Finding Mixed-Strategy Equilibria of Continuous-Action Games without Gradients Using Randomized Policy Networks

Carlos Martin\(^1\) and Tuomas Sandholm\(^{1,2,3,4}\)

\(^1\)Carnegie Mellon University
\(^2\)Strategy Robot, Inc.
\(^3\)Optimized Markets, Inc.
\(^4\)Strategic Machine, Inc.

\{cgmartin, sandholm\}@cs.cmu.edu

Abstract

We study the problem of computing an approximate Nash equilibrium of continuous-action games without access to gradients. Such game access is common in reinforcement learning settings, where the environment is typically treated as a black box. To tackle this problem, we apply zeroth-order optimization techniques that combine smoothed gradient estimators with equilibrium-finding dynamics. We model players’ strategies using artificial neural networks. In particular, we use randomized policy networks to model mixed strategies. These take noise in addition to an observation as input and can flexibly represent arbitrary observation-dependent, continuous-action distributions. Being able to model such mixed strategies is crucial for tackling continuous-action games that lack pure-strategy equilibria. We evaluate the performance of our method using an approximation of the Nash convergence metric from game theory, which measures how much players can benefit from unilaterally changing their strategy. We apply our method to continuous Colonel Blotto games, single-item and multi-item auctions, and a visibility game. The experiments show that our method can quickly find a high-quality approximate equilibrium. Furthermore, they show that the dimensionality of the input noise is crucial for performance. To our knowledge, this paper is the first to solve general continuous-action games with unrestricted mixed strategies and without any gradient information.

1 Introduction

Most work on computing equilibria of games has focused on settings with finite, discrete action spaces. Yet many games involving space, time, money, etc. actually have continuous action spaces. Examples include continuous resource allocation games [Ganzfried, 2021], security games in continuous spaces [Kamra et al., 2017; Kamra et al., 2018; Kamra et al., 2019], network games [Ghosh and Kundu, 2019], simulations of military scenarios and wargaming [Marchesi et al., 2020], and most video games [Berner et al., 2019; Vinyals et al., 2019]. Furthermore, even if the action space is discrete, it may be fine-grained enough to treat as continuous for computational efficiency purposes [Borel, 1938; Chen and Ankenman, 2006; Ganzfried and Sandholm, 2010].

The typical approach to computing an equilibrium of a game with continuous action spaces involves discretizing the action space. That entails loss in solution quality [Kroer and Sandholm, 2015]. Also, it does not scale well; for one, in multidimensional action spaces it entails a combinatorial explosion of discretized points (exponential in the number of dimensions). Therefore, other approaches are called for. Furthermore, in many applications, explicit gradient information about the game is not available.

This paper is, to our knowledge, the first to solve general continuous-action games with unrestricted mixed strategies and without any gradient information. In §2, we introduce some background needed to formulate the problem. In §3, we describe related research that tackles the problem of computing an approximate equilibrium of such games. In §4, we describe our method and its components, including smoothed gradient estimators, equilibrium-finding dynamics, and representation of mixed strategies using randomized policy networks. (We use the terms policy and strategy interchangeably. The former is common in reinforcement learning, the latter in game theory.) In §5, we describe the various games that we use as benchmarks, and present our experimental results and discussion. In §6, we present our conclusions and suggest directions for future research.

2 Problem Description

First, we introduce some notation: \(\Delta X\) is the set of probability measures on \(X\), \(L(X)\) is the uniform probability measure on \(X\), and \([\cdot]\) is an Iverson bracket, which is 1 if its argument is true and 0 otherwise. A strategic-form game is a tuple \((I, S, u)\) where \(I\) is a set of players, \(S_i\) a set of strategies for player \(i\), and \(u_i : \prod_{j \neq i} S_j \rightarrow \mathbb{R}\) a utility function for player \(i\). A strategy profile \(s : \prod_{i \in I} S_i\) maps players to strategies and \(s_{-i}\) denotes \(s\) excluding \(i\)'s strategy. Player \(i\)'s best response utility \(b_i(s_{-i}) = \sup_{s_i \in S_i} u_i(s)\) is the highest utility they can attain given the other players’ strategies. Their utility gap \(g_i(s) = b_i(s_{-i}) - u_i(s)\) is the highest utility they can gain from unilaterally changing their strategy, and \(s\) is an \(\varepsilon\)-equilibrium iff \(\sup_{i \in I} g_i(s) \leq \varepsilon\). A 0-equilibrium is called a
Nash equilibrium. A common measure of closeness to Nash equilibrium is NashConv, defined as \( g = \int_{0}^{\mu} g_i \), where \( \mu \) is some measure on \( I \). Typically, \( I \) is finite and \( \mu \) is the counting measure, making \( g \) a finite sum of utility gaps. However, some games may have infinitely many players, such as mean field games. A game is called zero-sum if \( \int_{0}^{\mu} u_i = 0 \), which makes \( g = \int_{0}^{\mu} b_i \). In a two-player zero-sum game, \( g \) reduces to the so-called “duality gap” [Grnarova et al., 2019]:

\[
\hat{g}(s) = \sup_{\omega} u(s_1, s_2) - \inf_{\omega} u(s_1, s_2).
\]

In many games, the \( S_i \) are infinite. The following theorems apply to such games. If for all \( i, S_i \) is nonempty and compact, and \( u_i \) is continuous in \( s \), a mixed strategy Nash equilibrium exists [Glicksberg, 1952]. If for all \( i, S_i \) is nonempty, compact, and convex, and \( u_i \) is continuous in \( s \) and quasi-concave in \( s_i \), a pure strategy Nash equilibrium exists [Fudenberg and Tirole, 1991, p. 34]. Other results include the existence of a mixed strategy Nash equilibrium for games with discontinuous utilities under some mild semicontinuity conditions on the utility functions [Dasgupta and Maskin, 1986], and the uniqueness of a pure Nash equilibrium for continuous games under diagonal strict concavity assumptions [Rosen, 1965].

A Bayesian game is a game of incomplete information, that is, a game in which players have only partial information about the game and other players. Formally, it is a tuple \((I, \Omega, \mu, \tau, A, R)\), where \( I \) is a set of players, \( \Omega \) a set of states, \( \mu : \Delta \Omega \rightarrow [0, 1] \) a distribution over states, \( \tau \) a set of observations for \( i \), \( \tau_i : \Omega \rightarrow \Omega_i \) an observation function for \( i \), \( A_i \) a set of actions for \( i \), and \( r_i : \Omega \times \prod_{j \neq i} A_j \rightarrow \mathbb{R} \) a payoff function for \( i \). A strategy for player \( i \) is a function \( s_i : \Omega_i \rightarrow \Delta A_i \). Given strategy profile \( s \), player \( i \)'s expected payoff is \( u_i(s) = \mathbb{E}_{\omega \sim \mu} E_{A_j \sim s_j(\tau_j(\omega))} r_i(\omega, a) \) and their best response payoff is \( b_i(s^{-i}) = \sup_{s_i} u_i(s) = \sup_{s_i} \mathbb{E}_\omega E_a r_i(\omega, a) = \mathbb{E}_\omega \sup_{s_i} \mathbb{E}_\omega \mathbb{E}_a r_i(\omega, a) = \mathbb{E}_\omega \sup_{s_i} \mathbb{E}_\omega r_i(\omega, a) \), where \( \omega | a_i \) conditions \( \omega \) on player \( i \)'s observation being \( a_i \).

3 Related Research
McMahan et al. [2003] introduced the double oracle algorithm for normal-form games and proved its convergence. Adam et al. [2021] extended it to two-player zero-sum continuous games. Kroupa et al. [2023] extend it to \( n \)-player continuous games. Their algorithm maintains finite strategy sets for each player and iteratively extends them with best responses to an equilibrium of the induced finite sub-game. It “converges fast when the dimension of strategy spaces is small, and the generated subgames are not large.” For example, in the two-player zero-sum case: “The best responses were computed by selecting the best point of a uniform discretization for the one-dimensional problems and by using a mixed-integer linear programming reformulation for the Colonel Blotto games.” This approach does not scale to high-dimensional games with general payoffs where best-response computation is difficult. Moreover, if the game is stochastic, estimating the finite subgame can be difficult and require many samples. Furthermore, this approach does not learn observation-dependent strategies that generalize across observations.

Ganzfried et al. [2021] introduced an algorithm for approximating equilibria in continuous games called “redundant fictitious play” and apply it to a continuous Colonel Blotto game. Kamra et al. [2019] presented DeepFP, an approximate extension of fictitious play to continuous action spaces. They demonstrate stable convergence to equilibrium on several classic games and a large forest security domain. DeepFP represents players’ approximate best responses via generative neural networks. The authors state that such models cannot be trained directly in the absence of gradients, and thus employ a game-model network that is a differentiable approximation of the game’s payoff function, training these networks end-to-end in a model-based learning regime. Our approach shows, however, that these generative models can be trained directly.

Li et al. [2021] extended the double oracle approach to \( n \)-player general-sum continuous Bayesian games. They represent agents as neural networks and optimize them using natural evolution strategies (NES) [Wierstra et al., 2008; Wierstra et al., 2014]. To approximate a pure-strategy equilibrium, they formulate the problem as a bi-level optimization and employ NES to implement both inner-loop best response optimization and outer-loop regret minimization.

Bichler et al. [2021] represented strategies as neural networks and applied simultaneous gradients to provably learn local equilibria. They focus on symmetric auction models, assuming symmetric prior distributions and symmetric equilibrium bidding strategies. Bichler et al. [2023] extended this to asymmetric auctions, where one needs to train multiple neural networks. The previous two papers restrict their attention to pure strategies.

Fichl et al. [2022] computed distributional strategies [Milogrom and Weber, 1985] (a form of mixed strategies for Bayesian game) on a discretized version of the game via online convex optimization, specifically simultaneous online dual averaging, and show that the equilibrium of the discretized game approximates an equilibrium in the continuous game. That is, they discretize the type and action spaces and implement gradient dynamics in the discretized version of the game without using neural networks. In contrast, our approach does not use discretization, which can work well for small games but does not scale to high-dimensional observation and action spaces. In the appendix, we discuss additional related research.

4 Proposed Method
We now describe our game-solving technique.

4.1 Gradient Estimation
Consider the problem of maximizing \( f : \mathbb{R}^d \rightarrow \mathbb{R} \) with access to its values but not derivatives. This setting is called zeroth-order optimization. One approach to this problem is to compute estimates of the gradient \( g(x) \approx \nabla f(x) \) and apply gradient-based optimization. The gradient could be estimated via finite differences as \( g(x) = \frac{1}{2}(f(x + \sigma e_i) - f(x)) \) for all \( i \in [d] \), where \( e_i \) is the \( i \)th standard basis vector and \( \sigma \) is a small number. However, the number of queries needed scales linearly with the number of dimensions \( d \). Another approach is to evaluate the function at randomly-sampled points and estimate the gradient as a sum of estimates of directional derivatives along random directions [Duchi et al., 2015; Nesterov
4.2 Equilibrium-Finding Dynamics

Several gradient-based algorithms exist for finding equilibria in continuous games, as described in the appendix. Their convergence is analyzed in various works [Balduzzi et al., 2018; Letcher et al., 2019; Merikopulos and Zhou, 2019; Gmarova et al., 2019; Mazumdar et al., 2019; Hsieh et al., 2021]. In the games we tested, simultaneous gradient ascent was sufficient to obtain convergence to equilibrium and the other dynamics did not yield further improvements. Merikopulos et al. [2019] analyze the conditions under which simultaneous gradient ascent converges to Nash equilibria. They prove that, if the game admits a pseudoconcave potential or if it is monotone, the players’ actions converge to Nash equilibrium, no matter the level of uncertainty affecting the players’ feedback. Bichler et al. [2021] write that most auctions in the literature assume symmetric bidders and symmetric equilibrium bid functions [Krishna, 2002]. This symmetry creates a potential game, and simultaneous gradient dynamics provably converge to a pure local Nash equilibria in finite-dimensional continuous potential games [Mazumdar et al., 2020]. Thus in any symmetric and smooth auction game, symmetric gradient ascent with appropriate (square-summable but not summable) step sizes almost surely converges to a local ex-ante approximate Bayes-Nash equilibrium [Bichler et al., 2021, Proposition 1]. These results apply to most of our experiments, except for the asymmetric-information auction.

Algorithm 1 Distributed multiagent pseudogradient ascent

Input: $I$ is the set of players, $u$ is the utility function
initialize PRNG state with fixed seed
$x \leftarrow$ initial strategy profile
for $i \in I$ do
  $S_i \leftarrow$ init$(x_i)$ \Comment*[r]{$i$-th state of optimizer}
loop
  $J \leftarrow$ set of available workers
  for $j \in J$ do
    $a_j \leftarrow$ player $\in I$ \Comment*[r]{$i$-th player}
    $\varepsilon_j \leftarrow$ scale $\in \mathbb{R}_{>0}$ \Comment*[r]{$i$-th step size}
    $z_i \sim N(0, I_{\dim x_i})$ where $i = a_j$
  end
  $j \leftarrow$ own worker ID
  $\delta_j \leftarrow \frac{u(x_i + \varepsilon_j z_i, x_{-i}) - \varepsilon_j z_i}{2\sigma_j z_i}$ where $i = a_j$
  send $\delta_j$ to coordinator, receive $\delta$ from coordinator
for $i \in I$ do
  $K \leftarrow \{ j \in J \mid a_j = i \}$ \Comment*[r]{$i$-th set of workers}
  $v_i \leftarrow \frac{1}{|K|} \sum_{j \in K} \delta_j z_j$ \Comment*[r]{$i$-th gradient}
  $S_i, x_i \leftarrow$ step$(S_i, v_i)$ \Comment*[r]{$i$-th step optimizer}

4.3 Distributed Training Algorithm

Our algorithm for training strategy profiles can also be efficiently distributed, as we now describe. We present the pseudocode as Algorithm 1.

On any iteration, there is a set of available workers $J$. Each worker is assigned the task of computing a pseudogradient for a particular player. The vector $\{a_j\}_{j \in J}$ contains the assignment of a player for each worker. Each worker’s pseudorandom number generator (PRNG) is initialized with the same fixed seed. On any iteration, one of the workers is the coordinator. Initially, or when the current coordinator goes offline, the workers choose a coordinator by running a leader election algorithm. On each iteration, each worker evaluates the utility function (generally the most expensive operation and bottleneck for training) twice to compute the finite difference required for the pseudogradient. It then sends this computed finite difference (a single scalar) to the coordinator. The coordinator then sends the vector of these scalars to every worker, ensuring that all workers see each other’s scalars (an “allgather” operation). Thus the information that needs to be passed between workers is minimal. This greatly reduces the required cross-worker bandwidth compared to schemes that pass parameters or gradients between workers, which can be prohibitively expensive for large models.

This massively parallelizes Algorithm 1 of Bichler et al. [2021] (“NPGA using ES gradients”). Simultaneously, it generalizes Algorithm 2 of Salimans et al. [2017] (“Parallelized Evolution Strategies”), which also uses shared seeds, to the multiplayer setting, with separate gradient evaluations and optimizers for each player. Furthermore, it allows for the possibility of setting the worker-player assignments $a_j$ and perturbation noise scales $\varepsilon_j$ dynamically over time, provided that this is done consistently across workers (for example, based on their common state variables). Vanilla gradient descent, momentum gradient descent, optimistic gradient descent, or other optimization algorithms can be used. The
appendix contains some examples of such algorithms.

The set of available workers can also change dynamically over time. If a worker leaves or joins the pool, the coordinator notifies all workers of its ID so they can remove it from, or add it to, their $J$ sets. The new worker is brought up to speed by passing it the current PRNG state, strategy profile parameters, and optimizer states (what state information is needed depends on the algorithm used, for example, whether momentum is used).

4.4 Policy Representation

Another key design choice is how the players’ strategies are modeled. Bichler et al. [2021] model strategies using neural networks. Each player’s policy network takes as input a player’s observation and outputs an action. These policy networks were then trained using neural pseudogradient ascent, which uses Gaussian smoothing and applies simultaneous gradient ascent. As the authors note, their policy networks can only model pure strategies, since the output action is deterministic with respect to the input observation.

We also model strategies using neural networks, with one crucial difference: our policy network $f_0$ takes as input the player’s observation $o$ together with noise $z$ from some fixed latent distribution, such as the standard multivariate Gaussian distribution. Thus the output $a = f_0(o,z)$ of the network is random with respect to $o$. The network can then learn to transform this randomness into a desired action distribution. This lets us model mixed strategies, which is especially desirable in games that lack pure-strategy equilibria. Some approaches in the literature use the output of a policy network to parameterize some parametric distribution on the action space, such as a Gaussian mixture model. However, taking the randomness as input and letting the neural network reshape it as desired allows us to model arbitrary distributions more flexibly.

Figure 1 illustrates the high-level structure of a randomized policy network. It takes as input an observation and random noise, concatenates them, passes the result through a feedforward neural network, and outputs an action. The dimensionality of noise fed into a randomized policy network is an important hyperparameter. In the appendix, we review the literature that studies the relation between input noise dimension and representational power in neural network-based generative models.

5 Experiments

To initialize our networks, we use He initialization [He et al., 2015], which is widely used for feedforward networks with ReLU-like activation functions. It initializes bias vectors to zero and weight matrices with normally-distributed entries scaled by $\sqrt{2/n}$, where $n$ is the layer’s input dimension. We use the ELU activation function [Clevert et al., 2016] for hidden layers. Like Bichler et al. [2021], we use 2 hidden layers with 10 neurons each.

We illustrate the performance of our method by plotting the NashConv over $10^8$ optimization steps. Each hyperparameter setting is labeled in the legend and shown in a different color. Each individual setting is run 20 times. Solid lines indicate means across trials. Bands indicate a confidence interval for this mean with a confidence level of 0.95. These confidence intervals are computed using bootstrapping [Efron, 1979], specifically the bias-corrected and accelerated (BCa) method [Efron, 1987].

For the gradient estimator, we use the Gaussian distribution with scale $\sigma = 10^{-2}$, $N = 1$ samples, and the central-difference stencil (2 evaluations per step). For the optimizer, we use standard gradient descent with a learning rate of $10^{-6}$. To estimate NashConv (see the end of §2), we use 100 observation samples and 300 state samples (given each observation). We use a 100-point discretization of the action space for the auctions and visibility game. For Colonel Blotto games, we use a 231-point discretization of the action space. It is obtained by enumerating all partitions of the integer 20 into 3 parts and renormalizing them to sum to 1.

We now describe our benchmark games and experimental results for each game. The appendix contains figures illustrating analytically-derived equilibria for these games, in cases where they are known.

5.1 Colonel Blotto Games

The original Colonel Blotto game was introduced by Borel et al. [1953]. It is a two-player zero-sum game in which two players distribute resources over several battlefields. A battlefield is won by whoever devotes the most resources to it. A player’s payoff is the number of battlefields they win. This game models many real-world situations of conflict or competition that involve resource allocation, such as political campaigns, research and development, national security, and systems defense. Various variants have been studied in the literature. More information about these can be found in the appendix. We describe the general case with continuous allocations, heterogeneous budgets, heterogeneous battlefield values across both players and battlefields, and several players. Suppose there are $J$ battlefields. Let $b_i$ be the budget of player $i$. Let $v_{ij}$ be the value to player $i$ of battlefield $j$. Player $i$’s action space is the standard $J$-simplex dilated by their budget: $A_i = \{ a_{ij} : 0 \leq a_{ij} \leq b_i, \sum_j a_{ij} = b_i \}$. Player $i$’s reward function is $r_i(a) = \sum_j v_{ij} w_{ij}(a)$ where $w_{ij}$ is the probability that $i$ wins $j$. Ties are broken uniformly at random.

Actions in the continuous Colonel Blotto game are points on the standard simplex. Thus we use a softmax activation function for the output layer of the randomized policy network. Figure 2 illustrates the performance of our method on the continuous Colonel Blotto game with 2 players and 3 battlefields. Since the game has no pure-strategy Nash equilib-
and are used in a wide range of contexts. In a single-item sealed bid auction, bidders simultaneously submit bids and the highest bidder wins the item. Let $w_i(a)$ be the probability $i$ wins given action profile $a$, where ties are broken uniformly at random. Let $v_i(\omega)$ be the item’s value for the $i$th player given state $\omega$. In a $k$-th winner-pay auction, the winner pays the $k$th highest bid: $v_i(\omega, a) = w_i(a)(v_i(\omega) - a_k)$, where $a_k$ is the $k$th highest bid. In an all-pay auction, each player always pays their bid: $v_i(\omega, a) = w_i(a)v_i(\omega) - a_i$. This auction is widely used to model lobbying for rents in regulated and trade-protected industries, technological competition and R&D races, political campaigns, job promotions, and other contests [Baye et al., 1996]. The all-pay complete-information auction lacks pure-strategy equilibria [Baye et al., 1996]. The 2-player 1st-price winner-pay asymmetric-information auction also lacks pure-strategy equilibria [Krishna, 2002, section 8.3]. In particular, the second player must randomize. More details about each type of auction can be found in the appendix.

To ensure the output is non-negative, we use a squaring function in the output layer, rather than a ReLU function like Bichler et al. [2021]. The reason is that, as we found in our experiments, ReLU can easily cause degenerate initializations: if the randomly-initialized neural network happens to map all of the unit interval (the observation space) to negative values, then the gradient signal can be weakened or nonexistent. By default, auctions are 2-player, 1st-price, and winner-pay unless otherwise noted. The all-pay complete-information auction lacks pure-strategy equilibria [Baye et al., 1996]. The 2-player 1st-price winner-pay asymmetric-information auction also lacks pure-strategy equilibria [Krishna, 2002, section 8.3]. In particular, the second player must randomize. More details about each type of auction can be found in the appendix.

To ensure the output is non-negative, we use a squaring function in the output layer, rather than a ReLU function like Bichler et al. [2021]. The reason is that, as we found in our experiments, ReLU can easily cause degenerate initializations: if the randomly-initialized neural network happens to map all of the unit interval (the observation space) to negative values, then the gradient signal can be weakened or nonexistent. By default, auctions are 2-player, 1st-price, and winner-pay unless otherwise noted. The all-pay complete-information auction lacks pure-strategy equilibria [Baye et al., 1996]. The 2-player 1st-price winner-pay asymmetric-information auction also lacks pure-strategy equilibria [Krishna, 2002, section 8.3]. In particular, the second player must randomize. More details about each type of auction can be found in the appendix.

To ensure the output is non-negative, we use a squaring function in the output layer, rather than a ReLU function like Bichler et al. [2021]. The reason is that, as we found in our experiments, ReLU can easily cause degenerate initializations: if the randomly-initialized neural network happens to map all of the unit interval (the observation space) to negative values, then the gradient signal can be weakened or nonexistent. By default, auctions are 2-player, 1st-price, and winner-pay unless otherwise noted. The all-pay complete-information auction lacks pure-strategy equilibria [Baye et al., 1996]. The 2-player 1st-price winner-pay asymmetric-information auction also lacks pure-strategy equilibria [Krishna, 2002, section 8.3]. In particular, the second player must randomize. More details about each type of auction can be found in the appendix.

To ensure the output is non-negative, we use a squaring function in the output layer, rather than a ReLU function like Bichler et al. [2021]. The reason is that, as we found in our experiments, ReLU can easily cause degenerate initializations: if the randomly-initialized neural network happens to map all of the unit interval (the observation space) to negative values, then the gradient signal can be weakened or nonexistent. By default, auctions are 2-player, 1st-price, and winner-pay unless otherwise noted. The all-pay complete-information auction lacks pure-strategy equilibria [Baye et al., 1996]. The 2-player 1st-price winner-pay asymmetric-information auction also lacks pure-strategy equilibria [Krishna, 2002, section 8.3]. In particular, the second player must randomize. More details about each type of auction can be found in the appendix.

5.3 Multi-Item Auctions

Multi-item auctions are of great importance in practice, for example in strategic sourcing [Sandholm, 2013] and radio spectrum allocation [Milgrom and Segal, 2014; Milgrom and Segal, 2020]. However, deriving equilibrium bidding strategies for multi-item auctions is notoriously elusive.
notable instance where equilibrium strategies have been derived is the chopstick auction [Szentes and Rosenthal, 2003b; Szentes and Rosenthal, 2003a]. In this auction, 3 chopsticks are sold simultaneously in separate first-price sealed-bid auctions. There are 2 bidders, and it is common knowledge that a pair of chopsticks is worth $1, a single chopstick is worth nothing by itself, and 3 chopsticks are worth the same as 2. Here, pure strategies are triples of non-negative real numbers (bids). This game has an interesting equilibrium: let the tetrahedron $T$ be defined as the convex hull of the four points $(1/2, 1/2, 0), (0, 1/2, 1/2), (0, 0, 1), (0, 0, 0)$. Then the uniform probability measure on the 2-dimensional surface of $T$ generates a symmetric equilibrium. (Furthermore, all points inside the tetrahedron are pure best responses to this equilibrium mixture.) We benchmark on the chopstick auction since it is a rare case of a multi-item auction with a known analytic equilibrium, so we can compare our output to an exact equilibrium. It is also a canonical case of simultaneous separate auctions under combinatorial preferences.

Figures 8 and 9 illustrate performances and strategies for the chopstick auction. The latter figure shows that, with more epochs, the strategies better approximate a tetrahedron, which is the analytic equilibrium (as discussed in the appendix). Here we encounter an interesting phenomenon. Recall that this game has a symmetric equilibrium generated by the uniform measure on the surface of a tetrahedron. Although the tetrahedron itself is 3-dimensional, its surface is only 2-dimensional. Thus one may wonder whether 2-dimensional noise is sufficient, that is, whether the network can learn to project this lower-dimensional manifold out into the third dimension while “folding” it in the way required to obtain the surface of the tetrahedron. Through our experiments, we observe that 2-dimensional noise suffices to (approximately) match the performance of higher-dimensional noise. Thus the intrinsic dimension of the equilibrium action distribution (as opposed to the extrinsic dimension of the ambient space in which it is embedded) seems to be the decisive factor.

5.4 Visibility Game
Lotker et al. [2008] introduced the visibility game, a noncooperative, complete-information strategic game. In this game, each player $i$ chooses a point $x_i \in [0, 1]$. Their payoff is the distance to the next higher point, or to 1 if $x_i$ is the highest. This game models a situation where players seek to maximize their visibility time, and is a variant of the family of “timing games” [Fudenberg and Tirole, 1991]. Lotker et al. [2008] prove that the $n$-player visibility game has no pure equilibrium, but has a unique mixed equilibrium, which is symmetric. In the 2-player case, up to a set of measure zero, there is a unique equilibrium whose strategies have probability densities $p(x) = 1/(1 - x)$ when $0 \leq x \leq 1 - 1/e$ and 0 otherwise. Each player’s expected payoff is $1/e$.

Figures 10 illustrates performances on the 2-player visibil-
Figure 9: Chopstick auction strategies, based on $10^5$ action samples. X, Y, and Z axes denote bid for each item. Left to right: Players 1 and 2. Top to bottom: Epochs 30, 60, and 90.

ity game. Figure 11 illustrates strategies during training for a trial with 1-dimensional noise. The players’ distributions converge to the expected distribution (there is a distinctive cutoff at $1 - 1/e \approx 0.632$). As expected, 0-dimensional noise, which yields deterministic strategies, performs very poorly. More interestingly, there is a noticeable gap in performance between 1-dimensional noise, which matches the dimensionality of the action space, and higher-dimensional noise. That is, using noise of higher dimension than the action space accelerates convergence in this game.

6 Conclusions and Future Research

We presented, to our knowledge, the first paper to solve general continuous-action games with unrestricted mixed strategies and without any gradient information. We accomplished this using zeroth-order optimization techniques that combine smoothed gradient estimators with equilibrium-finding gradient dynamics. We modeled players’ strategies using randomized policy networks that take noise as input and can flexibly represent arbitrary observation-dependent, continuous-action distributions. Being able to model such mixed strategies is crucial for tackling continuous-action games that can lack pure-strategy equilibria.

We evaluated our method on various games, including continuous Colonel Blotto games, single-item and multi-item auctions, and a visibility game. The experiments showed that our method can quickly compute a high-quality approximate equilibrium for these games. Furthermore, they showed that the dimensionality of the input noise is crucial for representing and converging to equilibrium. In particular, noise of too low dimension (or no noise, which yields a deterministic policy) results in failure to converge. Randomized policy networks flexibly model observation-dependent action distributions. Thus, in contrast to prior work, we can flexibly model mixed strategies and directly optimize them in a “black-box” game with access only to payoffs.

This work opens many directions for tackling even more complex multiagent environments. In multi-step environments, the current observation might not contain all information about past observations and actions that is relevant to choosing an action. To give agents memory, one can use recurrent networks. In that case, the policy network would receive as input an observation, source of randomness, and memory state and output an action and new memory state. One can also consider games with more complex observation and action spaces, including high-dimensional arrays like images. Convolutional networks can be used to process such inputs. Very complex environments, including real-time strategy games like StarCraft II, may require more sophisticated neural architectures.
Acknowledgements

This material is based on work supported by the National Science Foundation under grants IIS-1901403 and CCF-1733556 and by the ARO under award W911NF2210266.

References


