijing, China
xiaotie@pku.edu.cn

ABSTRACT

In this paper, we investigate the learnability of the function approximator that approximates Nash equilibrium (NE) for games generated from a distribution. First, we offer a generalization bound using the Probably Approximately Correct (PAC) learning model. The bound describes the gap between the expected loss and empirical loss of the NE approximator. Afterward, we prove the agnostic PAC learnability of the Nash approximator. In addition to theoretical analysis, we demonstrate an application of NE approximator in experiments. The trained NE approximator can be used to warm-start and accelerate classical NE solvers. Together, our results show the practicability of approximating NE through function approximation.

CCS CONCEPTS

• **Theory of computation → Exact and approximate computation of equilibria**; **Sample complexity and generalization bounds**; **Solution concepts in game theory**.

KEYWORDS

Game Theory; Normal-Form Games; Nash Equilibrium; Function Approximation; Generalization Bound; PAC Learnability

ACM Reference Format:
Zhijian Duan, Wenhan Huang, Dinghuai Zhang, Yali Du, Jun Wang, Yaodong Yang, and Xiaotie Deng. 2023. Is Nash Equilibrium Approximator Learnable? . In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 9 pages.

1 INTRODUCTION

Nash equilibrium (NE) [49], in which each agent's strategy is optimal given the strategies of all other agents, is one of the most important solution concepts in game theory. It can be used to analyze the outcome of strategic interactions among rational agents. An NE or ϵ-approximate Nash equilibrium (ϵ-NE) strategy can also be a good guide for agents in the game since agents have no or negligible incentive to disobey individually. There has been increasing interest in NE due to its broad applications in Generative Adversarial Networks (GAN) [28], Multi-Agent Reinforcement Learning (MARL) [61], multi-agent systems [56], economics [19, 20], and online advertising [18]. Although NE always exists in normal-form games [49], finding an NE is PPAD-complete even for 2-player games [9] and 3-player games [14]. Such negative results lead to increased attention on developing algorithms to approximate NE.

While many algorithms were proposed to find ϵ-NE for some approximation $\epsilon > 0$ [7, 12, 15–17, 37, 38, 58], these works focus on solving a single game in isolation. However, many similar games usually need to be solved in practice or in some multi-agent learning algorithms. For instance, in repeated contextual games such as traffic routing [54], the utility function depends on contextual information generated from a distribution. The Nash Q-learning [34] algorithm, which solves Markov games via value-based reinforcement learning, needs to compute NE for a normal-form game every time it updates the Q-value. In these settings, traditional solvers have to compute from scratch for every game, ignoring the similarity among those games. As an improvement, it can be preferable to construct a function approximator that predicts NE from game utility [26, 46]. The NE approximator is trained through the historical data and can provide an approximate solution quickly at the test time.

Several critical theoretical issues arise in developing algorithms to predict NE from samples. First, the NE approximator is learned

from training data and will be evaluated by unseen games in testing. Therefore, its generalization ability, i.e., its performance in testing, needs to be clarified. Moreover, people also care about the sample complexity (how many training samples we need) to get a reasonable approximator.

In this paper, we make the first step to study the learnability of predicting NE by function approximation. We consider general n-player normal-form games with fixed action space. We follow the standard Probably Approximately Correct (PAC) learning [31, 59] model, in which game utilities are independently generated from an identical distribution, both in training and testing. One challenge is the non-uniqueness issue of exact NE, which brings difficulty for naively adopting supervised learning techniques. Inspired by the definition of ϵ-NE, we set up a self-supervised loss function to evaluate the performance of an NE approximator. Such Nash approximation loss is Lipschitz continuous to game utility and players' strategies. Based on that, we present a generalization bound for any NE approximators. The bound provides a confidence interval on the expected loss based on the empirical loss in training. Furthermore, based on a mild assumption of the NE approximator function class, we prove that it is agnostic PAC learnable to predict NE from samples. To the best of our knowledge, this is the first result that addresses the PAC learnability of Nash equilibrium.

In addition to the theoretical analysis, we demonstrate a practical application of the learned NE approximator. We show that it can warm-start other classic approximate NE solvers. By doing so, we combine both advantages of the function approximation method and the traditional approach. The former helps to provide an effective initial solution in batches with low computational costs, and the latter provides theoretical guarantees. Specifically, we conduct numerical experiments in bimatrix games. We train a neural network-based NE approximator and use the predicted solutions as the pre-solving initialization for the algorithm of Tsaknakis and Spirakis [58] and the start-of-the-art approximate NE solver proposed by Deligkas et al. [17]. In both cases, we report faster convergence.

Our paper is organized as follows: In Section 2 we describe related works; In Section 3 we introduce the preliminary of game theory; In Section 4 we set up the PAC learning framework for predicting NE from samples; We present our learnability results in Section 5; We conduct numerical experiments and demonstrate the application in Section 6; We draw our conclusion in Section 7.

2 RELATED WORK

Classic solvers with feasibility guarantee. For 2-player games, there are algorithms with a theoretical guarantee for maximum Nash approximation loss (See definition in Equation (1)). Kontogiannis et al. [37] and Daskalakis et al. [16] introduced simple polynomial-time algorithms based on searching small supports to reach an approximation loss of $3/4$ and $1/2$, respectively. Daskalakis et al. [15] provided an algorithm of approximation loss 0.38 by enumerating arbitrarily large supports, and this approximation loss is also achieved by Czumaj et al. [12] with a different approach. Bosse et al. [7] proposed an algorithm based on Kontogiannis and Spirakis [38] to reach an approximation loss of 0.36. TS algorithm [58] achieves an approximation loss of 0.3393, and Chen et al. [10] proved that the bound is tight. Recently, DFM algorithm [17], an improved version of Tsaknakis and Spirakis [58], establishes the best currently known approximation loss of $1/3$. However, computing approximate NE for even arbitrary constant approximation is PPAD-hard [13].

Learning approaches. Learning is another paradigm to compute approximate NE by repeatedly proposing temporal strategies and updating them with feedback rewards. Fictitious play [48] is the most well-known learning-based algorithm to approximate NE, and Conitzer [11] proves that it reaches an approximate loss of $1/2$ when given constant rounds. Double Oracle methods [21, 47] and PSRO methods [40, 51], though effective, target solving zero-sum games only. Online learning methods, including regret matching [30], Hedge [4] and Multiplicative weight update [3], are proved to converge to (approximate) coarse correlated equilibrium [8].

Data-driven approaches. In addition to traditional methods, many works have proposed to approximate NE through data-driven approaches. Some of them make use of the historical game-playing data and learn the game utility functions [1, 6, 62] or game gradients [32, 42, 43] from the observed (approximate) NE. By doing so, they can predict approximate NE solutions for a class of games (e.g., contextual games [32, 54]). Another way is to learn a function approximator that maps game utility to an approximate solution [46]. Such NE approximator can be applied in PSRO [26]. Recently, Harris et al. [29] introduce meta-learning algorithms for equilibrium finding. In our paper, we study the generalization ability of the NE approximator and the PAC learnability of NE.

Learnability. As for learnability analysis in games, Viqueira et al. [60] and Marchesi et al. [45] provide the PAC analysis of learning the game utility in simulation-based games, in which the utility is obtained by query and would potentially be disturbed by noise. A *Nash Oracle*, which can output the exact NE for arbitrary games directly, is assumed in these papers. Similarly, Fele and Margellos [25] considers games with noisy utilities and studies the learnability of NE, given the strong assumption of Nash Oracle. As a comparison, we do not assume any Nash Oracles in our paper. Some other works consider query complexity to approximate NE [22, 24], while we focus on the sample complexity to learn a generalizable NE approximator. Moreover, while Jin et al. [36] and Bai et al. [5] propose PAC learnable algorithm to approximate NE in a zero-sum Markov game, and to approximate Coarse Correlated Equilibria (CCE) or Correlated Equilibria (CE) in a general-sum Markov game, we must highlight the difference that we consider the PAC analysis of NE in *general-sum* games sampled from a same *arbitrary distribution*, instead of approximating NE for one specific game instance.

3 GAME THEORY PRELIMINARIES

Normal-form games. We denote a normal-form game with joint utility function u as $\Gamma_u = (N, A, u)$ and explain each item as follows.

- $N = \{1, 2, \ldots, n\}$ is the set of all the n players. Each player is represented by the index $i \in N$.
- $A = A_1 \times A_2 \times \cdots \times A_n$ is the combinatorial action space of all players, in which A_i is the action space for player i. For player $i \in N$, let $a_i \in A_i$ be a specific action and $|A_i|$

be the number of actions (An action is also referred to as a pure strategy). An action profile $a = (a_1, a_2, \ldots, a_n) \in A$ represents one play of the game in which the player i takes her corresponding action $a_i \in A_i$. The action space A is a Cartesian product that contains all possible action profiles. Therefore, we have $|A| = \prod_{i \in N} |A_i|$.

- $u = (u_1, \ldots, u_n)$ is the game utility (payoff), in which $u_i : A \to \mathbb{R}$ is the utility function (or utility matrix, equivalently) for player i. u_i describes the utility of player i on each possible action profile $a = (a_1, a_2, \ldots, a_n) \in A$. We have $|u_i| = |A|$ and $|u| = n|A|$. In our paper, we assume each utility is in the range of $[0, 1]$ without loss of generality. Such an assumption is widely-used in previous literatures [17, 58].

A mixed strategy of player i, denoted by σ_i, is a distribution over her action set A_i. Specifically, $\sigma_i(a_i)$ represents the probability that player i chooses action a_i. Under such definition, we have $\sum_{a_i \in A_i} \sigma_i(a_i) = 1$. Denote $\Delta A_i \ni \sigma_i$ be the set of all the possible mixed strategies for player i. A mixed strategy profile $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ is a joint strategy for all the players. Based on σ, the probability of action profile $a = (a_1, a_2, \ldots, a_n)$ being played is $\sigma(a) := \prod_{i \in N} \sigma_i(a_i)$. Notice that an action profile a (i.e., a pure strategy profile) can also be seen as a mixed strategy profile σ with $\sigma_i(a_i) = 1$ for all $i \in N$. The expected utility of player i under σ is

$$u_i(\sigma) = \mathbb{E}_{a \sim \sigma}[u_i(a)] = \sum_{a \in A} \sigma(a) u_i(a).$$

Besides, on behalf of player i, the other players' strategy profile is denoted as $\sigma_{-i} = (\sigma_1, \ldots, \sigma_{i-1}, \sigma_{i+1}, \ldots, \sigma_n)$.

(ϵ-approximate) Nash equilibrium. Nash equilibrium is one of the most important solution concepts in game theory. A (mixed) strategy profile $\sigma^* = (\sigma_1^*, \sigma_2^*, \ldots, \sigma_n^*)$ is called a Nash equilibrium if and only if for each player $i \in N$, her strategy is the best response given the strategies σ_{-i}^* of all the other players. Formally,

$$u_i(\sigma_i, \sigma_{-i}^*) \le u_i(\sigma_i^*, \sigma_{-i}^*), \quad \forall i \in N, \sigma_i \in \Delta A_i \qquad \text{(NE)}$$

However, computing NE for even general 2-player or 3-player games is PPAD-hard [9, 14]. Given such hardness, many works focus on finding approximate solutions. For arbitrary $\epsilon > 0$, we say a strategy profile $\hat{\sigma}$ is an *ϵ-approximate Nash equilibrium* (ϵ-NE) if no one can achieve more than ϵ utility gain by deviating from her current strategy. Formally,

$$u_i(\sigma_i, \hat{\sigma}_{-i}) \le u_i(\hat{\sigma}_i, \hat{\sigma}_{-i}) + \epsilon, \quad \forall i \in N, \sigma_i \in \Delta A_i \qquad (\epsilon\text{-NE})$$

The definition of ϵ-NE reflects the idea that players might not be willing to deviate from their strategies when the amount of utility they could gain by doing so is tiny (not more than ϵ).

4 LEARNING FRAMEWORK

In this section, we set up the PAC learning framework of predicting NE in n-player normal-form games with fixed players and fixed action space. The learning framework includes a domain set $\mathcal{U}$, a game-generation distribution $\mathcal{D}$, a hypothesis class $\mathcal{H}$ of the NE approximator, a training set S, and evaluation metrics to evaluate the performance of any NE approximators.

Domain set is defined as the set of all the possible input games. In our paper, the domain set $\mathcal{U}$ includes all the possible game utilities given the fixed players and action space. Following the standard PAC learning paradigm, we assume each game utility $u \in \mathcal{U}$ is sampled independent and identically from a game-generation distribution $\mathcal{D}$ with $\text{supp}(\mathcal{D}) \subseteq \mathcal{U}$. The generated games may belong to a specific game class (e.g., symmetric games). We make no assumption about $\mathcal{D}$. The learner does not know the exact form of $\mathcal{D}$, but she can access the generated samples.

Table 1: An example illustrating the non-uniqueness issue of exact NE, in which $A_1 = \{L, R\}$ and $A_2 = \{U, D\}$. Each element (x, y) in the table represents $u_1(\cdot,\cdot) = x$ and $u_2(\cdot,\cdot) = y$ for the corresponding joint action profile. There are two pure NE (bolded) and one mixed NE in the example.

	U	D
L	$(0, 0)$	$\mathbf{(1, 0.5)}$
R	$\mathbf{(0.5, 1)}$	$(0, 0)$

The learner should choose in advance (before seeing the data) a class of functions $\mathcal{H}$, where each function $h : \mathcal{U} \to \Delta A_1 \times \Delta A_2 \times \cdots \times \Delta A_n$ in $\mathcal{H}$ maps a game utility to a joint strategy of n players. We call such function class $\mathcal{H}$ the *hypothesis class*. In our paper, we consider hypothesis classes with infinite size. We will describe how we measure the capacity of $\mathcal{H}$ in Section 5. During learning, a *training set* S of size m is provided to the learner. $S = \{u^{(1)}, u^{(2)}, \ldots, u^{(m)}\}$ contains m game utilities drawn i.i.d. from domain set $\mathcal{U}$ according to $\mathcal{D}$.

One challenge for learning to predict NE is the *non-uniqueness issue*: There may be multiple NEs for a game (See Table 1 for an illustration). Such an issue brings trouble for applying supervised learning. The equilibrium selection problem is nontrivial and many works made some assumptions to ensure the uniqueness of NE [6, 32, 41, 62]. To deal with the issue, we use Nash approximation loss to measure the level of approximation of a mixed strategy to NE. The metrics is widely used in the literature for approximating NE[17, 58]. Nash approximation is defined as follows [1]:

Definition 4.1 (Nash approximation, NashApr)**.** For normal-form game $\Gamma_u = (N, A, u)$, the Nash approximation loss of strategy profile σ with respect to game utility u is the maximum utility gain each player can obtain by deviating from her strategy. Formally,

$$\begin{aligned} \text{NashApr}(\sigma, u) &:= \max_{i \in N} \max_{\sigma_i' \in \Delta A_i} [u_i(\sigma_i', \sigma_{-i}) - u_i(\sigma_i, \sigma_{-i})] \\ &= \max_{i \in N} \max_{a_i \in A_i} [u_i(a_i, \sigma_{-i}) - u_i(\sigma_i, \sigma_{-i})]. \end{aligned} \tag{1}$$

The computation of $\text{NashApr}(\sigma, u)$ only involves σ and u. Thus we do not need any NE or side information. Besides, as we will discuss in Section 5, the Nash approximation loss is Lipschitz continuous with respect to both inputs, which helps to derive our results.

For finite $\mathcal{H}$, it is trivial to provide a PAC learnable result [55]. The learning algorithm $\mathcal{A} : \mathcal{U}^m \to \mathcal{H}$ aims to learn a good NE approximator $h \in \mathcal{H}$ from the training data S, aiming to minimize

[1] Another similar concept is called Nash exploitability [44]: $\text{NashExpl}_i(\sigma, u) := \max_{\sigma_i' \in \Delta A_i} u_i(\sigma_i', \sigma_{-i}) - u_i(\sigma)$.

Algorithm 1 NE approximator Learning via minibatch SGD

Input: Training set S of size m
Parameters: Number of iterations $T > 0$, batch size $B > 0$, learning rate $\eta > 0$, initial parameters $w_0 \in \mathbb{R}^d$ of the NE approximator model.
for $t = 0$ **to** T **do**
  Receive minibatch $S_t = \{u^{(1)}, \ldots, u^{(B)}\} \subset S$
  Compute the empirical average loss of S_t:
  $L_{S_t}(h^{w_t}) \leftarrow \frac{1}{B}\sum_{i=1}^{B} \textsc{NashApr}(h^{w_t}(u^{(i)}), u^{(i)})$
  Update model parameters:
  $w_{t+1} \leftarrow w_t - \eta \nabla_{w_t} L_{S_t}(h^{w_t})$
end for

the *true risk* $L_{\mathcal{D}}(h)$ of using h. The true risk is the expected Nash approximation of h under distribution $\mathcal{D}$:

$$L_{\mathcal{D}}(h) := \mathbb{E}_{u\sim\mathcal{D}}\Big[\textsc{NashApr}(h(u), u)\Big], \tag{2}$$

We also define the *empirical risk* $L_S(h)$ on the data set S as:

$$L_S(h) := \frac{1}{|S|}\sum_{u\in S} \textsc{NashApr}(h(u), u) \tag{3}$$

Given enough samples, the true risk can be estimated by the empirical risk (See Theorem 5.7). Therefore, *empirical risk minimization* (ERM) can be applied to learn an NE approximator h from hypothesis class $\mathcal{H}$:

$$\text{ERM}_{\mathcal{H}}(S) \in \arg\min_{h\in\mathcal{H}} L_S(h) \tag{4}$$

However, in practice, it is usually intractable to implement the ERM algorithm, especially when $\mathcal{H}$ is infinite. Following the standard approach in deep learning community [27], we can approximate ERM by *minibatch Stochastic Gradient Descent* (minibatch SGD). Specifically, we parameterize the NE approximator as h^w with d-dimensional parameter variable $w \in \mathbb{R}^d$ (e.g., the weights of a neural network). We optimize w by the standard minibatch SGD algorithm (See Algorithm 1). This method is feasible since $\textsc{NashApr}(\sigma, u)$ is differentiable almost everywhere, except for some minor points on a zero-measure set. Those minor points appear when one of the two maximum operations in $\textsc{NashApr}(\sigma, u)$ has multiple maximum inputs. We can set one of them according to any tie-breaking rule as the outcome to compute the corresponding gradient.

We must emphasize that the empirical risk minimization algorithm will only be used in the PAC learnability analysis in Section 5.3. Moreover, Algorithm 1 will only be used to demonstrate the application of NE approximator in Section 6. The generalization bound in Section 5.2 is unrelated to the learning algorithm we apply. Instead, we can use any learning algorithms such as the approach in Marris et al. [46] to obtain the NE approximator, and the generalization bound still holds.

5 THEORETICAL LEARNABILITY RESULTS

In this section, we present our theoretical learnability result for predicting NE from samples. We first analyze the Lipschitz property of Nash approximation loss. Based on that, we provide a generalization bound for the NE approximator. We further show that Nash equilibrium is agnostic PAC learnable under a mild assumption on $\mathcal{H}$. All the omitted proofs are presented in Appendix.

Table 2: An example illustrating the non-smooth issue of exact NE, in which $A_1 = \{L, R\}$ and $A_2 = \{U, D\}$. Each element (x, y) in the table represents $u_1(\cdot,\cdot) = x$ and $u_2(\cdot,\cdot) = y$ for the corresponding joint action. Minor changes in the utility of game Γ_u (into game Γ_v) can cause different exact NE solutions. (a): Game Γ_u. The unique NE is (L, U). (b): Game Γ_v. The unique NE is (R, U). The only difference between Γ_u and Γ_v is the utility $u_1(R, U)$, which only differs by arbitrary small 2ϵ.

(a)

	U	D
L	$\mathbf{(0.5, 0.5)}$	$(1, 0)$
R	$(0.5-\epsilon, 1)$	$(0, 0)$

(b)

	U	D
L	$(0.5, 0.5)$	$(1, 0)$
R	$\mathbf{(0.5+\epsilon, 1)}$	$(0, 0)$

5.1 Lipschitz Property of Nash Approximation

We start with deriving the Lipschitz continuity of $\textsc{NashApr}(\sigma, u)$ with respect to its first input: the joint strategy profile σ. We get the following lemma, which indicates that $\textsc{NashApr}(\sigma, u)$ is 2-Lipschitz continuous with respect to σ under ℓ_1-distance.

Lemma 5.1. *For arbitrary strategy profile σ and σ', we have*

$$|\textsc{NashApr}(\sigma, u) - \textsc{NashApr}(\sigma', u)| \le 2\|\sigma - \sigma'\|_1,$$

where

$$\|\sigma - \sigma'\|_1 := \sum_{i\in N}\sum_{a_i\in A_i} |\sigma_i(a_i) - \sigma'_i(a_i)|$$

is the ℓ_1-distance between two mixed strategy profiles $\sigma, \sigma' \in \Delta A_1 \times \Delta A_2 \times \cdots \times \Delta A_n$.

We also analyze the Lipschitz property of $\textsc{NashApr}(\sigma, u)$ with respect to the game utility u, and get the following result:

Lemma 5.2. *For strategy profile σ and arbitrary normal-form game $\Gamma_u = (N, A, u)$ and $\Gamma_v = (N, A, v)$ with $u, v \in \mathcal{U}$, we have*

$$|\textsc{NashApr}(\sigma, u) - \textsc{NashApr}(\sigma, v)| \le 2\|u - v\|_{\max},$$

where

$$\|u - v\|_{\max} := \max_{i\in N}\max_{a\in A} |u_i(a) - v_i(a)| \tag{5}$$

is the $\ell_{\max}$-distance between game utilities u and v.

Remark 5.3. While minor changes in game utility may cause different NEs (See Table 2 for illustration of such non-smooth issue), as we can see in Lemma 5.2 the Nash approximation loss is 2-Lipschitz continuous with respect to the game utility. Such a continuity result plays a critical role in the theoretical analysis of game utility learning in simulation-based games [60], in which the goal is to recover the actual game utility through the noisy query data and to compute an approximate NE for the underlying game.

5.2 Generalization Bound

We measure the generalizability of an NE approximator by generalization bound. Such a bound depends on the complexity of hypothesis class $\mathcal{H}$. We characterize such complexity through (external) covering numbers [55], a standard technique in PAC analysis [2]. We first define the distance between two different approximators.

Definition 5.4 ($\ell_{\infty,1}$-distance)**.** The $\ell_{\infty,1}$-distance between two NE approximators h_1, h_2 is:

$$\|h_1 - h_2\|_{\infty,1} := \max_{u \in \mathcal{U}} \|h_1(u) - h_2(u)\|_1,$$

Under $\ell_{\infty,1}$-distance, we define the r-cover and the r-covering number for hypothesis class $\mathcal{H}$:

Definition 5.5 (r-cover)**.** We say function class $\mathcal{H}_r$ r-covers $\mathcal{H}$ under $\ell_{\infty,1}$-distance if for all function $h \in \mathcal{H}$, there exists h_r in $\mathcal{H}_r$ such that $\|h - h_r\|_{\infty,1} \leq r$.

Definition 5.6 (r-covering number)**.** The r-covering number of $\mathcal{H}$, denoted by $\mathcal{N}_{\infty,1}(\mathcal{H}, r)$, is the cardinality of the smallest function class $\mathcal{H}_r$ that r-covers $\mathcal{H}$ under $\ell_{\infty,1}$-distance.

We then derive the generalization bound of NE approximators. It describes the gap between the NE approximator's true risk $L_{\mathcal{D}}(h)$ and empirical risk $L_S(h)$ on the training set S.

Theorem 5.7. *[Generalization bound] For hypothesis class $\mathcal{H}$ of NE approximator and distribution $\mathcal{D}$, with probability at least $1-\delta$ over draw of the training set S from $\mathcal{D}$, $\forall h \in \mathcal{H}$ we have*

$$L_{\mathcal{D}}(h) - L_S(h) \leq 2\Delta_m + 4\sqrt{\frac{2\ln(4/\delta)}{m}}$$

where $\Delta_m := \inf_{r>0}\{\sqrt{\frac{2\ln \mathcal{N}_{\infty,1}(\mathcal{H},r)}{m}} + 2r\}$.

Theorem 5.7 is quite general and orthogonal to the learning algorithm we use. It characterizes the generalization ability of all the NE approximators in normal-form games with fixed action space. As we can see, with a large enough training set, the bound goes to zero (if $\mathcal{N}_{\infty,1}(\mathcal{H}, r)$ is bounded) so that we can estimate the true risk through the empirical risk.

5.3 Agnostic PAC Learnable

If for arbitrary $r > 0$ the covering number $\mathcal{N}_{\infty,1}(\mathcal{H}, r)$ can be bounded, then the bound in Theorem 5.7 goes to zero as the training set size $m \to \infty$. Inspired by this, we make the following assumption to limit the representativeness of $\mathcal{H}$:

Assumption 5.8. For hypothesis class $\mathcal{H}$, we assume the logarithm of its r-covering number grows as a polynomial with respect to $1/r$. i.e.,

$$\ln \mathcal{N}_{\infty,1}(\mathcal{H}, r) \leq \texttt{Poly}(\frac{1}{r})$$

for $r > 0$.

Assumption 5.8 is a standard assumption in PAC analysis [2]. It holds for many widely used machine learning models, including the classical linear model [63] and kernel method [64]. Moreover, as we will prove, Assumption 5.8 also holds for the Lipschitz hypothesis class, which includes neural networks with parameters of bounded ranges [52, 57].

Definition 5.9 (Lipschitz hypothesis class)**.** We say $\mathcal{H}$ is a Lipschitz hypothesis class if there is a constant $L_{\mathcal{H}} > 0$ such that for each function $h \in \mathcal{H}$ and game utility $u, v \in \mathcal{U}$, we have $\|h(u) - h(v)\|_1 \leq L_{\mathcal{H}}\|u - v\|_{\max}$,

Lemma 5.10. *Assumption 5.8 holds For Lipschitz hypothesis class $\mathcal{H}$ since we have*

$$\mathcal{N}_{\infty,1}(\mathcal{H}, r) \leq O\left((\frac{L_{\mathcal{H}}}{r})^{n|A|}\ln\frac{1}{r}\right).$$

Based on Assumption 5.8, Lemma 5.1 and Lemma 5.2, we prove the *uniform convergence* of hypothesis class $\mathcal{H}$ with respect to Nash approximation loss. It characterizes the sample complexity to probably obtain an *ϵ-representative* training set S. That is, for an arbitrary function $h \in \mathcal{H}$, the empirical risk $L_S(h)$ on S is close to the true risk $L_{\mathcal{D}}(h)$ up to ϵ.

Theorem 5.11. *[Uniform convergence] Fix $\epsilon, \delta \in (0,1)$, for hypothesis class $\mathcal{H}$ and distribution $\mathcal{D}$, with probability at least $1-\delta$ over draw of the training set S with*

$$m \geq m^{UC}_{\mathcal{H}}(\epsilon, \delta) := \frac{9}{2\epsilon^2}\left(\ln\frac{2}{\delta} + \ln \mathcal{N}_{\infty,1}(\mathcal{H}, \frac{\epsilon}{6})\right)$$

games from $\mathcal{D}$, we have

$$|L_S(h) - L_{\mathcal{D}}(h)| \leq \epsilon$$

for all $h \in \mathcal{H}$. $m^{UC}_{\mathcal{H}}(\epsilon, \delta)$ grows as a polynomial of $1/\epsilon$ and $\ln(1/\delta)$ under Assumption 5.8.

Theorem 5.11 is the sufficient condition for *agnostic PAC learnable*, which provides the learnability guarantee of predicting NE from samples.

Theorem 5.12. *Fix $\epsilon, \delta \in (0,1)$, for hypothesis class $\mathcal{H}$ and distribution $\mathcal{D}$, with probability at least $1-\delta$ over draw of the training set S with*

$$m \geq m_{\mathcal{H}}(\epsilon, \delta) := \frac{18}{\epsilon^2}\left(\ln\frac{2}{\delta} + \ln \mathcal{N}_{\infty,1}(\mathcal{H}, \frac{\epsilon}{6})\right)$$

games from $\mathcal{D}$, when running empirical risk minimization on Nash approximation loss, we have

$$L_{\mathcal{D}}(\text{ERM}_{\mathcal{H}}(S)) \leq \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) + \epsilon.$$

The sample complexity $m_{\mathcal{H}}(\epsilon, \delta)$ grows as a polynomial of $1/\epsilon$ and $\ln(1/\delta)$ under Assumption 5.8.

Theorem 5.12 provides the (agnostic) PAC learnability of NE. Under Assumption 5.8, when using a training set with size larger than a polynomial of $1/\epsilon$ and $\ln(1/\delta)$, with probability at least $1-\delta$ the learned NE approximator can reach the near-optimal performance in $\mathcal{H}$ up to ϵ. As we will demonstrate by experiments in Section 6, even equipped with the most simple neural architectures, the learned NE approximator can efficiently compute approximate NE solutions for games under the same distribution.

Remark 5.13. While realizability assumption [55], i.e., the assumption that $\min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) = 0$, is adopted in many PAC analyses [35, 39], however, it is not feasible in our case. Due to the non-smooth issue, we discussed in Table 2, it remains an open question whether there is a hypothesis class that satisfies the realizability assumption with limited complexity. As a result, we consider *agnostic* PAC learnability.

6 EXPERIMENTS AND APPLICATION

In this section, we first provide numerical experiments to verify the practicality of our PAC result. Specifically, we construct a parameterized model as our hypothesis class and train an NE approximator via Algorithm 1. We show that the learned NE approximator is computation-efficient with low generation loss. Afterward, we demonstrate an application for the NE approximator: It can warm-start other NE solvers in bimatrix games by providing effective initializing points. All of our experiments are run on a Linux machine with 48 core Intel(R) Xeon(R) CPU (E5-2650 v4@2.20GHz) and 4 TITAN V GPU. Each experiment is run by 5 times, and the average results are presented.

6.1 Experimental Setup

We use GAMUT[2] [50], a suite of game generators designated for testing game-theoretic algorithms, to generate the game instances. We select 5 game classes as our data distribution since they are nontrivial for TS algorithm [58] to solve [23]:

- *TravelersDilemma:* Each player simultaneously requests an amount of money and receives the lowest of the requests submitted by all players.
- *GrabTheDollar:* A price is up for grabs, and both players have to decide when to grab the price. The action of each player is the chosen times. If both players grab it simultaneously, they will rip the price and receive a low payoff. If one chooses a time earlier than the other, she will receive the high payoff, and the opposing player will receive a payoff between the high and the low.
- *WarOfAttrition:* In this game, both players compete for a single object, and each chooses a time to concede the object to the other player. If both concede at the same time, they share the object. Each player has a valuation of the object, and each player's utility is decremented at every time step.
- *BertrandOligopoly:* All players in this game are producing the same item and are expected to set a price at which to sell the item. The player with the lowest price gets all the demand for the item and produces enough items to meet the demand to obtain the corresponding payoff.
- *MajorityVoting:* This is an *n*-player symmetric game. All players vote for one of the $|A_1|$ candidates. Players' utilities for each candidate being declared the winner are arbitrary. If there is a tie, the winner is the candidate with the lowest number. There may be multiple Nash equilibria in this game.

For bimatrix games, we set the game size as 300×300. For multiplayer games, we generate the 3 and 4 player versions of *BertrandOligopoly* and *MajorityVoting* (The suffix -3 and -4 represent the 3 and 4 player versions, respectively). We set the game size as $30 \times 30 \times 30$ for 3-player games and $15 \times 15 \times 15 \times 15$ for 4-player games. For each game class, we generate 2×10^4 game instances with different random seeds, and we randomly divide 2000 and 200 instances for validation and testing.

As for the NE approximator, we construct a fully connected neural network as the hypothesis class due to the universal approximation theorem of it [33]. We apply ReLU as the activation function and add batch normalization (without learnable parameters) before the activation function. We use 4 hidden layers with 1024 nodes of each layer in our neural network. We learn our model using the Adam optimizer, and we restrict the parameters of our model in the range of $[0, 1]$. By doing so, we make our model a Lipschitz hypothesis class so that it satisfies Assumption 5.8.

[2] http://gamut.stanford.edu/

6.2 Generalization and Efficiency

Generalization. Table 3 and Table 4 report the average Nash approximation loss of the trained NE approximator. We observe that the Nash approximation loss in the test set is sufficiently small and much lower than the random solutions. Such a comparison result inspires us to use the predicted solution as the initial point for classical solvers. Moreover, we can also see a small gap between the training and testing performance, which gives the feasibility of estimating the true risk of the NE approximator through its empirical risk on the training set. It also verifies the generalization bound in Theorem 5.7.

Efficiency. Notice that the NE approximator has never seen the test game instances in training. The approximate solution is obtained by just a simple feed-forward neural network computation. As a result, it can be used to infer approximate solutions quickly. To better demonstrate the efficiency of the trained NE approximator, we record the time and iterations traditional algorithms spent (on the test set) to reach the same performance. We use the following algorithms:

- *Fictitious play (FP)* [48]: The most well-known learning algorithm to approximate Nash equilibrium;
- *Regret matching (RM)* [30]: Representative method of no-regret learning, and it leads to coarse correlated equilibrium.
- *Replicator dynamics (RD)* [53]: A system of differential equations that describe how a population of strategies, or replicators, evolve through time.
- *TS Algorithm* (TS): The algorithm proposed by Tsaknakis and Spirakis [58]. It reaches an approximation ratio $\epsilon = 0.3393$ for bimatrix games.
- *DFM algorithm* (DFM): The algorithm proposed by Deligkas et al. [17]. It is an improved version of the TS algorithm, and reaches the current best approximation ratio $\epsilon = 1/3$ for bimatrix games.

During implementation, we use GPU to speed up the computation of the baselines. TS and DFM algorithm cannot be accelerated by GPU, so we run them on CPU. For FP, RM and RD, we set the maximum number of iterations to 100000 and terminate the algorithm once it reaches the same performance. TS and DFM algorithm terminates with probability 1, so we stop them early if the same performance has already been reached.

We present the efficiency results of bimatrix game in Table 5 and multiplayer game in Table 6. While the NE approximator efficiently comes up with an approximate solution, the baseline methods spend much more time to reach the same performance. Sometimes the learning approaches FP, RM and RD even fail to converge to the same performance as NE approximator.

Table 3: The average Nash approximation loss (and the corresponding standard deviation across random seeds) of the learned NE approximator on training and testing, compared with random solutions. All the games are 300×300 bimatrix games.

	TravelersDilemma NashApr	*GrabTheDollar* NashApr	*WarOfAttrition* NashApr	*BertrandOligopoly* NashApr	*MajorityVoting* NashApr
Random	0.2644 ± 2.57e-4	0.2603 ± 2.78e-4	0.3396 ± 2.65e-4	0.3208 ± 3.86e-4	0.4727 ± 5.98e-4
Train	1.013e-6 ± 1.07e-7	8.328e-5 ± 7.87e-5	2.984e-7 ± 1.65e-8	3.402e-4 ± 5.48e-5	4.416e-6 ± 1.02e-6
Test	0.991e-6 ± 1.04e-7	4.823e-5 ± 5.36e-5	2.871e-7 ± 1.89e-8	3.338e-4 ± 4.90e-5	5.526e-6 ± 2.04e-6

Table 4: The average Nash approximation loss (and the corresponding standard deviation across random seeds) of the learned NE approximator on training and testing, compared with random solutions. The game dimension is $30 \times 30 \times 30$ for 3-player games and $15 \times 15 \times 15 \times 15$ for 4-player games.

	BertrandOligopoly-3 NashApr	MajorityVoting-3 NashApr	BertrandOligopoly-4 NashApr	MajorityVoting-4 NashApr
Random	0.1145 ± 9.16e-4	0.3534 ± 1.11e-3	0.0573 ± 5.41e-4	0.2428 ± 1.34e-3
Train	4.046e-6 ± 7.22e-6	1.018e-3 ± 3.45e-4	1.619e-7 ± 3.51e-8	3.881e-4 ± 1.45e-4
Test	2.525e-6 ± 4.20e-6	0.612e-3 ± 2.54e-4	1.643e-7 ± 3.53e-8	2.359e-4 ± 4.36e-4

Table 5: The average time and iterations traditional algorithms spent on each test set to reach the same performance as the NE approximator (NEA) in 300×300 bimatrix games. * represents the method fails to reach the same performance under the limitation of the maximum iterations in some of the 5 runs.

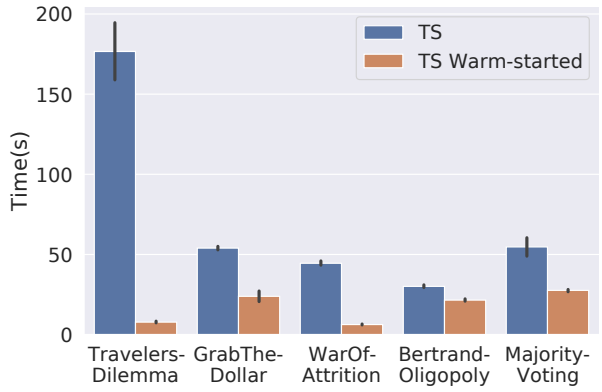
	TravelersDilemma		*GrabTheDollar*		*WarOfAttrition*		*BertrandOligopoly*		*MajorityVoting*	
	Time	Iteration	Time	Iteration	Time	Iteration	Time	Iteration	Time	Iteration
FP	*99.9s	*100000	*95.2s	*100000	57.6s	59919.6	61.5s	63794.8	77.1s	80452.6
RM	162.2s	85442.2	*190.3s	*98788.2	149.3s	79151.2	28.1s	14544.0	*189.4s	*100000
RD	4.3s	3813.4	2.5s	2212.4	2.8s	2482.8	1.0s	826.2	*118.7s	*100000
TS	149.9s	--	47.1s	--	41.9s	--	26.6s	--	44.1s	--
DFM	147.6s	--	45.8s	--	39.8s	--	27.1s	--	44.7s	--
NEA	**<0.5s**	**1.0**	**<0.5s**	**1.0**	**<0.5s**	**1.0**	**<0.5s**	**1.0**	**<0.5s**	**1.0**

Table 6: The average time and iterations traditional algorithms spent on each test set to reach the same performance as the NE approximator (NEA) in $30 \times 30 \times 30$ and $15 \times 15 \times 15 \times 15$ games. * represents the method fails to reach the same performance under the limitation of the maximum iterations in some of the 5 runs.

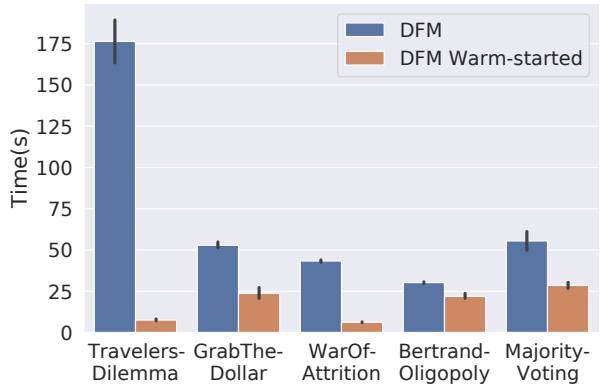
Methods	BertrandOligopoly-3		MajorityVoting-3		BertrandOligopoly-4		MajorityVoting-4	
	Time	Iteration	Time	Iteration	Time	Iteration	Time	Iteration
FP	*186.3s	*100000	2.5s	1273.8	*311.6s	*100000	183.8s	59324.0
RM	*258.6s	*100000	19.8s	7395.6	*393.3s	*100000	106.5s	27331.6
RD	33.0s	28629.0	0.9s	607.2	50.6s	23100.2	143.3s	61431.2
NEA	**<0.5s**	**1.0**	**<0.5s**	**1.0**	**<0.5s**	**1.0**	**<0.5s**	**1.0**

6.3 Application: Warm-Start Classical Solvers

As we can see from the previous experiments, the NE approximator could be efficient for the games under the same distribution. Moreover, it can achieve a better Nash approximation loss than random solutions. Meanwhile, the classical NE solvers, such as the TS and DFM algorithm, usually set random strategies as the starting point. Therefore, it is promising to warm-start those algorithms with the NE approximator. By doing so, we benefit from both advantages of the function-approximation method and the traditional approach. The NE approximator can infer initial solutions in batches with low computational costs, and the classical solvers can provide theoretical guarantees.



(a) Warm-starting TS algorithm.



(b) Warm-starting DFM algorithm.

Figure 1: Experimental results of warm-starting TS algorithm and DFM algorithm with NE approximator. Each experiment is run by 5 times. Average results and 95% confidence intervals are shown.

Figure 1a and Figure 1b report the experimental results of warm-starting TS and DFM algorithm, respectively. We can observe that by taking the output strategies of the NE approximator as the pre-solving initialization, both TS and DFM algorithms spend less time to terminate, especially in game *TravelersDilemma* and *WarOfAttrition*. Notice that both algorithms ensure that the final solutions will be better than the initial solutions. Thus, it would always be helpful to provide a good starting point for them.

7 CONCLUSION

In this paper, we study the learnability of predicting NE in n-player normal-form games with fixed action space. Theoretically, we provide a generalization bound for the NE approximator under Nash approximation loss. The bound gives a theoretical guarantee of the generalization ability. We then prove that Nash equilibrium is agnostic PAC learnable. Such a result provides the feasibility of obtaining a good NE approximator via empirical risk minimization. Empirically, we conduct numerical experiments to verify the learned NE approximator's generalization ability and efficiency. Afterward, we demonstrate the application of the NE approximator to warm-start other classical solvers, and we report fast convergence. Our theoretical and empirical results show the practicability of learning an NE approximator via data-driven approach. As for future work, we are interested in extending the learnability results of the NE approximator to settings beyond normal-form games, and exploring a more efficient hypothesis class for the NE approximator.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (Grant No. 62172012). We thank Xiang Yan, Dongge Wang, David Mguni and Kun Shao for various helpful discussions. We thank all anonymous reviewers for their helpful feedback.

REFERENCES

[1] Stephanie Allen, Steven A Gabriel, and John P Dickerson. 2022. Using inverse optimization to learn cost functions in generalized Nash games. *Computers & Operations Research* 142 (2022), 105721.
[2] Martin Anthony, Peter L Bartlett, Peter L Bartlett, et al. 1999. *Neural network learning: Theoretical foundations*. Vol. 9. cambridge university press Cambridge.
[3] Sanjeev Arora, Elad Hazan, and Satyen Kale. 2012. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing* 8, 1 (2012), 121–164.
[4] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. 1995. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*. IEEE, 322–331.
[5] Yu Bai, Chi Jin, and Tiancheng Yu. 2020. Near-optimal reinforcement learning with self-play. *Advances in neural information processing systems* 33 (2020), 2159–2170.
[6] Dimitris Bertsimas, Vishal Gupta, and Ioannis Ch Paschalidis. 2015. Data-driven estimation in equilibrium using inverse optimization. *Mathematical Programming* 153, 2 (2015), 595–633.
[7] Hartwig Bosse, Jaroslaw Byrka, and Evangelos Markakis. 2007. New algorithms for approximate Nash equilibria in bimatrix games. In *International Workshop on Web and Internet Economics*. Springer, 17–29.
[8] Nicolo Cesa-Bianchi and Gábor Lugosi. 2006. *Prediction, learning, and games*. Cambridge university press.
[9] Xi Chen, Xiaotie Deng, and Shang-Hua Teng. 2009. Settling the complexity of computing two-player Nash equilibria. *Journal of the ACM (JACM)* 56, 3 (2009), 1–57.
[10] Zhaohua Chen, Xiaotie Deng, Wenhan Huang, Hanyu Li, and Yuhao Li. 2021. On tightness of the Tsaknakis-Spirakis algorithm for approximate Nash equilibrium. In *Algorithmic Game Theory: 14th International Symposium, SAGT 2021, Aarhus, Denmark, September 21–24, 2021, Proceedings 14*. Springer, 97–111.
[11] Vincent Conitzer. 2009. Approximation guarantees for fictitious play. In *2009 47th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 636–643.
[12] Artur Czumaj, Argyrios Deligkas, Michail Fasoulakis, John Fearnley, Marcin Jurdziński, and Rahul Savani. 2019. Distributed methods for computing approximate equilibria. *Algorithmica* 81, 3 (2019), 1205–1231.
[13] Constantinos Daskalakis. 2013. On the complexity of approximating a Nash equilibrium. *ACM Transactions on Algorithms (TALG)* 9, 3 (2013), 1–35.
[14] Constantinos Daskalakis, Paul W Goldberg, and Christos H Papadimitriou. 2009. The complexity of computing a Nash equilibrium. *SIAM J. Comput.* 39, 1 (2009), 195–259.
[15] Constantinos Daskalakis, Aranyak Mehta, and Christos Papadimitriou. 2007. Progress in approximate Nash equilibria. In *Proceedings of the 8th ACM Conference on Electronic Commerce*. 355–358.
[16] Constantinos Daskalakis, Aranyak Mehta, and Christos Papadimitriou. 2009. A note on approximate Nash equilibria. *Theoretical Computer Science* 410, 17 (2009), 1581–1588.
[17] Argyrios Deligkas, Michail Fasoulakis, and Evangelos Markakis. 2022. A Polynomial-Time Algorithm for 1/3-Approximate Nash Equilibria in Bimatrix Games. In *30th Annual European Symposium on Algorithms, ESA 2022, September 5-9, 2022, Berlin/Potsdam, Germany (LIPIcs, Vol. 244)*, Shiri Chechik, Gonzalo Navarro, Eva Rotenberg, and Grzegorz Herman (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 41:1–41:14. https://doi.org/10.4230/LIPIcs.ESA.2022.41
[18] Xiaotie Deng, Tao Lin, and Tao Xiao. 2020. Private data manipulation in optimal sponsored search auction. In *Proceedings of The Web Conference 2020*. 2676–2682.
[19] Xiaotie Deng, Tao Xiao, and Keyu Zhu. 2017. Learn to play maximum revenue auction. *IEEE Transactions on Cloud Computing* 7, 4 (2017), 1057–1067.
[20] Xiaotie Deng and Keyu Zhu. 2019. On bayesian epistemology of myerson auction. *IEEE Transactions on Cloud Computing* 9, 3 (2019), 1172–1179.

[21] Le Cong Dinh, Stephen Marcus McAleer, Zheng Tian, Nicolas Perez-Nieves, Oliver Slumbers, David Henry Mguni, Jun Wang, Haitham Bou Ammar, and Yaodong Yang. 2022. Online Double Oracle. *Transactions on Machine Learning Research* (2022). https://openreview.net/forum?id=rrMK6hYNSx

[22] John Fearnley, Martin Gairing, Paul W Goldberg, and Rahul Savani. 2015. Learning equilibria of games via payoff queries. *J. Mach. Learn. Res.* 16 (2015), 1305–1344.

[23] John Fearnley, Tobenna Peter Igwe, and Rahul Savani. 2015. An empirical study of finding approximate equilibria in bimatrix games. In *International Symposium on Experimental Algorithms*. Springer, 339–351.

[24] John Fearnley and Rahul Savani. 2016. Finding approximate Nash equilibria of bimatrix games via payoff queries. *ACM Transactions on Economics and Computation (TEAC)* 4, 4 (2016), 1–19.

[25] Filiberto Fele and Kostas Margellos. 2020. Probably approximately correct Nash equilibrium learning. *IEEE Trans. Automat. Control* (2020), 4238–4245.

[26] Xidong Feng, Oliver Slumbers, Ziyu Wan, Bo Liu, Stephen McAleer, Ying Wen, Jun Wang, and Yaodong Yang. 2021. Neural auto-curricula in two-player zero-sum games. *Advances in Neural Information Processing Systems* 34 (2021), 3504–3517.

[27] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

[28] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems* 27 (2014).

[29] Keegan Harris, Ioannis Anagnostides, Gabriele Farina, Mikhail Khodak, Steven Wu, and Tuomas Sandholm. 2023. Meta-Learning in Games. In *International Conference on Learning Representations*. https://openreview.net/forum?id=uHaWaNhCvZD

[30] Sergiu Hart and Andreu Mas-Colell. 2000. A simple adaptive procedure leading to correlated equilibrium. *Econometrica* 68, 5 (2000), 1127–1150.

[31] David Haussler. 1990. *Probably approximately correct learning*. University of California, Santa Cruz, Computer Research Laboratory.

[32] Howard Heaton, Daniel McKenzie, Qiuwei Li, Samy Wu Fung, Stanley Osher, and Wotao Yin. 2021. Learn to Predict Equilibria via Fixed Point Networks. *arXiv preprint arXiv:2106.00906* (2021).

[33] Kurt Hornik, Maxwell B. Stinchcombe, and Halbert L. White. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks* 2 (1989), 359–366.

[34] Junling Hu and Michael P Wellman. 2003. Nash Q-learning for general-sum stochastic games. *Journal of machine learning research* 4, Nov (2003), 1039–1069.

[35] Chi Jin, Qinghua Liu, and Sobhan Miryoosefi. 2021. Bellman eluder dimension: New rich classes of rl problems, and sample-efficient algorithms. *Advances in neural information processing systems* 34 (2021), 13406–13418.

[36] Chi Jin, Qinghua Liu, Yuanhao Wang, and Tiancheng Yu. 2022. V-Learning – A Simple, Efficient, Decentralized Algorithm for Multiagent RL. In *ICLR 2022 Workshop on Gamification and Multiagent Solutions*.

[37] Spyros C Kontogiannis, Panagiota N Panagopoulou, and Paul G Spirakis. 2006. Polynomial algorithms for approximating Nash equilibria of bimatrix games. In *International Workshop on Internet and Network Economics*. Springer, 286–296.

[38] Spyros C Kontogiannis and Paul G Spirakis. 2007. Efficient algorithms for constant well supported approximate equilibria in bimatrix games. In *International Colloquium on Automata, Languages, and Programming*. Springer, 595–606.

[39] Akshay Krishnamurthy, Alekh Agarwal, and John Langford. 2016. PAC reinforcement learning with rich observations. *Advances in Neural Information Processing Systems* 29 (2016).

[40] Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. 2017. A unified game-theoretic approach to multiagent reinforcement learning. *Advances in neural information processing systems* 30 (2017).

[41] Jiayang Li, Jing Yu, Yu Nie, and Zhaoran Wang. 2020. End-to-end learning and intervention in games. *Advances in Neural Information Processing Systems* 33 (2020).

[42] Chun Kai Ling, Fei Fang, and J. Zico Kolter. 2018. What Game Are We Playing? End-to-end Learning in Normal and Extensive Form Games. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, 396–402. https://doi.org/10.24963/ijcai.2018/55

[43] Chun Kai Ling, Fei Fang, and J Zico Kolter. 2019. Large scale learning of agent rationality in two-player zero-sum games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 6104–6111.

[44] Edward Lockhart, Marc Lanctot, Julien Pérolat, Jean-Baptiste Lespiau, Dustin Morrill, Finbarr Timbers, and Karl Tuyls. 2019. Computing Approximate Equilibria in Sequential Adversarial Games by Exploitability Descent.. In *IJCAI*, Sarit Kraus (Ed.). ijcai.org, 464–470.

[45] Alberto Marchesi, Francesco Trovò, and Nicola Gatti. 2020. Learning Probably Approximately Correct Maximin Strategies in Simulation-Based Games with Infinite Strategy Spaces. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*. 834–842.

[46] Luke Marris, Ian Gemp, Thomas Anthony, Andrea Tacchetti, Siqi Liu, and Karl Tuyls. 2022. Turbocharging Solution Concepts: Solving NEs, CEs and CCEs with Neural Equilibrium Solvers. In *Advances in Neural Information Processing Systems*, Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (Eds.). https://openreview.net/forum?id=RczPtvlaXPH

[47] H Brendan McMahan, Geoffrey J Gordon, and Avrim Blum. 2003. Planning in the presence of cost functions controlled by an adversary. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*. 536–543.

[48] Dov Monderer and Lloyd S Shapley. 1996. Fictitious play property for games with identical interests. *Journal of economic theory* 68, 1 (1996), 258–265.

[49] John F Nash et al. 1950. Equilibrium points in n-person games. *Proceedings of the national academy of sciences* 36, 1 (1950), 48–49.

[50] Eugene Nudelman, Jennifer Wortman, Yoav Shoham, and Kevin Leyton-Brown. 2004. Run the GAMUT: A comprehensive approach to evaluating game-theoretic algorithms. In *AAMAS*, Vol. 4. 880–887.

[51] Nicolas Perez-Nieves, Yaodong Yang, Oliver Slumbers, David H Mguni, Ying Wen, and Jun Wang. 2021. Modelling Behavioural Diversity for Learning in Open-Ended Games. In *International Conference on Machine Learning*. PMLR, 8514–8524.

[52] Kevin Scaman and Aladin Virmaux. 2018. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In *NeurIPS*. 3839–3848.

[53] Peter Schuster and Karl Sigmund. 1983. Replicator dynamics. *Journal of theoretical biology* 100, 3 (1983), 533–538.

[54] Pier Giuseppe Sessa, Ilija Bogunovic, Andreas Krause, and Maryam Kamgarpour. 2020. Contextual games: Multi-agent learning with side information. *Advances in Neural Information Processing Systems* 33 (2020), 21912–21922.

[55] Shai Shalev-Shwartz and Shai Ben-David. 2014. *Understanding machine learning: From theory to algorithms*. Cambridge university press.

[56] Yoav Shoham and Kevin Leyton-Brown. 2008. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press.

[57] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.).

[58] Haralampos Tsaknakis and Paul G Spirakis. 2007. An optimization approach for approximate Nash equilibria. In *International Workshop on Web and Internet Economics*. Springer, 42–56.

[59] Leslie G Valiant. 1984. A theory of the learnable. *Commun. ACM* 27, 11 (1984), 1134–1142.

[60] Enrique Areyan Viqueira, Cyrus Cousins, Eli Upfal, and Amy Greenwald. 2019. Learning equilibria of simulation-based games. *arXiv preprint arXiv:1905.13379* (2019).

[61] Yaodong Yang and Jun Wang. 2020. An Overview of Multi-Agent Reinforcement Learning from Game Theoretical Perspective. *arXiv preprint arXiv:2011.00583* (2020).

[62] Jing Zhang and Ioannis Ch Paschalidis. 2017. Data-driven estimation of travel latency cost functions via inverse optimization in multi-class transportation networks. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 6295–6300.

[63] Tong Zhang. 2002. Covering number bounds of certain regularized linear function classes. *Journal of Machine Learning Research* 2, Mar (2002), 527–550.

[64] Ding-Xuan Zhou. 2002. The covering number in learning theory. *Journal of Complexity* 18, 3 (2002), 739–767.