Learning to Optimize Differentiable Games

Xuxi Chen [1] **Nelson Vadori** [2] **Tianlong Chen** [1] **Zhangyang Wang** [1]

Abstract

Many machine learning problems can be abstracted in solving game theory formulations and boil down to optimizing nested objectives, such as generative adversarial networks (GANs) and multi-agent reinforcement learning. Solving these games requires finding their stable fixed points or Nash equilibrium. However, existing algorithms for solving games suffer from empirical instability, hence demanding heavy ad-hoc tuning in practice. To tackle these challenges, we resort to the emerging scheme of *Learning to Optimize* (L2O), which discovers problem-specific efficient optimization algorithms through data-driven training. Our customized L2O framework for differentiable game theory problems, dubbed *"Learning to Play Games"* (L2PG), seeks a stable fixed point solution, by predicting the fast update direction from the past trajectory, with a novel gradient stability-aware, sign-based loss function. We further incorporate curriculum learning and self-learning to strengthen the empirical training stability and generalization of L2PG. On test problems including quadratic games and GANs, L2PG can substantially accelerate the convergence, and demonstrates a remarkably more stable trajectory. Codes are available at `https://github.com/VITA-Group/L2PG`.

1. Introduction

While a substantial fraction of recent progress in machine learning is based on optimizing neural networks with respect to a single objective function, there is an increasing set of problems that require optimizing nested objectives, such as generative adversarial networks (Goodfellow et al., 2014) and multi-level optimization (Pfau & Vinyals, 2016), among many of them. These multi-objective problems can be effectively modeled as *differentiable games* (Balduzzi et al., 2018), where *players* correspond to internal parts of the problems and different objective functions are introduced for different players.

The solutions of differentiable games can be generally defined as equilibrium points, such as (local) Nash equilibrium or stable equilibrium (stable fixed points) (Balduzzi et al., 2018). From the perspective of optimization, the latter is more appealing (Balduzzi et al., 2018) and will be our main focus. Although various algorithms have been proposed to locate stable fixed points, they suffer from empirical instability. Concretely, hyperparameters such as learning rates and coefficients of regularization can significantly affect the convergence speed of these algorithms. Moreover, a pre-chosen set of hyperparameters cannot consistently achieve satisfactory convergence speed across different game instances. Therefore, these algorithms demand ad-hoc tuning, generally done by heavy trial-and-error experiments, making them less efficient in practice.

We propose an automatic and flexible algorithm for solving differentiable games to tackle these challenges. An optimization paradigm, termed "Learning to Learn" (Lv et al., 2017; Andrychowicz et al., 2016) or "Learning to Optimize" (L2O) (Chen et al., 2022a), is a well-suited candidate. It *learns* problem-specific efficient optimization algorithms, usually parameterized by a neural network (called an "optimizer"), from its performance on a set of optimization problems. The optimizer produced through data-driven training is capable of efficiently solving optimization problems unseen yet similar to those encountered during training.

However, several questions need to be answered before extending L2O into solving differentiable games: Firstly, L2O trains an optimizer based on its "performance" (*e.g.* training loss) on a series of optimization problems, but there is no off-the-shelf metric that characterizes both the distance to fixed points and stability; Secondly, how to represent the game dynamics for L2O models, and how to generate updates to each player's parameters, remain unexplored; Lastly, the game dynamics can sometimes be noisy and cyclic, which could further hinder the learning of L2O optimizers that are already known to be unstable.

In this work, we present an L2O framework called "*Learning to Play Games*" (L2PG), which attempts to tackle the

[1]University of Texas at Austin [2]J.P. Morgan AI Research. Correspondence to: Xuxi Chen <xxchen@utexas.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

above challenges and extends L2O to solving differentiable games. L2PG learns to predict updates to players' parameters from historic trajectories via minimizing a novel stability-aware sign-based loss function that leads the players' parameters to be stable yet away from unstable fixed points. As L2O optimizers often suffer from unstable training and poor generalization (Chen et al., 2020a), we also incorporate customized curriculum learning and self-learning techniques, which provide substantial remediation to these pitfalls. We summarize our contributions as follows:

1. We propose a new L2O framework, called *Learning to Play Games* (L2PG), that aims at solving differentiable games by seeking stable fixed points. A novel stability-aware, sign-based loss function is leveraged during the training stage, which helps the optimizer to locate stable solutions and avoid unstable fixed points.
2. We customize new curriculum learning and self-training techniques, to enhance the generalization ability and stabilize the learning process of L2PG.
3. Extensive experiments show that L2PG outperforms baseline methods in terms of the speed of finding stable fixed points. On two-player and four-player quadratic games, L2PG substantially accelerates the convergence to stable fixed points by significant margins, while avoiding the convergence to unstable fixed points. We also show that L2PG can achieve a faster convergence speed in optimizing GANs.

2. Related Works

2.1. Learning to Optimize

Learning to optimize (L2O) is one specific instance of meta-learning that aims at learning a network to solve optimization tasks. The first L2O framework is introduced by Andrychowicz et al. (2016), where a recurrent neural network (RNN) is utilized to predict updates for the optimization problem's variables from the objective value and the gradients on the variables. Later on, Li & Malik (2016) proposed a reinforcement learning framework that used the historic gradient information as well as the objective values to predict updates on problems' variables. To enhance the generalization of learned optimizers, Wichrowska et al. (2017) proposed a hierarchical RNN structure with additional features, which shows better generalization on longer training horizons; Chen et al. (2020a); Yang et al. (2023) introduce more training techniques that enhance the learned optimizer's generalization.

Regarding the applications of L2O, Chen et al. (2017); You et al. (2020); Chen et al. (2020b) applied L2O methods in black-box optimization, graph neural networks, and domain adaptation, respectively. Cao et al. (2019) used multiple L2O optimizers to optimize swarm particles. Zheng et al. (2022) distilled the learned optimization rules with symbolic regression and enhanced the explainability of the learned rules. Chen et al. (2022b) applied L2O to train the neural network in a subspace more efficiently.

One closely related prior art to ours is (Shen et al., 2020), which extends the RNN-based L2O in (Andrychowicz et al., 2016) to solving the minimax optimization - often considered as one particular type of game. However, their method is tailored for two-player zero-sum games, and cannot easily extend to finding stable fixed points in general games.

2.2. Learning in Differentiable Games

The game formulations have been introduced for modeling machine learning problems, such as computer vision (Goodfellow et al., 2014) and reinforcement learning (Busoniu et al., 2008). Gradient-based methods have been developed to find the stationary points (either Nash equilibrium or stable fixed points). Daskalakis et al. (2017) proposed Optimistic Mirror Descent that extrapolates the next gradient by using historic information and Gidel et al. (2018) explores several variants of the extrapolation algorithms. Various methods have been proposed and designed with guaranteed convergence on specific games, such as Consensus Optimization (Mescheder et al., 2017) that has convergence guarantee on two-player bi-linear zero-sum games and Competitive Gradient Descent (Schäfer & Anandkumar, 2019) that has convergence guarantee on two-player zero-sum games. Vadori et al. (2022) applied reinforcement learning in solving the constrained two-player zero-sum games.

Several works have studied the general games: Foerster et al. (2017) proposed learning with opponent-learning awareness (LOLA) leverages other players' information to calculate the anticipated parameter update of one player. Letcher et al. (2018) proposed Stable Opponent Shaping that maintains the theoretical guarantee of convergence. Balduzzi et al. (2018); Letcher et al. (2019) proposed the Symplectic Gradient Adjustment (SGA) that aims at adjusting the direction of updates towards the stable fixed points using generalized Helmholtz decomposition, and Ramponi & Restelli (2021) further developed a Newton-like method that can find stable fixed points in multi-agent reinforcement learning problems. In this work, we propose to use L2O as a novel tool to find stable fixed points for general differential games.

3. Methodology

3.1. Preliminaries

Differentiable Games A general differentiable game (Balduzzi et al., 2018) consists of n players $= \{1, 2, \ldots, n\}$ and corresponding twice differentiable losses for each player $= \{l_i : \mathbb{R}^d \mapsto \mathbb{R}\}_{i=1}^n$. The parameters for the players are

defined as $\boldsymbol{w} = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \dots, \boldsymbol{w}_n\}$, where the i-th player controls $\boldsymbol{w}_i \in \mathbb{R}^{d_i}$ and $\sum_{i=1}^n d_i = d$.

The simultaneous gradient of the game is defined as the concatenated gradient of each player's loss (l_i) with respect to its corresponding parameters ($\boldsymbol{w}_i$), *i.e.*, $\boldsymbol{g}(\boldsymbol{w}) = (\boldsymbol{g}_1(\boldsymbol{w}_1), \boldsymbol{g}_2(\boldsymbol{w}_2), \dots, \boldsymbol{g}_n(\boldsymbol{w}_n)) = (\nabla_{\boldsymbol{w}_1} l_1, \nabla_{\boldsymbol{w}_2} l_2, \dots, \nabla_{\boldsymbol{w}_n} l_n)$. Besides the simultaneous gradient, we will also utilize the Jacobian of a game, which is a $d \times d$ matrix containing second-order derivatives:

$$\mathbf{J}(\boldsymbol{w}) = \begin{pmatrix} \nabla^2_{\boldsymbol{w}_1} l_1 & \nabla_{\boldsymbol{w}_1,\boldsymbol{w}_2} l_1 & \dots & \nabla_{\boldsymbol{w}_1,\boldsymbol{w}_n} l_1 \\ \nabla_{\boldsymbol{w}_2,\boldsymbol{w}_1} l_2 & \nabla^2_{\boldsymbol{w}_2} l_2 & \dots & \nabla_{\boldsymbol{w}_2,\boldsymbol{w}_n} l_2 \\ \vdots & \vdots & & \vdots \\ \nabla^2_{\boldsymbol{w}_n,\boldsymbol{w}_1} l_n & \nabla_{\boldsymbol{w}_n,\boldsymbol{w}_2} l_n & \dots & \nabla^2_{\boldsymbol{w}_n} l_n \end{pmatrix},$$

where $\nabla_{\boldsymbol{w}_i,\boldsymbol{w}_j} l_k$ is a $d_i \times d_j$ matrix and $\nabla^2_{\boldsymbol{w}_i} l_i$ is a $d_i \times d_i$ matrix. We adopt the definition of stable and unstable fixed points from Balduzzi et al. (2018), *i.e.*, $\boldsymbol{w}$ is called a fixed point if $\boldsymbol{g}(\boldsymbol{w})$ is $\mathbf{0}$; a fixed point $\boldsymbol{w}^*$ is called *stable* if $\boldsymbol{J}(\boldsymbol{w}) \succeq 0$; and is called *unstable* if $\boldsymbol{J}(\boldsymbol{w}) \prec 0$ for every $\boldsymbol{w}$ in the neighborhood of $\boldsymbol{w}^*$, where $\succeq$ denotes positive semi-definiteness and $\prec$ denotes negative definiteness. With this definition, *every* stable fixed points is a (local) Nash equilibrium of a game (Balduzzi et al., 2018).

Learning to Optimize In L2O, an optimizer is usually a neural network `opt` parameterized by $\boldsymbol{\phi}$. An observation vector $\boldsymbol{z}^{(t)}$ is constructed for players' parameters $\boldsymbol{w}^{(t)}$, where t indicates that the number of update steps.

Generally, the pipeline of L2O can be split into two stages. The first one is called *meta-training*, where the parameters of the optimizer $\boldsymbol{\phi}$ are being updated by minimizing the following loss (Chen et al., 2022a):

$$\mathcal{L}(\boldsymbol{\phi}) = \mathbb{E}_{l_i}\left[\sum_{t=1}^{T} a_t \mathcal{L}_t(\{l_i\}_{i=1}^n, \boldsymbol{w}^{(t)})\right], \tag{1}$$

with $\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \texttt{opt}(\boldsymbol{z}^{(t)}; \boldsymbol{\phi})$, where $\mathcal{L}_t(\cdot,\cdot)$ is the function for evaluating the performance of the L2O optimizer at time step t. The coefficients $\{a_t\}_{t=1}^T$ are introduced to balance the losses between different time steps, which we by default set as 1 to distribute the same importance. In each training rounds, the optimizer will be required to solve new games with newly initialized players' parameters.

The second stage is called *meta-testing*, where the parameters of the L2O optimizer are fixed. The optimizer is utilized to solve new unseen game problems at this stage, which is expected to find stable fixed points rapidly and amortize the meta-training cost.

3.2. Formulating L2PG

We now explain the framework of L2PG in details. We have summarized the workflow of L2PG in Algorithm 1.

3.2.1. Featurizing Game Dynamics for L2O

The first critical design is how to construct the observation vector $\boldsymbol{z}^{(t)}$ for each time step. While previously L2O optimizers primarily use the gradient information (*i.e.*, $\boldsymbol{g}(\boldsymbol{w}^{(t)})$) and its momentums to predict the updates for networks' parameters, they are not sufficiently representing the game dynamics due to the potential existence of "rotational forces" in certain games (Balduzzi et al., 2018). Using only gradients to update the players' parameters fails to, or slowly, reach convergence on some games (Balduzzi et al., 2018), and is often not optimal on others. Therefore, we involve additional information to depict the game dynamics, *i.e.*, the combinations of second-order derivatives. This additional information is found to be beneficial and accelerate the empirical convergence (see Section 4.4).

Specifically, we consider the generalized Helmholtz decomposition (Letcher et al., 2019) which decomposes the Jacobian into symmetric and antisymmetric components $\mathbf{S}(\boldsymbol{w}) = (\mathbf{J}(\boldsymbol{w}) + \mathbf{J}(\boldsymbol{w})^\top)/2$ and $\mathbf{A}(\boldsymbol{w}) = (\mathbf{J}(\boldsymbol{w}) - \mathbf{J}(\boldsymbol{w})^\top)/2$. These components are respectively associated to the potential and Hamiltonian (cyclic) dynamics of the game. Consensus optimization considers an update rule based on $\mathbf{J}^\top(\boldsymbol{w})\boldsymbol{g}(\boldsymbol{w})$, whereas SGA considers $\mathbf{A}^\top(\boldsymbol{w})\boldsymbol{g}(\boldsymbol{w})$. We will generalize these update rules by considering both $\mathbf{A}^\top(\boldsymbol{w})\boldsymbol{g}(\boldsymbol{w})$ and $\mathbf{S}(\boldsymbol{w})\boldsymbol{g}(\boldsymbol{w})$ independently. The three components $\boldsymbol{g}, \mathbf{A}^\top\boldsymbol{g}$ and $\mathbf{S}\boldsymbol{g}$ are stacked to construct a feature matrix $\boldsymbol{o}^{(t)} = [\boldsymbol{g}(\boldsymbol{w}^{(t)}); \mathbf{S}(\boldsymbol{w}^{(t)})\boldsymbol{g}(\boldsymbol{w}^{(t)}); \mathbf{A}^\top(\boldsymbol{w}^{(t)})\boldsymbol{g}(\boldsymbol{w}^{(t)})] \in \mathbb{R}^{n\times 3}$. Following Zheng et al. (2022), we further include the momentum terms (Lv et al., 2017), *i.e.*, $\boldsymbol{m}^{(t)}, \tilde{\boldsymbol{m}}^{(t)}$ and $\tilde{\boldsymbol{o}}^{(t)}$:

$$\begin{aligned} &\boldsymbol{m}^{(t)} = \beta_1 \boldsymbol{m}^{(t-1)} + (1-\beta_1)\boldsymbol{o}^{(t)}, \\ &(\boldsymbol{v}^{(t)})^2 = \beta_2 (\boldsymbol{v}^{(t-1)})^2 + (1-\beta_2)(\boldsymbol{o}^{(t)})^2, \\ &\hat{\boldsymbol{m}}^{(t)} = \boldsymbol{m}^{(t)}/(1-\beta_1^t), \hat{\boldsymbol{v}}^{(t)} = \boldsymbol{v}^{(t)}/(1-\beta_1^t), \\ &\tilde{\boldsymbol{m}}^{(t)} = \hat{\boldsymbol{m}}^{(t)} \odot (1/\boldsymbol{v}^{(t)}), \tilde{\boldsymbol{o}}^{(t)} = \boldsymbol{o}^{(t)} \odot (1/\boldsymbol{v}^{(t)}), \end{aligned} \tag{2}$$

to construct $\boldsymbol{z}^{(t)} = [\boldsymbol{o}^{(t)}; \boldsymbol{m}^{(t)}; \tilde{\boldsymbol{m}}^{(t)}; \tilde{\boldsymbol{o}}^{(t)}]$ as the input features to the L2O optimizer. These auxiliary features are found to benefit the meta-training process (Chen et al., 2020a; Zheng et al., 2022) as they capture historic information along the training process.

The second design is how the L2O optimizer updates the players' parameters. Although most existing L2O works (Chen et al., 2022a) directly predict the exact update amounts, they require an additional factor to scale the outputs to a reasonable range, which also requires tuning for every optimization tasks. Instead, we propose to utilize the optimizer's predictions to interpolate the three components aforementioned. In our design, the L2O optimizer outputs three *update coefficients*, namely $\boldsymbol{c}_{\boldsymbol{g}}, \boldsymbol{c}_{\mathbf{A}}$ and $\boldsymbol{c}_{\mathbf{S}} \in \mathbb{R}^d$, and

derive the *actual update* by the following formula:

$$\begin{aligned}\texttt{opt}(\boldsymbol{z}^{(t)}, \boldsymbol{\phi}) = \boldsymbol{c_g} \odot \boldsymbol{g}(\boldsymbol{w}^{(t)}) \\ + \boldsymbol{c_A} \odot (\mathbf{A}^\top(\boldsymbol{w}^{(t)})\boldsymbol{g}(\boldsymbol{w}^{(t)})) \\ + \boldsymbol{c_S} \odot (\mathbf{S}(\boldsymbol{w}^{(t)})\boldsymbol{g}(\boldsymbol{w}^{(t)})),\end{aligned} \tag{3}$$

where $\odot$ means the point-wise multiplication operation. A $\texttt{tanh}$ activation function precedes $\boldsymbol{c_g}, \boldsymbol{c_A}$ and $\boldsymbol{c_S}$, so the values of elements in them are in $(-1, 1)$. It is noteworthy that our L2O algorithm generalizes the SGA and ConOpt: (1) $\boldsymbol{c_S} = 0$ corresponds to SGA; (2) $\boldsymbol{c_S} = \boldsymbol{c_A}$ corresponds to ConOpt. Our L2O formulation has higher flexibility by relaxing these pre-defined constraints, and can achieve higher performance (refer to Section 4.4).

3.2.2. Stability-aware Meta-Training Loss

Training an L2O optimizer is to minimize an objective function $\mathcal{L}_t$. $\mathcal{L}_t$ is critical in solving games as it characterizes the solution's properties. An ideal $\mathcal{L}_t$ should lead to fixed points while being stability-aware: (1) it should find players' parameters where the simultaneous gradient $\boldsymbol{g}(\boldsymbol{w})$ is close to zero (when applicable), and (2) it should repel the players' parameters from unstable fixed points.

To this end, we propose to use the following stability-aware sign-based loss function for meta-training:

$$\mathcal{L}_t(\{l_i\}_{i=1}^n, \boldsymbol{w}^{(t)}) = \frac{1}{2}\text{sign}(\boldsymbol{g}^\top \mathbf{S}\boldsymbol{g})\|\boldsymbol{g}\|_2^2, \tag{4}$$

where $\boldsymbol{g} = \boldsymbol{g}(\boldsymbol{w}^{(t)})$ and $\mathbf{S} = \mathbf{S}(\boldsymbol{w}^{(t)})$.

Proposition 3.1. *Any stable fixed point $\boldsymbol{w}_s^*$ is a local minimum of the loss $\mathcal{L}_t$ in (4). Any unstable fixed point $\boldsymbol{w}_u^*$ satisfies that for any closed ball $\mathcal{B}$ centered at $\boldsymbol{w}_u^*$, $\mathcal{L}_t$ attains its minimum at a point $z \neq \boldsymbol{w}_u^*$. To wit, minimizing $\mathcal{L}_t$ leads the players' parameters to $\boldsymbol{w}_s^*$, but away from $\boldsymbol{w}_u^*$.*

Proof. For the first claim, we have by definition of a stable fixed point that $\boldsymbol{g}^\top \mathbf{S}\boldsymbol{g} \geq 0$ in a neighborhood of $\boldsymbol{w}_\text{s}^*$, hence in that neighborhood, $\mathcal{L}_t = \frac{1}{2}\|\boldsymbol{g}\|_2^2$. Since $\|\boldsymbol{g}\|_2 = 0$ at $\boldsymbol{w}_\text{s}^*$, the claim is true. For the second claim, simply observe that $\mathcal{L}_t = -\frac{1}{2}\|\boldsymbol{g}\|_2^2$ in a neighborhood $\mathcal{N}$ of $\boldsymbol{w}_\text{u}^*$ since $\boldsymbol{g}^\top \mathbf{S}\boldsymbol{g} < 0$ for $g \neq 0$ by unstability. Since $S \prec 0$, $\boldsymbol{g}$ cannot be identically 0 on $\mathcal{N}$ hence $\mathcal{L}_t$ is not minimized at $\boldsymbol{w}_\text{u}^*$ for any closed ball centered at $\boldsymbol{w}_\text{u}^*$ (a closed ball is compact, hence the infimum is always attained in the ball). □

Remark. In Equation 4 the stability constraint is imposed in a *multiplicative* form. An alternative is to pursue an *additive* regularization form, *i.e.*,

$$\mathcal{L}_t'(\{l_i\}_{i=1}^n, \boldsymbol{w}^{(t)}) = \frac{1}{2}\|\boldsymbol{g}\|_2^2 - \min\{\boldsymbol{g}^\top \mathbf{S}\boldsymbol{g}, 0\}, \tag{5}$$

which has similar effects on the training procedure at the first sight. $\mathcal{L}_t'$ is smaller in the neighborhood of stable fixed points compared to unstable fixed points when the norms of the gradient $\boldsymbol{g}$ are the same, so stable fixed points are preferred. However, if a game has only unstable fixed points $\boldsymbol{w}_\text{u}^*$, then $\boldsymbol{w}^{(t)}$ will be inevitably attracted to $\boldsymbol{w}_\text{u}^*$. As $\mathbf{S}$ will be negative definite around the unstable fixed points, $\mathcal{L}_t'$ can be re-expressed as $\frac{1}{2}\|\boldsymbol{g}\|_2^2 + \boldsymbol{g}^\top(-\mathbf{S})\boldsymbol{g}$, which is positive definite. Moreover, the exact magnitude of $\boldsymbol{g}^\top \mathbf{S}\boldsymbol{g}$ is less meaningful compared to the signs, which already indicates the stability around fixed points. Therefore, we choose to use the multiplicative regularization by default, and we compare these two regularization forms in Section 4.4 to empirically validate our proposition.

3.3. Generalizing and stabilizing L2PG

Lastly, as similarly shown in existing L2O literature, training L2O optimizer often faces the challenge of training instability, and a simply trained L2O optimizer often struggles when generalizing to unseen problems at the meta-testing time (See Table 4).

Self-Learning. The unstable behaviors occur at the meta-training stage and harm the learning stability of the optimizers. To remedy this, we propose a *self-learning* (SL) technique that enhances the stability of L2PG. We introduce an additional L2O optimizer with the same structure, to "mimic" the behaviors of the optimizer normally trained by $\mathcal{L}$. We update the parameters of this newly introduced "*student*" optimizer $\boldsymbol{\phi}'$ via two paths: similarity losses and exponential moving average. In each epoch, the two optimizers are required to solve the same set of games with the same initial players' parameters, where the student attempts to imitate the accumulated updates to players' parameters generated by the original optimizer. Mathematically speaking, let the players' parameters updated by the student optimizer be $\boldsymbol{w}'$, we update $\boldsymbol{\phi}'$ with respect to $\mathcal{L}_\text{sim} = \sum_{t=1}^{[T/K]} \|\boldsymbol{w}'^{(t)} - \boldsymbol{w}^{(Kt)}\|_2^2$, where K is a hyper-parameter that controls the number of accumulation steps. In practice, we found $K = 5$ works well (refer to Table 14). Additionally, we transfer knowledge from the original optimizer to the student by performing an averaged weight update: $\boldsymbol{\phi}' \leftarrow \beta_\phi \boldsymbol{\phi}' + (1 - \beta_\phi)\boldsymbol{\phi}$. The value of β_ϕ is set to 0.95 (we show in Table 13 it is not highly sensitive).

Curriculum Learning. To enhance the generalization of the learned optimizers, we propose to leverage the concept of curriculum learning (CL). Curriculum learning (Bengio et al., 2009) is originally introduced to optimize neural networks progressively: first only train on subsets of training data containing only "easy" data points, and then gradually switch to use all data for training. While existing CL techniques for L2O solely focused on adjusting the number of training iterations (Shen et al., 2020; Chen et al., 2020a), the training process of L2PG is actually impeded by the nature of optimization problems (games) themselves. For example, games that only have unstable solutions ideally

require L2PG to predict updates to diverge from any fixed point, while other games that have stable solutions demand L2PG to find these stable points. Motivated by this, we propose a new CL technique called "*curriculum learning on data*" (Data-CL). Specifically, we progressively involve games with more diverse dynamics (*e.g.*, magnitude of coefficients, stability of games) in the meta-training stage, to let L2PG gradually learn comprehensive knowledge.

Applying CL and SL. We by default apply the aforementioned Data-CL and self-learning method during the meta-training stage. We perform ablation studies in Section 4.4 and empirically prove the effectiveness of our proposals. Additionally, we follow Chen et al. (2020a) to progressively increase the number of training iterations in each epoch (we call this technique "*Training-CL*"). This technique helps mitigate the dilemma between the truncation bias and gradient explosion (Chen et al., 2020a) and hence improves the quality of the trained L2O optimizer. In practice, we apply the SL technique after 10% of the whole training period for better supervision from the "teacher" model. We also clip the loss with excessively large magnitude to avoid overflow.

4. Experiments

4.1. Implementation Details

Baselines. We compare against two analytical baseline optimizers (*i.e.*, not learned optimizers): SGA (Letcher et al., 2019) and consensus optimization (ConOpt) (Mescheder et al., 2017). SGA updates the players' parameters by the following formula: $\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha(\boldsymbol{g} + \lambda \mathbf{A}^\top \boldsymbol{g})$, where α is the learning rate and λ is the coefficient for adjustment. ConOpt updates the parameters by $\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha(\boldsymbol{g} + \lambda \mathbf{J}^\top \boldsymbol{g})$, with the same meaning of α and λ.

Game Types and Generations. We study two types of games: (1) quadratic games, whose loss functions are in the quadratic forms, *i.e.*, $l_i(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}_i^\top \boldsymbol{M}_{ii}\boldsymbol{w}_i + \sum_{j\neq i} \boldsymbol{w}_i^\top \boldsymbol{M}_{ij}\boldsymbol{w}_j + \boldsymbol{w}_i^\top \boldsymbol{b}_i$. We consider two variants of the quadratic games, *i.e.*, a simple two-player game (Ibrahim et al., 2020), and a more complicated four-player game (Balduzzi et al., 2018). Each player controls only one parameter; (2) two-dimensional GANs problem studied in Metz et al. (2016). We sample data for training the generator and discriminator from a mixture of 16 Gaussian distributions arranged in a 4×4 grid. We follow Letcher et al. (2019) to set the generator and discriminator network to both have 6 MLP layers (64 neurons each with ReLU activation (Nair & Hinton, 2010)). The generator will output a two-dimensional vector while the discriminator outputs a scalar.

Meta-training details. We use an LSTM optimizer with a hidden dimension of 32 in all experiments. A detailed explanation of the structure of the L2O optimizer can be found in Section B. The unroll length (*i.e.,* the value of T) is set to 10. We batch-ify the training process by simultaneously training on 128 different games, and we train the optimizer for 300 epochs. The number of training iterations in each epoch takes values from $\{50, 100, 200, 500, 1000\}$ increasingly if the Training-CL technique is applied, otherwise we set the number of training iterations to be 100. We train the parameters in L2PG (*i.e.*, $\boldsymbol{\phi}$) by using the Adam optimizer (Kingma & Ba, 2014), with an initial learning rate of 1×10^{-3}. We decay the learning rate by 10 every $1/3$ of the total number of training epochs. We evaluate the L2O optimizer on a fixed set of game instances with the same type (*i.e.,* quadratic or GANs) every 5 epochs, and the optimizer with the highest evaluation performance will be used at the meta-testing stage.

To apply the Data-CL technique, we linearly increase the probability to sample games with unstable fixed points when meta-training on quadratic games from 0 to 10%. For the generative networks, we set the standard deviation for initialization to be $\frac{2}{\sqrt{d_{\text{out}}}}(\texttt{epoch}/300 + 0.5)$ where `epoch` means the current epoch and d_{out} means the output dimension.

Metrics. We slightly adapt the definition of convergence from Letcher et al. (2019). For quadratic games, we say an algorithm has achieved convergence if the average gradient norm of the last 10 steps is smaller than 1×10^{-3}. The convergence speed is then measured by the total number of steps. If the convergence is towards the stable fixed point, then a smaller step is preferred; otherwise, we wish the step to be large, or even not converging. In practice, we end the algorithms after 1000 steps no matter the algorithm converged or not. For GANs, we run 10000 steps.

Evaluation and (meta-)testing details. For the quadratic games, we sample 20 different game coefficients $\{\boldsymbol{M}_{ij}, \boldsymbol{b}_i\}$ with stable solutions to constitute the evaluation set. The elements in $\boldsymbol{M}_{ij}$ and $\boldsymbol{b}_i$ are sampled from the standard normal distribution. We evaluate the performance of L2PG and tune the hyperparameters of SGA and ConOpt on this set. Afterward, we compare the convergence speed of L2PG with other baselines on hold-out testing sets, whose coefficients are also sampled from standard normal distributions. We construct two testing sets, each of which contains 20 games with stable and unstable fixed points, respectively. All the sampled games' coefficients can be found in the Appendix. For each game, we randomly draw four different initial parameters of the players from the unit disk (*i.e.*, $\|\boldsymbol{w}^{(0)}\|_2 \leq 1$). All the methods will start with the same initial parameters for a fair comparison, and the average of steps to convergence is recorded.

4.2. Main Results

L2PG converges faster to stable fixed points. We first focus on two-player quadratic games. For SGA and ConOpt, we search the values of hyper-parameters(α and λ) over

$\{0.1, 0.2, \ldots, 2\}$ on the evaluation set of games to find those that can lead to the fastest convergence speed.

Table 1 shows the number of updates required before convergence to the stable fixed points, by using L2PG, SGA, and ConOpt. L2PG brings a significant boost to the convergence speed on the evaluation set by over 20 compared to both baseline methods, and also generalizes well on the testing set, demonstrating a substantial performance gap. We additionally provide the convergence speed of SGA and ConOpt with different combinations of hyperparameters on both the evaluation and testing set in Section A.3, where we can observe that the hyperparameters show great influence on empirical convergence, and necessitate the ad-hoc tuning process. L2PG, however, can predict the updates to players' parameters automatically and achieve fast convergence at the same time. We also used the Adam (Kingma & Ba, 2014) optimizer to solve the two-player games. The convergence step on the testing set is 248.71, which clearly shows that utilizing adaptive variants of traditional optimization techniques (Adam) does not yield satisfactory results.

Table 1. Convergence steps on the testing set of quadratic games with stable fixed points.

Method	Best Evaluation Steps	Transfer Test Steps
SGA	61.450	35.963
ConOpt	76.063	44.938
L2PG	**41.025**	**26.375**

L2PG can avoid convergence to unstable fixed points. We then investigate how L2PG and other algorithms update players' parameters when the fixed points are unstable. The convergence speeds are collected in Table 2. On the 20 quadratic games with unstable fixed points, L2PG almost certainly avoids convergence to unstable fixed points. This empirically proves that L2PG is able to differentiate the different dynamics of stable and unstable equilibrium and predict suitable updates to players.

Table 2. Convergence steps on testing sets of quadratic games with different types of fixed points (stable and unstable fixed points).

Method	Stable Fixed Points	Unstable Fixed Points
SGA	35.963	968.1
ConOpt	44.938	1000.0
L2PG	**26.375**	951.4

L2PG on four-player games. We continue to apply L2PG to solving the aforementioned four-player games and compare with SGA and ConOpt. Each player controls 1 parameter in this series of the game, and the parameters of the games are characterized as:

$$M_{ij} = \begin{cases} 1, & i < j \\ \epsilon/2, & i = j \\ -1, & i > j \end{cases}, \quad \boldsymbol{b}_i = 0, \tag{6}$$

for $i, j \in \{1, 2, 3, 4\}$. We vary the value of ϵ when training the optimizer. We use $\epsilon = \{0.1, 0.3, 0.5, 0.7, 0.9\}$ for the evaluation set, and we benchmark all methods on the same testing set ($\epsilon \in \{0.2, 0.4, 0.6, 0.8\} \cup \{1.0, 1.1, 1.2, \ldots, 2.5\}$) . We first report the convergence speed on the evaluation set and on three unseen ϵ in Table 3.

Table 3. Convergence steps on four-player games with different values of ϵ. We report the performance on three different ϵ, which controls the value of M_{ij} in the Eqn. 6.

Method	Best Evaluation Steps	Transfer Test Steps		
		$\epsilon = 0.6$	$\epsilon = 1$	$\epsilon = 2$
SGA	47.60	32.25	27.25	1000
ConOpt	53.10	32.00	29.50	1000
L2PG	**47.15**	**31.50**	**25.00**	**61.00**

We observe a similar trend in the four-player games, *i.e.*, the updates to players' parameters predicted by L2PG lead to the fastest convergence speed. It is also noteworthy that with fixed settings of λ and α, both SGA and ConOpt are not able to find solutions within 1000 steps when $\epsilon = 2$. Figure 1 visually demonstrates the relationship between the convergence speed of SGA and ConOpt and the values of ϵ, with the fixed hyperparameters respectively optimal on the evaluation set. Their convergence speeds are greatly decelerated as the value of ϵ increases, while L2PG exhibits more stable empirical convergence behaviors. We also provide the variance analysis in Table 11 to show that the performance gains are significant.

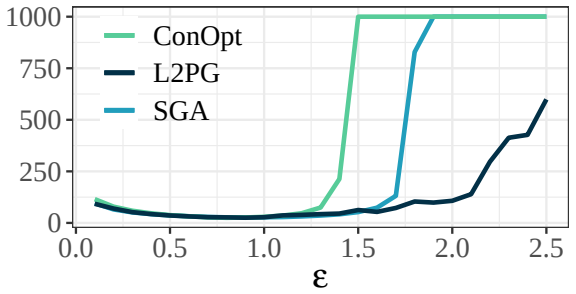

Figure 1. The relationship between algorithms' convergence steps and the values of ϵ. The results are truncated at 1000 steps for better visualization.

Performance on GANs. We follow the pipeline described in Letcher et al. (2019) and train L2PG along with the RMSprop optimizer to optimize a generative adversarial network. The GAN is optimized for 10000 steps. The full results are shown in Figure 6. While L2PG accelerates the

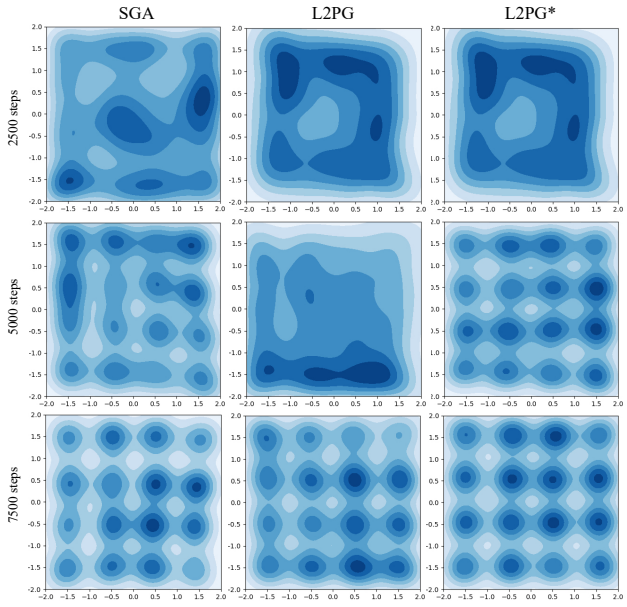

Figure 2. Generation results of models updated by SGA, L2PG, and L2PG*. Results at 2.5K, 5K and 7.5K steps are reported. L2PG* shows better performance, in terms of the distribution of generated data and convergence steps

training process at the early stage, it faces difficulties in optimizing GANs later, due to the notorious and unsolved problem of "longer horizons" in existing L2O work. To provide a remedy, we combine analytic gradients, which has also been applied in Vicol et al. (2021), with L2PG to this challenge: we first apply L2PG in the early stage of training (first 2K steps), and switch to use $\boldsymbol{g} + \mathbf{A}\boldsymbol{g}$ afterward. We call this hybrid pipeline "L2PG*". In Figure 2 we observe that L2PG* already shows higher convergence quality at 5K steps (the peaks are clearer compared to SGA), and shows a better pattern at 7.5K steps (the grids are neater compared to SGA). These results show the potential of applying L2PG to more complicated games. A full comparison between SGA, L2PG, and L2PG* can also be found in Section A.3.

4.3. Visualizations

To better understand the optimization behaviors of L2PG, we extract the players' parameters from the two-player quadratic games and visualize it in Figure 3. We study two quadratic games with distinct dynamics: (1) $l_1(x, y) = 0.492x^2 - 1.081xy - 0.048x, l_2(x, y) = 0.330y^2 + 0.001xy - 0.560y$; and (2) $l_1(x, y) = -0.584x^2 - 1.012xy + 0.826x, l_2(x, y) = -0.826y^2 - 1.184xy - 2.312y$. The former has a stable fixed point located at $(0.979, 0.846)$, while the latter has an unstable fixed point.

Generated Updates Coefficients. We first visualize the prediction of the three update coefficients from L2PG, *i.e.*, the values of $c_{\boldsymbol{g}}, c_{\mathbf{A}}$ and $c_{\mathbf{S}}$. We collect the predicted update coefficients from the aforementioned two games and present them in Figure 5. The figure demonstrates that: (1) On the games with a stable fixed point as the solution, the output coefficients of L2PG are substantially different. Specifically, $c_{\boldsymbol{g}}$ and $c_{\mathbf{A}}$ seems to be emphasized while $c_{\mathbf{S}}$ is close to 0. This in fact demonstrates that L2PG has learned certain update patterns; (2) For the games with an unstable fixed point as the solution, L2PG produces different patterns compared to those for the games with stable fixed points, which again demonstrates that L2PG is able to differentiate games by the fixed point's stability.

Trajectories of L2PG and SGA. To visually understand the update behaviors of L2PG and SGA, we draw the trajectories of players' parameters in the former game (*i.e.*, with a stable fixed point) updated by the two methods in Figure 3. As we can observe, while the two methods are approaching the same solution where $\boldsymbol{g} = \mathbf{0}$, L2PG updates the players' parameters significantly faster. In detail, L2PG takes merely two update steps to reach around $(0.75, 0.75)$, while SGA takes approximately 9 steps to approach the same region. In total, L2PG takes around 20 steps to converge, while SGA converges much slower (needs around 60 steps). We also provide more visualization in Section C, where we show the update behaviors on different games.

4.4. Ablation Studies

Curriculum Learning and Self-Learning We perform an ablation study to understand the practical benefits brought by curriculum learning (CL) and self-learning (SL) techniques. We meta-train the optimizers with different combinations of techniques (Data-CL, Training-CL, and SL) on quadratic games, and report the steps to converge on both the evaluation and testing set we previously used. Table 4 shows that: (1) Although a naively trained L2O can already achieve convergence in practice, the convergence speed is not satisfactory; (2) Both curriculum learning techniques, *i.e.*, Data-CL and Training-CL are useful, and combining them brings more performance gain; (3) Applying Self-learning solely can improve the convergence speed, and further combining the CL techniques is more beneficial. In conclusion, both the CL techniques and SL are vital to the learned optimizers with better quality.

Different Update Components One of the key designs of L2PG is using the three components ($\boldsymbol{g}, \mathbf{A}^\top\boldsymbol{g}$, and $\mathbf{S}\boldsymbol{g}$) to construct the input features to the optimizer, and calculate the actual updates to players' parameters by interpolating them with the predicted update coefficients $c_{\boldsymbol{g}}, c_{\mathbf{A}}$ and $c_{\mathbf{S}}$. To examine the critical roles of these components, we alternatively remove one or multiple components from both the inputs features and update coefficients, at both the meta-training and meta-testing stages.

The performance is shown in Table 5, where we can observe

Table 4. Performance on the evaluation and testing set of quadratic games. We report the convergence steps on the evaluation set and two testing sets aforementioned, one containing games with stable fixed points and another containing games with unstable fixed points.

CL		SL	Performance	
Data-CL	Training-CL		Evaluation	Testing
✗	✗	✗	402.575	358.763
✓	✗	✗	54.313	31.163
✗	✓	✗	91.113	79.025
✓	✓	✗	51.875	32.463
✗	✗	✓	153.938	248.525
✓	✓	✓	**41.025**	**26.375**

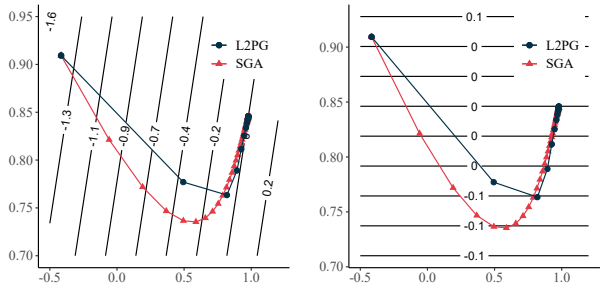

Figure 3. Trajectories of players' parameters updated by L2PG and SGA. We separately visualize the gradient of the two players' parameters in two plots. The contour lines in the background show the levels of gradients of each player.

that enabling all update components is crucial to the best performance. Combining all three components for constructing inputs will help capture richer information for training the L2PG, and using all of them to generate updates to players' will give the optimizer more flexibility and hence leads to better performance.

Different Regularization Form As aforementioned, the additive regularization form cannot avoid convergence to unstable fixed points if they are the only fixed points. To provide empirical support to our claim, we compare L2PG trained with the additive and the multiplicative regularization, and an additional baseline that does not use any regularization term for stability. Figure 4 shows the players' update trajectory on the game with $l_1(x, y) = -0.073x^2 + 0.409xy - 0.277x, l_2(x, y) = -0.074y^2 - 0.539xy + 1.007y$, which has an unstable fixed point located at $(-5.049, 5.015)$. We can observe that L2PG trained without regularization or with additive regularization form both show convergence to the unstable fixed point where $\boldsymbol{g}$ is close to 0. Although the additive regularization seems to help the players' parameters to diverge at the first few

Table 5. Ablation study on different update components. The optimizer are trained with different update components, and predict updates based on the same set of components.

Components			Performance	
$c_{\boldsymbol{g}}$	$c_{\mathbf{A}}$	$c_{\mathbf{S}}$	Evaluation	Testing
✓	✗	✗	67.975	64.813
✗	✓	✗	291.163	354.100
✗	✗	✓	189.988	372.263
✓	✓	✗	45.350	31.313
✓	✗	✓	52.838	35.688
✗	✓	✓	106.463	166.775
✓	✓	✓	**41.025**	**26.375**

iterations (similar to the first few iterations of using the multiplicative regularization), it eventually converges to the unstable point. Meanwhile, the multiplicative regularization is able to avoid convergence to the unstable fixed point and show clear divergent patterns.

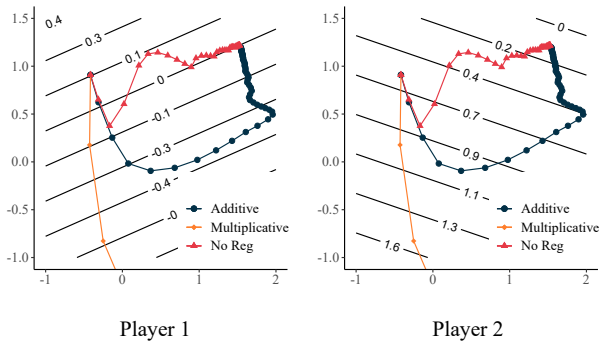

Figure 4. Comparison on the effects of different forms of regularization for avoiding unstable fixed points. We find that not using regularization ("No Reg") and using additive regularization ("Additive") may result in convergence to the unstable fixed point, while using the multiplicative form ("Multiplicative") leads to a clear divergent pattern from the unstable fixed point. Contour lines in the background show the levels of gradients of each player.

More Ablations We provide more ablation results in Section A.3. Specifically, we demonstrate the effectiveness of stability-awareness loss, decomposed update, and regularization techniques for training separately.

5. Conclusions

In this work, we formulate an L2O framework for solving general games that aim at finding stable fixed points for players' parameters. We propose a novel stability-aware sign-based loss for meta-training, which helps the optimizer to find stable fixed points and avoid unstable fixed points. To further overcome the training instability and improve the generalization ability, we propose two additional techniques,

data curriculum learning, and self-learning. Extensive experiments on quadratic games and GANs show that our proposed L2PG can find stable fixed points faster than baseline methods (SGA, ConOpt). We further justify by experiments that our proposed techniques are vital to fast convergence, and more importantly, our method can empirically avoid convergence to unstable fixed points.

Limitations. Our work has applied learned optimization rules to find stable fixed points in differentiable games. While focusing solely on empirical results, the theoretical understanding of our method is out of scope and is overall challenging for L2O-based methods. Therefore, we leave the theoretical guarantees for future works.

Acknowledgement

Z. Wang is in part supported by a J. P. Morgan AI Faculty Research Award in 2021, and an NSF CCSS grant (Award Number: 2113904).

References

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and De Freitas, N. Learning to learn by gradient descent by gradient descent. *Advances in neural information processing systems*, 29, 2016.

Balduzzi, D., Racaniere, S., Martens, J., Foerster, J., Tuyls, K., and Graepel, T. The mechanics of n-player differentiable games. In *International Conference on Machine Learning*, pp. 354–363. PMLR, 2018.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

Busoniu, L., Babuska, R., and De Schutter, B. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008.

Cao, Y., Chen, T., Wang, Z., and Shen, Y. Learning to optimize in swarms. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 15018–15028, 2019.

Chen, T., Zhang, W., Jingyang, Z., Chang, S., Liu, S., Amini, L., and Wang, Z. Training stronger baselines for learning to optimize. *Advances in Neural Information Processing Systems*, 33:7332–7343, 2020a.

Chen, T., Chen, X., Chen, W., Wang, Z., Heaton, H., Liu, J., and Yin, W. Learning to optimize: A primer and a benchmark. *The Journal of Machine Learning Research*, 23(1):8562–8620, 2022a.

Chen, W., Yu, Z., Wang, Z., and Anandkumar, A. Automated synthetic-to-real generalization. In *International Conference on Machine Learning (ICML)*, pp. 1746–1756, 2020b.

Chen, X., Chen, T., Cheng, Y., Chen, W., Awadallah, A., and Wang, Z. Scalable learning to optimize: A learned optimizer can train big models. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIII*, pp. 389–405. Springer, 2022b.

Chen, Y., Hoffman, M. W., Colmenarejo, S. G., Denil, M., Lillicrap, T. P., Botvinick, M., and De Freitas, N. Learning to learn without gradient descent by gradient descent. In *International Conference on Machine Learning (ICML)*, pp. 748–756, 2017.

Daskalakis, C., Ilyas, A., Syrgkanis, V., and Zeng, H. Training gans with optimism. *arXiv preprint arXiv:1711.00141*, 2017.

Foerster, J. N., Chen, R. Y., Al-Shedivat, M., Whiteson, S., Abbeel, P., and Mordatch, I. Learning with opponent-learning awareness. *arXiv preprint arXiv:1709.04326*, 2017.

Gidel, G., Berard, H., Vignoud, G., Vincent, P., and Lacoste-Julien, S. A variational inequality perspective on generative adversarial networks. *arXiv preprint arXiv:1802.10551*, 2018.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. *stat*, 1050:10, 2014.

Ibrahim, A., Azizian, W., Gidel, G., and Mitliagkas, I. Linear lower bounds and conditioning of differentiable games. In *International conference on machine learning*, pp. 4583–4593. PMLR, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Letcher, A., Foerster, J., Balduzzi, D., Rocktäschel, T., and Whiteson, S. Stable opponent shaping in differentiable games. *arXiv preprint arXiv:1811.08469*, 2018.

Letcher, A., Balduzzi, D., Racaniere, S., Martens, J., Foerster, J., Tuyls, K., and Graepel, T. Differentiable game mechanics. *The Journal of Machine Learning Research*, 20(1):3032–3071, 2019.

Li, K. and Malik, J. Learning to optimize. *arXiv preprint arXiv:1606.01885*, 2016.

Lv, K., Jiang, S., and Li, J. Learning gradient descent: Better generalization and longer horizons. In *International Conference on Machine Learning*, pp. 2247–2255. PMLR, 2017.

Mescheder, L., Nowozin, S., and Geiger, A. The numerics of gans. *Advances in neural information processing systems*, 30, 2017.

Metz, L., Poole, B., Pfau, D., and Sohl-Dickstein, J. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2016.

Nair, V. and Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

Pfau, D. and Vinyals, O. Connecting generative adversarial networks and actor-critic methods. *arXiv preprint arXiv:1610.01945*, 2016.

Ramponi, G. and Restelli, M. Newton optimization on helmholtz decomposition for continuous games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

Santurkar, S., Schmidt, L., and Madry, A. A classification-based study of covariate shift in gan distributions. In *International Conference on Machine Learning*, pp. 4480–4489. PMLR, 2018.

Schäfer, F. and Anandkumar, A. Competitive gradient descent. *Advances in Neural Information Processing Systems*, 32, 2019.

Shen, J., Chen, X., Heaton, H., Chen, T., Liu, J., Yin, W., and Wang, Z. Learning a minimax optimizer: A pilot study. In *International Conference on Learning Representations*, 2020.

Vadori, N., Savani, R., Spooner, T., and Ganesh, S. Consensus multiplicative weights update: Learning to learn using projector-based game signatures. *International Conference on Machine Learning*, 2022.

Vicol, P., Metz, L., and Sohl-Dickstein, J. Unbiased gradient estimation in unrolled computation graphs with persistent evolution strategies. In *International Conference on Machine Learning*, pp. 10553–10563. PMLR, 2021.

Wichrowska, O., Maheswaranathan, N., Hoffman, M. W., Colmenarejo, S. G., Denil, M., de Freitas, N., and Sohl-Dickstein, J. Learned optimizers that scale and generalize. In *International Conference on Machine Learning (ICML)*, 2017.

Yang, J., Chen, X., Chen, T., Wang, Z., and Liang, Y. M-l2o: Towards generalizable learning-to-optimize by test-time fast self-adaptation. In *The Eleventh International Conference on Learning Representations*, 2023.

You, Y., Chen, T., Wang, Z., and Shen, Y. L2-gcn: Layer-wise and learned efficient training of graph convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2127–2135, 2020.

Zheng, W., Chen, T., Hu, T.-K., and Wang, Z. Symbolic learning to optimize: Towards interpretability and scalability. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=ef0nInZHKIC.

A. Additional Experiments and Details

A.1. Algorithms

We provide a summary to L2PG's pipeline in Algorithm 1.

Algorithm 1 L2PG

Input: optimizer $\texttt{opt}(\cdot;\cdot)$, current training step t, a training set of games $\mathcal{D}_{\text{train}}$, a fixed evaluation set of games $\mathcal{D}_{\text{eval}}$, unroll length T, training steps N for each epoch, the evaluation frequency N_{E} and the number of epochs E.
Output: Optimal optimizer weights $\boldsymbol{\phi}$
Initialize $\boldsymbol{\phi} = \boldsymbol{\phi}_0$ and $\texttt{epoch} = 0$
for $\texttt{epoch} < E$ **do**
  Sample losses $\{l_i\}_{i=1}^n$ from $\mathcal{D}_{train}$
  Initialize players' parameters $\boldsymbol{w}_0$
  for $j = 0, T, 2T \ldots, ([N/T] - 1) \times T$ **do**
    Set $\mathcal{L} \leftarrow 0$
    for $k = 0, 1, 2 \ldots, T-1$ **do**
      $t := j + k$
      Calculate $\boldsymbol{o}^{(t)} = [\boldsymbol{g}; \mathbf{A}^\top \boldsymbol{g}; \mathbf{S}\boldsymbol{g}]$ at $\boldsymbol{w}^{(t)}$.
      Calculate $\boldsymbol{m}^{(t)}, \tilde{\boldsymbol{m}}^{(t)}, \tilde{\boldsymbol{o}}^{(t)}$ by Equation 2.
      Construct $\boldsymbol{z}^{(t)}$ from $\boldsymbol{o}^{(t)}, \boldsymbol{m}^{(t)}, \tilde{\boldsymbol{m}}^{(t)}, \tilde{\boldsymbol{o}}^{(t)}$.
      Update $\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} - \texttt{opt}(\boldsymbol{z}^{(t)}; \boldsymbol{\phi})$
      $\mathcal{L} \leftarrow \mathcal{L} + \mathcal{L}_t(\{l_i\}_{i=1}^n, \boldsymbol{w}^{(t+1)})$
    end for
    Update $\boldsymbol{\phi}$ for one step by minimizing $\mathcal{L}$.
  end for
  if $\texttt{epoch} \ \% \ N_{\text{E}} == 0$ **then**
    Evaluate $\texttt{opt}$ on $\mathcal{D}_{\text{eval}}$ and find the $\boldsymbol{\phi}$ with the highest evaluation performance.
  end if
end for

A.2. Sampled Game Coefficients

As aforementioned, we have sampled a fixed evaluation set and two testing sets of quadratic games. The coefficients of the 60 games are provided in the three files: "evaluation.txt", "test_stable.txt" and "test_unstable.txt". Each line in the file represents a game, containing 6 numbers that represent $M_{11}, M_{22}, M_{12}, M_{21}, b_1, b_2$, respectively.

A.3. Additional Experimental Results

Grid Search on the hyper-parameters of SGA and ConOpt We report the convergence steps of SGA and ConOpt on the evaluation set of games using different α and λ. Table 6 and Table 7 report the 20 settings of α and λ that yield the fastest convergence speed by using SGA and ConOpt, respectively. On each individual game, the optimal setting is not consistent: for example, on Game 1, SGA achieves the fastest convergence speed with $\alpha = 0.4$ and $\lambda = 1$, while on Game 2 SGA achieves the fastest convergence speed with $\alpha = 0.6$ and $\lambda = 0.1$. ConOpt shows a similar phenomenon and exhibits a larger variety of performance when applying the same hyperparameters to different games, and when applying different values of hyperparameters. More "1000" can be observed in the results, which necessitates careful hyperparameter selection.

Table 8 and Table 9 provide the convergence steps of the two algorithms on the testing set of games with stable fixed points. We can observe that the optimal choices of hyperparameters on the evaluation set are not achieving the fastest convergence speed on the testing samples.

Finally, we report the convergence steps of L2PG on these two sets of games in Table 10. L2PG achieves faster convergence in general, achieving a performance gain of more than 20 steps on the evaluation set of games and also performing better on unseen testing games.

Table 6. Convergence steps of SGA on games in the evaluation set.

α	λ	Game																				Mean
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
0.5	0.3	24.25	22.5	94.0	23.0	15.5	26.5	29.0	44.0	19.0	22.0	28.25	80.5	37.25	17.0	13.75	19.0	28.5	26.25	62.25	596.5	61.45
0.5	0.2	26.75	22.75	92.0	22.0	15.25	26.5	28.75	40.25	19.0	29.75	26.5	80.25	36.0	17.25	13.0	19.0	28.25	26.0	73.5	595.75	61.92
0.5	0.4	21.75	22.5	95.5	24.0	15.5	26.75	29.25	61.0	19.25	18.5	30.25	81.25	38.25	17.0	15.0	19.25	28.25	26.5	53.75	597.25	62.04
0.5	0.1	30.25	22.25	89.25	20.5	15.5	26.5	29.25	39.5	18.0	49.0	30.25	79.75	34.0	17.25	14.75	18.5	28.25	25.5	90.75	595.25	63.71
0.5	0.5	19.75	22.0	97.0	24.5	15.5	26.75	29.5	141.5	19.0	16.0	31.75	82.0	39.25	17.0	16.25	19.5	28.25	26.25	47.25	597.25	65.81
0.6	0.1	28.25	19.25	75.0	18.5	13.5	23.0	25.25	208.75	44.25	79.5	26.25	66.75	29.25	15.25	15.25	16.5	24.5	22.0	85.75	496.25	66.65
0.4	0.9	18.0	26.5	125.0	28.25	17.25	32.75	37.25	44.0	13.0	13.5	40.75	107.0	49.0	19.5	14.75	22.25	33.25	28.5	39.25	744.5	72.71
0.4	1.0	18.25	26.25	125.25	27.0	17.75	32.5	37.75	48.75	13.0	15.75	40.25	108.25	49.25	19.0	15.0	22.25	33.0	27.0	36.0	743.0	72.76
0.4	0.8	18.75	26.0	124.25	28.75	17.0	32.5	36.25	45.0	13.25	12.0	40.75	106.0	49.25	19.75	14.75	22.75	33.5	30.0	42.25	745.25	72.90
0.4	0.7	20.75	25.25	123.5	29.5	17.75	32.5	35.5	46.75	13.25	13.0	40.75	104.25	49.0	19.75	15.25	23.0	34.0	31.0	45.25	745.5	73.28
0.4	1.1	18.5	26.25	125.25	26.25	17.5	32.5	38.5	69.25	13.0	18.75	40.0	109.25	48.75	18.75	15.25	21.5	32.75	26.0	33.75	741.5	73.66
0.4	0.6	22.25	25.25	122.25	29.25	18.25	32.5	35.75	47.75	13.25	14.5	40.0	103.25	48.75	20.0	15.25	23.25	34.25	31.5	50.75	746.0	73.70
0.4	0.5	23.25	26.75	120.75	29.75	18.5	32.5	36.0	48.5	13.25	16.0	39.0	102.25	48.0	20.0	16.25	23.0	34.25	32.0	55.75	746.0	74.09
0.4	0.4	25.25	27.0	118.75	29.25	17.5	32.5	35.75	47.5	13.5	18.25	37.0	101.0	47.25	20.0	16.25	23.0	34.5	32.25	63.0	746.0	74.28
0.4	0.3	28.25	27.0	116.75	28.0	18.0	32.5	35.0	44.75	13.5	22.0	34.5	100.25	45.75	20.0	16.5	22.75	34.5	32.0	71.75	745.0	74.44
0.4	0.2	30.25	27.0	114.0	26.25	18.0	32.25	35.25	48.0	13.25	28.0	32.25	99.25	44.0	20.0	15.75	22.75	34.5	31.75	84.5	744.75	75.09
0.4	0.1	34.75	26.75	111.25	24.75	18.0	32.0	35.75	48.5	12.75	41.0	36.75	99.25	41.5	20.0	16.0	22.25	34.5	31.0	101.0	743.0	76.54
0.4	1.2	18.75	25.75	125.25	24.5	17.0	32.5	38.75	130.5	13.0	23.5	39.25	110.25	48.0	18.75	15.25	21.25	32.0	26.25	32.25	740.0	76.64
0.3	1.5	22.0	31.25	163.75	29.25	20.25	43.0	50.0	42.25	14.0	17.0	48.25	150.25	60.25	21.25	14.0	24.75	39.25	31.0	35.5	977.25	91.72
0.3	1.4	22.75	32.0	164.75	29.75	19.75	43.0	50.5	44.25	13.75	15.75	49.25	149.25	61.5	22.25	13.0	25.25	39.0	32.0	38.0	980.25	92.30

More Ablation Results In Table 11 we present the mean and the confidence intervals (CI) of the gain for 10 random initial players' parameters on four-player quadratic games. We run experiments with two different ϵ values. From the table we can observe that the gains are significantly larger than 0.

In Table 12 we show the ablation on the components we introduce, which verifies the importance of modules to expedite the convergence.

In Table 13 we provide the ablation studies on the choices of β_ϕ. We show that either too small ($0.1 \sim 0.9$) or too large β_ϕ (0.99) has inferior performance. In Table 14 we provide the ablation studies on the choices of K. We show that 5 is an appropriate choice as it leads to the highest performance among the values we have studied. Overall, the performance is not highly sensitive to the choice of hyperparameters.

We tried to use exponential moving average techniques (*i.e.,* soft update) to update the weights of the original optimizer without using SL. In this case, the convergence step is 28.625, which shows that EMA is efficient than our self-learning approach in expediting convergence.

Other GAN problems we trained a GAN network to learn a 75-dimensional uni-modal Gaussian. This challenge has also been explored in research by Santurkar et al. (Santurkar et al., 2018) and Balduzzi et al. (Balduzzi et al., 2018). The difficulty arises from boundary distortion, which is a type of covariate shift that causes the generator to struggle in accurately modeling the true data distribution. In Table 15, we present the mean absolute error (MAE) of the eigenvalues of the covariate matrix of data generated by a three-layer GAN used in Balduzzi et al. (2018), where we employed L2PG* and SGA to train the network, respectively. Our observations indicate that: (1) L2PG attains lower error in the initial stage; (2) By utilizing the analytical optimizer following L2PG, L2PG* ultimately outperforms SGA by a considerable margin. These findings demonstrate the effectiveness of our approach.

B. Additional Details on Methods

L2O Optimizer Structures. The structure of the learned optimizer in L2PG is a single-layer LSTM network with a hidden dimension of 32. The optimizer is being applied in a *coordinate-wise* way, *i.e.* the LSTM network will generate updates to every single position of the variables to update, which is a canonical way in L2O literature (Chen et al., 2022a). At the meta-training stage, the "hidden" and "cell" variables in the LSTM network are only reset after each epoch, and at the meta-testing stage, these two variables are never re-initialized.

Table 7. Convergence steps of ConOpt on games in the evaluation set.

α	λ	Game																				Mean
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
0.4	0.1	33.25	25.5	112.25	24.25	16.5	30.75	34.25	48.5	20.5	43.0	32.75	97.25	42.75	19.0	14.75	21.25	33.0	30.0	98.75	743.0	76.06
0.3	0.2	34.75	31.75	153.25	33.75	19.75	38.5	42.75	62.0	17.25	30.0	42.0	128.0	58.0	22.25	16.75	26.0	41.25	38.75	98.0	991.0	96.29
0.3	0.1	40.5	33.0	149.25	31.25	20.5	39.75	43.75	63.0	12.75	41.0	41.25	129.25	55.75	23.5	17.0	27.0	43.0	39.0	118.0	990.25	97.94
0.2	0.5	33.75	40.75	239.5	49.5	23.25	50.25	58.0	98.25	20.0	21.75	70.25	191.25	87.0	25.75	20.25	32.75	54.0	52.25	93.5	1000.0	113.10
0.2	0.4	37.25	42.5	237.0	49.75	23.5	52.0	61.25	85.25	15.25	25.0	69.5	191.25	87.25	27.25	21.5	34.25	56.0	54.0	102.0	1000.0	113.59
0.2	0.3	43.5	44.5	233.75	49.75	25.25	53.75	62.0	85.0	13.0	29.5	67.25	191.25	86.5	29.0	22.5	35.5	58.5	55.25	119.0	1000.0	115.24
0.2	0.2	48.5	46.0	229.0	49.0	27.25	56.0	62.5	91.25	14.25	36.5	62.25	192.25	85.25	31.5	23.25	37.0	60.75	56.75	138.5	1000.0	117.39
0.2	0.1	55.25	47.75	222.75	45.5	28.75	58.25	64.5	93.25	15.5	48.5	58.0	193.25	81.75	33.5	23.25	38.25	63.25	57.5	165.25	1000.0	119.70
0.3	0.3	31.0	30.75	156.25	34.25	18.5	37.0	42.5	1000.0	35.5	24.5	45.5	127.75	59.0	20.5	16.5	25.0	40.0	38.0	83.25	991.25	142.85
0.4	0.2	28.25	25.0	115.25	26.0	16.25	29.75	33.0	1000.0	648.25	30.0	32.25	96.25	44.25	17.75	13.25	20.75	32.0	29.75	80.75	743.75	153.12
0.6	0.2	23.5	22.0	77.5	19.0	23.0	21.0	23.0	1000.0	1000.0	63.5	22.5	64.5	30.5	54.75	41.75	24.5	22.75	21.25	66.25	496.25	155.88
0.1	1.2	39.5	57.5	482.0	81.0	32.5	78.75	110.75	161.5	16.75	21.25	129.75	378.5	159.75	33.75	26.5	47.0	83.25	81.5	103.25	1000.0	156.24
0.6	0.1	28.0	18.75	75.25	17.25	15.0	22.0	24.0	1000.0	1000.0	135.25	24.25	65.5	29.75	20.0	16.0	16.0	23.5	21.5	84.25	496.25	156.62
0.2	0.6	31.75	39.0	241.0	48.25	22.75	48.5	57.0	1000.0	33.25	19.5	70.5	191.5	87.0	24.25	19.25	31.5	52.0	50.5	82.75	1000.0	157.51
0.1	1.1	41.75	60.0	483.5	83.0	33.5	81.25	112.0	164.0	14.75	22.25	132.0	379.5	162.0	35.25	27.5	48.75	86.25	83.75	106.75	1000.0	157.89
0.5	0.2	25.25	20.75	92.75	22.0	15.25	24.5	26.75	1000.0	1000.0	36.75	26.5	77.5	36.0	19.25	19.0	17.5	26.25	25.0	70.75	595.5	158.86
0.1	1.0	43.75	63.0	484.5	85.75	35.25	83.25	112.75	166.25	14.0	23.5	134.25	380.5	164.25	37.0	28.5	50.75	89.0	86.75	112.25	1000.0	159.76
0.5	0.1	29.25	21.75	90.0	20.25	14.5	25.5	27.75	1000.0	1000.0	57.0	27.75	78.5	35.0	16.25	14.5	18.0	27.25	25.0	88.75	594.75	160.59
0.1	1.3	38.0	54.5	480.25	78.75	31.5	77.0	109.0	291.5	19.0	20.5	127.5	377.0	157.5	32.5	25.5	46.0	81.25	78.75	100.0	1000.0	161.30
0.1	0.9	46.25	65.5	484.75	88.0	36.75	85.75	113.5	168.75	13.5	25.25	136.0	381.5	166.25	38.75	30.5	52.75	91.75	89.5	126.25	1000.0	162.06

C. Additional Visualization

We visualize the values of $\boldsymbol{c_g}, \boldsymbol{c_A}$ and $\boldsymbol{c_S}$ predicted by L2PG in Figure 5. L2PG exhibits different behaviors on different types of games. In Figure 6 we visualize the mixed Gaussian distribution generated by SGA, L2PG and L2PG*. Overall,

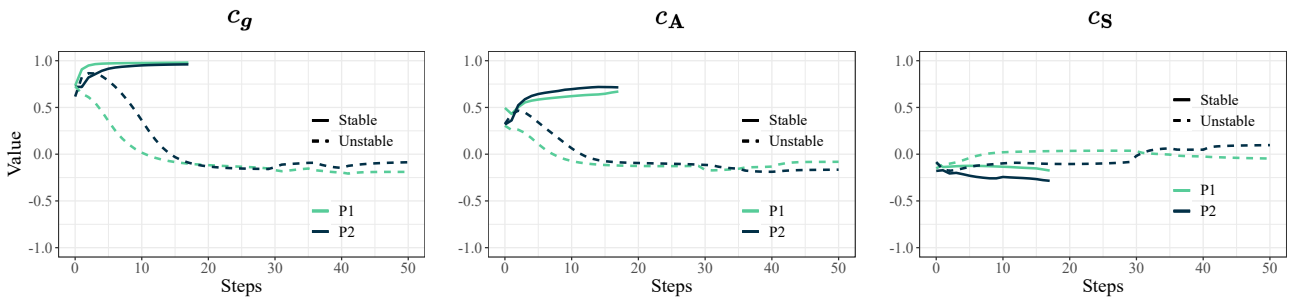

Figure 5. The prediction of L2PG on the two quadratic games (with stable and unstable fixed points). The update coefficients, *i.e.*, $c_{\boldsymbol{g}}, c_{\mathbf{A}}$ and $c_{\mathbf{S}}$ are shown in the three figures. The figures are truncated at 50 steps for better visualization quality. The stable fixed games are converged at 19 steps.

L2PG* produces the best performance.

Table 8. Convergence steps of SGA on games in the testing set of stable games.

α	λ	Game																				Mean
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
0.7	0.4	21.5	20.75	92.75	31.0	39.5	13.75	58.25	58.5	23.25	19.25	16.5	16.0	15.25	37.5	29.25	22.25	21.0	17.5	20.0	15.0	29.44
0.7	0.3	20.75	20.25	92.75	36.0	39.5	11.75	57.5	58.5	23.25	13.5	15.5	17.75	14.75	49.75	26.25	22.25	20.25	16.75	20.0	14.75	29.59
0.7	0.5	22.25	21.75	92.75	33.75	39.5	18.5	58.5	58.5	23.25	34.0	17.0	14.75	15.5	30.75	32.25	22.5	21.75	18.0	20.0	15.75	30.55
0.6	0.5	24.75	23.25	108.0	19.0	45.5	15.0	68.5	68.0	26.5	18.5	19.25	15.0	17.25	30.75	17.75	25.5	24.25	17.75	22.25	14.75	31.07
0.6	0.4	24.25	23.25	108.0	20.0	45.5	14.0	67.5	68.0	26.5	15.5	18.75	16.0	17.0	37.5	17.25	25.5	23.75	18.0	22.25	14.5	31.15
0.6	0.6	25.25	23.0	108.0	20.5	45.5	17.5	68.75	67.75	26.5	26.25	19.5	13.75	17.75	26.75	18.5	25.75	24.25	17.25	22.25	14.75	31.48
0.6	0.3	23.5	22.75	107.75	23.0	45.5	13.25	66.5	67.75	26.5	15.25	17.5	18.0	16.5	47.25	16.5	25.25	22.75	18.25	22.25	14.25	31.51
0.7	0.2	19.75	19.5	92.5	58.25	39.5	15.0	56.5	58.25	23.25	14.5	15.5	20.25	15.5	74.0	23.5	21.75	19.0	16.75	20.0	14.0	31.86
0.6	0.2	22.25	22.25	107.75	32.0	45.5	15.0	65.5	67.75	26.25	15.25	16.5	20.25	17.0	65.5	16.0	24.75	21.75	18.25	22.25	14.5	32.81
0.6	0.7	25.25	22.75	107.75	60.0	45.5	26.0	69.5	67.5	26.5	53.75	19.25	13.0	18.0	23.75	18.75	25.75	24.25	17.25	22.25	14.75	35.08
0.5	0.6	29.5	27.0	129.0	14.75	54.25	16.0	82.5	81.0	31.25	17.25	22.25	15.0	20.25	28.75	14.5	30.25	28.75	19.75	26.0	16.75	35.24
0.5	0.7	29.75	26.75	129.0	20.0	54.25	16.0	83.25	80.5	31.25	19.5	22.0	14.5	20.5	25.5	14.5	30.25	28.5	19.5	26.0	16.75	35.41
0.5	0.5	29.0	27.25	129.0	15.0	54.25	16.5	81.5	81.25	31.25	16.75	22.25	16.0	20.0	33.0	14.5	30.25	28.25	20.25	26.0	16.5	35.44
0.5	0.4	28.5	27.25	129.0	16.25	54.25	16.0	80.5	81.25	31.25	17.25	21.5	17.75	19.5	38.75	14.5	29.5	27.75	20.5	26.0	16.0	35.66
0.5	0.3	27.25	26.75	129.0	19.0	54.25	15.0	79.5	81.0	31.0	17.25	20.0	19.0	19.0	48.0	14.75	29.5	26.5	20.5	26.0	16.0	35.96
0.5	0.8	29.75	25.75	129.0	40.75	54.0	18.0	83.75	79.75	31.25	25.5	21.75	14.25	21.0	23.0	14.5	30.25	28.0	18.75	26.0	16.25	36.56
0.5	0.2	26.0	25.75	129.0	24.75	54.25	16.25	78.5	80.5	30.5	16.25	18.25	21.75	19.25	63.25	14.75	29.25	25.0	20.75	26.0	16.25	36.81
0.6	0.1	20.0	21.0	107.75	65.25	45.5	18.75	64.5	66.75	26.25	18.75	18.75	23.75	18.25	109.5	15.0	24.5	22.75	18.25	22.25	14.25	37.09
0.8	0.3	18.75	22.0	81.5	139.75	35.0	12.25	50.5	51.75	20.75	16.25	14.25	18.75	13.75	55.5	110.75	20.0	18.0	24.25	18.0	18.25	38.00
0.5	0.1	23.5	24.5	128.75	39.0	54.0	19.5	77.5	80.0	30.5	19.75	21.0	24.5	20.25	94.75	14.5	28.5	26.25	21.0	26.0	16.0	39.49

Table 9. Convergence steps of ConOpt on games in the testing set of stable games.

α	λ	Game																				Mean
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
0.5	0.2	25.75	25.25	126.0	34.0	52.25	13.75	77.25	79.5	29.0	16.25	18.75	20.75	17.0	65.0	29.75	27.25	25.25	19.25	23.5	14.5	37.00
0.5	0.1	24.25	24.5	127.25	47.5	53.0	17.75	76.5	79.5	29.5	17.5	19.0	23.75	18.5	96.0	16.0	27.5	23.75	20.0	25.0	15.25	39.10
0.6	0.1	21.0	21.0	106.5	147.5	44.5	17.0	64.5	66.75	25.5	16.25	17.0	23.75	17.0	114.25	27.25	23.5	20.5	17.5	21.0	14.0	41.31
0.4	0.3	32.25	30.25	155.75	21.0	62.75	17.75	96.25	98.0	34.75	19.75	24.5	19.75	19.25	52.25	22.75	32.25	32.0	21.5	27.25	15.75	41.79
0.4	0.2	31.25	30.5	157.25	25.25	64.25	16.0	96.25	98.75	35.5	19.0	22.25	22.75	20.0	66.5	15.5	33.0	30.75	22.5	28.5	17.0	42.64
0.4	0.4	32.0	30.25	154.25	19.25	61.75	18.25	96.25	97.25	33.75	19.75	25.25	18.0	19.0	43.5	61.0	31.25	32.0	21.0	26.0	15.75	42.77
0.4	0.1	29.75	29.75	158.75	34.0	65.5	19.0	95.5	98.75	36.0	19.75	22.25	25.75	21.75	92.75	15.75	33.75	28.75	23.5	30.0	17.75	44.94
0.3	0.6	40.75	37.5	201.25	14.75	78.75	22.5	127.25	126.5	41.75	22.75	32.5	18.0	22.75	38.5	34.25	38.25	41.5	24.25	31.0	17.25	50.60
0.3	0.5	41.5	38.5	203.25	15.75	80.0	22.75	127.75	128.0	42.75	24.0	32.5	19.75	23.75	43.75	20.75	39.5	41.75	25.25	32.25	18.0	51.08
0.3	0.4	41.5	39.25	205.25	17.75	81.75	22.75	128.0	129.25	43.75	24.5	32.25	21.25	24.0	51.0	16.25	40.75	41.75	26.5	33.5	17.75	51.94
0.3	0.3	41.5	39.75	207.25	19.75	83.0	22.0	128.0	130.25	45.0	24.75	31.25	23.5	24.5	61.0	16.75	41.75	41.5	27.5	35.25	19.5	53.19
0.3	0.2	40.5	39.5	209.25	24.0	84.75	19.75	127.75	130.75	46.25	23.75	28.75	26.75	25.25	75.75	18.0	42.75	39.75	28.75	37.0	21.0	54.50
0.3	0.7	40.25	36.75	199.25	14.25	77.25	22.0	126.25	125.5	40.75	24.5	32.0	16.75	22.5	34.75	151.5	37.25	40.75	23.5	29.75	17.5	55.65
0.3	0.1	38.0	38.5	211.25	32.0	86.25	23.0	127.0	131.0	47.5	24.0	27.25	30.75	27.25	100.75	19.25	44.0	37.0	30.5	39.0	22.0	56.81
0.2	1.1	54.75	48.25	286.25	12.75	106.75	27.5	184.75	178.75	53.75	27.75	43.0	18.75	28.25	34.0	36.0	48.25	54.75	28.25	37.25	20.25	66.50
0.2	1.0	56.0	49.5	289.25	12.5	108.5	28.5	185.75	180.75	55.0	27.5	43.75	19.25	29.0	36.75	24.25	49.5	56.0	29.25	38.5	20.5	67.00
0.2	1.2	53.5	46.75	283.5	14.75	105.0	26.75	182.75	176.5	52.5	30.0	42.0	18.0	27.75	31.75	76.0	47.0	53.5	27.25	36.0	19.5	67.54
0.2	0.9	57.0	51.0	292.0	12.5	110.5	29.5	186.75	182.75	56.5	27.25	44.75	20.0	30.0	39.5	19.25	51.0	57.25	30.25	39.75	21.25	67.94
0.2	0.8	58.0	52.5	295.0	13.75	112.5	30.25	188.0	184.75	58.0	27.0	45.75	20.75	31.25	42.75	17.75	52.75	58.5	31.25	41.5	22.25	69.21
0.2	0.7	59.0	53.5	298.0	14.75	114.5	30.75	189.75	187.0	59.25	29.5	46.5	22.0	32.25	47.0	18.5	54.25	59.75	33.0	43.25	23.0	70.78

Table 10. Convergence steps of L2PG on games in the evaluation set and the testing set of stable games.

Dataset	Game																				Mean
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
Evaluation	22.00	16.75	50.25	17.25	14.25	17.75	20.00	23.75	16.00	24.50	24.50	49.50	23.50	15.00	13.00	15.25	19.50	19.25	58.50	360.00	41.03
Testing (Stable)	18.25	19.25	74.25	19.50	33.00	14.75	49.00	46.25	21.75	15.25	17.50	19.00	16.75	53.75	15.00	20.25	22.25	17.75	19.00	15.00	26.38

Table 11. Mean and 90% CIs of gains in convergence steps on four-player quadratic games. The experiments are repeated 10 times with different initial players' parameters.

Method	Mean (CI) of performance gap	
	$\epsilon = 0.6$	$\epsilon = 1$
L2PG v.s. SGA	0.70±0.47	2.20±1.31
L2PG v.s. ConOpt	0.50±0.30	4.90±1.51

Table 12. Ablation studies on different categories of techniques (Update decomposition, stability-aware training loss, and training techniques). The best performance is marked in **bold**.

Components			Testing	
Update Decomposition	Stability-Aware Loss	CL & SL	Stable	Unstable
✗	✗	✗	220.375	195.500
✓	✗	✗	140.250	158.213
✗	✓	✗	1000	**1000**
✗	✗	✓	69.788	66.900
✓	✓	✓	**26.375**	**951.400**

Table 13. Ablation studies on β_ϕ on two-player quadratic games.

β_ϕ	Testing Performance
0.1	28.448
0.5	26.512
0.9	26.825
0.95	26.375
0.99	27.133

Table 14. Ablation studies on K on two-player quadratic games.

K	Testing Performance
2	29.118
5	26.375
10	27.122

Table 15. Mean absolute error of the eigenvalues of the covariate matrix of data generated by a three-layer GAN.

Steps	SGA	L2PG*
500	0.7049	0.7472
1000	0.7197	0.6268
1500	0.6583	0.6318
5000	0.4705	0.7094
6000	0.4326	0.6069
7000	0.4009	0.5006
8000	0.3678	0.3991
9000	0.3500	0.2956
10000	0.3177	0.2166

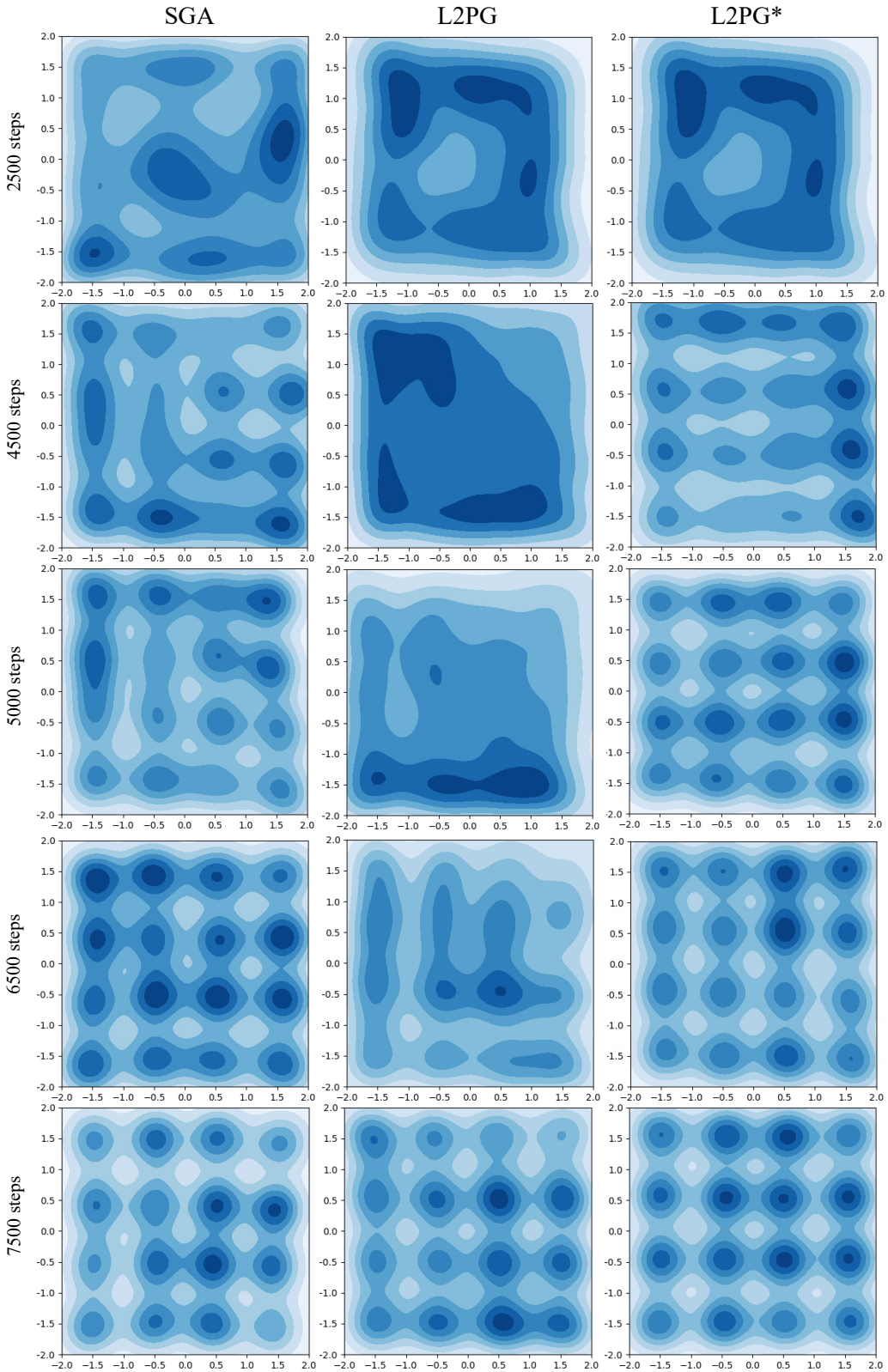

Figure 6. Generation results of models updated by SGA, L2PG and L2PG*. Results at $\{2.5, 4.5, 5, 6.5, 7.5\}$K steps are reported. L2PG* shows better quality after optimizing for 4.5K steps.