DeepFP for Finding Nash Equilibrium in Continuous Action Spaces

Nitin Kamra[1(✉)], Umang Gupta[1], Kai Wang[1], Fei Fang[2], Yan Liu[1], and Milind Tambe[3]

[1] University of Southern California, Los Angeles, CA 90089, USA
{nkamra,umanggup,wang319,yanliu.cs}@usc.edu
[2] Carnegie Mellon University, Pittsburgh, PA 15213, USA
feifang@cmu.edu
[3] Harvard University, Cambridge, MA 02138, USA
milind.tambe11@gmail.com

Abstract. Finding Nash equilibrium in continuous action spaces is a challenging problem and has applications in domains such as protecting geographic areas from potential attackers. We present DeepFP, an approximate extension of fictitious play in continuous action spaces. DeepFP represents players' approximate best responses via generative neural networks which are highly expressive implicit density approximators. It additionally uses a game-model network which approximates the players' expected payoffs given their actions, and trains the networks end-to-end in a model-based learning regime. Further, DeepFP allows using domain-specific oracles if available and can hence exploit techniques such as mathematical programming to compute best responses for structured games. We demonstrate stable convergence to Nash equilibrium on several classic games and also apply DeepFP to a large forest security domain with a novel defender best response oracle. We show that DeepFP learns strategies robust to adversarial exploitation and scales well with growing number of players' resources.

Keywords: Security games · Nash equilibrium · Fictitious Play

1 Introduction

Computing equilibrium strategies is a major computational challenge in game theory and finds numerous applications in economics, planning, security domains etc. We are motivated by security domains which are often modeled as Stackelberg Security Games (SSGs) [5,13,24]. Since Stackelberg Equilibrium, a commonly used solution concept in SSGs, coincides with Nash Equilibrium (NE) in zero-sum security games and in some structured general-sum games [17], we focus on the general problem of finding mixed strategy Nash Equilibrium. Security domains often involve protecting geographic areas thereby leading to *continuous action spaces* [3,26]. Most previous approaches discretize players' continuous

T. Alpcan et al. (Eds.): GameSec 2019, LNCS 11836, pp. 238–258, 2019.
https://doi.org/10.1007/978-3-030-32430-8_15

actions [9,11,27] to find equilibrium strategies using linear programming (LP) or mixed-integer programming (MIP). However, a coarse discretization suffers from low solution quality and a fine discretization makes it intractable to compute the optimal strategy using mathematical programming techniques, especially in high-dimensional action spaces. Some approaches exploit spatio-temporal structural regularities [4,6,28] or numerically solve differential equations in special cases [13], but these do not extend to general settings.

We focus on algorithms more amenable to tractable approximation in continuous action spaces. Fictitious Play (FP) is a classic algorithm studied in game theory and involves players repeatedly playing the game and best responding to each other's history of play. FP converges to a NE for specific classes of discrete action games [18] and its variants like Stochastic Fictitious Play and Generalized Weakened Fictitious Play converge under more diverse settings [20,23,25] under reasonable regularity assumptions over underlying domains. While FP applies to discrete action games with exact best responses, it does not trivially extend to continuous action games with arbitrarily complex best responses.

In this work, we present DeepFP, an approximate fictitious play algorithm for two-player games with continuous action spaces. The key novelties of DeepFP are: (a) It represents players' approximate best responses via state-of-the-art generative neural networks which are highly expressive implicit density approximators with no shape assumptions on players' action spaces, (b) Since implicit density models cannot be trained directly, it also uses a game-model network which is a differentiable approximation of the players' payoffs given their actions, and trains these networks end-to-end in a model-based learning regime, and (c) DeepFP allows replacing these networks with domain-specific oracles if available. This allows working in the absence of gradients for player/(s) and exploit techniques from research areas like mathematical programming to compute best responses. We also apply DeepFP to a forest security problem with a novel defender best response oracle designed for this domain. The proposed oracle is another novel contribution of this work and may also be of interest in its own right for the forest protection domain.

Related Work: Previous approaches to find equilibria in continuous action spaces have employed variants of cournot adjustment strategy [1] but suffer from convergence issues [14]. Variants of FP either require explicit maximization of value functions over the action set as in Fictitious Self-Play [12] or maintaining complex hierarchies of players' past joint strategies as in PSRO [19], which is only feasible with finite and discrete action sets (e.g. poker) and does not generalize to continuous action spaces. Since it is challenging to maintain distributions over continuous action spaces, recent works in multiagent reinforcement learning (RL) [21] often assume explicit families of distributions for players' strategies which may not span the space of strategies to which NE distributions belong. More recently update rules which modify gradient descent using second-order dynamics of multi-agent games have been proposed [2]. The closest method to our approach is OptGradFP [15] which assumes a multivariate logit-normal distribution for players' strategies. We show that due to explicit shape assumptions,

it suffers from lack of representational power and is prone to diverging since logit-normal distributions concentrate in some parts of the action space often yielding $-\infty$ log-probabilities in other parts. DeepFP addresses the lack of representational power by using flexible implicit density approximators. Further, our model-based training proceeds without any likelihood estimates and hence does not yield $-\infty$ log-likelihoods in any parts of the action space, thereby converging stably. Moreover, unlike OptGradFP, DeepFP is an off-policy algorithm and trains significantly faster by directly estimating expected rewards using the game model network instead of replaying previously stored games.

2 Game Model

We consider a two-player game with continuous action sets for players 1 and 2. We will often use the index $p \in \{1, 2\}$ for one of the players and $-p$ for the other player. U_p denotes the compact, convex action set of player p. We denote the probability density for the mixed strategy of player p at action $u_p \in U_p$ as $\sigma_p(u_p) \geq 0$ s.t. $\int_{U_p} \sigma_p(u_p) du_p = 1$. We denote player p sampling an action $u_p \in U_p$ from his mixed strategy density σ_p as $u_p \sim \sigma_p$. We denote joint actions, joint action sets and joint densities without any player subscript i.e. as $u = (u_1, u_2), U = U_1 \times U_2$ and $\sigma = (\sigma_1, \sigma_2)$ respectively.

Each player has a bounded and Lipschitz continuous reward function $r_p : U \to \mathbb{R}$. For zero-sum games, $r_p(u) + r_{-p}(u) = 0 \; \forall u \in U$. With players' mixed strategy densities σ_p and σ_{-p}, the expected reward of player p is:

$$\mathbb{E}_{u \sim \sigma}[r_p] = \int_{U_p} \int_{U_{-p}} r_p(u) \sigma_p(u_p) \sigma_{-p}(u_{-p}) du_p du_{-p}.$$

The best response of player p against player $-p$'s current strategy σ_{-p} is defined as the set of strategies which maximizes his expected reward:

$$BR_p(\sigma_{-p}) := \arg\max_{\sigma_p} \left\{ \mathbb{E}_{u \sim (\sigma_p, \sigma_{-p})}[r_p] \right\}.$$

A pair of strategies $\sigma^* = (\sigma_1^*, \sigma_2^*)$ is said to be a Nash equilibrium if neither player can increase his expected reward by changing his strategy while the other player sticks to his current strategy. In such a case both these strategies belong to the best response sets to each other:

$$\sigma_1^* \in BR_1(\sigma_2^*) \text{ and } \sigma_2^* \in BR_2(\sigma_1^*).$$

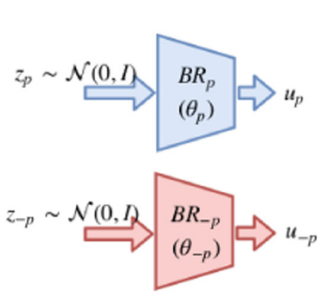
3 Deep Fictitious Play

To compute NE for a game, we introduce an approximate realization of fictitious play in high-dimensional continuous action spaces, which we call Deep Fictitious

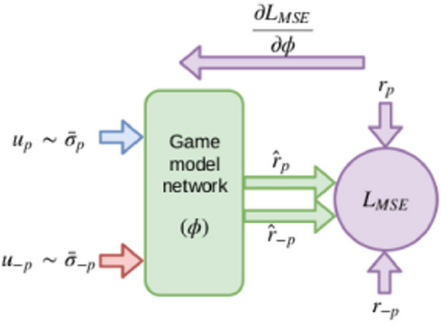
Play (DeepFP). Let the density function corresponding to the empirical distribution of player p's previous actions (a.k.a. belief density) be $\bar{\sigma}_p$. Then fictitious play involves player p best responding to his opponent's belief density $\bar{\sigma}_{-p}$:

$$BR_p(\bar{\sigma}_{-p}) := \arg\max_{\sigma_p} \left\{ \mathbb{E}_{u \sim (\sigma_p, \bar{\sigma}_{-p})}[r_p] \right\}.$$

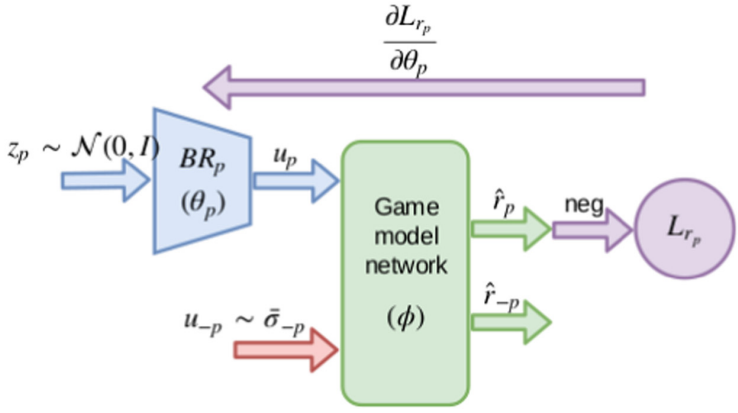
Repeating this procedure for both players is guaranteed to converge to the Nash equilibrium densities for both players for certain classes of games [18]. Hence extending Fictitious Play to continuous action spaces requires approximations for two essential ingredients: (a) belief densities over players' actions, and (b) best responses for each player.



(a) Sampling actions from the best response network



(b) Learning game model network parameters ϕ



(c) Learning best response network parameters θ_p

Fig. 1. Neural network models for DeepFP; blue color denotes player p, red denotes his opponent $-p$, green shows the game model network and violet shows loss functions and gradients. (Color figure online)

3.1 Approximating Belief Densities

Representing belief densities compactly is challenging in continuous action spaces. However with an appropriate approximation to Fictitious Play, one can get away with a representation which only requires sampling from the belief density but never explicitly calculating the density at any point in the action space. Our DeepFP is one such approximation and hence we maintain the belief density $\bar{\sigma}_p$ of each player p via a non-parameterized population based estimate i.e. via a simple memory of all actions played by p so far. Directly sampling u_p from the memory gives an unbiased sample from $\bar{\sigma}_p$.

3.2 Approximating Best Responses

Computing exact best responses is intractable for most games. But when the expected reward for a player p is differentiable w.r.t. the player's action u_p and admits continuous and smooth derivatives, approximate best responses are feasible. One way is to use the gradient of reward to update the action u_p iteratively using gradient ascent till it converges to a best response. Since the best response needs to be computed per iteration of FP, employing inner iterations of gradient descent can be expensive. However since the history of play for players doesn't change too much between iterations of FP, we expect the same of best responses. Consequently we approximate best responses with function approximators (e.g., neural networks) and keep them updated with a single gradient ascent step (also done by [8]). We propose a *best response network* for each player p which maps an easy to sample d_p-dimensional random variable $Z_p \in \mathbb{R}^{d_p}$ (e.g. $Z_p \sim \mathcal{N}(0, I_{d_p})$) to the player's action u_p. By learning an appropriate mapping $BR_p(z_p; \theta_p)$ parameterized by weights θ_p, it can approximate any density in the action space U_p (Fig. 1a). Note that this is an implicit density model i.e. one can draw samples of u_p by sampling $z_p \sim \mathcal{P}_{Z_p}(\cdot)$ and then computing $BR_p(z_p; \theta_p)$, but no estimate of the density is explicitly available. Further, best response networks maintain stochastic best responses since they lead to smoother objectives for gradient-based optimization. Using them is common practice in policy-gradient and actor-critic based RL since deterministic best responses often render the algorithm unstable and brittle to hyperparameter settings (also shown by [10]).

To learn θ_p we need to approximate the expected payoff of player p given by $\mathbb{E}_{(u_p \sim BR_p(\cdot;\theta_p), u_{-p} \sim \bar{\sigma}_{-p})}[r_p]$ as a differentiable function of θ_p. However a differentiable game model is generally not available a priori, hence we also maintain a *game model network* which takes all players' actions i.e. $\{u_p, u_{-p}\}$ as inputs and predicts rewards $\{\hat{r}_p, \hat{r}_{-p}\}$ for each player. This can either be pre-trained or learnt simultaneously with the best response networks directly from gameplay data (Fig. 1b). Coupled with a shared game model network, the best response networks of players can be trained to approximate best responses to their opponent's belief densities ($\bar{\sigma}_{-p}$) (Fig. 1c). The training procedure is discussed in detail in Sect. 3.3.

When the expected reward is not differentiable w.r.t. players' actions or the derivatives are zero in large parts of the action space, DeepFP can also employ

approximate best response oracle (BRO_p) for player p. The oracle can be a non-differentiable approximation algorithm employing Linear Programming (LP) or Mixed Integer Programming (MIP) and since it will not be trained, it can also be deterministic. In many security games, Mixed-integer programming based algorithms are proposed to compute best responses and our algorithm provides a novel way to incorporate them as subroutines in a deep learning framework, as opposed to most existing works which require end-to-end differentiable policy networks and cannot utilize non-differentiable solutions even when available.

3.3 DeepFP

Algorithm 1 shows the DeepFP pseudocode. DeepFP randomly initializes any best response networks and game model network (if needed) and declares an empty memory (`mem`) to store players' actions and rewards [lines 1–2].

Algorithm 1. DeepFP

Data: `max_games`, batch sizes (m_1, m_2, m_G), memory size E, game simulator and oracle BRO_p for players with no gradient
Result: Final belief densities $\bar{\sigma}_p^*$ in `mem` $\forall$ players p

1 Initialize all network parameters $(\theta_1, \theta_2, \phi)$ randomly;
2 Create empty memory `mem` of size E;
3 **for** *game* $\in \{1, \ldots,$ `max_games`$\}$ **do**
  `/* Obtain best responses */`
4 **for** *each player p* **do**
5 **if** *grad_avlbl(p)* **then**
6 Sample $z_p \sim \mathcal{N}(0, I)$;
7 Approx. best response $u_p = BR_p(z_p; \theta_p)$;
8 **else**
9 $u_p = BRO_p(\bar{\sigma}_{-p})$ with $\bar{\sigma}_{-p}$ from `mem`;
  `/* Play game and update memory */`
10 Play with $u = \{u_1, u_2\}$ to get $r = \{r_1, r_2\}$;
11 Store sample $\{u, r\}$ in `mem`;
  `/* Train shared game model net */`
12 **if** *grad_avlbl(p) for any* $p \in \{1, 2\}$ **then**
13 Draw samples $\{u^i, r^i\}_{i=1:m_G}$ from `mem`;
14 $\phi :=$ Adam.min$(L_{MSE}, \phi, \{u^i, r^i\}_{i=1:m_G})$;
  `/* Train best response nets */`
15 **for** *each player p with grad_avlbl(p)* **do**
16 Draw samples $\{u^i\}_{i=1:m_p}$ from `mem`;
17 $\theta_p :=$ Adam.min$(L_{r_p}, \theta_p, \{u^i_{-p}\}_{i=1:m_p})$;

Then it iteratively makes both players best respond to the belief density of their opponent. This best response can be computed per player p via a forward

pass of the best response network BR_p or via a provided oracle BRO_p or if gradients are not available [lines 4–9]. The best response moves and the rewards obtained by playing them are stored in `mem` [lines 10–11]. Samples from *exact* belief density $\bar{\sigma}$ of both players are available from `mem`.

The game model network is also trained simultaneously to learn a differentiable reward model of the game [lines 12–14]. It takes all players' actions u as input and predicts the game rewards $\hat{r}(u;\phi)$ for all players. Its parameters ϕ can be learnt by minimizing the mean square error loss over a minibatch of samples $\{u^i\}_{i=1:m_G}$ from `mem`, using any optimizer (we use Adam [16]):

$$L_{MSE}(\phi) = \frac{1}{2m_G} \sum_{p\in\{1,2\}} \sum_{i=1}^{m_G} (\hat{r}_p(u^i;\phi) - r_p^i)^2.$$

The advantage of estimating this differentiable reward model independent of playing strategies is that it can be trained from the data in replay memory without requiring importance sampling, hence it can be used as a proxy for the game simulator to train the best response networks. An alternative could be to replay past actions of players using the game simulator as done by [15], but it is much slower (see Sect. 4.2).

Finally each player updates their best response network to keep it a reasonable approximation to the best response to his opponent's belief density [lines 15–17]. For this, each player p maximizes his expected predicted reward $\hat{r}_p$ (or minimizes expected $-\hat{r}_p$) against the opponent's belief density $\bar{\sigma}_{-p}$ (see Fig. 1c) using any optimizer (we use Adam):

$$L_{r_p}(\theta_p) = -\mathbb{E}_{(z_p\sim\mathcal{N}(0,I), u_{-p}\sim\bar{\sigma}_{-p})}[\hat{r}_p(BR_p(z_p;\theta_p), u_{-p};\phi)].$$

The expectation is approximated using a minibatch of samples $\{u^i_{-p}\}_{i=1:m_p}$ drawn from `mem` and $\{z^i_p\}_{i=1:m_p}$ independently sampled from a standard normal distribution. In this optimization, ϕ is held constant and the gradient is only evaluated w.r.t. θ_p and the updates applied to the best response network. In this sense, the game model network acts like a critic to evaluate the best responses of player p (actor) against his opponent's belief density $\bar{\sigma}_{-p}$ similar to actor-critic methods [22]. However, unlike actor-critic methods we train the best response and the game model networks in separate decoupled steps which potentially allows replacing them with pre-trained models or approximate oracles, while skipping their respective learning steps.

3.4 Connections to Boltzmann Actor-Critic and Convergence of DeepFP

DeepFP is closely related to the Boltzmann actor-critic process proposed by Generalized Weakened Fictitious Play (GWFP) [20], which converges to the NE under certain assumptions. But it differs in two crucial aspects: (i) GWFP requires assuming explicit probability densities and involves weakened ϵ-best

responses which are updated via a Boltzmann actor-critic process. Since we store the empirical belief densities and best responses as implicit densities, a Boltzmann-style strategy update is infeasible, (ii) GWFP also requires the ϵ-best responses to eventually become exact (i.e. when $\lim_{n\to\infty} \epsilon_n \to 0$). Since we are approximating stochastic best responses via generative neural networks (or with approximate oracles), this assumption may not always hold exactly. Nevertheless, with our approximate best responses and with the one-step gradient updates best response networks, we empirically observed that DeepFP converges for multiple games with continuous reward functions wherever GWFP converges. At convergence, the belief density $\bar{\sigma}^*$ in `mem` is a non-parametric approximation to a NE density for both players.

4 Experimental Evaluation

4.1 Simple Games

We first evaluate DeepFP on two simple games where traditional fictitious play is known to converge, as potential sanity checks and to demonstrate convergence to nash equilibrium.

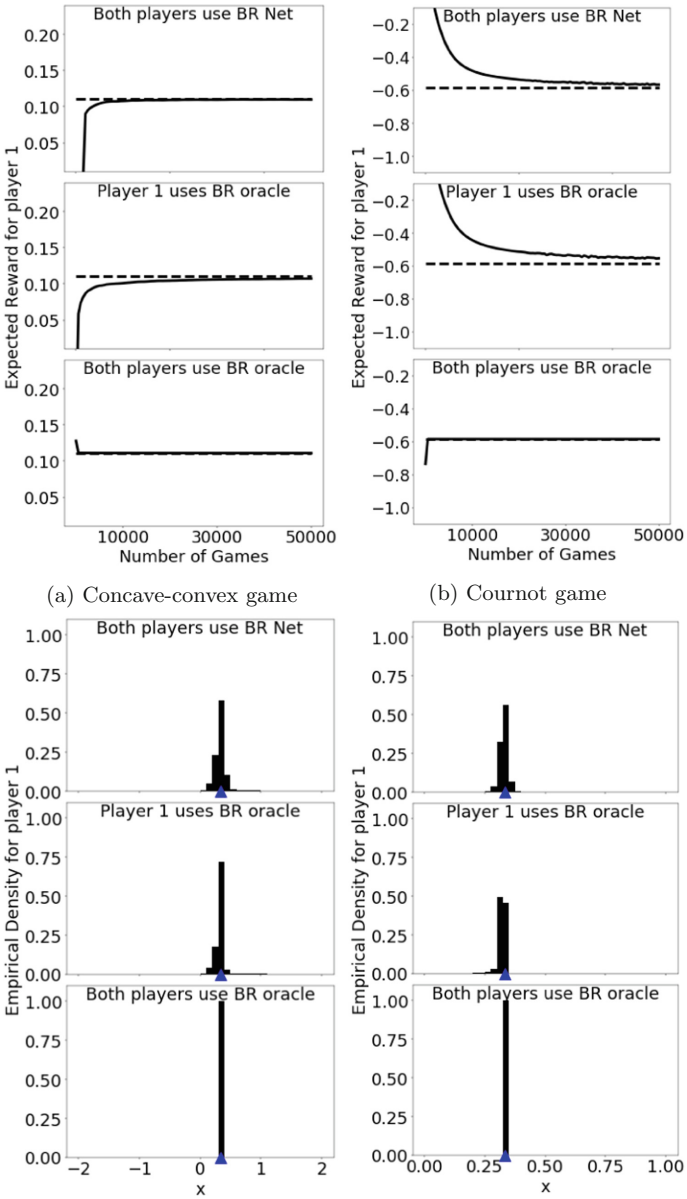
Concave-Convex Game: Two players 1 and 2 with scalar actions $x, y \in [-2, 2]$ respectively play to maximize their rewards: $r_1(x, y) = -2x^2 + 4xy + y^2 - 2x - 3y + 1$ and $r_2(x, y) = -r_1(x, y)$. The game is concave w.r.t. x and convex w.r.t. y and admits a pure strategy NE which can be computed using standard calculus. The NE strategies are $x = 1/3, y = 5/6$, the expected equilibrium rewards are $r_1^* = -r_2^* = -7/12$ and the best responses of players to each others' average strategies are $BR_1(\bar{y}) = \bar{y} - 1/2$ and $BR_2(\bar{x}) = 3/2 - 2\bar{x}$.

Cournot Game: It is a classic game [7] with two competing firms (1 and 2) producing a quantity ($q_1 \geq 0$ and $q_2 \geq 0$ resp.) of a product. The price of the product is $p(q_1, q_2) = a - q_1 - q_2$ and the cost of manufacturing quantity q is $C(q) = cq$, where $c, a > 0$ are constants. Reward for a firm p is $R_p(q_1, q_2) = (a - q_1 - q_2)q_p - cq_p$, $p \in \{1, 2\}$ and the best response against the competing firm's choice can be analytically computed as q_{-p} is $BR_p(q_{-p}) = \frac{a-c-q_{-p}}{2}$. The NE strategy can be computed as $q_1^* = q_2^* = \frac{a-c}{3}$. We use $a = 2$ and $c = 1$ for our experiment so that $q_1^* = q_2^* = 1/3$.

Figure 2 shows the results of DeepFP to these games and its convergence to the NE for all variants i.e. when both, exactly one, or no player uses the best response oracle. Note that both players using the best response oracle (bottom case in all subfigures) is the same as exact fictitious play and converges very fast as opposed to other cases (top and mid in all subfigures) since the latter variants require estimating the best responses from repeated gameplay.

4.2 Forest Protection Game

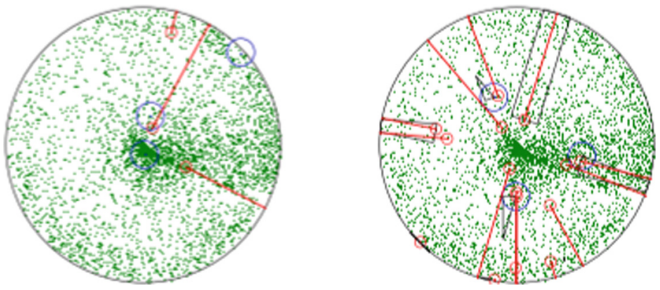
For a large application of DeepFP, we choose the forest protection game as proposed by [15] with a Defender (D) and an Adversary (A). Consider a circular



(a) Concave-convex game (b) Cournot game

(c) Concave-convex game (d) Cournot game

Fig. 2. DeepFP on simple games under three settings: when both players learn BR nets (top), player 1 uses BR oracle (mid), and when both players use BR oracle (bottom); (a) and (b) expected reward of player 1 converges to the true equilibrium value (shown by dashed line) for both games; (c) and (d) final empirical density for player 1 approaches NE strategy for both games (shown by blue triangle on horizontal axis).



(a) (b)

Fig. 3. Forest game with trees (green dots), guards (blue dots), guard radii R_g (blue circles), lumberjacks (red dots), lumberjack chopping radii R_l (red circles), lumberjacks' paths (red lines) and black polygons (top weighted capture-sets for guards): (a) with $\mathrm{m} = \mathrm{n} = 3$, (b) best response oracle for 3 guards and 15 lumberjacks. (Color figure online)

forest with an arbitrary tree distribution. The adversary has n lumberjacks who cross the boundary and move straight towards the forest center. They can stop at any point on their path, chop trees in a radius R_l around the stopping point and exit back from their starting location. The adversary's action is then all the stopping points for lumberjacks (which fully specifies their trajectories). The defender has m guards whose locations in the forest can be chosen to ambush the lumberjacks. A lumberjack whose trajectory comes within R_g distance from any guard's location is considered ambushed and loses all his chopped trees and bears a penalty r_{pen}. The final reward for adversary ($r_A \in \mathbb{R}$) is the number of trees jointly stolen by the lumberjacks plus the total negative penalty incurred. The defender's reward is $r_D = -r_A$. A full game is shown in Fig. 3a. In our experiments we use the following settings for the game: $r_{pen} = 4.0, R_g = 0.1, R_l = 0.04$.

Approximate Best Response Oracle: Note that if guards' locations do not overlap significantly with those of lumberjacks then changing them by a small amount does not affect the rewards for either player since no extra lumberjacks are ambushed. Hence, the gradient of reward w.r.t. defender's parameters ($\nabla_{\theta_D} r$) ≈ 0 over most of the action space. But the gradients for the adversary are continuous and non-zero because of the dense tree distribution. Hence we apply DeepFP to this game with a best response network for the adversary and an approximate domain-specific best response oracle for the defender. Devising a defender's best response to the adversary's belief distribution is non-trivial for this game, and we propose a greedy approximation for it[1]. Briefly, the oracle algorithm involves creating capture-sets for lumberjack locations l encountered so far in `mem` and intersecting these capture-sets to find those which cover multiple lumberjacks. Then it greedily allocates guards to the top m such capture-sets one

[1] The full oracle algorithm and the involved approximations are detailed in the appendix to keep the main text concise and continuous.

at a time, while updating the remaining capture-sets simultaneously to account for the lumberjacks ambushed by the current guard allocation. We illustrate an oracle best response in Fig. 3b.

Baselines: Since the forest protection game involves arbitrary tree density patterns, the ground truth equilibria are intractable. So we evaluate DeepFP by comparing it with OptGradFP [15] and to another approximate discrete linear programming method (henceforth called DLP).

DLP Baseline: We propose DLP which discretizes the action space of players and solves a linear programming problem to solve the game approximately (but only for small m and n). The DLP method discretizes the action space in cylindrical coordinates with 20 radial bins and 72 angular bins, which gives a joint action space of size $(72 \times 20)^{m+n}$. For even a single guard and lumberjack, this implies about 2 million pure strategies. Hence, though DLP gives the approximate ground truth for $\mathrm{m} = \mathrm{n} = 1$ due to our fine discretization, going beyond m or $n > 1$ is infeasible with DLP. The DLP baseline proceeds in two steps:

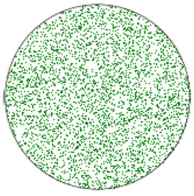
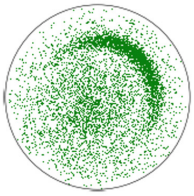
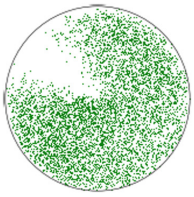
1. We generate $72 \times 20 = 1440$ cylindrically discretized bins and compute a matrix $R \in \mathbb{R}^{1440 \times 1440}$ where R_{ij} characterizes the defender's reward with a guard in the i-th bin and a lumberjack in the j-th bin. Each entry R_{ij} is computed by averaging the game simulator's reward over 20 random placements of the guard and lumberjack inside the bins.
2. Next we solve the following optimization problem for the defender:

$$\sigma^*, \chi^* = \arg \max_{\sigma \geq 0, \chi} \chi$$
$$s.t.\ \sigma^T R_{:j} \geq \chi \, \forall j$$
$$\sum_{i=1}^{1440} \sigma_i = 1$$

Note that χ represents the defender's reward, σ_i is the i-th element of $\sigma \in [0,1]^{1440}$ i.e. the probability of placing the guard in the i-th bin and $R_{:j}$ is the j-th column of R corresponding to the adversary taking action j. The above problem maximizes the defender's reward subject to the constraints that σ has all non-negative elements summing to 1 (since it's a distribution over all bins) and the defender's reward χ is least exploitable regardless of the adversary's placement in any bin j. Solving it gives us the optimal defender distribution σ^* over all bins to place the guard and the equilibrium reward for the defender χ^* when $\mathrm{m} = \mathrm{n} = 1$.

Fixed Hyperparameters: We set `max_games` $= E = 40000$ to provide enough iterations to DeepFP and OptGradFP for convergence. The batch sizes for DeepFP are set to $m_D = 3$ (kept small to have a fast oracle), $m_A = 64, m_G = 128$ (large for accurate gradient estimation). For full neural network architectures used, please refer to the appendix.

Table 1. Results on four representative forests for m = n = 1. Green dots: trees, blue dots: guard locations sampled from defender's strategy, red dots: lumberjack locations sampled from adversary's strategy. The exploitability metric shows that DLP which is approximately the ground truth NE strategy is the least exploitable followed by DeepFP, while OptGradFP's inflexible explicit strategies make it heavily exploitable.

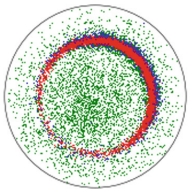
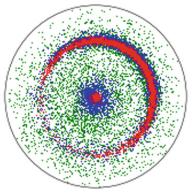
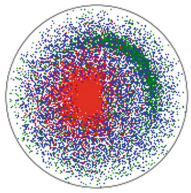
Forest structure	DeepFP	OptGradFP	DLP (approx. ground truth)
 F1	$\epsilon = 88.57 \pm 23.1$	$\epsilon = 174.08 \pm 21.04$	$\epsilon = 95.60 \pm 10.82$
 F2	$\epsilon = 16.90 \pm 0.13$	$\epsilon = 17.09 \pm 0.39$	$\epsilon = 16.38 \pm 0.86$
 F3	$\epsilon = 88.72 \pm 24.09$	$\epsilon = 115.02 \pm 0.86$	$\epsilon = 72.95 \pm 1.21$
 F4	$\epsilon = 30.72 \pm 1.65$	$\epsilon = 32.21 \pm 0.52$	$\epsilon = 23.97 \pm 0.64$

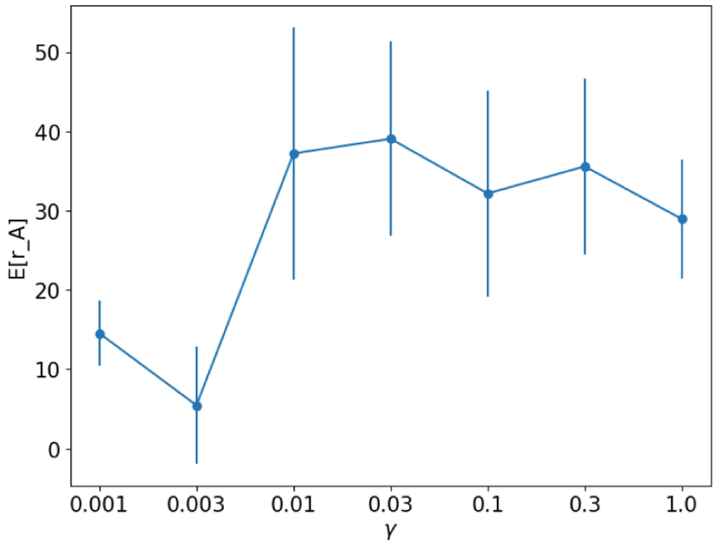
Exploitability Analysis: Since direct computation of the ground truth equilibrium is infeasible for a forest, we compare all methods by evaluating the exploitability of the defender's final strategy as NE strategies are least exploitable. For this, we designed an evolutionary algorithm to compute the adversary's best response to the defender's final strategy. It maintains a population (size 50) of adversary's actions and iteratively improves it by selecting

the best 10 actions, duplicating them four-fold, perturbing the duplicate copies with gaussian noise (whose variance decays over iterations) and re-evaluating the population against the defender's final strategy. This evolutionary procedure is independent of any discretization or neural network and outputs the adversary action which exploits the defender's final strategy most heavily. We denote the reward achieved by the top action in the population as the exploitability ϵ and report the exploitability of the defender's strategy averaged across 5 distinct runs of each method (differing only in the initial seed). Since rewards can differ across forests due to the number of trees in the forest and their distribution, the exploitability of each forest can differ considerably. Also, since the evolutionary algorithm requires $150K$–$300K$ game plays per run, it is quite costly and only feasible for a single accurate post-hoc analysis rather than using it to compute best responses within DeepFP.

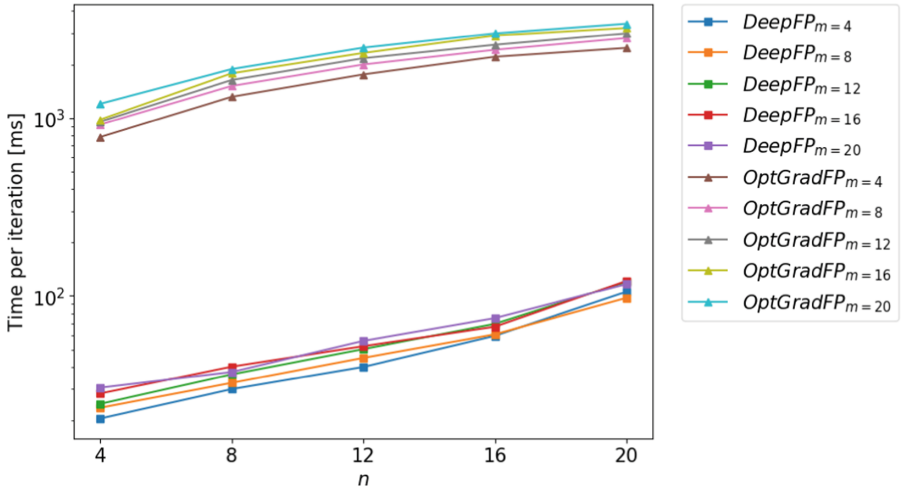
Single Resource Case: Table 1 shows results on four representative forests when $m = n = 1$. We observe that both DLP and DeepFP find strategies which intuitively cover dense regions of the forest (central forest patch for F1, nearly the whole forest for uniform forest F2, dense arch of trees for F3 and ring for the forest F4 with a tree-less sector). On the uniform forest F2, the expected NE strategy is a ring at a suitable radius from the center, as outputted by DeepFP. However, DLP has a fine discretization and is able to sense minute deviations from uniform tree structure induced due to the sampling of trees from a uniform distribution, hence it forms a circular ring broken and placed at different radii. A similar trend is observed on F4. On F3, DeepFP finds a strategy strongly covering the dense arch of trees, similar to that of DLP. Note that sometimes DeepFP even finds less exploitable strategies than DLP (e.g. on F1), since DLP while being close to the ground truth still involves an approximation due to discretization. Overall, as expected DLP is in general the least exploitable method and is the closest to the NE, followed by DeepFP. OptGradFP is more exploitable than DeepFP for nearly uniform tree densities (F2 and F4) and heavily exploitable

Table 2. Results on forest F3 for $m = n = \{2, 3\}$. Green dots: trees, blue dots: guard locations sampled from defender's strategy, red dots: lumberjack locations sampled from adversary's strategy. DeepFP is always less exploitable than OptGradFP.

DeepFP (m=n=2)	DeepFP (m=n=3)	OptGradFP (m=n=2)	OptGradFP (m=n=3)
			
$\epsilon = 135.49 \pm 15.24$	$\epsilon = 137.53 \pm 8.63$	$\epsilon = 186.58 \pm 23.71$	$\epsilon = 190.00 \pm 23.63$



(a)



(b)

Fig. 4. (a) Adversary's average reward with memory size E as a fraction of total games played. Even for a 1% fraction of memory size i.e. $\gamma = 0.01$, the average rewards are close to $\gamma = 1$ case. (b) Time per iteration vs. players' resources. DeepFP is orders of magnitude faster than OptGradFP (y-axis has log scale).

for forests with concentrated tree densities (F1 and F3), since unlike DeepFP, it is unable to approximate arbitrary strategy shapes.

Multiple Resource Case: Since DLP cannot scale for m or $n > 1$, we compute the strategies and exploitability for $\mathrm{m} = \mathrm{n} = \{2, 3\}$ on F3 in table 2 for DeepFP and OptGradFP only (more forests in appendix). We consistently observe that DeepFP accurately covers the dense forest arch of F3 and OptGradFP spreads both players out more uniformly (due to explicit strategies). For $\mathrm{m} = \mathrm{n} = 3$ case, DeepFP also allots a guard to the central patch of F3. Overall, DeepFP is substantially less exploitable than OptGradFP.

Effect of Memory Size: In Algorithm 1, we stored and best responded to all games in the replay memory. Figure 4a shows the expected reward ($\mathbb{E}[r_A]$) achieved by the adversary's final strategy against the defender's final strategy, when the replay memory size E is varied as a fraction γ of `max_games`. Only the most recent γ fraction of `max_games` are stored and best responded to, and the previous ones are deleted from `mem`. We observe that DeepFP is fairly robust to memory size and even permits significantly small replay memories (upto 0.01 times `max_games`) without significant deterioration in average rewards.

Running Time Analysis: Given the same total number of iterations, we plot the time per iteration for DeepFP and OptGradFP in Fig. 4b with increasing m and n (y-axis has log scale). OptGradFP's training time increases very fast with increasing m and n due to high game replay time. With our approximate best-response oracle and estimation of payoffs using the game model network, DeepFP is orders of magnitude faster. For a total 40K iterations, training DeepFP takes about 0.64 ± 0.34 h (averaged over values of m and n) as opposed to 22.98 ± 8.39 h for OptGradFP.

5 Conclusion

We have presented DeepFP, an approximate fictitious play algorithm for games with continuous action spaces. DeepFP implicitly represents players' best responses via generative neural networks without prior shape assumptions and optimizes them using a learnt game-model network with gradient-based training. It can also utilize approximate best response oracles whenever available, thereby harnessing prowess in approximation algorithms from discrete planning and operations research. DeepFP provides significant speedup in training time and scales well with growing number of resources.

DeepFP can be easily extended to multi-player applications, with each player best responding to the joint belief density over all other players using an oracle or a best response network. Like most gradient-based optimization algorithms, DeepFP and OptGradFP can sometimes get stuck in local nash equilibria (see appendix for experiments). While DeepFP gets stuck less often than OptGradFP, principled strategies to mitigate local optima for gradient-based equilibrium finding methods remains an interesting direction for future work.

Acknowledgments. This research was supported in part by NSF Research Grant IIS-1254206, NSF Research Grant IIS-1850477 and MURI Grant W911NF-11-1-0332.

A Appendix

A.1 Approximate Best Response Oracle for Forest Protection Game

Algorithm 2. Approximate best response oracle

Data: `mem`, batch size m_D, game simulator, m, n
Result: Guard assignments approximating $BRO_D(\bar{\sigma}_A)$
1 Draw batch of adversary actions $\{u_A^i\}_{i=1:m_D}$ from $\bar{\sigma}_A$ (stored in `mem`);
2 Extract all $m_D \times n$ lumberjack locations $l \in \{u_A^i\}_{i=1:m_D}$;
`/* Capture-set for each lumberjack */`
3 Initialize empty capture-set list S;
4 **for** $l \in \{u_A^i\}_{i=1:m_D}$ **do**
5 Create a capture-set $s(l)$ (approximated by a convex polygon) i.e. as the set of all guard locations which are within radius R_g from any point on the trajectory of the lumberjack stopping at l;
6 Query reward $w(l)$ of ambushing at l (using simulator);
7 Append (s, w, l) to S.
`/* Output max reward capture-sets */`
8 Find all possible intersections of sets $s \in S$ while assigning a reward $w' = \sum_j w_j$ and lumberjacks $l' = \cap_j l_j$ to $s' = \cap_j s_j$ and append all new (s', w', l') triplets to S;
9 Pop the top m maximum reward sets in S one at a time and assign a single guard to each, while updating all remaining sets' weights to remove lumberjacks covered by the guard allotment;
10 Output the guard assignments.

Devising a defender's best response to the adversary's belief distribution is non-trivial for this game. So we propose a greedy approximation to the best response. (see Algorithm 2). We define a capture-set for a lumberjack location l as the set of all guard locations within a radius R_g from any point on the trajectory of the lumberjack. The algorithm involves creating capture-sets for lumberjack locations l encountered so far in `mem` and intersecting these capture-sets to find those which cover multiple lumberjacks. Then it greedily allocates guards to the top m such capture-sets one at a time, while updating the remaining capture-sets simultaneously to account for the lumberjacks ambushed by the current guard allocation. Our algorithm involves the following approximations:

1. *Mini-batch approximation*: Since it is computationally infeasible to compute the best response to the full set of actions in `mem`, we best-respond to a small mini-batch of actions sampled randomly from `mem` to reduce computation (line 1).

2. *Approximate capture-sets*: Initial capture-sets can have arbitrary arc-shaped boundaries which can be hard to store and process. Instead, we approximate them using convex polygons for simplicity (line 5). Doing this ensures that all subsequent intersections also result in convex polygons.
3. *Bounded number of intersections*: Finding all possible intersections of capture-sets can be reduced to finding all cliques in a graph with capture-sets as vertices and pairwise intersections as edges. Hence it is an NP-hard problem with complexity growing exponentially with the number of polygons. We compute intersections in a pairwise fashion while adding the newly intersected polygons to the list. This way the k^{th} round of intersection produces uptil all $k+1$-polygon intersections and we stop after $k=4$ rounds of intersection to maintain polynomial time complexity (implemented for line 8, but not shown explicitly in Algorithm 2).
4. *Greedy selection*: After forming capture-set intersections, we greedily select the top m sets with the highest rewards (line 9).

A.2 Supplementary Experiments with m, n > 1

Table 3. More results on forests F1 and F4 for m = n = 2.

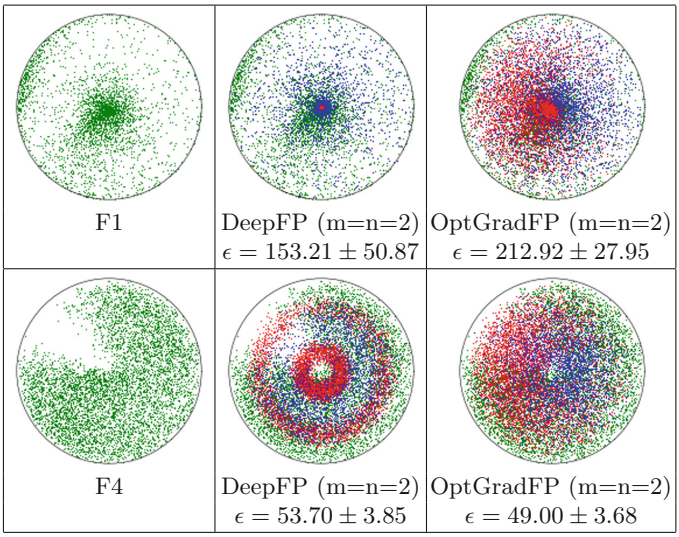
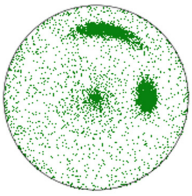
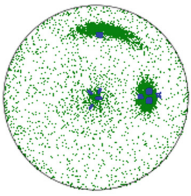
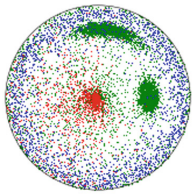
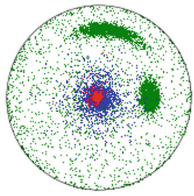
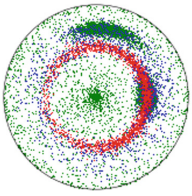
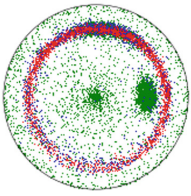
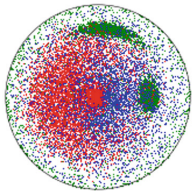
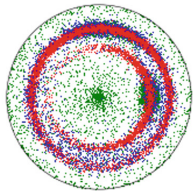


Table 3 shows more experiments for DeepFP and OptGradFP with m,n>1. We see that DeepFP is able to cover regions of importance with the players' resources but OptGradFP suffers from the zero defender gradients issue due to logit-normal strategy assumptions which often lead to sub-optimal results and higher exploitability.

Table 4. Demonstrating getting stuck in locally optimal strategies.

 F5	 C1: DLP (m=n=1)	 C2: OptGradFP (m=n=1)	 C3: OptGradFP (m=n=1)
 C4: DeepFP (m=n=1)	 C5: DeepFP (m=n=1)	 C6: OptGradFP (m=n=3)	 C7: DeepFP (m=n=3)

A.3 Locally Optimal Strategies

To further study the issue of getting stuck in locally optimal strategies we show experiments with another forest F5 in Table 4. F5 has three dense tree patches and very sparse and mostly empty other parts. The optimal defender's strategy computed by DLP for $m = n = 1$ is shown in C1. In such a case, due to the tree density being broken into patches, gradients for both players would be zero at many locations and hence both algorithms are expected to get stuck in locally optimal strategies depending upon their initialization. This is confirmed by configurations C2, C3, C4 and C5 which show strategies for OptGradFP and DeepFP with $m = n = 1$ covering only a single forest patch. Once the defender gets stuck on a forest patch, the probability of coming out of it is small since the tree density surrounding the patches is negligible. However, with more resources for the defender and the adversary $m = n = 3$, DeepFP is mostly able to break out of the stagnation and both players eventually cover more than a single forest patch (see C7), whereas OptGradFP is only able to cover additional ground due to random initialization of the 3 player resources but otherwise remains stuck around a single forest patch (see C6). DeepFP is partially able to break out because the defender's best response does not rely on gradients but rather come from a non-differentiable oracle. This shows how DeepFP can break out of local optima even in the absence of gradients if a best response oracle is provided, however OptGradFP relies purely on gradients and cannot overcome such situations.

A.4 Neural Network Architectures

All our models were trained using TensorFlow v1.5 on a Ubuntu 16.04 machine with 32 CPU cores and a Nvidia Tesla K40c GPU.

Cournot Game and Concave-Convex Game. Best response networks for the Cournot game and the Concave-convex game consist of single fully connected layer with a sigmoid activation, directly mapping the 2-D input noise $z \sim \mathcal{N}([0,0], I_2)$ to a scalar output q_p for player p. Best response networks are trained with Adam optimizer [16] and learning rate of 0.05. To estimate payoffs, we use exact reward models for the game model networks. Maximum games were limited to 30,000 for Cournot game and 50,000 for Concave-convex game.

Forest Protection Game. The action u_p of player p contains the cylindrical coordinates (radii and angles) for all resources of that player. So, the best response network for the Forest protection game maps $Z_A \in \mathbb{R}^{64}$ to the adversary action $u_A \in \mathbb{R}^{n \times 2}$. It has 3 fully connected hidden layers with $\{128, 64, 64\}$ units and ReLU activations. The final output comes from two parallel fully connected layers with n (number of lumberjacks) units each: (a) first with sigmoid activations outputting n radii $\in [0, 1]$, and (b) second with linear activations outputting n angles $\in [-\infty, \infty]$, which are modulo-ed to be in $[0, 2\pi]$ everywhere. All layers are L2-regularized with coefficient 10^{-2}:

$$x_A = relu(FC_{64}(relu(FC_{64}(relu(FC_{128}(Z_A))))))$$
$$u_{A,rad} = \sigma(FC_n(x_A)); \quad u_{A,ang} = FC_n(x_A)$$

The game model takes all players' actions as inputs (i.e. matrices u_D, u_A of shapes $(m, 2)$ and $(n, 2)$) respectively) and produces two scalar rewards r_D and r_A. It internally converts the angles in the second columns of these inputs to the range $[0, 2\pi]$. Since the rewards should be invariant to the permutations of the defender's and adversary's resources (guards and lumberjacks resp.), we first pass the input matrices through non-linear embeddings to interpret their rows as sets rather than ordered vectors (see Deep Sets [29] for details). These non-linear embeddings are shared between the rows of the input matrix and are themselves deep neural networks with three fully connected hidden layers containing $\{60, 60, 120\}$ units and ReLU activations. They map each row of the matrices into a 120-dimensional vector and then add all these vectors. This effectively projects the action of each player into a 120-dimensional action embedding representation invariant to the ordering of the resources. The players' embedding networks are trained jointly as a part of the game model network. The players' action embeddings are further passed through 3 hidden fully connected layers with $\{1024, 512, 128\}$ units and ReLU activations. The final output rewards are produced by a last fully connected layer with 2 hidden units and linear activation. All layers are L2-regularized with coefficient 3×10^{-4}:

$$emb_p = \sum_{dim=row} (DeepSet_{60,60,120}(u_p)) \quad \forall p \in \{D, A\}$$
$$\hat{r}_D, \hat{r}_A = FC_2(relu(FC_{128}(relu(FC_{512}(relu(FC_{1024}(emb_D, emb_A)))))))$$

The models are trained with Adam optimizer [16]. Note that the permutation invariant embeddings are not central to the game model network and only help to incorporate an inductive bias for this game. We also tested the game model network without the embedding networks and achieved similar performance with about 2x increase in the number of iterations since the game model would need to infer permutation invariance from data.

References

1. Amin, K., Singh, S., Wellman, M.P.: Gradient methods for stackelberg security games. In: UAI, pp. 2–11 (2016)
2. Balduzzi, D., Racaniere, S., Martens, J., Foerster, J., Tuyls, K., Graepel, T.: The mechanics of n-player differentiable games. In: International Conference on Machine Learning (2018)
3. Basilico, N., Celli, A., De Nittis, G., Gatti, N.: Coordinating multiple defensive resources in patrolling games with alarm systems. In: Proceedings of the 16th Conference on Autonomous Agents and Multiagent Systems, pp. 678–686 (2017)
4. Behnezhad, S., Derakhshan, M., Hajiaghayi, M., Seddighin, S.: Spatio-temporal games beyond one dimension. In: Proceedings of the 2018 ACM Conference on Economics and Computation, pp. 411–428 (2018)
5. Cermák, J., Bošanský, B., Durkota, K., Lisý, V., Kiekintveld, C.: Using correlated strategies for computing stackelberg equilibria in extensive-form games. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI 2016, pp. 439–445 (2016)
6. Fang, F., Jiang, A.X., Tambe, M.: Optimal patrol strategy for protecting moving targets with multiple mobile resources. In: AAMAS, pp. 957–964 (2013)
7. Ferguson, T.S.: Game Theory, vol. 2 (2014). https://www.math.ucla.edu/~tom/Game_Theory/Contents.html
8. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. arXiv preprint arXiv:1703.03400 (2017)
9. Gan, J., An, B., Vorobeychik, Y., Gauch, B.: Security games on a plane. In: AAAI, pp. 530–536 (2017)
10. Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. arXiv preprint arXiv:1801.01290 (2018)
11. Haskell, W., Kar, D., Fang, F., Tambe, M., Cheung, S., Denicola, E.: Robust protection of fisheries with compass. In: IAAI (2014)
12. Heinrich, J., Lanctot, M., Silver, D.: Fictitious self-play in extensive-form games. In: International Conference on Machine Learning, pp. 805–813 (2015)
13. Johnson, M.P., Fang, F., Tambe, M.: Patrol strategies to maximize pristine forest area. In: AAAI (2012)
14. Kamra, N., Fang, F., Kar, D., Liu, Y., Tambe, M.: Handling continuous space security games with neural networks. In: IWAISe: First International Workshop on Artificial Intelligence in Security (2017)

15. Kamra, N., Gupta, U., Fang, F., Liu, Y., Tambe, M.: Policy learning for continuous space security games using neural networks. In: AAAI (2018)
16. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
17. Korzhyk, D., Yin, Z., Kiekintveld, C., Conitzer, V., Tambe, M.: Stackelberg vs. Nash in security games: an extended investigation of interchangeability, equivalence, and uniqueness. JAIR **41**, 297–327 (2011)
18. Krishna, V., Sjöström, T.: On the convergence of fictitious play. Math. Oper. Res. **23**(2), 479–511 (1998)
19. Lanctot, M., et al.: A unified game-theoretic approach to multiagent reinforcement learning. In: Advances in Neural Information Processing Systems, pp. 4190–4203 (2017)
20. Leslie, D.S., Collins, E.J.: Generalised weakened fictitious play. Games Econ. Behav. **56**(2), 285–298 (2006)
21. Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, O.P., Mordatch, I.: Multi-agent actor-critic for mixed cooperative-competitive environments. In: Advances in Neural Information Processing Systems, pp. 6379–6390 (2017)
22. Mnih, V., et al.: Asynchronous methods for deep reinforcement learning. In: International Conference on Machine Learning, pp. 1928–1937 (2016)
23. Perkins, S., Leslie, D.: Stochastic fictitious play with continuous action sets. J. Econ. Theory **152**, 179–213 (2014)
24. Rosenfeld, A., Kraus, S.: When security games hit traffic: optimal traffic enforcement under one sided uncertainty. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-2017, pp. 3814–3822 (2017)
25. Shamma, J.S., Arslan, G.: Unified convergence proofs of continuous-time fictitious play. IEEE Trans. Autom. Control **49**(7), 1137–1141 (2004)
26. Wang, B., Zhang, Y., Zhong, S.: On repeated stackelberg security game with the cooperative human behavior model for wildlife protection. In: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, pp. 1751–1753 (2017)
27. Yang, R., Ford, B., Tambe, M., Lemieux, A.: Adaptive resource allocation for wildlife protection against illegal poachers. In: AAMAS (2014)
28. Yin, Y., An, B., Jain, M.: Game-theoretic resource allocation for protecting large public events. In: AAAI, pp. 826–833 (2014)
29. Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R.R., Smola, A.J.: Deep sets. In: Advances in Neural Information Processing Systems, pp. 3394–3404 (2017)