A dynamical neural network approach for solving stochastic two-player zero-sum games

Dawen Wu, Abdel Lisser

A dynamical neural network for solving stochastic two-player zero-sum games

Dawen Wu and Abdel Lisser

Laboratoire Signaux & Systèmes, CentraleSupélec, Université Paris-Saclay

3, rue Joliot Curie, 91190 Gif-sur-Yvette, France

dawen.wu@centralesupelec.fr (Dawen Wu)

abdel.lisser@l2s.centralesupelec.fr (Abdel Lisser)

A dynamical neural network approach for solving stochastic two-player zero-sum games

Dawen Wu[a,*], Abdel Lisser[a]

[a] *Centralesupelec, Université Paris-Saclay*

Abstract

This paper aims at solving a stochastic two-player zero-sum nash game problem studied in (Singh & Lisser, 2019). The main contribution of our paper is that we model this game problem as a dynamical neural network (DNN for short). In this paper, we show that the saddle point of this game problem is the equilibrium point of the DNN model, and we study the globally asymptotically stable of the DNN model. In our numerical experiments, we present the time-continuous feature of the DNN model and compare it with the state-of-the-art convex solvers, i.e., Splitting conic solver (SCS for short) and Cvxopt. Our numerical results show that our DNN method has two advantages in dealing with this game problem. Firstly, the DNN model can converge to a better optimal point. Secondly, the DNN method can solve all problems, even when the problem size is large.

Keywords: Stochastic two-player zero-sum game, Saddle point, Dynamical neural network,

1. Introduction

In 1928, von Neumann (1928) studied the equilibrium concept in game theory and showed that there always exists a saddle point equilibrium for a finite action two-player zero-sum games. Nash (1950) generalizes this result to n-player games and shows that there always exists a Nash equilibrium for any finite action and finite players general-sum game. Charnes (1953) studied a new type of two-player zero-sum games where the mixed strategy of each player is linearly constrained. The saddle point of such games can be obtained by solving the primal-dual pair of linear programs. Recently, Singh & Lisser (2019) studied a stochastic version of two-player zero-sum games, namely chance-constrained two-player zero-sum games. They show a saddle point exists if the random vectors defining stochastic linear constraints follow elliptically symmetric distributions.

There is a vast literature on saddle points or Nash equilibrium in game theory. Dantzig (1963) shows that solving the saddle point of a two-player zero-sum game can be formulated as a linear program. Lemke & Howson (1964) provide a method to solve two-player general-sum games, namely the Lemke-Howson algorithm. For n-player general-sum cases, van der Laan et al. (1987); Govindan & Wilson (2003); Blum et al. (2006) studied several algorithms to find Nash equilibrium. However, all the above-mentioned learning methods can only obtain the Nash equilibrium of deterministic games. There is a lack of dedicated learning algorithms for games with stochastic payoff.

The state-of-the-art approach for solving such stochastic games consists in converting the problem into a convex constraint problem. Then solve it by well-studied convex optimization algorithms (Boyd et al., 2004; Nocedal & Wright, 2006).

*Corresponding author
Email address: dawen.wu@centralesupelec.fr, abdel.lisser@l2s.centralesupelec.fr (Abdel Lisser)

Besides the convex algorithms, there are many other alternatives to solve optimization problems. Bangyal et al. (2020, 2021) study various types of metaheuristic algorithms to solve combinatorial optimization problems. Hopfield & Tank (1985) pioneered the study of using dynamical systems in optimization, i.e., dynamical neural networks. Since then, DNNs have been used on various mathematical problems. For example, DNNs were used to solve Linear programming problems (Wang, 1993; Xia, 1996b), Quadratic programming problems (Xia, 1996a; Nazemi, 2014; Feizi et al., 2021), Second-order cone programming problems (Nazemi & Sabeghi, 2020), Nonlinear programming problems (Kennedy & Chua, 1988; Forti et al., 2004; Nazemi & Tahmasbi, 2013), Minimax problems (Gao et al., 2004; Nazemi, 2011), and Nonlinear complementarity problems (Liao et al., 2001; Dang et al., 2004), Manipulator problems (Zhang et al., 2020). DNN models usually expect two properties: 1. The equilibrium points of the DNN models should coincide with the solutions to the original problems. 2. The DNN models should be globally convergent. Compared to convex optimization algorithms, DNN methods use a different strategy, deemed more straightforward. Once the DNN model and initial points are known, this approach converges towards the optimal solutions. It does not require decomposing the original problem into multiple sub-problems and solving them iteratively. DNN models can achieve an arbitrary required accuracy independent of the problem size thanks to the global convergence property. With regards to numerical solutions of DNN models, Runge-Kutta and backward differentiation are two classical methods (Curtiss & Hirschfelder, 1952; Gottlieb & Shu, 1998; Teschl, 2012). Furthermore, there are emerging studies of deep learning on differential equations (Lagaris et al., 1998; Raissi et al., 2019; Flamant et al., 2020), which make it possible to connect our DNN models with deep learning.

This paper uses a DNN method to obtain the saddle point of stochastic two-player zero-sum games. To the best of our knowledge, DNN methods have not been used to solve this game problem in the literature. The main contributions are provided as follows: 1) We formulate a stochastic two-players zero-sum game as a DNN model. We show that the equilibrium point of the DNN model coincides with the saddle point of the stochastic two-players zero-sum game. Then, we prove that the equilibrium of the DNN models is global asymptotic stability. 2) Our experimental results show that the DNN method can provide high accuracy solutions and remains robust to the problem size.

The remaining of the paper is organized as follows. Section 2 shows the stochastic two-player zero-sum game problem together with its SOCP reformulation. Section 3 gives the KKT conditions related to the SOCP problem and the DNN model. Numerical experiments are given in Section 4.

The following notations are used in the remainder of the paper.

- x and y denote the strategies of player 1 and player 2 respectively. m and n denote the sizes of the action set of the player1 and the player2, respectively.
- $\mathcal{J}_1$, $\mathcal{J}_2$ denote the probabilistic constraints sets for player 1 and player 2, respectively. J_1, J_2 denote the sets sizes.
- $A, \mu^1, \mu^2, \Sigma^1, \Sigma^2, b, d$ denotes the data for a stochastic two-players zero-sum game. A denotes the payoff matrix. $\mu^1, \mu^2, \Sigma^1, \Sigma^2$ are the means and the variances of the probability distributions, respectively. $\varphi_k^1\left(t^2\right)$ and $\varphi_l^2\left(t^2\right)$ are the characteristic functions of the probability distributions, respectively.
- α^1 and α^2 are the settings of the players' confidence levels.
- $\Psi_{\xi_l^2}^{-1}\left(\alpha_l^2\right), \Psi_{\xi_k^1}^{-1}\left(\alpha_k^1\right)$ are quantile functions of 1-dimensional distribution functions induced by characteristic functions $\varphi_k^1\left(t^2\right)$ and $\varphi_l^2\left(t^2\right)$, respectively.

- $s = (y, v, \delta, \lambda)$ are the decision variables of the optimization problem. u is the dual variable of the optimization problem. $r = (s, u) = (y, v, \delta, \lambda, u)$ are the variables of the neural network.
- nr, ns, nu are the number of r, s, u variables, respectively. Moreover, nu also denotes the number of constraints of the optimization problem.
- $f(s) = f(y, v, \delta, \lambda)$ and $g(s) = g(y, v, \delta, \lambda)$ are the objective function and the constraints of the optimization problem respectively. Also, $f(s), g(s), \nabla f(s), \nabla g(s), \nabla^2 g(s)$ are abbreviated as $f, g, \nabla f, \nabla g, \nabla^2 g$, respectively.
- $\Phi(r) = \frac{dr}{dt}$ denotes the DNN model.

2. Problem formulation

In this section, we present the stochastic two-player zero-sum game with chance constraints. A two-player zero-sum game involves two persons called player 1 and player 2, respectively. These games are described by a matrix A with m rows and n columns. Matrix A represents the payoffs of player 1, and matrix $-A$ represents the payoffs of player 2. Let $I = \{1, 2, \ldots, m\}$ be the action set of player 1 and $J = \{1, 2, \ldots, n\}$ be the action set of player 2. A pure strategy refers to a single action from the action set. A mixed strategy refers to a probability distribution defined over the action set. Let $X = \{x \in \mathbb{R}^m | \mathbf{1}^T x = 1, x \geq 0\}$ and $Y = \{y \in \mathbb{R}^n | \mathbf{1}^T y = 1, y \geq 0\}$ the sets of mixed strategies of player 1 and player 2, respectively. The payoffs of player 1 and player 2 are defined by $x^T A y$ and $x^T(-A)y$, respectively, for a given strategy pair $(x, y) \in X \times Y$. von Neumann (1928) showed that there exists a saddle point equilibrium in mixed strategies in zero-sum games. Dantzig (1951) showed that the saddle point equilibrium is a solution of primal-dual pair of linear programs. Charnes (1953) studied a linear constrained two-player zero-sum game problem. For a given strategy y of player 2, the objective of player 1 is to choose a strategy x which solves the linear programming problem (1).

$$\begin{cases} \max\limits_{x} x^T A y \\ \text{s.t.} \\ \quad Bx \leq b \\ \quad \mathbf{1}^T x = 1 \\ \quad x \geq 0, \end{cases} \tag{1}$$

Similarly, the aim of player 2 is to choose a strategy y that solves problem (2) for a given strategy x of player 1.

$$\begin{cases} \min\limits_{y} x^T A y \\ \text{s.t.} \\ \quad Dy \geq d \\ \quad \mathbf{1}^T y = 1 \\ \quad y \geq 0, \end{cases} \tag{2}$$

where $B \in \mathbb{R}^{J_1 \times m}, D \in \mathbb{R}^{J_2, n}, b \in \mathbb{R}^{J_1}$ and $d \in \mathbb{R}^{J_2}$. A strategy pair (x, y) is said a saddle point equilibrium for the above constrained zero-sum game if x is an optimal solution of (1) for a given y, and y is an optimal solution of (2) for the given x.

Singh & Lisser (2019) consider the problem where each row vector B_k and D_l follows an elliptical distribution i.e. $B_k^w \sim \text{Ellip}_m\left(\mu_k^1, \Sigma_k^1, \varphi_k^1\right)$ and $D_l^w \sim \text{Ellip}_n\left(\mu_l^2, \Sigma_l^2, \varphi_l^2\right)$. $\Psi_{\xi_l^2}^{-1}\left(\alpha_l^2\right)$ and $\Psi_{\xi_k^1}^{-1}\left(\alpha_k^1\right)$ are the quantile functions of 1-dimensional distribution functions induced by characteristic functions $\varphi_k^1\left(t^2\right)$ and $\varphi_l^2\left(t^2\right)$, respectively. The chance constrained optimization problem can be written as

$$\begin{cases} \max\limits_{x} x^T A y \\ \text{s.t.} \\ \quad P\left\{B_k^w x \leq b_k\right\} \geq \alpha_k^1, \quad \forall k \in \mathcal{J}_1 \\ \quad \mathbf{1}^T x = 1 \\ \quad x \geq 0, \end{cases} \tag{3}$$

and

$$\begin{cases} \min\limits_{y} x^T A y \\ \text{s.t.} \\ \quad P\left\{D_l^w y \geq d_l\right\} \geq \alpha_l^2, \quad \forall l \in \mathcal{J}_2 \\ \quad \mathbf{1}^T y = 1 \\ \quad y \geq 0. \end{cases} \tag{4}$$

We use the SOCP reformulation from Kataoka (1963) to rewrite the probabilistic constraints in (3) and (4) as follows

$$x^T \mu_k^1 + \Psi_{\xi_k^1}^{-1}(\alpha_k^1) \|(\Sigma_k^1)^{\frac{1}{2}} x\| \leq b_k, \quad \forall k \in \mathcal{J}_1, \tag{5}$$

and

$$-y^T \mu_l^2 + \Psi_{\xi_l^2}^{-1}(\alpha_l^2) \|(\Sigma_l^2)^{\frac{1}{2}} y\| \leq -d_l, \quad \forall l \in \mathcal{J}_2. \tag{6}$$

We denote the stochastic two-players zero-sum game by $G(\alpha)$ and the feasible strategy sets of the two players by $S_1\left(\alpha^1\right)$ and $S_2\left(\alpha^2\right)$,

$$S_1\left(\alpha^1\right) = \left\{x \in \mathbb{R}^m \mid \mathbf{1}^T x = 1, x \geq 0, x^T \mu_k^1 + \Psi_{\xi_k^1}^{-1}(\alpha_k^1) \|(\Sigma_k^1)^{\frac{1}{2}} x\| \leq b_k, \quad \forall k \in \mathcal{J}_1\right\}, \tag{7}$$

and

$$S_2\left(\alpha^2\right) = \left\{y \in \mathbb{R}^n \mid \mathbf{1}^T y = 1, y \geq 0, -y^T \mu_l^2 + \Psi_{\xi_l^2}^{-1}(\alpha_l^2) \|(\Sigma_l^2)^{\frac{1}{2}} y\| \leq -d_l, \quad \forall l \in \mathcal{J}_2\right\}. \tag{8}$$

Assumption 1.

1. *The set $S_1(\alpha^1)$ is strictly feasible, i.e., there exists an $x \in \mathbb{R}^m$ which is a feasible point of $S_1(\alpha^1)$ and the inequality constraints of $S_1(\alpha^1)$ are strictly satisfied by x.*
2. *The set $S_2(\alpha^2)$ is strictly feasible, i.e., there exists an $x \in \mathbb{R}^n$ which is a feasible point of $S_2(\alpha^2)$ and the inequality constraints of $S_2(\alpha^2)$ are strictly satisfied by y.*

In the remaining of the paper, we suppose that assumption 1 holds. Otherwise, the game would not have a saddle point.

(x^*, y^*) is called a saddle point equilibrium of $G(\alpha)$ if the following inequality holds:

$$x^T A y^* \leq x^{*T} A y^* \leq x^{*T} A y, \forall x \in S_1\left(\alpha^1\right), y \in S_2\left(\alpha^2\right). \tag{9}$$

The following theorem shows the saddle point existence of the stochastic two-player zero-sum game problem.

Theorem 1 (Singh & Lisser (2019), Theorem 3.5)**.** *Consider a constrained zero-sum matrix game where the matrices B^w and D^w defining the constraints of both the players, respectively, are random. Let the row vectors $B_k^w \sim Ellip_m\left(\mu_k^1, \Sigma_k^1, \varphi_k^1\right), k \in \mathcal{J}_1$, and $D_l^w \sim \text{Ellip}_n\left(\mu_l^2, \Sigma_l^2, \varphi_l^2\right), l \in \mathcal{J}_2$. For all k and l, $\Sigma_k^1 \succ 0$ and $\Sigma_l^2 \succ 0$. Then, there exists a saddle point equilibrium for the game $G(\alpha)$ for all $\alpha \in (0.5, 1]^{J_1} \times (0.5, 1]^{J_2}$.*

We refer the reader to Singh & Lisser (2019) for more details about the proof of this theorem and the related results.

Proposition 1. *The chance constrained optimization problems* (3) *and* (4) *can be reformulated as the following SOCP problems* $(\mathcal{P})$ *and* $(\mathcal{D})$*.*

$$\left.\begin{array}{l} \min_{y, v^1, \left(\delta_k^1\right)_{k \in \mathcal{J}_1}, \lambda^1} v^1 + \sum_{k \in \mathcal{J}_1} \lambda_k^1 b_k \\ s.t. \\ (i) Ay - \sum_{k \in \mathcal{J}_1} \lambda_k^1 \mu_k^1 - \sum_{k \in \mathcal{J}_1} \left(\Sigma_k^1\right)^{\frac{1}{2}} \delta_k^1 \leq v^1 \mathbf{1}_m \\ (ii) - y^T \mu_l^2 + \Psi_{\xi_l^2}^{-1}\left(\alpha_l^2\right) \left\| \left(\Sigma_l^2\right)^{\frac{1}{2}} y \right\| \leq -d_l, \quad \forall l \in \mathcal{J}_2 \\ (iii) \left\|\delta_k^1\right\| \leq \lambda_k^1 \Psi_{\xi_k^1}^{-1}\left(\alpha_k^1\right), \quad \forall k \in \mathcal{J}_1 \\ (iv) \mathbf{1}^T y = 1 \\ (v) y \geq 0 \\ (vi) \lambda_k^1 \geq 0, \quad \forall k \in \mathcal{J}_1 \end{array}\right\} \quad (\mathcal{P})$$

$$\left.\begin{array}{l} \max_{x, v^2, \left(\delta_l^2\right)_{l \in \mathcal{J}_2}, \lambda^2} v^2 + \sum_{l \in \mathcal{J}_2} \lambda_l^2 d_l \\ s.t. \\ (i) A^T x - \sum_{l \in \mathcal{J}_2} \lambda_l^2 \mu_l^2 - \sum_{l \in \mathcal{J}_2} \left(\Sigma_l^2\right)^{\frac{1}{2}} \delta_l^2 \geq v^2 \mathbf{1}_n \\ (ii) x^T \mu_k^1 + \Psi_{\xi_k^1}^{-1}\left(\alpha_k^1\right) \left\| \left(\Sigma_k^1\right)^{\frac{1}{2}} x \right\| \leq b_k, \quad \forall k \in \mathcal{J}_1 \\ (iii) \left\|\delta_l^2\right\| \leq \lambda_l^2 \Psi_{\xi_l^2}^{-1}\left(\alpha_l^2\right), \quad \forall l \in \mathcal{J}_2 \\ (iv) \mathbf{1}^T x = 1 \\ (v) x \geq 0 \\ (vi) \lambda_l^2 \geq 0, \quad \forall l \in \mathcal{J}_2 \end{array}\right\} \quad (\mathcal{D})$$

Proof. We show the process that generate $(\mathcal{P})$ from (3).

The chance constrained optimization problem (3) with the second-order cone reformulation (5) is

$$\begin{cases} \max_x x^T A y \\ \text{s.t.} \\ \quad \text{(i)} x^T \mu_k^1 + \Psi_{\xi_k^1}^{-1}(\alpha_k^1) \|(\Sigma_k^1)^{\frac{1}{2}} x\| \le b_k, \quad \forall k \in \mathcal{J}_1 \\ \quad \text{(iii)} \mathbf{1}^T x = 1 \\ \quad \text{(v)} x \ge 0. \end{cases} \tag{10}$$

Given a strategy y of player 2, the problem can be written as the following SOCP problem, where $(t_k^1)_{k\in\mathcal{J}_1}$ are auxiliary variables,

$$\begin{cases} \max_{x,(t_k^1)_{k\in\mathcal{J}_1}} x^T A y \\ \text{s.t.} \\ \quad \text{(i)} - x^T \mu_k^1 - \Psi_{\xi_k^1}^{-1}(\alpha_k^1) \|t_k\| + b_k \ge 0, \quad \forall k \in \mathcal{J}_1 \\ \quad \text{(ii)} t_k^1 - (\Sigma_k^1)^{\frac{1}{2}} x = 0, \quad \forall k \in \mathcal{J}_1 \\ \quad \text{(iii)} \mathbf{1}^T x = 1 \\ \quad \text{(iv)} x \ge 0 \end{cases} \tag{11}$$

The saddle point of the lagrangian of (11) is

$$\min_{v^1,(\delta_k)_{k\in\mathcal{J}_1}^1,\lambda^1\ge 0} \max_{x,(t_k^1)_{k\in\mathcal{J}_1}} \left[x^T A y + v^1 \mathbf{1}^T x + \sum_{k\in\mathcal{J}_1} \left(\delta_k^1\right)^T \left(t_k^1 - (\Sigma_k^1)^{\frac{1}{2}} x\right) + \sum_{k\in\mathcal{J}_1} \lambda_k^1 \left(-x^T \mu_k^1 - \Psi_{\xi_k^1}^{-1}\left(\alpha_k^1\right) \|t_k\| + b_k\right)\right] \tag{12}$$

For the fixed $v^1, (\delta_k)_{k\in\mathcal{J}_1}^1, \lambda^1$, we have

$$\begin{aligned} &\max_{x,(t_k^1)_{k\in\mathcal{J}_1}} \left[x^T A y + v^1 \mathbf{1}^T x + \sum_{k\in\mathcal{J}_1} \left(\delta_k^1\right)^T \left(t_k^1 - (\Sigma_k^1)^{\frac{1}{2}} x\right) + \sum_{k\in\mathcal{J}_1} \lambda_k^1 \left(-x^T \mu_k^1 - \Psi_{\xi_k^1}^{-1}\left(\alpha_k^1\right) \|t_k\| + b_k\right)\right] \\ =& \max_x \left[x^T \left(A y - \sum_{k\in\mathcal{J}_1} \lambda_k^1 \mu_k^1 - \sum_{k\in\mathcal{J}_1} \left(\Sigma_k^1\right)^{\frac{1}{2}} \delta_k^1 - v^1 \mathbf{1}\right)\right] + \max_{(t_k^1)_{k\in\mathcal{J}_1}} \left[\sum_{k\in\mathcal{J}_1} \left(\left(\delta_k^1\right)^T t_k^1 - \lambda_k^1 \Psi_{\xi_k^1}^{-1}\left(\alpha_k^1\right) \|t_k^1\|\right)\right] + v^1 + \sum_{k\in\mathcal{J}_1} \lambda_k^1 b_k \end{aligned} \tag{13}$$

The first and the second maximization problems are unbounded unless the following conditions hold,

$$A y - \sum_{k\in\mathcal{J}_1} \lambda_k^1 \mu_k^1 - \sum_{k\in\mathcal{J}_1} \left(\Sigma_k^1\right)^{\frac{1}{2}} \delta_k^1 \le v^1 \mathbf{1} \tag{14}$$

$$\|\delta_k^1\| \le \lambda_k^1 \Psi_{\xi_k^1}^{-1}\left(\alpha_k^1\right), \quad \forall k \in \mathcal{J}_1 \tag{15}$$

The Lagrangian dual of the inner maximum problem is given by the following SOCP problem,

$$\begin{cases} \min_{v^1,(\delta_k^1),\lambda^1} v^1 + \sum_{k\in\mathcal{J}_1} \lambda_k^1 b_k \\ \text{s.t.} \\ \quad \text{(i)} Ay - \sum_{k\in\mathcal{J}_1} \lambda_k^1 \mu_k^1 - \sum_{k\in\mathcal{J}_1} \left(\Sigma_k^1\right)^{\frac{1}{2}} \delta_k^1 \leq v^1 \mathbb{1}_m \\ \quad \text{(ii)} \left\|\delta_k^1\right\| \leq \lambda_k^1 \Psi_{\xi_k^1}^{-1}\left(\alpha_k^1\right), \forall k \in \mathcal{J}_1 \\ \quad \text{(iii)} \lambda_k^1 \geq 0, \forall k \in \mathcal{J}_1 \end{cases} \tag{16}$$

With the second-order cone constraint (6) for y, we finally get the first SOCP $(\mathcal{P})$.

$$\left.\begin{array}{l} \min_{y,v^1,\left(\delta_k^1\right)_{k\in\mathcal{J}_1},\lambda^1} v^1 + \sum_{k\in\mathcal{J}_1} \lambda_k^1 b_k \\ \text{s.t.} \\ \quad \text{(i)} Ay - \sum_{k\in\mathcal{J}_1} \lambda_k^1 \mu_k^1 - \sum_{k\in\mathcal{J}_1} \left(\Sigma_k^1\right)^{\frac{1}{2}} \delta_k^1 \leq v^1 \mathbf{1}_m \\ \quad \text{(ii)} - y^T \mu_l^2 + \Psi_{\xi_l^2}^{-1}\left(\alpha_l^2\right) \left\|\left(\Sigma_l^2\right)^{\frac{1}{2}} y\right\| \leq -d_l, \quad \forall l \in \mathcal{J}_2 \\ \quad \text{(iii)} \left\|\delta_k^1\right\| \leq \lambda_k^1 \Psi_{\xi_k^1}^{-1}\left(\alpha_k^1\right), \quad \forall k \in \mathcal{J}_1 \\ \quad \text{(iv)} \mathbf{1}^T y = 1 \\ \quad \text{(v)} y \geq 0 \\ \quad \text{(vi)} \lambda_k^1 \geq 0, \quad \forall k \in \mathcal{J}_1 \end{array}\right\} \quad (\mathcal{P})$$

The dual problem $(\mathcal{D})$ can be generated similarly. □

The following theorem shows the existence of the saddle point for the chance constrained zero-sum game $G(\alpha)$.

Theorem 2 (Singh & Lisser (2019), Theorem 3.7)**.** *Consider a constrained zero-sum game where the matrices B^w and D^w defining the constraints of player 1 and player 2, respectively, are random. Let the row vector $B^w \sim$ Ellip $_m\left(\mu_k^1, \Sigma_k^1, \varphi_k^1\right), k \in \mathcal{J}_1$, where $\Sigma_k^1 \succ 0$, and the row vector $D_l^w \sim \text{Ellip}_n\left(\mu_l^2, \Sigma_l^2, \varphi_l^2\right), l \in \mathcal{J}_2$ where $\Sigma_l^2 \succ 0$. Let Assumption 1 holds. Then, for a given $\alpha \in (0.5, 1]^p \times (0.5, 1]^q$, (x^*, y^*) is a saddle point equilibrium of the game $G(\alpha)$ if and only if there exist $\left(v^{1*}, \left(\delta_k^{1*}\right)_{k\in\mathcal{J}_1}, \lambda^{1*}\right)$ and $\left(v^{2*}, \left(\delta_l^{2*}\right)_{l\in J_2}, \lambda^{2*}\right)$ such that $\left(y^*, v^{1*}, \left(\delta_k^{1*}\right)_{k\in J_1}, \lambda^{1*}\right)$ and $\left(x^*, v^{2*}, \left(\delta_l^{2*}\right)_{l\in\mathcal{J}_2}, \lambda^{2*}\right)$ are optimal solutions of primal-dual pair of SOCPs $(\mathcal{P})$ and $(\mathcal{D})$, respectively.*

We refer the reader to Singh & Lisser (2019) for more details about the proof of this theorem and the related results.

3. Methodology

This section studies a DNN approach to solve the second-order cone programming problem given in section 2. We provide the necessary and sufficient KKT conditions of problem $(\mathcal{P})$. We use a DNN to solve the KKT conditions problem as the equilibrium point of the DNN model corresponds to the KKT point. Then, we study the stability of the equilibrium point by analyzing a Lyapunov function.

We transform the equality constraint $\mathbf{1}^T y = 1$ in $(\mathcal{P})$ into inequality $\mathbf{1}^T y - 1 \leq 0, 1 - \mathbf{1}^T y \leq 0$. For sake of simplicity, we consider only the primal problem. Denote $s = (y, v, \delta, \lambda) = (y, v^1, (\delta_k^1)_{k\in\mathcal{J}_1}, \lambda^1)$, where $\delta = (\delta_k^1)_{k\in\mathcal{J}_1} = [{\delta_1^1}^T, \ldots, {\delta_{J_1}^1}^T]^T$

and $\lambda = \lambda^1 = [\lambda_1^1, \ldots, \lambda_{J_1}^1]^T$. The optimization problem $(\mathcal{P})$ can be written as

$$\begin{aligned} &\min_s f(s) \\ &\text{s.t.} \\ &\quad g(s) \leq 0, \end{aligned} \tag{17}$$

where the objective function $f : \mathbb{R}^{ns} \to \mathbb{R}$, and the constraints $g : \mathbb{R}^{ns} \to \mathbb{R}^{nu}$.

3.1. KKT conditions

Since the SOCP constraints of $g(s)$ are not differentiable, we introduce the following perturbation $\epsilon = 10^{-6}$, i.e., $\sqrt{\|s\|^2 + \epsilon^2}$. Thanks to this smoothness technique, the KKT conditions of the SOCP are necessary and sufficient optimality conditions.

The KKT conditions of the SOCP problem $(\mathcal{P})$ are

$$\begin{aligned} \nabla f(s) + \nabla g(s)^T u = 0 \\ g(s) \leq 0, \quad u^T \geq 0, \quad u^T g(s) = 0 \end{aligned} \tag{18}$$

where the $\nabla f, u, g, \nabla g$ are as follows

$$\nabla f(s) = \begin{bmatrix} \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial v} \\ \frac{\partial f}{\partial \delta} \\ \frac{\partial f}{\partial \lambda} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ b \end{bmatrix} \tag{19}$$

$$u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_{41} \\ u_{42} \\ u_5 \\ u_6 \end{bmatrix} \tag{20}$$

$$g(s) = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_{41} \\ g_{42} \\ g_5 \\ g_6 \end{bmatrix} = \begin{bmatrix} Ay - v\mathbf{1} - \sum_{k \in \mathcal{J}_1} \left(\Sigma_k^1\right)^{\frac{1}{2}} \delta_k - \sum_{k \in \mathcal{J}_1} \lambda_k \mu_k^1 \\ (-y^T \mu_l^2 + \Psi_{\xi_l^2}^{-1}\left(\alpha_l^2\right) \left\| \left(\Sigma_l^2\right)^{\frac{1}{2}} y \right\| + d_l)_{l \in \mathcal{J}_2} \\ (\|\delta_k\| - \Psi_{\xi_k^1}^{-1}\left(\alpha_k^1\right) \lambda_k)_{k \in \mathcal{J}_1} \\ \mathbf{1}^T y - 1 \\ -\mathbf{1}^T y + 1 \\ -y \\ -\lambda \end{bmatrix} \tag{21}$$

$$\nabla g(s) = \begin{bmatrix} \frac{\partial g_1}{\partial y} & \frac{\partial g_1}{\partial v} & \frac{\partial g_1}{\partial \delta} & \frac{\partial g_1}{\partial \lambda} \\ \frac{\partial g_2}{\partial y} & \frac{\partial g_2}{\partial v} & \frac{\partial g_2}{\partial \delta} & \frac{\partial g_2}{\partial \lambda} \\ \frac{\partial g_3}{\partial y} & \frac{\partial g_3}{\partial v} & \frac{\partial g_3}{\partial \delta} & \frac{\partial g_3}{\partial \lambda} \\ \frac{\partial g_{41}}{\partial y} & \frac{\partial g_{41}}{\partial v} & \frac{\partial g_{41}}{\partial \delta} & \frac{\partial g_{41}}{\partial \lambda} \\ \frac{\partial g_{42}}{\partial y} & \frac{\partial g_{42}}{\partial v} & \frac{\partial g_{42}}{\partial \delta} & \frac{\partial g_{42}}{\partial \lambda} \\ \frac{\partial g_5}{\partial y} & \frac{\partial g_5}{\partial v} & \frac{\partial g_5}{\partial \delta} & \frac{\partial g_5}{\partial \lambda} \\ \frac{\partial g_6}{\partial y} & \frac{\partial g_6}{\partial v} & \frac{\partial g_6}{\partial \delta} & \frac{\partial g_6}{\partial \lambda} \end{bmatrix} = \begin{bmatrix} A & -\mathbf{1} & (-(\Sigma_k^1)^{\frac{1}{2}})_{k\in\mathcal{J}_1} & (-\mu_k^1)_{k\in\mathcal{J}_1} \\ (-(\mu_l^2) + \Psi_{\xi_l^2}^{-1}(\alpha_l^2)\frac{{\Sigma_l^2}^{\frac{1}{2}^T}\Sigma_l^{2\frac{1}{2}}y}{\|\Sigma_l^{2\frac{1}{2}}y\|})_{l\in\mathcal{J}_2}^T & 0 & 0 & 0 \\ 0 & 0 & (\frac{\delta_k}{\|\delta_k\|})_{k\in\mathcal{J}_1} & (-\Psi_{\xi_k^1}^{-1}\left(\alpha_k^1\right))_{k\in\mathcal{J}_1} \\ \mathbf{1}^T & 0 & 0 & 0 \\ -\mathbf{1}^T & 0 & 0 & 0 \\ -I & 0 & 0 & 0 \\ 0 & 0 & 0 & -I \end{bmatrix} \tag{22}$$

The stationarity, primal feasibility, dual feasibility, and complementary slackness can be written as follows,

$$\begin{bmatrix} 0 \\ 1 \\ 0 \\ b \end{bmatrix} + \begin{bmatrix} A & -\mathbf{1} & (-(\Sigma_k^1)^{\frac{1}{2}})_{k\in\mathcal{J}_1} & (-\mu_k^1)_{k\in\mathcal{J}_1} \\ (-(\mu_l^2)^T + \Psi_{\xi_l^2}^{-1}\frac{{\Sigma_l^2}^{\frac{1}{2}^T}\Sigma_l^{2\frac{1}{2}}y}{\|\Sigma_l^{2\frac{1}{2}}y\|})_{l\in\mathcal{J}_2} & 0 & 0 & 0 \\ 0 & 0 & (\frac{\delta_k}{\|\delta_k\|})_{k\in\mathcal{J}_1} & (-\Psi_{\xi_k^1}^{-1}\left(\alpha_k^1\right))_{k\in\mathcal{J}_1} \\ \mathbf{1}^T & 0 & 0 & 0 \\ -\mathbf{1}^T & 0 & 0 & 0 \\ -I & 0 & 0 & 0 \\ 0 & 0 & 0 & -I \end{bmatrix}^T \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_{41} \\ u_{42} \\ u_5 \\ u_6 \end{bmatrix} = 0 \tag{23}$$

$$\begin{bmatrix} Ay - v\mathbf{1} - \sum_{k\in\mathcal{J}_1}\left(\Sigma_k^1\right)^{\frac{1}{2}}\delta_k - \sum_{k\in\mathcal{J}_1}\lambda_k\mu_k^1 \\ (-y^T\mu_l^2 + \Psi_{\xi_l^2}^{-1}\left(\alpha_l^2\right)\left\|\left(\Sigma_l^2\right)^{\frac{1}{2}}y\right\| + d_l)_{l\in\mathcal{J}_2} \\ (\|\delta_k\| - \lambda_k\Psi_{\xi_k^1}^{-1}\left(\alpha_k^1\right))_{k\in\mathcal{J}_1} \\ \mathbf{1}^Ty - 1 \\ -\mathbf{1}^Ty + 1 \\ -y \\ -\lambda \end{bmatrix} \leq 0 \tag{24}$$

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_{41} \\ u_{42} \\ u_5 \\ u_6 \end{bmatrix} \geq 0 \tag{25}$$

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_{41} \\ u_{42} \\ u_5 \\ u_6 \end{bmatrix}^T \begin{bmatrix} Ay - v\mathbf{1} - \sum_{k\in\mathcal{J}_1} \left(\Sigma_k^1\right)^{\frac{1}{2}} \delta_k - \sum_{k\in\mathcal{J}_1} \lambda_k \mu_k^1 \\ (-y^T\mu_l^2 + \Psi_{\xi_l^2}^{-1}\left(\alpha_l^2\right) \left\| \left(\Sigma_l^2\right)^{\frac{1}{2}} y \right\| + d_l)_{l\in\mathcal{J}_2} \\ (\|\delta_k\| - \lambda_k \Psi_{\xi_k^1}^{-1}\left(\alpha_k^1\right))_{k\in\mathcal{J}_1} \\ \mathbf{1}^T y - 1 \\ -\mathbf{1}^T y + 1 \\ -y \\ -\lambda \end{bmatrix} = 0 \tag{26}$$

The four decision variables of problem $(\mathcal{P})$, namely y, v, δ, λ, have $n, 1, J_1 * n, J_1$ components, respectively. The function g is composed of $g_1, g_2, g_3, g_{41}, g_{42}, g_5,$ and g_6, with $m, J_2, J_1, 1, 1, n, J_1$ components, respectively. The gradient ∇f is a $(J_1+1)*(n+1)-$vector. The Jacobian ∇g is $(2+m+n+2*J_1+J_2)\times(J_1+1)*(n+1)-$matrix.

3.2. Neural network model

We now propose a neural network model with a given initial value. Let $r = (y, v, \delta, \lambda, u)^T$ be the variables of the neural network. The DNN model is given as follows,

$$\frac{dr}{dt} = \begin{bmatrix} \frac{dy}{dt} \\ \frac{dv}{dt} \\ \frac{d\delta}{dt} \\ \frac{d\lambda}{dt} \\ \frac{du}{dt} \end{bmatrix} = \begin{bmatrix} -\left(\nabla f_y + \nabla g_y^T (u+g)^+\right) \\ -\left(\nabla f_v + \nabla g_v^T (u+g)^+\right) \\ -\left(\nabla f_\delta + \nabla g_\delta^T (u+g)^+\right) \\ -\left(\nabla f_\lambda + \nabla g_{\lambda^1}^T (u+g)^+\right) \\ (u+g)^+ - u \end{bmatrix}, \tag{27}$$

$$r\left(t_0\right) = r_0, \tag{28}$$

where $(u+g)^+ = \max\{0, u+g\}$.

The complexity for solving the neural network (27) is highly dependent on the number of variables. The number of the decision variables y, v, δ, λ is $ns = (J_1+1)*(n+1)$, and the number of the dual variables μ is $nu = 2+m+n+2*J_1+J_2$, leading to a total number of variables $nr = 3+m+2N+(3+n)*J_1+J_2$ for the neural network.

Theorem 3. *The point $r^* = (y^*, v^*, \delta^*, \lambda^*, u^*)^T$ is the equilibrium point of the neural network* (27) *if and only if it is also the KKT point of the SOCP problem.*

Proof. Let $r^* = (y^*, v^*, \delta^*, \lambda^*, u^*)^T$ be the equilibrium of the neural network (27). It follows that $\frac{dr^*}{dt} = 0$,

$$\begin{aligned} -\left(\nabla f_y^* + \nabla g_y^{T^*} (u^*+g^*)^+\right) &= 0 \\ -\left(\nabla f_v^* + \nabla g_v^{T^*} (u^*+g^*)^+\right) &= 0 \\ -\left(\nabla f_\delta^* + \nabla g_\delta^{T^*} (u^*+g^*)^+\right) &= 0 \\ -\left(\nabla f_\lambda^* + \nabla g_\lambda^{T^*} (u^*+g^*)^+\right) &= 0 \\ (u^*+g^*)^+ - u^* &= 0 \end{aligned} \tag{29}$$

Substituting the first four lines by the last line $u^* = (u^* + g^*)^+$, we have

$$\begin{aligned} -\left(\nabla f_y^* + \nabla g_y^{T^*} u^*\right) &= 0 \\ -\left(\nabla f_v^* + \nabla g_v^{T^*} u^*\right) &= 0 \\ -\left(\nabla f_\delta + \nabla g_\delta^{T^*} u^*\right) &= 0 \\ -\left(\nabla f_\lambda^* + \nabla g_\lambda^{T^*} u^*\right) &= 0, \end{aligned} \tag{30}$$

where the KKT conditions stationarity holds. Moreover, $u^* = (u^* + g^*)^+$ result in

$$g^* \leq 0, \quad u^* \geq 0, \quad u^{*\,\mathrm{T}} g^* = 0, \tag{31}$$

where the primal feasibility, the dual feasibility, and the complementary slackness hold.

Conversely, let $r^* = (y^*, v^*, \delta^*, \lambda^*, u^*)$ be the KKT point of the problem $(\mathcal{P})$, then we have

$$\begin{aligned} \nabla f_y^* + \nabla g_y^{T^*} u^* &= 0 \\ \nabla f_v^* + \nabla g_v^{T^*} u^* &= 0 \\ \nabla f_{\delta^*} + \nabla g_\delta^{T^*} u^* &= 0 \\ \nabla f_\lambda^* + \nabla g_\lambda^{T^*} u^* &= 0, \end{aligned} \tag{32}$$

$$g^* \leq 0, \quad u^* \geq 0, \quad u^{*\,\mathrm{T}} g^* = 0. \tag{33}$$

Conditions (33) lead to $u^* = (u^* + g^*)^+$. By substituting this into (32), we obtain

$$\begin{aligned} -\left(\nabla f_y^* + \nabla g_y^{T^*} (u^* + g^*)^+\right) &= 0 \\ -\left(\nabla f_v^* + \nabla g_v^{T^*} (u^* + g^*)^+\right) &= 0 \\ -\left(\nabla f_\delta^* + \nabla g_\delta^{T^*} (u^* + g^*)^+\right) &= 0 \\ -\left(\nabla f_\lambda^* + \nabla g_\lambda^{T^*} (u^* + g^*)^+\right) &= 0 \\ (u^* + g^*)^+ - u^* &= 0 \end{aligned} \tag{34}$$

which is the equilibrium point of the neural network.

□

3.3. Stability analysis

In this subsection, we study the uniqueness and the stability of the equilibrium point.

Lemma 4. *The equilibrium point of the proposed neural network* (27) *is unique.*

Proof. Since the problem $(\mathcal{P})$ has the unique optimal solution $(y^*, v^*, \delta^*, \lambda^*)$, the necessary and sufficient KKT conditions (18) have the corresponding unique solution. From Theorem 3, we see that the equilibrium point of the proposed neural network is a necessary and sufficient condition for being a KKT point (18). Thus the equilibrium point of the neural network is unique. □

Lemma 5. *For an initial value problem* (27) *and* (28)*, there exists a unique continuous solution* $r(t)$*.*

Proof. Since $g, \nabla f$ and ∇g are locally Lipschitz continuous, and the operations $+, \cdot, (\cdot)^+$ would not change the locally Lipschitz property, such that $\nabla f + \nabla g^T (u+g)^+$ and $(u+g)^+ - u$ are locally Lipschitz continuous. According to the Cauchy-Lipschitz theorem, the neural network (27) with an initial point $r(t_0) = r_0$ has a unique solution $r(t), t \in [t_0, T)$, for some $T > t_0$. Additionally, if $r(t)$ is globally bound, the interval $[t_0, T]$ expand to $[t_0, +\infty)$. □

Lemma 6. *The Jacobian matrix $\nabla\Phi(r), \forall r \in \mathbb{R}^{nr}$ is a negative semidefinite matrix.*

Proof. We separate the situations into three cases, depending on the different status of $(u+g)^+ \in \mathbb{R}_+^{nu}$, and show under all three situations $\nabla\Phi(r)$ is negative semidefinite .

For the case where $(u+g)^+$ has zero and non-zero components, such that $0 < p < nu$

$$(u+g)^+ = (\underbrace{u_1 + g_1, \ldots, u_p + g_p}_{p}, \underbrace{0, \ldots, 0}_{nu-p}), \tag{35}$$

the Jacobian matrix $\nabla\Phi(r), \forall r \in \mathbb{R}^{nr}$ is

$$\nabla\Phi(r) = \begin{bmatrix} -\left(\nabla^2 f + \sum_{k=1}^{p} \left((u_k + g_k)\nabla^2 g_k^p\right) + \nabla g^{pT} \nabla g^p\right) & -\nabla g^{pT} \\ \nabla g^p & S \end{bmatrix}, \tag{36}$$

where

$$S = \begin{bmatrix} 0_{p\times p} & 0_{p\times(nu-p)} \\ 0_{(nu-p)\times p} & -I_{(nu-p)\times(nu-p)} \end{bmatrix}. \tag{37}$$

∇g and $\nabla^2 g_k$ denote the Jacobian matrix of the function g and the Hessian matrix of the function g_k. ∇g^p and $\nabla^2 g_k^p$ are the same as ∇g and $\nabla^2 g_k$ for first p row but the remaining $nu - p$ row are all zero.

The matrix $\nabla g^{pT} \nabla g^p$ is positive semidefinite. Since the functions f and g are assumed to be convex and twice differentiable, the Hessian matrices $\nabla^2 f$ and $\nabla^2 g_k, k = 1, 2, \ldots, p$, are positive semidefinite matrices. Furthermore, the positive semidefiniteness of $\nabla^2 g_k$ implies that $\nabla^2 g_k^p$ is positive semidefinite matrix. Moreover, it is clear that matrix S is negative semidefinite matrix. Putting those all together, the Jacobian matrix $\nabla\Phi$ is a negative semidefinite matrix.

For the case where $(u+g)^+$ has all non-zero components, such that $p = nu$

$$(u+g)^+ = (u_1 + g_1, \ldots, u_{nu} + g_{nu}), \tag{38}$$

the Jacobian matrix $\nabla\Phi(r)$ is

$$\nabla\Phi(r) = \begin{bmatrix} -\left(\nabla^2 f + \sum_{k=1}^{m} \left((u_k + g_k)\nabla^2 g_k\right) + \nabla g^T \nabla g\right) & -\nabla g^T \\ \nabla g & 0_{nu\times nu} \end{bmatrix}, \tag{39}$$

Similar to the previous case, we can see that $\nabla\Phi(r)$ is a $(ns + nu) \times (ns + nu)$ negative semidefinite matrix.

For the case where $(u+g)^+$ has all zero components, such that $p = 0$

$$(u+g)^+ = (0, \ldots, 0), \tag{40}$$

the Jacobian matrix $\nabla\Phi(r)$ is

$$\nabla\Phi(r) = \begin{bmatrix} -\nabla^2 f & 0_{ns\times nu} \\ 0_{nu\times ns} & -I_{nu\times nu} \end{bmatrix}. \tag{41}$$

In this case, it is easy to see that $\nabla\Phi(r)$ is a negative semidefinite matrix. This completes the proof. □

Definition 1. *A mapping $F : \mathbb{R}^n \to \mathbb{R}^n$ is said to be monotonic if:*

$$(x-y)^T(F(x)-F(y)) \geq 0, \quad \forall x, y \in \mathbb{R}^n \tag{42}$$

Lemma 7 (Ortega & Rheinboldt (2000))**.** *A differentiable mapping $F : \mathbb{R}^n \to \mathbb{R}^n$ is monotonic, if and only if the Jacobian matrix $\nabla F(x), \forall x \in \mathbb{R}^n$ is positive semidefinite.*

Theorem 8. *The equilibrium point $r^* = (y^*, v^*, \delta^*, \lambda^*, u^*)$ of the proposed neural network (27) is globally asymptotically stable.*

Proof. Consider the following Lyapunov function

$$E(r) = \|\Phi(r)\|^2 + \frac{1}{2}\|r - r^*\|^2. \tag{43}$$

$E(r)$ is a positive definite function because $E(r^*) = 0$ and $E(r) > 0, \quad \forall r \neq r^*$.

$$\frac{d\Phi}{dt} = \frac{\partial\Phi}{\partial r}\frac{dr}{dt} = \nabla\Phi(r)\Phi(r) \tag{44}$$

$$\begin{aligned} \dot{E}(r(t)) &= \left(\frac{d\Phi}{dt}\right)^T \Phi + \Phi^T\left(\frac{d\Phi}{dt}\right) + (r-r^*)^T\frac{dr(t)}{dt} \\ &= \Phi^T\left(\nabla\Phi(r)^T + \nabla\Phi(r)\right)\Phi + (r-r^*)^T\Phi(r) \end{aligned} \tag{45}$$

By Lemma 6, we have

$$\Phi^T(r)\left(\nabla\Phi(r)^T + \nabla\Phi(r)\right)\Phi(r) \leq 0, \quad \forall r \neq r^*. \tag{46}$$

By lemma 7, we have

$$(r-r^*)^T\left(\Phi(r) - \Phi\left(r^*\right)\right) = (r-r^*)^T\Phi(r) \leq 0, \quad \forall r \neq r^*. \tag{47}$$

This means that $\dot{E}(r(t)) \leq 0$.

According to Lyapunov globally stable theorem, the equilibrium r^* of the neural network (27) is globally stable. Moreover, it follows from (27), (28), (46) and (47) that $\Phi(r) = 0 \Leftrightarrow \dot{E}(r) = 0$. This means that $\dot{E}(r) = 0$ is true only for the equilibrium point, such that $\dot{E}(r)$ is a negative definite function. Therefore, the equilibrium point of the neural network is globally asymptotically stable.

□

Figure 1 summarizes the complete process of obtaining a Nash equilibrium point of a stochastic two-player zero-sum game using the DNN method. The original game problem (3)-(4) can not be solved directly. It needs to be converted into a SOCP problem first. Then a DNN is constructed to come up with the optimal solution. Once a DNN model has been developed, the model needs an initial point and a solution interval as an input. The choice of the initial point

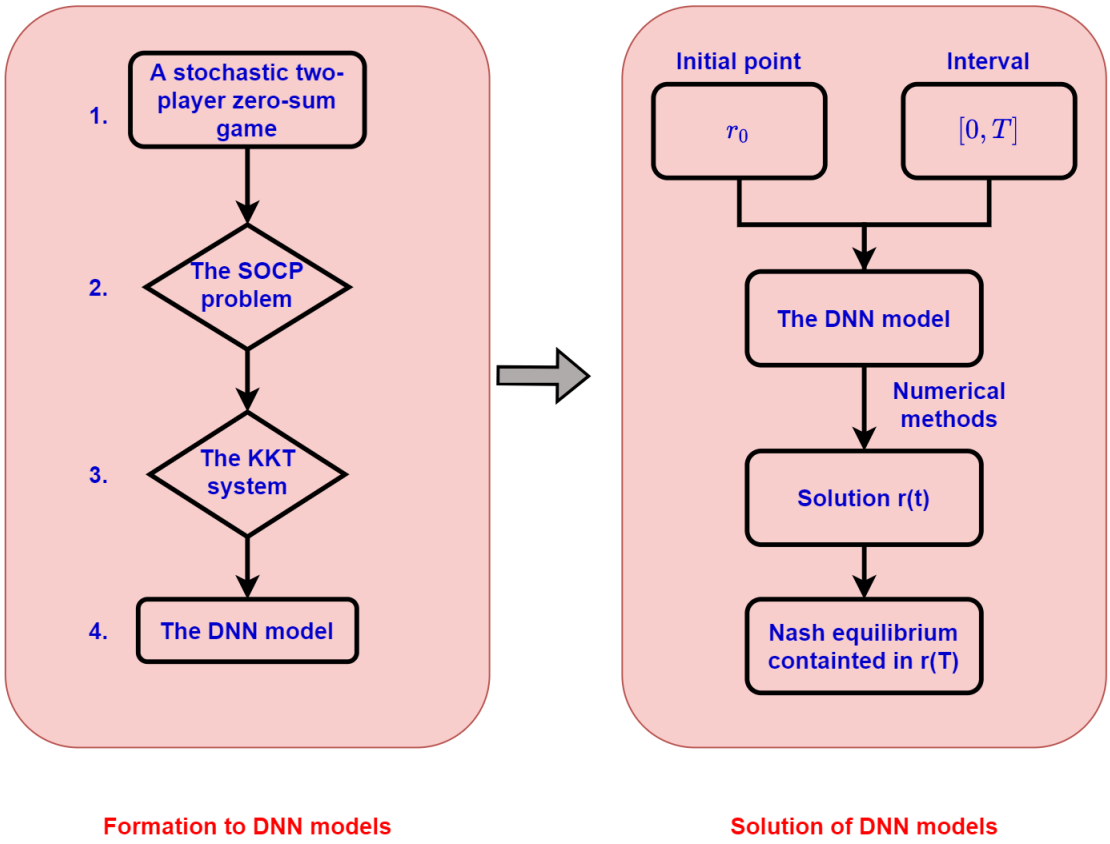


Figure 1: Overall flowchart for obtaining Nash equilibrium of a stochastic nash game using DNN method. The left side describes how to construct a DNN model for a stochastic nash game. The right side shows how to obtain a desired solution from the constructed DNN model.

significantly affects the convergence and quality of the solution. The closer the initial point is to the optimal point, the faster the convergence of the DNN model. The solution interval determines which part of $r(t)$ is considered. According to the globally asymptotically stable Theorem 8, the DNN model should provide a more accurate solution when T is large.

4. Numerical Experiments

In section 4.1, we give the setup for conducting the experiments, including software description, hardware description, the definition of KKT error, and the way to generate problem data. In section 4.2, we give experimental results of our DNN model and compare them with other solving methods, i.e., Splitting conic solver (SCS for short) and Cvxopt (O'donoghue et al., 2016; CVX Research, 2012).

4.1. Settings

The experimental computer has CPU i7-10610U and 16 GB RAM. We use Python3.8 as our programming language, Scipy 1.6 to solve differential equations.

The following definition is a metric to evaluate the accuracy of a solution point (Andreani et al., 2008; Conn et al., 2000)

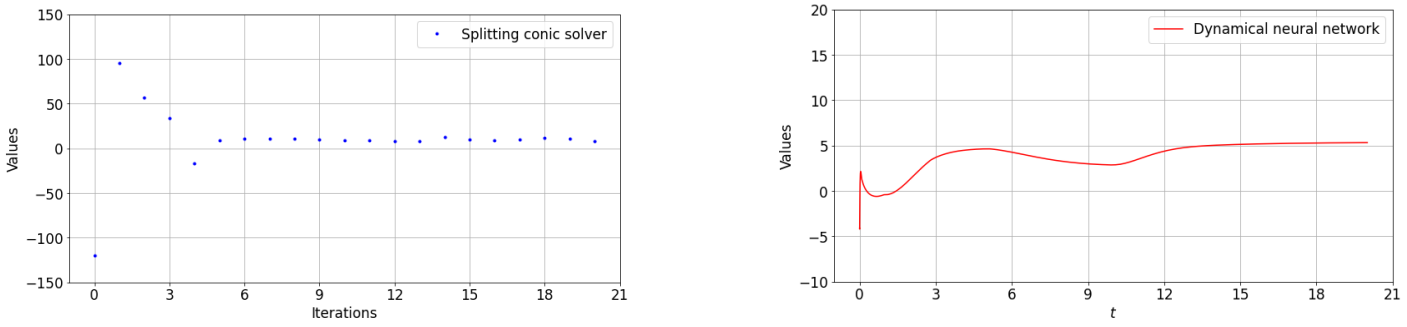
Definition 2. *The point (s, u) is an approximate KKT point with ϵ-error if it satisfies*

$$\begin{aligned} \left\|\nabla f(s) + \nabla g(s)^T u\right\| &\leq \epsilon, \\ ||u \circ g(s)|| &\leq \epsilon, \\ \left\|g(s)_+\right\| &\leq \epsilon, \\ \|u_-\| &\leq \epsilon, \end{aligned} \tag{48}$$

where $\circ$ is the element-wise product, $g(s)_+ = \max\{0, g(s)\}$, $u_- = \min\{u, 0\}$.

We generate the problem data through the following uniform distributions, i.e., $A \sim U(0, 10), b \sim U(7, 10), d \sim U(3, 6), \mu^1 \sim U(0, 10), \mu^2 \sim U(0, 10)$. For the sake of simplicity, we only consider uniformly distributed diagonal matrices Σ^1 and Σ^2, i.e., $\Sigma^1, \Sigma^2 \sim U(0, 3)$.

4.2. Numerical results



(a) Splitting conic solver

(b) Dynamical neural network

Figure 2: Solution processes of SCS and DNN for a $10 * 10$ game instance. Figure (a) shows the objective values with respect to iterations provided by the SCS method. Figure (b) shows the objective values with respect to t by our DNN model.

Figure 2 shows the difference between SCS and DNN solving processes. Both SCS and our DNN approaches find the optimal solution by addressing the KKT conditions. A key feature of our DNN model is that it is time-continuous. This advantage allows our DNN model to provide more information about how primal and dual variables move toward optimal solutions.

Table 1 shows the comparison of the DNN, SCS, and CVXOPT methods in terms of numerical performances. The different instances are generated as described in Section 4.1. Our DNN model chooses the initial point all-zero and the solution interval $[0, 100]$. Compared to SCS, our DNN model can reach a better solution point with smaller ϵ-error. Compared with Cvxopt, our DNN model can solve all instances, while Cvxopt fails to find a solution when the game sizes are large.

Figure 3 shows the quality of the convergence of our DNN model. Thanks to the globally asymptotically stable theorem, the KKT error of our DNN method can keep decreasing, and the model will converge to the optimal solution. The figure shows that our DNN model has a faster convergence rate and better numerical accuracy solution than the SCS method.

When facing tricky stochastic games, traditional convex methods might have numerical convergence difficulties. Our DNN approach remains robust in dealing with these hard problems thanks to the use of a completely different solution strategy. According to the shown CPU time, our DNN approach may be time-consuming. However, this is due to the numerical integration methods and the current standard ODE software. Our DNN method is highly promising, mainly

Game size	Probability distribution	α		DNN			SCS			CVXOPT		
		α_1	α_2	CPU time	Value	ϵ-Error	CPU time	Value	ϵ-Error	CPU time	Value	ϵ-Error
(4, 4)	Normal	0.8	0.8	1.79	4.59	0.12	0.015	5.03	8.20	0.031	4.39	0.00
		0.9	0.9	1.79	4.30	0.05	0.015	4.75	3.23	0.015	4.39	0.00
	Laplace	0.8	0.8	1.66	4.40	0.11	0.015	4.74	4.63	0.015	4.39	0.00
		0.9	0.9	1.28	4.67	0.02	0.015	4.80	2.86	0.015	4.64	0.00
(10, 10)	Normal	0.8	0.8	2.39	5.51	0.01	0.015	5.64	0.45	0.015	5.54	0.00
		0.9	0.9	2.67	5.78	0.09	0.015	5.80	0.25	0.015	5.88	0.00
	Laplace	0.8	0.8	2.61	5.53	0.03	0.015	5.66	1.00	0.015	5.59	0.00
		0.9	0.9	2.65	6.08	0.05	0.015	6.15	0.03	0.031	6.14	0.00
(50, 50)	Normal	0.8	0.8	56.33	5.26	0.18	0.063	4.71	4.59	0.271	5.15	0.00
		0.9	0.9	68.78	5.17	0.09	0.062	4.87	4.71	0.249	5.16	0.00
	Laplace	0.8	0.8	58.37	5.25	0.15	0.055	5.04	7.33	0.257	5.15	0.00
		0.9	0.9	47.07	5.16	0.08	0.046	5.17	2.18	0.249	5.18	0.00
(100, 100)	Normal	0.8	0.8	381.33	5.02	0.02	0.111	4.69	1.77	1.48	5.00	0.00
		0.9	0.9	369.88	4.99	0.04	0.105	4.97	1.56	1.59	5.00	0.00
	Laplace	0.8	0.8	319.04	5.00	0.02	0.109	4.82	1.63	Failed	Failed	Failed
		0.9	0.9	359.95	5.04	0.04	0.109	4.99	1.84	Failed	Failed	Failed
(200, 200)	Normal	0.8	0.8	8984.51	4.97	0.03	0.283	4.75	5.14	10.45	4.96	0.00
		0.9	0.9	8862.24	4.99	0.04	0.281	4.90	2.06	Failed	Failed	Failed
	Laplace	0.8	0.8	8381.08	4.98	0.04	0.252	4.77	4.18	Failed	Failed	Failed
		0.9	0.9	11218.82	5.00	0.05	0.265	4.96	0.94	Failed	Failed	Failed

Table 1: The numerical results of DNN, SCS, and Cvxopt. The game size refers to the size of the action set for each of the two players. The probability distribution represents the distribution followed by each row vector B_k and D_l in (3) and (4). α is the confidence level of the probability constraint. CPU time is calculated in seconds. The value represents the objective function value. The ϵ-Error measures the degree to which the solution point satisfies the KKT condition.

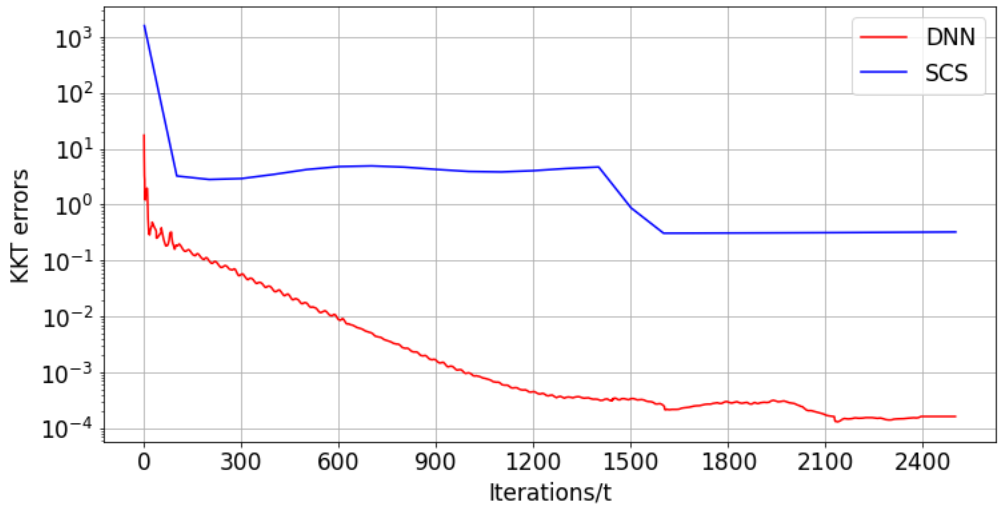


Figure 3: Comparison of the convergence of DNN and SCS methods. The instance is with game size $10 * 10$. The x-axis represents the number of iterations (SCS method) or time (DNN method). The y-axis represents the ϵ error for the KKT conditions.

with future efficient ODE numerical methods and solvers. The performance will be substantially improved in the future with the help of these studies of ODE.

5. Conclusion

In this paper, we studied a dynamical neural network approach to solve a two-player zero-sum game with stochastic linear constraints problem. We reformulated our problem as a second order conic problem. We show that the equilibrium point of the DNN model is the optimal solution for the original problem. By using the Lyapunov stability theory, we prove the globally asymptotically stable and the uniqueness of the equilibrium point of the proposed DNN. Our numerical experiments show the solution process of the DNN model. Our DNN model can give more accurate Nash equilibrium points and remains robust with respect to game size. Finally, our approach opens up new research directions for solving various types of game problems using dynamic neural networks.

Bibliography

Andreani, R., Birgin, E. G., Martínez, J. M., & Schuverdt, M. L. (2008). On augmented lagrangian methods with general lower-level constraints. *SIAM Journal on Optimization*, *18*, 1286–1309. URL: https://doi.org/10.1137/060654797. doi:10.1137/060654797. arXiv:https://doi.org/10.1137/060654797.

Bangyal, W. H., Ahmed, J., & Rauf, H. T. (2020). A modified bat algorithm with torus walk for solving global optimisation problems. *International Journal of Bio-Inspired Computation*, *15*, 1–13.

Bangyal, W. H., Nisar, K., Ag. Ibrahim, A. A. B., Haque, M. R., Rodrigues, J. J. P. C., & Rawat, D. B. (2021). Comparative analysis of low discrepancy sequence-based initialization approaches using population-based algorithms for solving the global optimization problems. *Applied Sciences*, *11*. URL: https://www.mdpi.com/2076-3417/11/16/7591. doi:10.3390/app11167591.

Blum, B., Shelton, C. R., & Koller, D. (2006). A continuation method for nash equilibria in structured games. *Journal of Artificial Intelligence Research*, *25*, 457–502.

Boyd, S., Boyd, S. P., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Charnes, A. (1953). Constrained games and linear programming. *Proceedings of the National Academy of Sciences of the United States of America*, *39*, 639.

Conn, A. R., Gould, N. I., & Toint, P. L. (2000). *Trust region methods*. SIAM.

Curtiss, C. F., & Hirschfelder, J. O. (1952). Integration of stiff equations. *Proceedings of the National Academy of Sciences of the United States of America*, *38*, 235.

CVX Research, I. (2012). CVX: Matlab software for disciplined convex programming, version 2.0. http://cvxr.com/cvx.

Dang, C., Leung, Y., Gao, X.-b., & Chen, K.-z. (2004). Neural networks for nonlinear and mixed complementarity problems and their applications. *Neural Networks*, *17*, 271–283.

Dantzig, G. (1951). A proof of the equivalence of the programming problem and the game problem. *Activity analysis of production and allocation*, (pp. 330–335).

Dantzig, G. B. (1963). *Linear Programming and Extensions*. Santa Monica, CA: RAND Corporation. doi:10.7249/R366.

Feizi, A., Nazemi, A., & Rabiei, M. R. (2021). Solving the stochastic support vector regression with probabilistic constraints by a high-performance neural network model. *Engineering with Computers*, (pp. 1–16).

Flamant, C., Protopapas, P., & Sondak, D. (2020). Solving differential equations using neural network solution bundles. arXiv:2006.14372.

Forti, M., Nistri, P., & Quincampoix, M. (2004). Generalized neural network for nonsmooth nonlinear programming problems. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *51*, 1741–1754.

Gao, X.-B., Liao, L.-Z., & Xue, W. (2004). A neural network for a class of convex quadratic minimax problems with constraints. *IEEE transactions on neural networks*, *15*, 622–628.

Gottlieb, S., & Shu, C.-W. (1998). Total variation diminishing runge-kutta schemes. *Mathematics of computation*, *67*, 73–85.

Govindan, S., & Wilson, R. (2003). A global newton method to compute nash equilibria. *Journal of Economic Theory*, *110*, 65–86.

Hopfield, J. J., & Tank, D. W. (1985). "neural" computation of decisions in optimization problems. *Biological cybernetics*, *52*, 141–152.

Kataoka, S. (1963). A stochastic programming model. *Econometrica*, *31*, 181–196. URL: http://www.jstor.org/stable/1910956.

Kennedy, M. P., & Chua, L. O. (1988). Neural networks for nonlinear programming. *IEEE Transactions on Circuits and Systems*, *35*, 554–562.

van der Laan, G., Talman, A., & Van der Heyden, L. (1987). Simplicial variable dimension algorithms for solving the nonlinear complementarity problem on a product of unit simplices using a general labelling. *Mathematics of operations research*, *12*, 377–397.

Lagaris, I., Likas, A., & Fotiadis, D. (1998). Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, *9*, 987–1000. URL: http://dx.doi.org/10.1109/72.712178. doi:10.1109/72.712178.

Lemke, C. E., & Howson, J. T., Jr. (1964). Equilibrium points of bimatrix games. *Journal of the Society for Industrial and Applied Mathematics*, *12*, 413–423. URL: https://doi.org/10.1137/0112033. doi:10.1137/0112033. arXiv:https://doi.org/10.1137/0112033.

Liao, L.-Z., Qi, H., & Qi, L. (2001). Solving nonlinear complementarity problems with neural networks: a reformulation method approach. *Journal of Computational and Applied Mathematics*, *131*, 343–359.

Nash, J. F. (1950). Equilibrium points in n-person games. *Proceedings of the national academy of sciences*, *36*, 48–49.

Nazemi, A. (2011). A dynamical model for solving degenerate quadratic minimax problems with constraints. *Journal of Computational and Applied Mathematics*, *236*, 1282–1295.

Nazemi, A. (2014). A neural network model for solving convex quadratic programming problems with some applications. *Engineering Applications of Artificial Intelligence*, *32*, 54–62.

Nazemi, A., & Sabeghi, A. (2020). A new neural network framework for solving convex second-order cone constrained variational inequality problems with an application in multi-finger robot hands. *Journal of Experimental & Theoretical Artificial Intelligence*, *32*, 181–203.

Nazemi, A., & Tahmasbi, N. (2013). A high performance neural network model for solving chance constrained optimization problems. *Neurocomputing*, *121*, 540–550.

von Neumann, J. (1928). Zur theorie der gesellschaftsspiele. *Mathematische annalen*, *100*, 295–320.

Nocedal, J., & Wright, S. (2006). *Numerical optimization*. Springer Science & Business Media.

Ortega, J. M., & Rheinboldt, W. C. (2000). *Iterative solution of nonlinear equations in several variables*. SIAM.

O'donoghue, B., Chu, E., Parikh, N., & Boyd, S. (2016). Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory and Applications*, *169*, 1042–1068.

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, *378*, 686–707.

Singh, V. V., & Lisser, A. (2019). A second-order cone programming formulation for two player zero-sum games with chance constraints. *European Journal of Operational Research*, *275*, 839–845.

Teschl, G. (2012). *Ordinary differential equations and dynamical systems* volume 140. American Mathematical Soc.

Wang, J. (1993). Analysis and design of a recurrent neural network for linear programming. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, *40*, 613–618.

Xia, Y. (1996a). A new neural network for solving linear and quadratic programming problems. *IEEE Transactions on Neural Networks*, *7*, 1544–1548. doi:10.1109/72.548188.

Xia, Y. (1996b). A new neural network for solving linear programming problems and its application. *IEEE Transactions on Neural Networks*, *7*, 525–529.

Zhang, J., Jin, L., & Cheng, L. (2020). Rnn for perturbed manipulability optimization of manipulators based on a distributed scheme: A game-theoretic perspective. *IEEE Transactions on Neural Networks and Learning Systems*, *31*, 5116–5126. doi:10.1109/TNNLS.2020.2963998.