Training Generative Adversarial Networks with Adaptive Composite Gradient

Huiqing Qi, Fang Li, Shengli Tan, Xiangyun Zhang†

School of Mathematical Sciences, East China Normal University, Shanghai 200241, China

Keywords: Generative adversarial networks; adaptive composite gradient; semi-gradient free; game theory; bilinear game

Citation: Qi H.Q., Li F., Tan S.L., et al.: Training Generative Adversarial Networks with Adaptive Composite Gradient. Data Intelligence 6(1), 120–157 (2024). doi: https://doi.org/10.1162/dint_a_00246
Submitted: June 10, 2023; Revised: November 15, 2023; Accepted: December 10, 2023

ABSTRACT

The wide applications of Generative adversarial networks benefit from the successful training methods, guaranteeing that an object function converges to the local minimum. Nevertheless, designing an efficient and competitive training method is still a challenging task due to the cyclic behaviors of some gradient-based ways and the expensive computational cost of acquiring the Hessian matrix. To address this problem, we proposed the Adaptive Composite Gradients (ACG) method, linearly convergent in bilinear games under suitable settings. Theory analysis and toy-function experiments both suggest that our approach alleviates the cyclic behaviors and converges faster than recently proposed SOTA algorithms. The convergence speed of the ACG is improved by 33% than other methods. Our ACG method is a novel Semi-Gradient-Free algorithm that can reduce the computational cost of gradient and Hessian by utilizing the predictive information in future iterations. The mixture of Gaussians experiments and real-world digital image generative experiments show that our ACG method outperforms several existing technologies, illustrating the superiority and efficacy of our method.

1. INTRODUCTION

Gradient descent-based machine learning and deep learning methods have been widely used in various computer science tasks over the past several decades. Optimizing a single objective problem with gradient descent is easy to converge to a saddle point in some cases [1]. However, there is a
Training Generative Adversarial Networks with Adaptive Composite Gradient

growing set of multi-objective problems that need to be optimized in numerous fields, such as deep reinforcement learning [2, 3], Game Theory, Machine Learning and Deep Learning. Generative Adversarial Networks [4] is a kind of classical multi-objective problem in Deep Learning. GANs have a wide range of applications [5] because of their capability, which can learn to generate complex and high dimensional target distribution. The existing literature about GANs can be divided into four categories, including music generation [6, 7, 8], natural languages [9, 10, 11, 12], methods of training GANs [13, 14, 15, 16], images processing [17, 18, 19, 20]. GANs have obtained remarkable progress in image processing, such as video generation [21, 22], noise removal [23], deblur [24], image to image translation [25, 26], image super-resolution [17], medical image processing [27].

Generative adversarial networks’ framework consists of two deep neural networks: generator network and discriminator network correspondingly. The generator network is given a noise sample from a simple known distribution as input, and it can produce a fake sample as output. The generator learns to make such fake samples, not by directly using real data, just by adversarial training with a discriminator network. Bilinear games are two-player, non-cooperative zero-sum games with compact polytopal strategy sets. If the generator and discriminator have no information communication, then training GANs is a non-cooperative zero-sum game. Therefore, GANs can be considered a bilinear game under suitable scenarios. The object function of GANs [4] is often formulated as a two-player min-max game with a Nash equilibrium at the saddle points:

\[ \min_{C} \max_{D} V(D, G) = \mathbb{E}_{x \sim P_{x}(x)}[\log D(x)] + \mathbb{E}_{z \sim P_{z}(z)}[\log(1 - D(G(z)))] \]  

Where \( x \sim P_{x}(x) \) denotes an actual data sample and \( z \sim P_{z}(z) \) denotes a sample from a noise distribution (often using uniform distribution or Gaussian distribution). More different forms of GANs object function are mentioned in [28]. Though GANs have achieved remarkable applications, training stable and fast GANs [29, 30] still is a challenging task. Since it suffers from the strongly associated gradient vector field rotating around a Nash equilibrium (see Figure 1). Moreover, those gradient descent ascent-based methods used to optimize object function of GANs tend to lead the limit oscillatory behavior because of imaginary components in the Jacobian eigenvalues.

The main idea of this work is to reduce the computing cost of the Hessian matrix in consensus optimization and SGA. Motivated by [15, 16] and [31], we propose a novel Adaptive Composite Gradient method, which can be used to calibrate and accelerate the traditional methods such as SGD, RMSProp, Adam, consensus optimization, and SGA. The ACG method exploits three aspects of information in the iteration process, which includes gradients information of the past iteration steps, adaptive and predictive information for future iteration steps, and the projection information of the current iteration step mentioned in [16]. We fuse this information as the composite gradient to update the scheme in our algorithm, which can be deployed in deep networks and used to train GANs. The main contributions of this paper are as follows:

- We propose a novel adaptive composite gradient (ACG) method, which can alleviate cyclic behaviors in training GANs. Meanwhile, ACG can reduce the computational consumption of gradients and improve convergence speed.
• For purely adversarial bilinear game problems, we prove that the ACG method is linearly convergent under suitable conditions. In addition, we extend the ACG method to three-player game problems and verify its effect and efficiency with toy models.

• The comprehensive experiments are conducted to test the effect of training GANs and Deep Convolutional Generative Adversarial Networks. The proposed method can obtain competitive results over state-of-the-art (SoTA) methods.

2. RELATED WORK

There are several distinctive approaches to improve the training of GANs, but they show more or fewer limitations in some cases. Some of these are dependent closely on the previous assumptions, which leads to these methods not being valid. Moreover, some of these need to payoff the computing cost of the Hessian in the dynamics. Next, we will discuss some related researches in this section.

Symplectic Gradient Adjustment (SGA) [32]: Compared with the traditional games, do not constrain the players’ parameter sets or require the loss functions to be convex. The general games can be decomposed into a potential game, and a Hamiltonian game in [32]. To introduce our method, we firstly review the SGA method as follows.

Definition 2.1 A game is a set of players \( p = \{1, 2, \ldots, n\} \), and the loss functions satisfy twice continuously differentiable \( \{\ell_j : \mathbb{R}^d \rightarrow \mathbb{R}\}_{i=1}^n \). Players’ parameters are \( w = (w_1, w_2, \ldots, w_n) \in \mathbb{R}^d \) with \( w_i \in \mathbb{R}^{d_i} \) where \( \sum_{i=1}^n d_i = d \). The \( j^{th} \) player can control \( w_j \).

Using the \( g(w) \) notes the simultaneous gradient which is the gradient of the losses withe respect to players’ parameters \( g(w) = (\nabla_{w_1} \ell_1, \nabla_{w_2} \ell_2, \ldots, \nabla_{w_n} \ell_n) \). For a bilinear game, it requires the losses to satisfy \( \sum_{j=1}^n \ell_j = 0 \) such as follow:
\( \ell_1(x, y) = x^T C_y \quad \text{and} \quad \ell_2(x, y) = -x^T C_y. \) \hspace{1cm} (2)

This kind of games have a Nash equilibrium at \((x,y) = (0,0)\). The simultaneous gradient \( g(x,y) = (C_y, -C^T x) \) rotates around the Nash equilibrium shown in Fig. 6.

We can derive the Hessian of a \( n \)-player game with the simultaneous gradient \( g(w) \). The formula of Hessian is \( H(w) = \nabla_w : g(w)^T = \left( \frac{\partial g_i(w)}{\partial w_j} \right)_{i,j=1}^d \), where \( H \in \mathbb{R}^{d \times d} \). Further, the matrix formula of Hessian is as follows:

\[
H(w) = 
\begin{bmatrix}
\nabla_{w_1}^2 \ell_1 & \nabla_{w_1}^2 \ell_2 & \cdots & \nabla_{w_1}^2 \ell_n \\
\nabla_{w_2}^2 \ell_1 & \nabla_{w_2}^2 \ell_2 & \cdots & \nabla_{w_2}^2 \ell_n \\
\vdots & \vdots & \ddots & \vdots \\
\nabla_{w_n}^2 \ell_1 & \nabla_{w_n}^2 \ell_2 & \cdots & \nabla_{w_n}^2 \ell_n
\end{bmatrix}.
\] \hspace{1cm} (3)

Applying the generalized Helmholtz decomposition \( \text{Lemma 1 in [32]} \) to the above mentioned Hessian of the game, we have \( H(w) = S(w) + A(w) \). David et al. (2018) [32] pointed that a game is a potential game if \( A(w) = 0 \). It is a Hamiltonian game if \( S(w) = 0 \). Potential games and Hamiltonian games are both well-studied, and they are easy to solve. Since the cyclic behavior around the Nash equilibrium is caused by simultaneous gradient, David et al. [32] proposed the Symplectic Gradient Adjustment method, which is as follows:

\[
g_{\lambda} := g + \lambda \cdot A^T g. \] \hspace{1cm} (4)

Where \( A \) is from the Helmholtz decomposition of Hessian. \( g_{\lambda} \) is used to replace the gradient among the iterates in the GAD-based methods, and using \( g_{\lambda} \) to train GANs can alleviate the cyclic behaviors. If we consider the players in a bilinear game as GANs, the SGA algorithm needs to pay an expensive computing cost of Hessian which can lower the algorithm efficiency.

**Centripetal Acceleration [15]:** The simultaneous gradient shows cyclic behaviors around the Nash. Hamiltonian games obey a conservation law in these gradient descent-based methods, so the cyclic behaviors can be considered as a uniform circular motion process. As well known, the direction of the centripetal acceleration in a consistent circular motion process points to the center of the circle. Using this characteristic modifies the direction of the simultaneous gradient vector field to alleviate the cyclic behaviors. Based on these observations, Peng et al. (2020) [15] propose the Centripetal acceleration methods, which are derived to two versions methods named Simultaneous Centripetal Acceleration (Grad-SCA) and Alternating Centripetal Acceleration (Grad-ACA), they are used to train GANs. In the next, we review the Centripetal Acceleration methods.

Given a bilinear game, the losses of this game are \( \ell_1(\theta, \phi), \ell_2(\theta, \phi) \) corresponding to player 1 and player 2. The parameter space is defined in \( \theta \times \phi \), where \( \theta, \phi \in \mathbb{R}^n \). Player 1 can control the parameter \( \theta \) and tries to minimize payoff function \( \ell_1 \), while player 2 can control parameter \( \phi \) and tries to minimize payoff function \( \ell_2 \) under the non-cooperative situation. This game is a process of the two players adjusting their
parameters to find a local Nash equilibrium that satisfies the following two requirements:

$$\theta' \in \text{argmin}_{\theta} \ell_1(\theta, \phi') \quad \text{and} \quad \phi' \in \text{argmin}_{\phi} \ell_2(\theta', \phi).$$  \hspace{1cm} (5)$$

The centripetal acceleration methods require that the two-player game is differentiable. Then, the above two payoff functions can be combined into a joint payoff function because of the zero-sum property of the game:

$$\langle \theta', \phi' \rangle \in \min_{\theta} \max_{\phi} V(\theta, \phi).$$  \hspace{1cm} (6)$$

The derivation of Eq. (1) leads a two-player game, which can be rewrite as $V(\theta, \phi)$. The problem becomes to finds a local Nash equilibrium:

$$\theta' \in \text{argmin}_{\theta} V(\theta, \phi') \quad \text{and} \quad \phi' \in \text{argmin}_{\phi} -V(\theta', \phi),$$  \hspace{1cm} (7)$$

where

$$V(\theta, \phi) = \mathbb{E}_{x \sim P(x)} [\log D(x ; \phi)] + \mathbb{E}_{z \sim P_z(z)} [\log(1 - D(G(z ; \theta); \phi))].$$  \hspace{1cm} (8)$$

In order to introduce the Centripetal Acceleration methods, we first review the simultaneous gradient descent method in [33] is

$$\theta_{t+1} = \theta_t - \alpha \nabla_{\theta} V(\theta_t, \phi), \quad \phi_{t+1} = \phi_t + \alpha \nabla_{\phi} V(\theta_t, \phi).$$  \hspace{1cm} (9)$$

And the simultaneous gradient descent based alternating version is

$$\theta_{t+1} = \theta_t - \alpha \nabla_{\theta} V(\theta_t, \phi), \quad \phi_{t+1} = \phi_t + \alpha \nabla_{\phi} V(\theta_{t+1}, \phi),$$  \hspace{1cm} (10)$$

where $\alpha$ is learning rate in the algorithms. While the Centripetal Acceleration methods directly utilize the item of centripetal acceleration to adjust the simultaneous gradient descent. Then gradient descent with simultaneous centripetal acceleration is introduced as:

$$\theta_{t+1} = \theta_t - \alpha \nabla_{\theta} V(\theta_t, \phi) - \beta_1 [\nabla_{\theta} V(\theta_t, \phi) - \nabla_{\theta} V(\theta_{t-1}, \phi_{t-1})]$$

$$\phi_{t+1} = \phi_t + \alpha \nabla_{\phi} V(\theta_t, \phi) + \beta_2 [\nabla_{\phi} V(\theta_t, \phi) - \nabla_{\phi} V(\theta_{t-1}, \phi_{t-1})].$$  \hspace{1cm} (11)$$

We can also obtain the gradient descent with the alternating centripetal acceleration method:

$$\theta_{t+1} = \theta_t - \alpha_1 \nabla_{\theta} V(\theta_t, \phi) - \beta_1 [\nabla_{\theta} V(\theta_t, \phi) - \nabla_{\theta} V(\theta_{t-1}, \phi_{t-1})]$$

$$\phi_{t+1} = \phi_t + \alpha_2 \nabla_{\phi} V(\theta_{t+1}, \phi) + \beta_2 [\nabla_{\phi} V(\theta_{t+1}, \phi) - \nabla_{\phi} V(\theta_{t+1-1}, \phi_{t-1})],$$  \hspace{1cm} (12)$$

where $\alpha_1, \beta_1, \alpha_2, \beta_2$ in above two versions of the centripetal acceleration methods are hyper parameters. The centripetal acceleration methods can calibrate other gradient-based methods. The intuitive understanding of the centripetal acceleration method is shown in Figure 2.

**Predictive Projection Centripetal Acceleration (PPCA) [16]:** For the centripetal acceleration methods, it uses the last iterative step information to update $\langle \theta_{t+1}, \phi_{t+1} \rangle$. However, there are some methods which utilize the predictive step information to update $\langle \theta_{t+1}, \phi_{t+1} \rangle$, such as MPM, OMD, OGDA. MPM is introduced by Liang et al. (2019) [34] and its dynamics are as follows:
Motivated by MPM and centripetal acceleration methods, Li et al. (2020) [16] propose the predictive projection centripetal acceleration methods. They also consider approximately the cyclic behavior around a Nash as a uniform circular motion process. However, it is different from the Grad-SCA and Grad-ACA. They construct the item of centripetal acceleration to use the predictive step information replacing that of the last step. Meanwhile, they argue that the centripetal acceleration term points to the matched center approximately. To make the centripetal acceleration item point to the center precisely, they proposed the projection centripetal acceleration methods. PPCA can modify the gradient descent ascent and the alternating gradient descent ascent directly. We can understand PPCA intuitively from Figure 2. The dynamics of predictive projection centripetal acceleration are the following formula:

**predictive step**: \[ \theta_{t+1}^{\frac{1}{2}} = \theta_t - \alpha \nabla_\theta \mathbf{V}(\theta_t, \phi_t), \quad \phi_{t+1}^{\frac{1}{2}} = \phi_t + \alpha \nabla_\phi \mathbf{V}(\theta_t, \phi_t); \] (13a)

**gradient step**: \[ \theta_{t+1} = \theta_t - \beta \nabla_\theta \mathbf{V}(t, \phi_t^{\frac{1}{2}}), \quad \phi_{t+1} = \phi_t + \beta \nabla_\phi \mathbf{V}(t, \phi_t^{\frac{1}{2}}) \] (13b)

Where \( \nabla \mathbf{V}(\theta_t, \phi_t) \) is the signed gradient vector at time \( t \). The sum \( \prod_{t \neq t+1} \left[ \nabla \mathbf{V}(t, \phi_t^{\frac{1}{2}}) - \nabla \mathbf{V}(t, \phi_t) \right] \) is the projection of the centripetal acceleration term \( [\nabla \mathbf{V}(t, \phi_t^{\frac{1}{2}}) - \nabla \mathbf{V}(t, \phi_t)] \) onto the vecto \( \nabla \mathbf{V}(t, \phi_t). \)
Li et al. (2020) proposed two versions of the PPCA methods by constraining the coefficient matrix which must be full rank in bilinear games under the specified situation (Lemma 3.2 in [16]). The form of PPCA method for bilinear game is

\[
\begin{align*}
\text{predictive step: } & \quad \theta_{t+1} = \theta_t - \gamma \nabla_{\theta} V(\theta_t, \phi_t); \\
& \quad \phi_{t+1} = \phi_t + \gamma \nabla_{\phi} V(\theta_t, \phi_t); \\
\text{gradient step: } & \quad \theta_{t+1} = \theta_t - \alpha \nabla_{\theta} V(\theta_t, \phi_t) - \beta [\nabla_{\theta} V(\theta_t, \phi_t) - \nabla_{\phi} V(\theta_t, \phi_t)]; \\
& \quad \phi_{t+1} = \phi_t + \alpha \nabla_{\phi} V(\theta_t, \phi_t) + \beta [\nabla_{\theta} V(\theta_t, \phi_t) - \nabla_{\phi} V(\theta_t, \phi_t)].
\end{align*}
\]

And also, we can get the alternating PPCA formula is as follows:

\[
\begin{align*}
\text{predictive step: } & \quad \theta_{t+1} = \theta_t - \gamma \nabla_{\theta} V(\theta_t, \phi_t); \\
& \quad \phi_{t+1} = \phi_t + \gamma \nabla_{\phi} V(\theta_t, \phi_t); \\
\text{gradient step: } & \quad \theta_{t+1} = \theta_t - \alpha \nabla_{\theta} V(\theta_t, \phi_t) - \beta [\nabla_{\theta} V(\theta_t, \phi_t) - \nabla_{\phi} V(\theta_t, \phi_t)]; \\
& \quad \phi_{t+1} = \phi_t + \alpha \nabla_{\phi} V(\theta_t, \phi_t) + \beta [\nabla_{\theta} V(\theta_t, \phi_t) - \nabla_{\phi} V(\theta_t, \phi_t)].
\end{align*}
\]

where the all of $\gamma$, $\alpha$, $\beta$ are hyper parameters.

Although the mentioned methods have achieved significant work in training GANs, some of them require high computing costs and computer memory. The rest of them depend closely on the approximate circular motion process. If the practical numerical experiments do not satisfy this approximate theory, these methods will not be valid. In contrast to our approach, we propose the adaptive composite gradient method, which can reduce the computing cost and solve the limitation brought by the approximate circular motion process.

### 3. MOTIVATION

#### 3.1 Limitation Analysis

These Hessian-based methods, such as consensus optimization and SGA, optimize game problems, bringing high computing costs. The centripetal acceleration algorithm reduces expensive computational costs but depends on the approximately uniform circular motion process assumption, shown in Figure 3. PPCA is an improved version of the centripetal acceleration algorithm, shown in Figure 4(b). However, PPCA needs a full rank coefficient matrix, causing the projection of the PPCA method to be zero (Lemma 3.2 in [16]). The proposed ACG is motivated by two aspects. Firstly, we consider cyclic behavior as a generally circular motion process but not a uniform one. Therefore, similar to the centripetal acceleration method, we modify directions by adding the projection of the centripetal acceleration term at time $t$. Secondly, A³DMM provides an idea that can use past iterations to predict future iterations because the trajectory of the sequence $Z_k$ is either straight linear or spiral. However, the cyclic behavior is also approximately a spiral. Motivated by the two aspects, we proposed a novel adaptive composite gradient (ACG) method to alleviate the cyclic behavior in
training GANs. ACG can reduce computing costs and accelerate iteration by predicting the future step iterative information, which is called a **Semi-Gradient-Free** method.

![Diagram](image)

**Figure 3.** The limitations of centripetal acceleration method. Left: $|\nabla V_{t+dt}| > |\nabla V_t|$, Right: $|\nabla V_{t+dt}| < |\nabla V_t|$.  

![Diagram](image)

**Figure 4.** (a): The spiral trajectory of $Z_k$. (b): Degenerated PPCA method in case $|\vec{b}| = 0$. The degenerated PPCA is the same as centripetal acceleration method, if $\delta t = 1/2$.

### 3.2 Motivational Theory

Our idea is motivated by **A³DMM** [31]. Then, we review the **A³DMM** method. Give an optimisation problem...
To derive the iteration scheme, considering the augmented Lagrangian and rewriting the optimisation problem, which reads

\[ \mathcal{L}(x, y; \Psi) = \mathbf{R}(x) + \mathbf{J}(y) + \langle \Psi, \mathbf{A}x + \mathbf{B}y - b \rangle + \frac{\gamma}{2} \| \mathbf{A}x + \mathbf{B}y - b \|^2, \]

where \( \gamma > 0 \) and \( \Psi \) is the Lagrangian multiplier, then we have iteration forms:

\[ x_k = \arg \min_{x \in \mathbb{R}^n} \mathbf{R}(x) + \frac{\gamma}{2} \| \mathbf{A}x + \mathbf{B}y_{k-1} - b \|^2 + \frac{1}{\gamma} \Psi_{k-1} \]  
\[ y_k = \arg \min_{y \in \mathbb{R}^m} \mathbf{J}(y) + \frac{\gamma}{2} \| \mathbf{A}x_k + \mathbf{B}y - b \|^2 + \frac{1}{\gamma} \Psi_{k-1} \]  
\[ \Psi_k = \Psi_{k-1} + \gamma (\mathbf{A}x_k + \mathbf{B}y_{k-1} - b) \]  

We can rewrite the above iteration into the following formula by introducing a new variable \( Z_{k}^{\text{def}} = \Psi_{k-1} + \gamma \mathbf{A}x_k \):

\[ x_k = \arg \min_{x \in \mathbb{R}^n} \mathbf{R}(x) + \frac{\gamma}{2} \| \mathbf{A}x - \frac{1}{\gamma} (Z_{k-1} - 2 \Psi_{k-1}) \|^2 \]  
\[ Z_k = \Psi_{k-1} + \gamma \mathbf{A}x_k \]  
\[ y_k = \arg \min_{y \in \mathbb{R}^m} \mathbf{J}(y) + \frac{\gamma}{2} \| \mathbf{B}y + \frac{1}{\gamma} (Z_k - \gamma b) \|^2 \]  
\[ \Psi_k = Z_k + \gamma (\mathbf{B}y_k - b) \]  

The trajectory of the sequence \( Z_k \) depends closely on the value of \( \gamma \), where \( k \in \mathbb{N} \). If selecting a proper \( \gamma \), the eventual trajectory of \( Z_k \) is spiral as shown in Figure 4. since the trajectory of \( Z_k \) has the characteristic of spiral, which provides a way that using the previous \( q \) iteration predicts the future \( s \) iterations. The \( Z_k \) of ADMM can be estimated by \( \tilde{Z}_{k,s} \) which is defined as follow:

\[ \tilde{Z}_k = \mathcal{F}(Z_k, Z_{k-1}, \ldots, Z_{k-q}) \]

for the choice of \( s = 1 \). Given a sequence \( Z_{k,i} \), \( i = 1, 2, \ldots, q+1 \), we can define \( v_i = Z_{i} - Z_{i-1} \) and obtain the past \( \nu_{k-1}, \nu_{k-2}, \ldots, \nu_{k-q} \), which can be used the \( v_i = \nu_{k-i} \) to approximate \( \nu_{k+1} \). Then we can use the \( V_k C_k \) to approximate \( v_{k+1} \), that is \( \tilde{Z}_{k+1} = \tilde{Z}_k + V_k C_k \). By iterating \( s \) times, we can obtain \( \tilde{Z}_{k,s} \) by this method proposed by Clarice Poon et al. [31].
4. ADAPTIVE COMPOSITE GRADIENT METHOD

To more easily understand our method, we make some symbol conventions throughout the paper. Let \( a \odot b \) is considered as the projection of \( a \) onto \( b \), where \( a, b \in \mathbb{R}^n \) and \( \odot \) denotes the projecting calculation between two vectors. \( W_t^i \) denotes the \( i^{th} \) player controlling parameter at time \( t \) and \( W_t = (w_1^t, w_2^t, \ldots, w_n^t) \). We use \( \{\ell_i : \mathbb{R}^d \to \mathbb{R}_{n=1}^n\} \) to denote the losses corresponding to \( n \) players which is same as mentioned in Definition 2.1. Then we can obtain \( \mathcal{L}(W_i) = (\ell_1(w_1^t), \ell_2(w_2^t), \ldots, \ell_n(w_n^t)) \), which is the payoff vector of the \( n \) players at time \( t \).

We consider a bilinear game problem with the following form

\[
\ell_i(w_1^t, w_2^t) = (w_1^t)^T A w_2^t, \quad \ell_2(w_1^t, w_2^t) = -(w_1^t)^T A w_2^t.
\]

**Desiderata.** This two players game has a Nash equilibrium which must satisfy

D1. The two losses satisfy: \( \sum_{i=1}^2 \ell_i(w_1^t, w_2^t) = 0 \); 
D2. For \( \ell_i \) is differentiable over the parameter space \( \Omega(w_1) \times \Omega(w_2) \), where \( \Omega(w_1) \times \Omega(w_2) \subseteq \mathbb{R}^n \times \mathbb{R}^n \).

The player 1 holds the parameter \( w_1 \) and tries to minimize the loss \( \ell_1 \), while the player 2 holds the parameter \( w_2 \) and tries to minimize the loss \( \ell_2 \). From the D1. we can get \( \ell_1 = -\ell_2 \), the formula (22) can be rewritten as

\[
\min_{w_1 \in \Omega(w_1)} \max_{w_2 \in \Omega(w_2)} \mathcal{L}(w_1^t, w_2^t).
\]

**Desiderata.**

The dynamics of gradient descent ascent based method is

\[
w_{t+1}^1 = w_t^1 - \alpha \nabla_{w_1} \mathcal{L}(w_t^1, w_t^2), \quad w_{t+1}^2 = w_t^2 + \alpha \nabla_{w_2} \mathcal{L}(w_t^1, w_t^2).
\]

The ACG method involves three parts which are consist of the composite gradients. Firstly, we introduce the predictive aspect. In this section, \( W_t = (w_1^t, w_2^t) \) is the parameter vectors at time \( t \). Similar to \( \text{A3DMM} \), we utilize the \( W \) of the previous \( q \) iterations to predict the future \( s \) iterations which can be denoted by \( W_{t,s} \). Then we can get the following formula of \( W_{t,s} \):

\[
W_{t,s}^k = \mathcal{F}(W_t, W_{t-1}, \ldots, W_{t-q}),
\]

for the value of \( s = 1 \). Define \( v_i = W_i - W_{i-1} \), \( W_i \) is from sequence \( \{W_{k-i}\}_{i=0}^q \). We can use the past \( v_{k-1}^1, v_{k-2}^1, \ldots, v_{k-q}^1 \) to approximate the last \( v_k^1 \). Then, \( c_i = \text{argmin}_{c_i \in \mathbb{R}} \|V_{k-1} - v_k^1\|^2 \). Finally, we can obtain \( W_{k,i} = W_k + V_k c_k \approx W_{k+1} \). By looping of \( s \) times, we can get \( W_{k,s} \approx W_{k+s} \). The second part and third part of our ACG method are the \( \nabla \mathcal{L}(w_1^t, w_2^t) \) and the projection of centripetal acceleration term. The dynamics of proposed ACG method is
Algorithm 1. ACG-Adaptive Composite Gradients method for Bilinear game.

Input: losses $\mathcal{L}(\ell_1, \ell_2)$ and $W = (w^1, w^2)$

Initial: Let $s \geq 1$, $q \geq 1$ be integers and $k = q + 1$, learning rate $\alpha$, adaptive rate $\beta_1, \beta_2$, $W_0 = (w^1_0, w^2_0)$.

while not converged do
for $t \geq 1$ do
if $\text{mod}(t,k) = 0$ then
Compute $C_t$ and $\bar{W}^t_{\alpha\beta} = (\bar{W}^1_{\alpha\beta}, \bar{W}^2_{\alpha\beta})$:
Composite gradients:
$$G_{w^1} = \nabla_{w^1} \mathcal{L}(W) + \frac{\beta_1}{\alpha} (\tilde{a}_1 - \tilde{b}_1) + \frac{\beta_2}{\alpha} \bar{W}^1_{\alpha\beta};$$
$$G_{w^2} = \nabla_{w^2} \mathcal{L}(W) + \frac{\beta_1}{\alpha} (\tilde{a}_2 - \tilde{b}_2) + \frac{\beta_2}{\alpha} \bar{W}^2_{\alpha\beta};$$
Gradient update: $w^1_{t+1} = w^1_t - \alpha G_{w^1}$; $w^2_{t+1} = w^2_t + \alpha G_{w^2}$.
else
OMD update:
$$w^1_{t+1} = w^1_t - \alpha \nabla_{w^1} \mathcal{L}(W_t) + \beta \nabla_{w^2} \mathcal{L}(W_t);$$
$$w^2_{t+1} = w^2_t + \alpha \nabla_{w^2} \mathcal{L}(W_t) - \beta \nabla_{w^1} \mathcal{L}(W_t).$$
end if
end for
end while

composite gradients:
$$G_{w^1} = \nabla_{w^1} \mathcal{L}(w^1_t, w^2_t) + \frac{\beta_1}{\alpha} (\tilde{a}_1 - \tilde{b}_1) + \frac{\beta_2}{\alpha} \bar{W}^1_{\alpha\beta};$$
$$G_{w^2} = \nabla_{w^2} \mathcal{L}(w^1_t, w^2_t) + \frac{\beta_1}{\alpha} (\tilde{a}_2 - \tilde{b}_2) + \frac{\beta_2}{\alpha} \bar{W}^2_{\alpha\beta};$$

gradient step:
$$w^1_{t+1} = w^1_t - \alpha G_{w^1}, \quad w^2_{t+1} = w^2_t + \alpha G_{w^2},$$

where the $\nabla_{w^1} \mathcal{L}(w^1_t, w^2_t)$ denotes the partial derivatives of $w_1$ corresponding to $\ell_1$ at time $t$. The $\tilde{a}_1$ denotes $\nabla_{w^1} \mathcal{L}(w^1_t, w^2_t) - \nabla_{w^1} \mathcal{L}(w^1_{t-1}, w^2_{t-1})$. The $\tilde{b}_1$ denotes $\tilde{a}_1 \odot \nabla_{w^2} \mathcal{L}(w^1_{t-1}, w^2_{t-1})$, which is the projection of $\tilde{a}_1$ onto the vector $\nabla_{w^1} \mathcal{L}(w^1_{t-1}, w^2_{t-1})$. The basic intuition of our proposed method is shown in Figure 5.
For clarity, we draft the scheme of proposed adaptive composite gradient method in Algorithm 1, which is applied in a two-player game problem. Note that the ACG method can calibrate any optimizer based on gradient descent ascent. Meanwhile, the ACG method can extend to a $n$-player game problem. Given a $n$-player game problem, let $g(W_t)$ be the gradient of losses for all players at $t$. It is worth noting that all loss functions must be differentiable. We adopt the same way of Algorithm 1 to compute $W_{t+1}$. The dynamics of the ACG method for $n$-players reads

$$W_{t+s} = F(W_t, W_{t-1}, \ldots, W_{t-s}),$$

$$\text{composite gradients} : G_W = g(W_t) + \frac{B}{\alpha} (\tilde{a} - \tilde{b}) + \frac{B}{\alpha} W_{t+s};$$

$$\text{gradient step} : W_{t+s} = W_t - \alpha G_W.$$  

Where the $\tilde{a}$ denotes $g(W_t) - g(W_{t-1})$, $\tilde{b}$ denotes $\tilde{a} \odot g(W_{t-1})$ which is the projection of $\tilde{a}$ onto $g(W_{t-1})$.

**Figure 5.** The basic intuition of our proposed Adaptive Composite Gradient method. To illustrate our approach, we chose the $s = 20$ in this figure.
Algorithm 2. ACG - Adaptive Composite Gradients method for the n-players game.

Input: losses $\mathcal{L}(W)$ and $W = (w^1, w^2, ..., w^n)$

Initial: Let $s \geq 1$, $q \geq 1$ be integers and $k = q + 1$, learning rate $\alpha$, adaptive rate $\beta_1, \beta_2$; $W_0 = (w_0^1, w_0^2, ..., w_0^n)$.

while not converged do
  for $t \geq 1$ do
    if mod($t$, $k$) = 0 then
      Compute $C_t$ and $W_{t+1}^\tau$: $W_{t+1}^\tau = \mathcal{F}(W_t, W_{t-1}, ..., W_{t-q})$;
      Compute composite gradients: $G_W = g(W_t) + \frac{\beta_1}{\alpha}(\bar{a} - \bar{b}) + \frac{\beta_2}{\alpha}W_{t+1}^\tau$;
      Gradient update: $W_{t+1} = W_t - \alpha G_W$.
    else
      Gradient update: $W_{t+1} = W_t - \alpha g(W_t)$.
    end if
  end for
end while

Remark 4.1 The value of $k$ can be controlled both in Algorithm 1 and Algorithm 2. Let $k = q + i$ where $i \in \mathbb{N}^*$, we can set different acceleration ratio of the algorithms by adjusting the values of $k,s$.

In Algorithm 1 and Algorithm 2, there is no need to calculate the gradient of each iteration because of $W_{t+1}^\tau$. Therefore, we also name it a Semi-Gradient-Free method, whose merit is that it can reduce the computational cost and convergent fast. $\beta_1, \beta_2$ can be used to control the convergence speed in our algorithms.

5. THE CONVERGENCE OF ADAPTIVE COMPOSITE GRADIENT METHOD

5.1 The Convergence Analysis for the Bilinear Game

In this subsection, we mainly discuss the convergence of Adaptive Composite Gradient Method in the bilinear game, which reads

$$\min_{\theta \in \mathbb{R}^d} \max_{\phi \in \mathbb{R}^d} \theta^T A \phi + \theta^T B + C^T \phi, A \in \mathbb{R}^{d \times d}, B, C \in \mathbb{R}^d.$$  \hspace{1cm} (29)

For any local Nash equilibrium of the bilinear game has the point $(\theta^*, \phi^*)$ satisfied the following conditions:

$$A \phi^* + B = 0, \hspace{1cm} (30a)$$
$$A^T \theta^* + C = 0. \hspace{1cm} (30b)$$

The local Nash equilibrium exists if and only if the ranks of $A$ and $A^T$ are the same as the dimension of $B, C$. By this way, without loss of generality, we can convert $(\theta, \phi)$ to $(\theta - \theta^*, \phi - \phi^*)$, which is used to rewrite bilinear game (29) as:
Training Generative Adversarial Networks with Adaptive Composite Gradient

Before analyzing the convergence property of the Adaptive Composite Gradient Method in this situation, we introduce some essential theorems and propositions.

**Theorem 5.1** Suppose $F \in \mathbb{R}^{d \times d}$ is from the iterative system $x_{k+1} = Fx_k$. If $F$ is nonsingular and satisfies the spectral radius $\rho(F) < 1$, then the $x_k$ of iterative system converges to 0 linearly.

**Theorem 5.2** (OMD) Consider a bilinear game $V(\theta, \phi) = \theta^T A \phi$, where $A \in \mathbb{R}^{d \times d}$. Assume $A$ is full rank. Then the following dynamics,

$$\begin{align*}
\theta_{t+1} &= \theta_t - 2\eta V'(\theta_t, \phi_t) + \eta V'(\theta_{t-1}, \phi_{t-1}), \\
\phi_{t+1} &= \phi_t + 2\eta V'(\theta_t, \phi_t) - \eta V'(\theta_{t-1}, \phi_{t-1}),
\end{align*}$$

with the learning rate

$$\eta = \frac{1}{2\sqrt{2\lambda_{\max}(AA^T)}},$$

obtain an $\varepsilon$-minimizer such that $(\theta_t, \phi_t) \in B_\varepsilon(\varepsilon)$, provided

$$T \geq T_{\text{OMD}} := \left\lceil \frac{4\sqrt{2\delta}}{\lambda_{\max}(AA^T)} \log \frac{4\sqrt{2\delta}}{\varepsilon} \right\rceil$$

under the assumption that $||| (\theta, \phi) ||| \leq \delta$.

To discuss the convergence of the ACG method for the bilinear game, we divided the analysis process into three parts (two cases). Without loss of generality, let $t$ represent the iterative step, and $k$ is the previous steps. The convergence property of Algorithm 1 is as the following.

**Case 1.** $\text{mod}(t,k) \neq 0$ or $\text{mod}(t,k) = 0$ & $\rho(C_k) \geq 1$. The ACG method adopts the dynamics:

$$\begin{align*}
w_{t+1}^1 &= w_t^1 - \alpha V_{w_1^1} \mathcal{L}(w_t^1, w_t^2); \\
w_{t+1}^2 &= w_t^2 + \alpha V_{w_2^1} \mathcal{L}(w_t^1, w_t^2) - \beta \nabla_{w_1^1} \mathcal{L}(w_t^1, w_t^2); \\
w_{t+1}^2 &= w_t^2 + \alpha V_{w_2^1} \mathcal{L}(w_t^1, w_t^2) - \beta \nabla_{w_1^1} \mathcal{L}(w_t^1, w_t^2).
\end{align*}$$

Taking $\alpha = 2\beta = 2\eta$ in **Case 1**, the dynamics scheme reduces to **OMD**, which can be written as:

$$\begin{align*}
w_{t+1}^1 &= w_t^1 - 2\eta V_{w_1^1} \mathcal{L}(w_t^1, w_t^2); \\
w_{t+1}^2 &= w_t^2 + 2\eta V_{w_2^1} \mathcal{L}(w_t^1, w_t^2).
\end{align*}$$

**Theorem 5.2** has assigned the condition about learning rate of **OMD** and it is exponential Convergence. The convergence of **OMD** can be found in [34] (Theorem 3).

**Case 2.** $\text{mod}(t,k) = 0$ & $\rho(C_k) < 1$. From the Algorithm 1, firstly we shall compute the $w_{t+s}$ by:

$$w_{t+s}^1 = F(w_t^1, w_{t-1}^1, \cdots, w_{t-q}^1), \quad w_{t+s}^2 = F(w_t^2, w_{t-1}^2, \cdots, w_{t-q}^2).$$
Using the fixed-point formulation of ADMM, (35) can be written as an unified \( \overline{\theta}_t = \epsilon(\overline{\theta}_{t-1}) \), let \( V_t \epsilon = \sigma_t \), then \( \overline{\theta}_t = \epsilon(\overline{\theta}_{t-1} + \sigma_t) \). We can obtain the convergence of (35), iff \( \sigma_t \) converges to 0. The convergence of (35) is based on the convergence of inexact Krasnosel’skii-Mann fixed-point iteration in [31] (Proposition 5.34). The detail convergence analysis of \( \overline{\theta}_t = \epsilon(\overline{\theta}_{t-1} + \sigma_t) \) has been discussed in [31] (Proposition 4.2).

Then we discuss the convergence of composite gradients update scheme, which is written as follow:

\[
w^1_{t+1} = w^1_t - \alpha G^1_{w^1}, w^2_{t+1} = w^2_t + \alpha G^2_{w^2},
\]

where the \( G^1_{w^1}, G^2_{w^2} \) are defined as:

\[
G^1_{w^1} = \nabla_{w^1} \mathcal{L}(w^1_t, w^2_t) + \frac{\beta_1}{\alpha}(\overline{a}_t - \overline{b}_t) + \frac{\beta_2}{\alpha} \overline{w}^1_{t+1};
\]

\[
G^2_{w^2} = \nabla_{w^2} \mathcal{L}(w^1_t, w^2_t) + \frac{\beta_1}{\alpha}(\overline{a}_t - \overline{b}_t) + \frac{\beta_2}{\alpha} \overline{w}^2_{t+1};
\]

(37)

**Proposition 5.3** Any given two vector \( \bar{a}, \bar{b} \), the projection of vector \( \bar{b} \) onto the vector \( \bar{a} \) can be denoted as \( \gamma \bar{a} \), where the \( \gamma \in \mathbb{R} \).

According to proposition 5.3, our dynamics of the bilinear game, the composite gradients update scheme is reduced to be

\[
\theta^{t+1}_1 = \theta_t - (\alpha + \beta_1) A \phi_t + \beta_1 (1 + \gamma) A \phi_{t-1} - \beta y \bar{q};
\]

\[
\phi^{t+1}_t = \phi_t + (\alpha + \beta_1) A^\top \theta_t - \beta_1 (1 + \gamma) A^\top \theta_{t-1} + \beta y \bar{q};
\]

(38)

We can obtain the iterative matrix as:

\[
F := \begin{bmatrix}
   I_d & -(\alpha + \beta_1) A & 0 & \beta_1 (1 + \gamma) A & -\beta_1 I_d & 0 \\
   (\alpha + \beta_1) A^\top & I_d & -\beta_1 (1 + \gamma) A^\top & 0 & 0 & \beta_1 I_d \\
   I_d & 0 & 0 & 0 & 0 & 0 \\
   0 & I_d & 0 & 0 & 0 & 0 \\
   0 & 0 & 0 & \tau I_d & 0 & 0 \\
   0 & 0 & 0 & 0 & \tau I_d & 0 
\end{bmatrix},
\]

(39)

where \( \tau \) is defined by (35).

According to the iterative matrix, it is easy to obtain that

\[
[\theta_{t+1}, \phi_{t+1}, \theta_t, \phi_t, \overline{w}_{q+1}, \overline{w}_{q+1}]^T = F [\theta_t, \phi_t, \theta_{t-1}, \phi_{t-1}, \overline{w}_{q}, \overline{w}_{q}]^T,
\]

where the \( (\theta, \phi) \) are generated by (38). With the assumption that \( A \) is square and nonsingular in Proposition 5.4, we use the well-known Theorem 5.1 to illustrate the linear convergence for the update scheme (38).

**Proposition 5.4** Suppose that \( A \) is square and nonsingular. Then, the eigenvalues of \( F \) are the roots of the sixth order polynomials:
where \( Sp(\cdot) \) denotes the collection of all eigenvalues.

**Proposition 5.5** Suppose that \( A \) is square and nonsingular. The \( 2 \sum_{t=0}^{T} \sum_{i=1}^{n} \left( \vartheta_{t+i}^2 \right) \) is linearly convergent to 0 for given \( \gamma \) with \( \alpha \) and \( \beta \) satisfy

\[
0 < \alpha + \beta \leq \frac{1}{\sqrt{\lambda_{\text{max}}(A^TA)}}, \quad \alpha + \beta | + |2\beta(1+\gamma)| \leq \frac{(\alpha + \beta)^2 \sqrt{\lambda_{\text{min}}(A^TA)}}{10}
\]

where the \( \lambda_{\text{max}}(\cdot) \), \( \lambda_{\text{min}}(\cdot) \) denote the largest and the smallest eigenvalues of \( A^TA \).

### 5.2 The Convergence Analysis for the n-player Game

This subsection mainly discusses the convergence of the Adaptive Composite Gradient method in the general \( n \)-player game. The problem is described as in Definition 2.1, according to Algorithm 2, the convergence analysis process in general \( n \) players game is the same as that of the bilinear game with three parts and two cases. Before analyzing convergence property, we introduce some basic definitions.

**Definition 5.6** Suppose that \( f : \mathbb{R}^n \to \mathbb{R} \) and it is convex, continuously. For \( \forall x, y \in \mathbb{R}^n \), the gradient of \( f \) is Lipschitz continuous with constant \( L \) such that:

\[
0 \leq f(y) - f(x) - \langle \nabla f(x), y - x \rangle \leq \frac{L}{2} || x - y ||^2,
\]

we define that \( f \) belongs to the class \( \mathcal{F}^{1,1}_L \). If \( f \) is strongly convex with modulus \( \mu > 0 \) and such that:

\[
\frac{\mu}{2} || x - y ||^2 \leq f(y) - f(x) - \langle \nabla f(x), y - x \rangle
\]

we define that \( f \) belongs to \( \mathcal{F}^{1,1}_{\mu,1} \).

Next, we suppose that the all \( \vartheta_{t+i}, i = 1, 2, \ldots, n \) are belonging to \( \mathcal{F}^{1,1}_L \). We can get the definition of fixed point which is also called Local Nash Equilibrium in game.

**Definition 5.7** \( W \) is a Local Nash Equilibrium (fixed point) if \( W^* \) satisfy \( g(W) = 0 \). We say that it is stable if \( g(W) \geq 0 \), unstable if \( g(W) \leq 0 \).

**Theorem 5.8** [Nesterov 1983] Let \( f \) be a convex and \( \beta \)-smooth function, and we can write the well-known Nesterov's Accelerated Gradient Descent update scheme as

\[
y_{t+1} = x_t - \alpha \nabla f(x_t), \quad x_{t+1} = y_{t+1} + \beta (y_{t+1} - y_t),
\]

Then Nesterov's Accelerated Gradient Descent satisfies

\[
f(x_t) - f(x^*) \leq \frac{2\beta || x_t - x^* ||^2}{t^2}.
\]
Nesterov (1983) proposed the accelerated gradient method which achieves the optimal $O\left(\frac{1}{t^2}\right)$ convergence rate.

The convergence of the ACG method for the general $n$ player game is also divided into two cases. Let $t$ represent the iterative step, and $k$ is the previous steps. The convergence property of Algorithm 2 is as the following.

**Case 1.** $\text{mod}(t,k)\neq0$ or $\text{mod}(t,k) = 0 \& \rho(C_k) \geq 1$. From Algorithm 2, if $t$ and $k$ satisfy the conditions of **Case 1**. Our proposed ACG method is the same as classical Gradient Descent and the update scheme is

$$W_{t+1} = W_t - \alpha g(W_t),$$

where the $\alpha$ is a positive step-size parameter. According to Definition 5.7, let the Local Nash Equilibrium $W$ and $\mathcal{L} = \mathcal{L}(W^*)$. Base the Definition 5.6, if $\ell, i = 1, 2, \ldots, n$ are belonging to $\mathcal{F}^{1,\ell}$, then $\mathcal{L}(W_i) - \mathcal{L}$ associated $(W_i)$ converges at rate $O\left(\frac{1}{t}\right)$. For more detail convergence of averaged iterations with generalized gradient descent update scheme in convex-concave games is analyzed in [36, 37].

**Case 2.** $\text{mod}(t,k) = 0 \& \rho(C_k) < 1$. According to Algorithm 2, firstly we should compute the $\overline{W}_{t+1}$ by:

$$\overline{W}_{t+1} = \mathcal{F}(W_t, W_{t-1}, \ldots, W_{t-q}).$$

The convergence of formula (44) is the same as that of **Case 2** in previous section 5.1. For more detail about convergence of (44) has been discussed in [30] (Proposition 4.2).

In **Case 2**, we mainly analyze the convergence of composite gradient update scheme in formula (28). Before illustrating our proposed method convergence property, The formula (42) can be equivalently written as

$$x_{t+1} = x_t - (\alpha + \beta)\nabla f(x_t) + \alpha \beta \nabla f(x_{t-1}) + \beta(x_t - x_{t-1}),$$

where the $\alpha$ and $\beta$ are step-size parameters. However, our proposed composite gradient method (28) can be transfer to the similar formula as (45) which is based on the Proposition 5.3. That is

$$W_{t+1} = W_t - (\alpha + \beta)g(W_t) + \beta(1 + \gamma)g(W_{t-1}) - \beta x_t - \overline{W}_{t+1},$$

where the $\overline{W}_{t+1}$ is equivalent to $(W_t - W_{t-1})$. Comparing (45) with (46) if the parameters are equivalently transformed, our proposed adaptive composite gradient method can be reduced to the Nesterov's Accelerated Gradient (NAG) method by this way. In 1983, Nesterov had given the convergence rate at $O\left(\frac{1}{t^2}\right)$ for convex smooth optimization in [38]. And also there had given the convergence bounds for convex, non-convex and smooth optimization in [39] (Theorem 1 ~ Theorem 4).

**Remark 5.1** In **Case 2**, our proposed Adaptive composite gradient method has the same convergence rate and the same convergence bounds as that of NAG method, since the assumption that all
$\ell, i = 1, 2, \ldots, n$ are belonging to $F^1_{\ell_i}$. We can naturally obtain the convergence rate of the ACG method, which is at $O\left(\frac{1}{t^2}\right)$ based on Theorem 5.8.

6. EXPERIMENTS

6.1 Toy Functions Simulation

We tested our ACG methods in Algorithm 1 and Algorithm 2 corresponding with the bilinear game and general game with 3 players. We tested the ACG method in Algorithm 1 on the following bilinear game, which can be written as

$$
\min_{\theta, \phi} \max_{d, d'} \theta \cdot \phi, \quad d = 1. \tag{47}
$$

It is obvious that the Nash Equilibrium (stationary point) is $(0,0)$. We compared our ACG with some other methods whose results are presented in Figure 6 (a). From the behaviors of Figure 6, the Sim-GDA method diverges, and the Alt-GDA method is rotating around the stationary point. However, the other methods all converge to the Nash Equilibrium. We proposed the ACG method converges faster than other convergence methods.

![Diagram showing the effects of various compared methods in two player games with 150 iterations.](Image)

Figure 6. The effects of various compared methods in two player games with 150 iterations. The parameters of compared methods are different in the three toy functions. In g1 function, the parameters are as following: Sim-GDA($\alpha = 0.1$), Alt-GDA($\alpha = 0.1$), Grad-SCA($\alpha = 0.1$, $\beta = 0.3$), Grad-ACA($\alpha = 0.1$, $\beta = 0.3$), OMD($\alpha = 0.1$, $\beta = 0.1$), MPM($\alpha = 0.1$, $\gamma = 1.0$), PPCA($\alpha = 0.1$, $\beta = 0.3$, $\gamma = 1.0$), APPCA($\alpha = 0.1$, $\beta = 0.3$, $\gamma = 1.0$), ACG($\alpha = 0.05$, $\beta_1 = 0.5$, $\beta_2 = 1.0$). In g2 function, the parameters are as following: Sim-GDA($\alpha = 0.1$), Alt-GDA($\alpha = 0.1$), Grad-SCA($\alpha = 0.1$, $\beta = 0.3$), Grad-ACA($\alpha = 0.1$, $\beta = 0.3$), OMD($\alpha = 0.1$, $\beta = 0.1$), MPM($\alpha = 0.3$, $\gamma = 0.2$), PPCA($\alpha = 0.1$, $\beta = 0.3$, $\gamma = 1.0$), APPCA($\alpha = 0.1$, $\beta = 0.3$, $\gamma = 0.2$), ACG($\alpha = 0.1$, $\beta_1 = 0.05$, $\beta_2 = 1.0$).

In Figure 6 (b), we test our ACG method on the following general zero-sum game

$$
\min_{\theta, \phi} \max_{d, d'} 3\theta^2 + \phi^2 + 4\theta \cdot \phi, \quad d = 1. \tag{48}
$$
The effects of the compared methods on this game show that all methods converge to the origin. Notably, the cyclic behavior of the Alt-GDA method has disappeared, and the Sim-GDA method converges. It is worth noting that the trajectory of our ACG method is the same as that of PPCA [16]. Both our ACG and PPCA [16] seem faster than others. We also compared ACG with other methods on the following general game

\[
\min_{\theta, \phi} \max_{d} \left( \theta^2 + \phi^2 - 4\theta \cdot \phi, \ d = 1 \right). \tag{49}
\]

Its effects are presented in Figure 6 (c), which shows that Sim-GDA and Grad-SCA diverge, the rest methods converge. The APPCA [16] is faster than our ACG method in this game.

We used the last general zero-sum game (49) to test the robustness of the proposed ACG method in Figure 7. As the learning rate \( \alpha \) increases through \( \{0.01, 0.05, 0.1\} \) and other parameters keeping same. ACG method converges faster when the learning rates setting with \( \alpha = 0.01 \) and \( \alpha = 0.05 \), although converge slower with learning rate \( \alpha = 0.1 \), ACG method still converge to the origin rapidly.

![Image of Figure 7](image_url)

(a) \( \alpha = 0.01, \beta_1 = 0.02, \beta_2 = 1 \)  
(b) \( \alpha = 0.05, \beta_1 = 0.02, \beta_2 = 1 \)  
(c) \( \alpha = 0.1, \beta_1 = 0.02, \beta_2 = 1 \)

**Figure 7.** The robustness of the Adaptive Composite Gradient method on \( g^3 \) toy function. It is worth noting that the proposed model has significant robustness, which means the ACG algorithm can quickly converge to the Nash point, whatever the value of \( \alpha \) changes.

The proposed Adaptive Composite Gradient method (ACG) is also suitable for the general game with \( n \) players. However, it is challenging to present the effect of the general \( n \) player game by toy function. To illustrate our proposed method of Algorithm 2 adaptive for \( n \) players game, we show the effects of Algorithm 2 by a general 3 players game. The payoff functions can be written as

\[
\ell_1(x, y, z) = \frac{1}{4} x^2 + xy + xz, \tag{50a}
\]

\[
\ell_2(x, y, z) = -xy + \frac{1}{10} y^2 + yz, \tag{50b}
\]

\[
\ell_3(x, y, z) = -xz - yz + \frac{1}{10} z^2. \tag{50c}
\]

Where the local Nash Equilibrium is \((0,0,0)\). The effects are shown in Figure 8, the top row and the bottom left of Figure 8 are the trajectories of the compared methods, and the bottom right of Figure 8 are
the Euclidean distances of each iteration away from the origin for compared methods. Figure 8 presented that SGD, SGA, and our ACG method all converge to origin. There is cyclic behavior in SGD, which leads to converging slowly. The second row of Figure 8 presents that the proposed ACG method approaches the origin faster than SGA and SGD.

![Figure 8](image)

**Figure 8.** The effects of SGD, SGA, proposed ACG method in general 3 players’ game. It shows projection trajectories of competing algorithms on three coordinate planes. The right subplot of the second row shows the euclidean distance from the current point to the Nash point as the iteration increased.

### 6.2 Mixtures of Gaussians

We also tested the ACG method by training a toy GAN model, and we compared our method with other well-known methods on learning 5 Gaussians and 16 Gaussians. All the mixture of 16 Gaussians and 5 Gaussians are appointed with a standard deviation of 0.02. The Ground truths for the mixture of 16 Gaussians and 5 Gaussians are present in Figure B1.

**Details on Network architecture.** GANs is consist of a generator network and a discriminator network. We set up both the generator and discriminator networks with six fully connected layers, and each layer has 256 neurons. We used the fully connected layer to replace the sigmoid function layer, appended to the
discriminator. We adopt the ReLU function layer appended to each of the six layers in the generator and
discriminator networks. The generator network has two output neurons, while the discriminator network
has one output. The input data of the generator is a random noise sampled from a standard Gaussian
distribution. The output of the generator is used as the input for the discriminator. The output of the
discriminator is used to evaluate the quality of the generated points by the generator.

**Experimental environments.** We deploy the mixture Gaussians experiments on the computer with **CPU**
AMD Ryzen 7 3700, **GPU** RTX 2060, **6GB RAM**, Python (version 3.6.7), Keras (version 2.3.1), TensorFlow
(version 1.13.1), PyTorch (version 1.3.1). We conducted each of the compared methods with 10,000
iterations.

We conducted the experiments on the mixture of 5 Gaussians with the proposed ACG method and
several other methods. The training set of the all compared algorithms are as follows:

- **RMSP**: We employ the TensorFlow to provide Simultaneous RMSPropOptimizer and set the learning
  rate $\alpha = 5 \times 10^{-4}$.
- **RMSP-alt**: The alternating RMSPropOptimizer realized by TensorFlow with learning rate $\alpha = 5 \times 10^{-4}$.
- **ConOpt [40]**: The Consensus Optimizer realized by TensorFlow with $h = 1 \times 10^{-4}$, $\gamma = 1$.
- **RMSP-SGA [32]**: The Symplectic Gradient Adjusted RMSPropOptimizer realized by TensorFlow with
  learning rate $\alpha = 1 \times 10^{-4}$, $\xi = 1$.
- **RMSP-ACA [15]**: The Alternating Centripetal Acceleration on RMSPropOptimizer realized by TensorFlow
  with learning rate $\alpha = 5 \times 10^{-4}$, $\beta = 0.5$.
- **SGA-ACG(ours)**: Our proposed ACG method in Algorithm 1 on the SGA Optimizer realized by PyTorch
  with learning rate $\alpha = 5 \times 10^{-4}$, $\beta_1 = 5 \times 10^{-7}$, $\beta_2 = \alpha$.

The mixture of 5 Gaussians numerical simulation results are shown in Figure 9. Figure 9 shows the
ConOpt, RMSP-SGA, SGA-ACG algorithms all converge. Meanwhile, the generated mixture of 5 Gaussians
is almost approaching the ground truth in Figure B1. It is observed that they have the same convergence
speed among ConOpt, RMSP-SGA, SGA-ACG (Ours). We also compared these methods with other SOTA
methods, such as RMSP, RMSP-alt, RMSP-ACA. The compared simulation results are shown in Figure B2.
To compare the convergence speed among all six algorithms, we employ the same training settings as the
mixture of 5 Gaussians to conduct the mixture of 16 Gaussians as shown in Figure 10.

From Figure 10, it is obvious that our proposed SGA-ACG method converges faster than ConOpt,
RMSP-SGA. More comparison results are shown in Figure B3. From Figure B3, RMSP, RMSP-alt, RMSP-ACA
still do not converge after 10,000 iterations. To compare the convergence speed, Figure 11 shows the time-
consuming of all the compared methods.
There is a parameter $s$ in our proposed algorithms in Algorithm 1 and Algorithm 2. We have explored the influence of $s$ on final results and conducted a series of experiments as the $s$ through \{50, 100, 150, ...

**Figure 9.** Compared results on the mixture of 5 Gaussians. Each row suggests a different method, and each column is the results at other iteration numbers through \{2000, 4000, 6000, 8000, 10000\}.

**Figure 10.** Compared results on the mixture of 16 Gaussians. Each row represents a kind of algorithms, and the columns are each algorithm in 2000, 4000, 6000, 8000, 10000 iterations, respectively.
200) on the mixture of 16 Gaussians, which is as shown in Figure 12. It shows that the proposed SGA-ACG method converges faster with the $s$ increasing.

**Figure 11.** The time consuming of compared methods on the mixture of 16 Gaussians in Figure B3. Our proposed method SGA-ACG takes more time than RMSProp, RMSProp-alt, and RMSProp-ACA. However, it takes less time than ConOpt and RMSProp-SGA methods.

**Figure 12.** Exploring of $s$, the mixture of 16 Gaussians. Each row shows results with $s$ at different values, and then each column shows the results with iteration number through [2000, 4000, 6000, 8000, 10000].
6.3 Experiments on Prevalent Datasets

This section conducts the third experiment that tested our proposed ACG method on image generation tasks. We employ four prevalent datasets to illustrate our ACG method can be applied in deep learning. We choose the standard MNIST [41], Fashion-MNIST [42], CIFAR-10 [43], CelebA [44] datasets to conduct realistic experiments.

**Network Architecture.** We choose two kinds of network architectures for GANs on the MNIST dataset. For the first kind of network structure, we employ 2 fully connected layers with 256 and 512 neurons for the generator network. Each of the 2 layers is appended to a LeakyReLU layer with $\alpha = 0.2$. We adopt a Tanh activation function layer as the last layer in the generator network. The input data for the generator is a random noise with dimensions 100 sampled from a standard Gaussian distribution. The output of the generator is an image with shape $28 \times 28 \times 1$. For the discriminator network, we also use 2 fully connected layers with 512 and 256 neurons. After each layer, there is appended to a LeakyReLU layer with $\alpha = 0.2$, which is the same as the generator network. However, in the last layer of the discriminator, we adopt a Sigmoid activation function. The input data of the discriminator includes the generated image and the ground truth image on the MNIST dataset. The output of the discriminator is used to evaluate the quality of the image made by the generator network. For the second kind of network structure, we used the architecture of DCGANs [45], we just used 4 layers of DCGANs [45] for both generator network and discriminator network.

**Experimental environments.** We conduct experiments of this section on a server equipped with CPU E5-2698, 4*GPU GTX 3090 aero, 24GB RAM, Python (version 3.6.13), PyTorch (version 1.8.0). We realized the all compared algorithms by the PyTorch, and the training set of these algorithms on the four datasets are as follows:

- **SGD:** The learning rate of linear GANs on MNIST dataset is $\alpha = 2 \times 10^{-4}$, excepted for the learning rate of DCGANs on MNIST is $\alpha = 5 \times 10^{-4}$, Fashion–MNIST (learning rate $\alpha = 2 \times 10^{-4}$), CIFAR–10 (learning rate $\alpha = 2 \times 10^{-4}$), CelebA (learning rate $\alpha = 2 \times 10^{-4}$).
- **Adam:** Linear GANs on MNIST (learning rate $\alpha = 3 \times 10^{-4}$), DCGANs on MNIST (learning rate $\alpha = 2 \times 10^{-4}$), Fashion–MNIST (learning rate $\alpha = 2 \times 10^{-4}$), CIFAR–10 (learning rate $\alpha = 2 \times 10^{-4}$), CelebA (learning rate $\alpha = 2 \times 10^{-4}$).
- **RMSP:** Linear GANs on MNIST (learning rate $\alpha = 2 \times 10^{-4}$), DCGANs on MNIST (learning rate $\alpha = 5 \times 10^{-4}$), Fashion–MNIST (learning rate $\alpha = 5 \times 10^{-4}$), CIFAR–10 (learning rate $\alpha = 5 \times 10^{-4}$), CelebA (learning rate $\alpha = 5 \times 10^{-4}$).
- **RMSP–ACG:** Only on the linear GANs (learning rate $\alpha = 5 \times 10^{-4}$, $\beta_1 = 5 \times 10^{-7}$, $\beta_2 = \alpha$).
- **Adam–ACG:** On the all datasets, our proposed Adam–ACG method applied in linear GANs and DCGANs with learning rate $\alpha = 5 \times 10^{-4}$, $\beta_1 = 5 \times 10^{-7}$, $\beta_2 = \alpha$.

In the experiment of linear GANs on MNIST data, we set the batch size as 64, and the epoch number is 324. The generation results of our proposed methods are shown in Figure 13. More comparisons among these algorithms on MNIST are shown in Figure B4.
For DCGANs experiments, the batch size is 64, and the epoch number is 110 on the MNIST dataset. The same batch size and epoch number are assigned to the Fashion-MNIST and CIFAR-10 data. In contrast, the batch size and number of epoch on CelebA data set are 128 and 70, respectively. The results of our methods on the four datasets are shown in Figure 14, and more comparisons results among several algorithms on the same datasets are shown in Figure B5.

Figure 13. Compared results of Linear GANs on MNIST dataset. The first and second rows are the results of RMSP-ACG, Adam-ACG. The first, second, third, and fourth columns are the results in 50000, 150000, 250000, and 300000 iterations, respectively.

Figure 14. Comparison of DCGAN for our proposed method on the four datasets. The first column is the results of the MNIST dataset in 100000 iterations. The second column is the results of the Fashion-MNIST dataset in 100000 iterations. The third column is the results of the CIFAR10 dataset in 80000 iterations, and the last column is the results of CelebA dataset in 100000 iterations.

7. CONCLUSION

We proposed the Adaptive Composite Gradients (ACG) method to find a local Nash Equilibrium in game problems. The ACG algorithm can alleviate the cyclic behaviors. It has robustness and can be easily inte-
Training Generative Adversarial Networks with Adaptive Composite Gradient

integrated with SGD, Adam, RMSP, SGA, and other gradient-based optimizers. Since the ACG method employs the predicted information in future s iterations, it is a novel semi-gradient-free algorithm. Our algorithm has a linear convergence rate. Furthermore, we offer that the SGA-ACG can be competitive to ConOpt and SGA methods on the mixture of Gaussians generated tasks. We prove the ACG can be applied in a general zero-sum game with n players by toy function experiment. The extensive generative image experiments show our method can optimize the generic deep learning networks model. However, our research objectives are limited to the convex and smoothness of simple zero-sum games. The non-convex and non-smooth games are more complex and challenging to find a local Nash Equilibrium. Therefore, optimizing and finding local solutions for non-convex and non-smooth games is still a challenging task worth researching in the future.

ACKNOWLEDGEMENT

This work is supported by the National Key Research and Development Program of China (No.2018AAA0101001), Science and Technology Commission of Shanghai Municipality (No.20511100200), and supported in part by the Science and Technology Commission of Shanghai Municipality (No.18dz2271000).

AUTHOR CONTRIBUTION STATEMENT

Conceptualization, methodology, algorithm designing, coding, original draft preparation and survey references: Huiqing Qi. Methodology, data analyses, manuscript review, funding acquisition, original draft preparation and manuscript revising: Fang Li. Methodology, data analyses, manuscript review and funding acquisition: Shengli Tan. Methodology, algorithm designing, manuscript review, funding acquisition: Xiangyun Zhang. All authors have read and agreed to the published version of the manuscript.

REFERENCES

[1] Maleknia M, Shamsi M.: A quasi-Newton proximal bundle method using gradient sampling technique for minimizing nonsmooth convex functions. Optimization Methods and Software, 37(4), 1415–1446 (2022)
[2] Li K, Zhang T, Wang R.: Deep reinforcement learning for multiobjective optimization. IEEE transactions on cybernetics, 51(6), 3103–3114 (2020)
[3] Vezhnevets A S, Osindero S, Schaul T, et al.: Feudal networks for hierarchical reinforcement learning. In: International conference on machine learning, pp. 3540–3549 (2017)
[4] Goodfellow I, Pouget-Abadie J, Mirza M, et al.: Generative adversarial networks. Communications of the ACM, 63(11), 139–144 (2020)
[5] Hong Y, Hwang U, Yoo J, et al.: How generative adversarial networks and their variants work: An overview. ACM Computing Surveys (CSUR), 52(1), 1–43 (2019)
[6] Guimaraes G L, Sanchez-Lengeling B, Outeiral C, et al.: Objective-reinforced generative adversarial networks (organ) for sequence generation models. arXiv preprint arXiv:1705.10843 (2017)
[7] Lee S, Hwang U, Min S, et al.: Polyphonic music generation with sequence generative adversarial networks. arXiv preprint arXiv:1710.11418 (2017)
[8] Yu L, Zhang W, Wang J, et al.: SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. arXiv preprint arXiv:1609.05473 (2016)

[9] Hsu C C, Hwang H T, Wu Y C, et al.: Voice conversion from unaligned corpora using variational autoencoding wasserstein generative adversarial networks. arXiv preprint arXiv:1704.00849 (2017)

[10] Lin K, Li D, He X, et al.: Adversarial ranking for language generation. Advances in neural information processing systems, 30 (2017)

[11] Akmal Haidar M, Rezagholizadeh M.: TextKD-GAN: Text Generation using KnowledgeDistillation and Generative Adversarial Networks. arXiv preprint arXiv: 1905.01976 (2019)

[12] Croce D, Castellucci G, Basili R.: GAN-BERT: Generative adversarial learning for robust text classification with a bunch of labeled examples. In: Proceedings of the 58th annual meeting of the association for computational linguistics, pp. 2114–2119 (2020)

[13] Neyshabur B, Bhojanapalli S, Chakrabarti A.: Stabilizing GAN training with multiple random projections. arXiv preprint arXiv:1705.07831 (2017)

[14] Qin C,Wu Y, Springenberg J T, et al.: Training generative adversarial networks by solving ordinary di erential equations. Advances in Neural Information Processing Systems, 33, 5599–5609 (2020)

[15] Peng W, Dai Y H, Zhang H, et al.: Training GANs with centripetal acceleration. Optimization Methods and Software, 35(5), 955–973 (2020)

[16] Keke L, Ke Z, Qiang L, et al.: Training GANS with predictive projection centripetal acceleration. arXiv preprint arXiv: 2010.03322 (2020)

[17] Ledig C, Theis L, Huszár F, et al.: Photo-realistic single image super-resolution using a generative adversarial network. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4681–4690 (2017)

[18] Wu J, Zhang C, Xue T, et al.: Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. Advances in neural information processing systems, 29 (2016)

[19] Zhu J Y, Park T, Isola P, et al.: Unpaired image-to-image translation using cycleconsistent adversarial networks. In: Proceedings of the IEEE international conference on computer vision, 2223–2232 (2017)

[20] Wang C, Xu C, Wang C, et al.: Perceptual adversarial networks for image-to-image transformation. IEEE Transactions on Image Processing, 27(8), 4066–4079 (2018)

[21] Walker J, Marino K, Gupta A, et al.: The pose knows: Video forecasting by generating pose futures. In: Proceedings of the IEEE international conference on computer vision, pp. 3332–3341 (2017)

[22] Tulyakov S, Liu M Y, Yang X, et al.: Mocogan: Decomposing motion and content for video generation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1526–1535 (2018)

[23] Yue Z, Zhao Q, Zhang L, et al.: Dual Adversarial Network: Toward Real-World Noise Removal and Noise Generation. In: European Conference on Computer Vision, pp. 41–58 (2020)

[24] Kupyn O, Budzan V, Mykhailych M, et al.: DeblurGAN: Blind motion deblurring using conditional adversarial networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 8183–8192 (2018)

[25] Kim T, Cha M, Kim H, et al.: Learning to discover cross-domain relations with generative adversarial networks. In: International conference on machine learning, pp. 1857–1865 (2017)

[26] Yi Z, Zhang H, Tan P, et al.: Dualgan: Unsupervised dual learning for image-to-image translation. In: Proceedings of the IEEE international conference on computer vision, pp. 2849–2857 (2017)

[27] Yang D, Xiong T, Xu D, et al.: Automatic vertebra labeling in large-scale 3D CT using deep image-to-image network with message passing and sparsity regularization. In: 25th International Conference on Information Processing in Medical Imaging, pp. 633–644 (2017)
[28] Wang Y.: A mathematical introduction to generative adversarial nets (GAN). arXiv preprint arXiv:2009.00169 (2020)
[29] Odena A.: Open questions about generative adversarial networks. Distill, 4(4), 18 (2019)
[30] Goodfellow I.: Nips 2016 tutorial: Generative adversarial networks. arXiv preprint arXiv:1701.00160 (2016)
[31] Poon C, Liang J.: Trajectory of alternating direction method of multipliers and adaptive acceleration. Advances in Neural Information Processing Systems, 32 (2019)
[32] Balduzzi D, Racaniere S, Martens J, et al.: The mechanics of n-player differentiable games. In: International Conference on Machine Learning, pp. 354–363 (2018)
[33] Nowozin S, Cseke B, Tomioka R.: f-gan: Training generative neural samplers using variational divergence minimization. Advances in neural information processing systems, 29 (2016)
[34] Liang T, Stokes J.: Interaction matters: A note on non-asymptotic local convergence of generative adversarial networks. In: The 22nd International Conference on Artificial Intelligence and Statistics, pp. 907–915 (2019)
[35] Bauschke H H, Combettes P L.: Convex Analysis and Monotone Operator Theory in Hilbert Spaces. corrected printing, (1997)
[36] Bruck R E.: On the weak convergence of an ergodic iteration for the solution of variational inequalities for monotone operators in Hilbert space. Journal of Mathematical Analysis and Applications, 61(1), 159–164 (1977)
[37] Nedic A, Ozdaglar A.: Subgradient methods for saddle-point problems. Journal of optimization theory and applications, 142, 205–228 (2009)
[38] Nesterov, Yuri.: A method for solving the convex programming problem with convergence rate O(1/k²). In: Proceedings of the USSR Academy of Sciences, 269, pp. 543–547 (1983)
[39] Yang T, Lin Q, Li Z.: Unified convergence analysis of stochastic momentum methods for convex and non-convex optimization. arXiv preprint arXiv:1604.03257 (2016)
[40] Mescheder L, Nowozin S, Geiger A.: The numerics of gans. Advances in neural information processing systems, 30 (2017)
[41] LeCun Y, Bottou L, Bengio Y, et al.: Gradient-based learning applied to document recognition. In: Proceedings of the IEEE, 86(11), pp. 2278–2324 (1998)
[42] Xiao H, Rasul K, Vollgraf R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017)
[43] Krizhevsky A, Hinton G.: Convolutional deep belief networks on cifar-10. Unpublished manuscript, 40(7), 1–9 (2010)
[44] Liu Z, Luo P, Wang X, et al.: Deep learning face attributes in the wild. In: Proceedings of the IEEE international conference on computer vision, pp. 3730–3738 (2015)
[45] Radford A, Metz L, Chintala S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015)
AUTHOR BIOGRAPHY

**Huiqing Qi** received the M.S. degree from Chongqing University, Chongqing, China, in 2019. He is currently pursuing the Ph.D. degree in the School of Mathematical Science, East China Normal University, Shanghai, China.
His main research interests are image processing, machine learning, and deep learning.

**Fang Li** received the Ph.D. degree from East China Normal University, Shanghai, China, in 2007. She is currently a Professor with the School of Mathematical Science, East China Normal University, Shanghai, China.
Her main research interests include image processing, ROF model, machine learning and deep learning.

**Shengli Tan** received the Ph.D. degree from East China Normal University, Shanghai, China, in 1991. He is currently a Professor with the School of Mathematical Science, East China Normal University, Shanghai, China. He is also a Cheung Kong Chair Professor and the editor of JP journal of Algebra and Number Theory.
His main research interests include moduli space of cuspidal curves on an algebraic surface, rank two vector bundles and binary cubic forms over a ring and linear systems on algebraic surfaces.

**Shengli Tan** received the Ph.D. degree from East China Normal University, Shanghai, China, in 2007. He is currently an Associate Professor with the School of Mathematical Science, East China Normal University, Shanghai, China.
His main research interests include parallel computing, machine learning, high-dimensional statistics, low-rank approximation of tensor and its application and deep learning.
8. APPENDICES

APPENDIX A. PROOFS IN SECTION 5

A.1 Proof of Proposition 5.3

Proof. Without loss of generality, let \( \hat{a} = (a_1, a_2, a_3, \ldots, a_n), \hat{b} = (b_1, b_2, b_3, \ldots, b_n) \), where \( n \geq 2, n \in \mathbb{N}^* \). Then, we have

\[
\cos \langle \hat{a}, \hat{b} \rangle = \frac{\hat{a} \cdot \hat{b}}{|\hat{a}| |\hat{b}|}.
\]  

(A1)

The projection \( \bar{\rho} \) of \( \hat{b} \) onto \( \hat{a} \) can be written as

\[
\bar{\rho} = |\hat{b}| \cos \langle \hat{a}, \hat{b} \rangle \frac{\hat{a}}{|\hat{a}|}.
\]  

(A2)

Incorporating 65 into 66, we have

\[
\bar{\rho} = |\hat{b}| \frac{\hat{a} \cdot \hat{b}}{|\hat{a}| |\hat{b}|} = \frac{\hat{a} \cdot \hat{b}}{||\hat{a}||^2} \hat{a}.
\]  

(A3)

Using the \( \gamma \) to replace \( \frac{\hat{a} \cdot \hat{b}}{||\hat{a}||^2} \), we can obtain \( \bar{\rho} = \gamma \hat{a} \), where the \( \gamma \in \mathbb{R} \).

A.2 Proof of Proposition 5.4

Proof. The characteristic polynomial of the matrix (39) is

\[
\begin{vmatrix}
(1 - \lambda)I_d & -(\alpha + \beta_1)A & 0 & \beta_1(1 + \gamma)A & -\beta_1J_d & 0 \\
(\alpha + \beta_1)A^T & (1 - \lambda)I_d & -\beta_1(1 + \gamma)A^T & 0 & 0 & \beta_1J_d \\
0 & 0 & -\lambda I_d & 0 & 0 & 0 \\
0 & I_d & 0 & -\lambda I_d & 0 & 0 \\
0 & 0 & 0 & 0 & (\tau - \lambda)I_d & 0 \\
0 & 0 & 0 & 0 & 0 & (\tau - \lambda)I_d
\end{vmatrix},
\]  

(A4)

which is equivalent to

\[
(\tau - \lambda)^2 \cdot \det
\begin{vmatrix}
(1 - \lambda)I_d & -(\alpha + \beta_1)A & 0 & \beta_1(1 + \gamma)A \\
(\alpha + \beta_1)A^T & (1 - \lambda)I_d & -\beta_1(1 + \gamma)A^T & 0 \\
0 & 0 & -\lambda I_d & 0 \\
0 & I_d & 0 & -\lambda I_d
\end{vmatrix}.
\]  

(A5)
From (A5) we can derive to

\[
(\tau - \lambda)^2 \cdot \begin{pmatrix}
\lambda(1-\lambda)l_d & -\lambda(\alpha + \beta_i)A + \beta_i(1+\gamma)A^T \\
\lambda(\alpha + \beta_i)A^T - \beta_i(1+\gamma)A^T & \lambda(1-\lambda)l_d
\end{pmatrix},
\]

(A6)

According to (A6), 0 and 1 cannot be its roots based on \(A\) is nonsingular and square. Eq. (A6) is equivalent to

\[
\det((\tau - \lambda)^2[\lambda^2(1-\lambda)^2 + (\lambda(\alpha + \beta_i) - \beta_i(1+\gamma))^2A^TA])
\]

(A7)

Then, we can obtain that the eigenvalues of \(F\) are the roots of the sixth order polynomials:

\[
(\tau - \lambda)^2[\lambda^2(1-\lambda)^2 + (\lambda(\alpha + \beta_i) - \beta_i(1+\gamma))^2\xi^2], \quad \xi^2 \in Sp(A^T A).
\]

(A8)

### A.3 Proof of Proposition 5.5

**Proof.** Let the characteristic polynomial of the matrix (39) to be 0, which is written as follows:

\[
(\tau - \lambda)^2[\lambda^2(1-\lambda)^2 + (\lambda(\alpha + \beta_i) - \beta_i(1+\gamma))^2\xi^2] = 0,
\]

\[
\xi^2 \in Sp(A^T A).
\]

(A9)

It is obvious that (A9) have 6 roots, and \(\lambda_1 = \lambda_2 = \tau\) are two of the 6 roots. According to the convergence of formula (35), we can obtain the \(\tau\) is almost small and \(|\tau| < 1\). We mainly discuss the following polynomial:

\[
[\lambda^2(1-\lambda)^2 + (\lambda(\alpha + \beta_i) - \beta_i(1+\gamma))^2\xi^2] = 0.
\]

(A10)

Using Proposition (5.4), we have

\[
(\lambda^2 - \lambda - i[\lambda(\alpha + \beta_i) - \beta_i(1+\gamma)]\xi)(\lambda^2 - \lambda + i[\lambda(\alpha + \beta_i) - \beta_i(1+\gamma)]\xi) = 0
\]

(A11)

Denote \(a : = \alpha + \beta_i\) and \(b : = \beta_i(1+\gamma)\), then (A11) can be written as:

\[
[\lambda^2 - \lambda - i(\lambda a - b)\xi][\lambda^2 - \lambda + i(\lambda a - b)\xi] = 0.
\]

(A12)

we can get the four roots of (A12) are

\[
\lambda_1^\pm = \frac{1-ia\xi \pm \sqrt{1-a^2\xi^2 - 2ia\xi + 4ib\xi}}{2},
\]

\[
\lambda_2^\pm = \frac{1+ia\xi \pm \sqrt{1-a^2\xi^2 + 2ia\xi - 4ib\xi}}{2}.
\]

(A13)

Let \(u : = a\xi + b\xi\) and \(v : = a\xi - b\xi\), then we can obtain
\[
\lambda_1^\pm = \frac{1 - \frac{u + v}{2} i \pm \sqrt{1 - \left(\frac{u + v}{2}\right)^2 - \left(3\nu - u\right)i}}{2},
\]
\[
\lambda_2^\pm = \frac{1 + \frac{u + v}{2} i \pm \sqrt{1 - \left(\frac{u + v}{2}\right)^2 + \left(3\nu - u\right)i}}{2}.
\] (A14)

Denote \( s := \frac{u + v}{2} \) and \( t := \frac{3\nu - u}{2} \), then we have
\[
\lambda_1^\pm = \frac{1 - si \pm \sqrt{1 - 2ti - s^2}}{2}; \quad \lambda_2^\pm = \frac{1 + si \pm \sqrt{1 + 2ti - s^2}}{2}.
\] (A15)

The following proof process is the same as (A.2) in [29]. For a given complex number \( z \), we can obtain the absolute value of the real part in \( z \) is \( \sqrt{\frac{|z| + R(z)}{2}} \) and the absolute value of the imaginary part in \( z \) is \( \sqrt{\frac{|z| - R(z)}{2}} \). However, According to this Proposition \( s \leq 1 \), all the real parts of the four roots lie in the interval \([-S, S]\), where
\[
S = \frac{1}{2} \sqrt{\frac{(1-s)^2 + 4t^2 + 1-s^2}{2} + \frac{1}{2}}.
\] (A16)

all the imaginary parts of roots lie in the interval \([-T, T]\), where
\[
T = \frac{1}{2} \sqrt{\frac{(1-s)^2 + 4t^2 - 1+s^2}{2} + \frac{s}{2}}.
\] (A17)

Using the Inequality \( \sqrt{x} + y \leq \sqrt{x} + \frac{y}{2\sqrt{x}} \), \((x > 0, y > 0)\), we can obtain
\[
S \leq \frac{1}{2} \sqrt{1-s^2 + \frac{t^2}{1-s^2} + \frac{1}{2}}.
\] (A18)
\[
T \leq \frac{s}{2} + \frac{|t|}{2\sqrt{1-s^2}}.
\] (A19)

Then, we analyze the \( s \) in \((0, \frac{1}{\sqrt{2}}]\) and \((\frac{1}{\sqrt{2}}, 1]\) two cases separately.

**Case 1**, we suppose \( 0 < s \leq \frac{1}{\sqrt{2}} \), According to this proposition \( |\alpha + \beta| + |2\beta_i(1 + \gamma)| / (\alpha + \beta)^2 \leq 0.1\xi \), for all \( \xi \in \mathcal{S}(A^TA) \), we have \(|t| \leq \frac{s^2}{10}\). Then, based on \( \frac{s^2}{2} \leq 1 - \sqrt{1-s^2} \), we have
\[
|t| \leq \frac{1 - \sqrt{1-s^2}}{5}.
\] (A20)
Integrating $s \leq \frac{1}{\sqrt{2}}$ with (A20), we can obtain

$$|t| \leq \frac{2(1-\sqrt{1-s^2})(1-s^2)}{5} \leq \frac{(1-\sqrt{1-s^2})(1-s^2)}{2\sqrt{1-s^2} + \frac{1}{2}},$$

which follows that

$$1 \geq \frac{|t|}{\sqrt{1-s^2}} + \frac{|t|}{\sqrt{1-s^2} + \frac{s}{\sqrt{1-s^2}}} \quad \text{(A21)}$$

$$\geq \frac{t^2}{1-s^2} + \frac{t^2}{1-s^2 + \frac{s}{\sqrt{1-s^2}}} + \frac{s|t|}{\sqrt{1-s^2}} \quad \text{(A22)}$$

$$\geq \frac{t^2}{1-s^2} + \frac{t^2}{1-s^2} + \frac{s|t|}{\sqrt{1-s^2}}. \quad \text{(A23)}$$

The inequality (A22) follows by the fact that $\frac{|t|}{\sqrt{1-s^2}} \leq \frac{s}{\sqrt{1-s^2}} \leq 1$ and the inequality (A23) uses $\sqrt{a+b} \leq \sqrt{a} + \frac{b}{2\sqrt{a}}$. The (A21 - A23) can be written equivalently to

$$\left(\frac{1}{2} \sqrt{1-s^2} + \frac{t^2}{1-s^2} + \frac{1}{2} \right)^2 + \frac{s^2}{2} + \frac{|t|}{2\sqrt{1-s^2}} \leq 1. \quad \text{(A24)}$$

According (A18) and (A19), we have

$$\rho(F) \leq \sqrt{S^2 + T^2} \leq 1. \quad \text{(A25)}$$

It is worth noting that $\sqrt{x+y} \leq \sqrt{x} + \frac{y}{2\sqrt{x}}$ holds equality if and only if $y = 0$. Then, the (A25) holds equality when $t = 0$ and $s = 0$. Since $s > 0$, we have the strict inequality $\rho(F) \leq 1$ which suggests for the linear convergence of unit time $\Delta_t$.

**Case 2.** we suppose $\frac{1}{\sqrt{2}} < s \leq 1$, since $t \leq \frac{s^2}{10} \leq 0.1$. Combining (A16) and (A17) directly, we can obtain

$$\rho(F) \leq \sqrt{S^2 + T^2} < 1. \quad \text{(A26)}$$

which is also linear convergence.
APPENDIX B. THE APPENDIX FIGURES OF EXPERIMENTS

This section shows more figures. There are two ground-truth in Figure B1. Figure B2 is the comparison results of our proposed method and other SOTA algorithms on the mixture of 5 Gaussians experiments. Figure B3 is the results of the compared methods on the mixture of 16 Gaussians experiments. Figure B4 is the experimental results of compared methods with linear GANs on the MNIST dataset. Figure B5 is the experimental results of compared methods with DCGANs on the four Datasets (MNIST, Fashion-MNIST, CIFAR10, and CelebA).

![Figure B1](image)

(a) A mixture of 5 Gaussians  
(b) A mixture of 16 Gaussians

**Figure B1.** Ground truth of Mixture Gaussians.
Figure B2. Compared results on the mixture of 5 Gaussians. Each row suggests a different method, and each column is the results at other iteration numbers through (2000, 4000, 6000, 8000, 10000). Although from the figure, we can obtain that RMSP, RMSP-alt, RMSP-ACA can not converge to the ground truth, and ConOpt and RMSP-SGA, SGA-ACG all converge to the ground truth, our proposed method seems competitive to ConOpt and RMSP-SGA.
Figure B3. Compared results on the mixture of 16 Gaussians. Each row suggests a different method, and then each column is the results at different iteration numbers through {2000, 4000, 6000, 8000, 10000}. The figure shows that RMSP, RMSP-alt, RMSP-ACA can not converge to the ground truth, and ConOpt and RMSP-SGA, SGA-ACG all converge to the ground truth. However, our methods converge faster than ConOpt and RMSP-SGA at iteration 2000, which seems competitive to ConOpt and RMSP-SGA.
Figure B4. Compared results of Linear GANs on MNIST dataset. Each row suggests a different compared method, and each column is the results of iteration number through {50000, 150000, 250000, 300000}. This figure shows SGD and Adam cannot generate correct handwritten numbers. While RMSP and RMSP-ACG (ours), Adam-ACG (ours) can generate the handwritten digits. However, all the compared methods, including our proposed, are faced with the mode collapse problem.
Figure B5. Comparison of DCGANs for several Algorithms on the four datasets. The first, second, third, and fourth rows are the results of SGD, RMSP, Adam, Adam-ACG(ours) on the four datasets. The first, second, third, and fourth columns are the results of the MNIST, Fashion-MNIST, CIFAR10, and CelebA datasets, respectively. We conduct 100000 iterations for all experiments on the MNIST dataset, 100000 iterations for all experiments on the Fashion-MNIST dataset, 80000 iterations for all experiments on the CIFAR10 dataset, and 100000 iterations for all experiments on the CelebA dataset, respectively. Thus, the SGD method is invalid on the MNIST and Fashion-MNIST datasets experiments. In contrast, SGD is valid on CIFAR10 and CelebA datasets experiments. Significantly, the RMSP method is invalid on the four datasets experiments. From this figure, our proposed method seems competitive to Adam on all four datasets experiments.