Extending Convergence Studies of GAN Training Algorithms with Duality Gap

Sandeep Kumar Swain[a], Mainak Bandyopadhyay[b]

[a]*School of Computer Engineering, Kalinga Institute of Industrial technology, Bhubaneswar, Odisha, India*
[b]*School of Computer Engineering, Kalinga Institute of Industrial technology, Bhubaneswar, Odisha, India*

Abstract

This paper studies the training and convergence of Generative Adversarial Networks (GANs) by applying the concept of duality gap. With many algorithms present for training and convergence of GANs , still there is scope of improving the convergence rate by focusing on the integration of the duality gap concept to enhance the convergence rates of several GAN algorithms. The duality gap, a key notion in convex optimization, measures the discrepancy between primal and dual objective values. This paper investigate applicability of duality gap in enhancing convergence of GAN algorithms such as Gradient Descent Ascent (GDA), Optimistic GDA, Extragradient, Symplectic Gradient Adjustment, Consensus Optimization, and Follow the Ridge. Our primary objective is to elevate the performance of these algorithms and gain valuable insights into their convergence behavior. Through a systematic analysis of experimental results, this paper demonstrate the effectiveness of incorporating the duality gap in improving the convergence rates of these GAN optimization algorithms. This research contributes to advancing the field of GAN optimization, paving the way for further research and practical implementations in a wide range of applications.

Keywords: Generative Adversarial Networks (GANs), Duality Gap, GAN Optimization Algorithms, Loss Function, Generator, Discriminator.

1. Introduction

Generative Adversarial Networks (GANs) have become increasingly popular as advanced models for creating realistic and high-quality synthetic data [1]. The core concept of GANs is to train the generator network to create synthetic data samples, using random noise as input such that they closely resemble real data and are nearly impossible to tell apart from genuine data. Concurrently, a discriminator network is trained to learn the ability to discern between the generated data and real data. The interplay between these two networks gives rise to an adversarial learning process, wherein both networks continuously enhance their respective capabilities through an iterative training process in which the generator and discriminator networks learn from each other, resulting in improved quality of the generated data.

GAN training entails addressing a non-convex optimization problem that encompasses multiple objectives. Optimizing GANs presents several challenges such as mode collapse (where the generator fails to explore the entire data space), instability, and slow convergence. GANs are trained using a min-max game framework, where the objectives of the generator and discriminator are in direct competition. This gives rise to a non-convex optimization problem with multiple equilibrium points and potentially unstable training dynamics. Mode collapse occur when generator in a GAN gets stuck generating a repetitive subset of samples, missing out on capturing the entire range of patterns and diversity present in the real data distribution. Additionally, due to adversarial nature of training, it typically exhibits slow convergence, necessitating careful tuning of hyper-parameters and architectural choices.

To solve the challenges various optimization algorithms like Gradient Descent Ascent (GDA), Optimistic GDA, Extragradient, Symplectic Gradient Adjustment, Consensus Optimization, and Follow the Ridge are specifically designed for GANs have been proposed. These algorithms aims to enhance the speed of convergence, stability, and generation quality in GAN training [2, 3].

The main focus of this paper is to further extend the studies carried out by [4, 5, 6, 7] on duality gap for convergence and minimizing competition between generator and discriminator for enhancing the capabilities of optimization algorithms for Generative Adversarial Networks (GANs). By incorporating duality gap, our aim is to study the effects on the convergence rates, stability, and generation quality of GANs. In particular, this paper focus on some specific optimization algorithms, namely Gradient Descent Ascent (GDA), Optimistic GDA, Extragradient, Symplectic Gradient Adjustment, Consensus Optimization, and Follow the Ridge. Through extensive experimental evaluations, the objective is to evaluate the capabilities of these algorithms when augmented with duality gap techniques, compare their performance, and gain deeper insights into their convergence behaviour.

Email addresses: 20051025@kiit.ac.in (Sandeep Kumar Swain), mainak.bandyopadhyayfcs@kiit.ac.in (Mainak Bandyopadhyay)

2. Related Background

2.1. GANs and it's training

Generative Adversarial Networks (GANs) have gained significant popularity as a class of deep learning models capable of generating synthetic data that closely resembles real data samples. They have demonstrated promising outcomes in various domains, including image synthesis, video prediction, and text generation. GANs are built upon the fundamental concept of training two neural networks, the generator and the discriminator, against each other in a competitive game.

In this game, the generator network creates synthetic data from random noise, while the discriminator network tries to indicate the authenticity of the data from both real and fake data sample. The generator aims to generate convincing data samples, while the discriminator aims to become better at distinguishing real from generated data. This competition drives the GAN's training and improves the quality of the generated data over time. Therefore, the GAN game objective is:

$$\min_G \max_D M(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D(\boldsymbol{x})] + \mathbb{E}_{z\sim p_z}[\log(1-D(G(z)))] \tag{1}$$

Through iterative training, the parameters weights of generator and discriminator networks are updated continually,thus learning from each other's feedback. This adversarial learning process drives the generator to produce increasingly convincing synthetic data, approaching the distribution of the real data.

2.2. Training Problems

Training Generative Adversarial Networks (GANs) is quite challenging because it is difficult to reach the Nash-Equilibrium state [8]. Predominantly, parameter weights corresponding to non-Nash Equilibrium states that provides satisfactory results are accepted. The training of GANs often encounters mode collapse which occurs when the generator instead of producing new and wide range of diverse and high-quality samples, ends up generating a limited and repetitive set of samples. It can occur if the generator learns too quickly compared to the discriminator, leading to an imbalanced optimization process.

On the other hand, if the discriminator learns too fast compared to the generator, it can become too effective at distinguishing real and generated samples, resulting in the generator receiving a high loss and encountering the vanishing gradient problem. This means the generator struggles to learn effectively, as the gradient updates during training become close to zero due to the discriminator's strong discriminatory power. The Wasserstein GAN is one method that addresses this problem.

If the generator and discriminator in a GAN have restricted options for their strategies, it may no longer be ensured that the GAN will reach a single global equilibrium point in its game. This is due to the constrained strategies that the generator and discriminator can employ. Finding an equilibrium in such cases becomes challenging. Instead of reaching a state of equilibrium, they may become trapped in non-Nash equilibrium, where neither side can improve further. This can hinder the overall convergence of the GAN model.

Furthermore, the training process of GANs requires careful tuning of various hyper-parameters, such as the learning rate and architecture. Finding the right combination of hyper-parameters can be a time-consuming and computationally expensive task, as it often involves iterative experimentation and fine-tuning to achieve optimal performance.

2.3. Nature of the Objective Functions

There are two notable points related to optimization of loss function in GANs:-

- The non-convex nature of the loss function.
- The inherent instability of the training process.

Training GANs presents challenges due to the non-convex nature of the objective function, which can result in multiple local minima, and the inherent instability caused by the adversarial learning process.

The loss function of GANs involves minimizing the Jensen-Shannon divergence between the real and generated data distributions [9]. This divergence measurement serves as a guide for the generator to produce samples that closely resemble the real data. However, this loss function is non-convex, meaning it can have multiple local minima instead of a single global minimum. As a result, rather than finding an ideal equilibrium state, sub-optimal states in the proximity of global optimal is accepted.

The instability during GAN training stems from the adversarial nature of the learning process. Any small update or adjustment to one network can have a significant impact on the other network, leading to a delicate balance that can easily be disrupted. This delicate equilibrium can result in oscillations, where the networks struggle to converge to a stable state. Consequently, the training process becomes less predictable and can exhibit periods of slow convergence or even divergence.

2.4. Optimization Approaches to GAN Training

Popularly, gradient-based optimization algorithms like stochastic gradient descent (SGD) or Adam is used for training. However, these algorithms can encounter issues such as vanishing or exploding gradients, which can impede convergence and result in getting stuck in local minima.

The improvements in GAN training are basically done into three areas:-

- Modification of the objective or loss function.
- Adjustments of the gradients propagated.
- Alterations of the update rules.

Modifications to the objective function aim to make it more amenable to optimization. For example, the Wasserstein distance (also known as Earth-Mover's distance) has been proposed as an alternative to the Jensen-Shannon divergence. The Wasserstein distance provides a smoother gradient, which can lead to more stable training and help avoid issues associated with mode collapse [10].

Adjusting the gradients propagated addresses the problems encountered due to instability and regularization in GAN training. One such technique is the gradient penalty, which enforces the Lipschitz continuity of the discriminator [11, 12]. By adding a penalty term to the loss function, the gradients are regulated, leading to improved stability during training. Another technique is gradient clipping that enforces the gradient values within a range.

Alterations to the update rules aim to enhance the convergence speed and quality of the generated samples. Momentum-based optimization algorithms like Nesterov's accelerated gradient (NAG) or Adagrad have been proposed to expedite the convergence of GAN training. These algorithms leverage past gradients to update the model parameters, enabling more efficient learning and better overall performance.

These fundamental modifications have been incorporated earlier to enhance the training process, enabling GAN models to generate more diverse, high-quality samples across various domains.

2.5. Duality Gap

Duality Gap is recently conceptualized in [4] [5] [6] [7] [13] to minimize the competition between generator and the discriminator and understanding the convergence. [7] provided proximal duality gap to monitor the convergence in case of non existing Nash Equilibrium.[13] proposed a measure based on Duality Gap to rank different GAN models and monitor the mode collapse and non convergence with low computational cost. Duality Gap is a variant of the primal-dual interior-point method that is used to optimize the GAN objective function which aims to minimize the duality gap between the primal and dual formulations of the GAN objective function [6].

In a zero-sum game, two players, P_1 and P_2, each have their own sets of strategies, denoted as U and V, respectively. The game's objective, denoted as M, defines the utilities of the players based on their chosen strategies $(u, v) \in U \times V$, respectively. The goal of each player is to maximize their worst-case utility; Thus,

$$\min_{u \in U} \max_{v \in V} M(u, v) \text{ is the goal of } P_1.$$
$$\max_{v \in V} \min_{u \in U} M(u, v) \text{ is the goal of } P_2.$$

For both players to converge to a joint solution, there must exist a pure equilibrium (u*, v*), where neither P1 nor P2 can improve their utility by unilaterally changing their strategy. A pure equilibrium satisfies the condition: $\underset{v \in V}{max} M(u*, v) = \underset{u \in U}{min} M(u, v*)$.

To assess the performance of a pure strategy $(u, v) \in U \times V$, we define the Duality Gap (DG) as follows:

$$DG(u, v) = \max_{v' \in V} M(u, v') - \min_{u' \in U} M(u', v) \tag{2}$$

In the context of Generative Adversarial Networks (GANs), it has been demonstrated that the duality gap is always positive when the generated distribution by GANs does not match the true distribution. Specifically, the duality gap is at least as large as the Jensen-Shannon divergence between the true and fake distributions, which is always non-negative [6]. When the GAN successfully generates samples that perfectly match the true distribution, it implies the existence of a discriminator that can no longer distinguish between real data and generated data, resulting in a duality gap of zero. [5]

3. Extending Training Algorithms with Duality Gap

In this section, the traditional algorithms are enhanced by utilizing the duality gap as a substitute objective function. This duality gap serves as an upper limit on the difference between the distributions of real data and the generated samples.

By optimizing this duality gap, the training process simplifies into a standard optimization problem, allowing us to bypass the complexities of the original GAN problem's game-theoretic formulation. The goal is to find the strategies $(u*, v*)$ that minimize the Duality Gap: $\min_{u \in U, v \in V}[DG(u, v)]$ instead of the stochastic minimax problem: $\min_{u \in U} \max_{v \in V}[M(u, v)]$.

3.1. Extending Gradient Descent Ascent (GDA) with Duality Gap

Gradient Descent Ascent (GDA) is the standard approach for training Generative Adversarial Networks (GANs). GDA involves updating the parameters of the generator and discriminator networks alternatively using gradient descent (for the generator) and gradient ascent (for the discriminator) to minimize/maximize the GAN's objective function until convergence is achieved.

The update rule for GDA is given by Eq. 3:

$$\begin{bmatrix} \theta_g^{(k+1)} \\ \theta_d^{(k+1)} \end{bmatrix} = \begin{bmatrix} \theta_g^{(k)} \\ \theta_d^{(k)} \end{bmatrix} - \alpha \begin{bmatrix} \nabla_{\theta_g} J_g(\theta_g^{(k)}, \theta_d^{(k)}) \\ -\nabla_{\theta_d} J_d(\theta_g^{(k)}, \theta_d^{(k)}) \end{bmatrix} \tag{3}$$

During each iteration, the generator's parameters are updated in the direction opposite to the gradient of its objective function, aiming to improve the quality of the generated samples. Simultaneously, the discriminator's parameters are updated in the direction of the gradient of its objective function to improve its capability to differentiate between real and fake samples.. This alternating update process continues until convergence is achieved, or a predefined stopping criterion is met.

Algorithm 1 Applying Duality Gap on GDA (GDA-DG)

Require: Number of steps T, game objective $M(u, v)$, k

for $t = 1$ to T **do**
  Set $u_{w_0} = u_t$ and $v_{w_0} = v_t$
  for $i = 1$ to k **do**
    # Computing u_{worst} and v_{worst}
    $u_{w_i} \leftarrow u_{w_{i-1}} - \alpha_i \nabla_{u_w} M(u_{w_{i-1}}, v_t)$
    $v_{w_i} \leftarrow v_{w_{i-1}} + \alpha_i \nabla_{v_w} M(u_t, v_{w_{i-1}})$
  end for
  Calculate: $DG(u_t, v_t) = M(u_t, v_{w_k}) - M(u_{w_k}, v_t)$
  $u_{t+1} \leftarrow u_t - \eta_t \nabla_u DG(u_t, v_t)$
  $v_{t+1} \leftarrow v_t - \eta_t \nabla_v DG(u_t, v_t)$
end for

Convergence guarantees for DG are established similar to those found in the GAN literature. However, computing DG exactly requires finding the worst-case players (generator and discriminator), which is often computationally inefficient. Thus,we estimate the worst-case values in a GAN using gradient-based optimization with a limited number of steps (k). This approach is commonly employed in GANs, where each player (generator and discriminator) approximates its loss function with a finite number of steps. To speed up the optimization process and ensure a non-negative duality gap (DG), the networks are initialized with the parameters of the adversary at the specific step under consideration.

Despite using a small value of k, this approximation remains accurate and results in effective generative models, as shown by previous analyses. Few update steps for the discriminator are enough to capture the generator's quality, confirming the effectiveness of this approximation technique.

3.2. Extending Alternating Gradient Descent Ascent (Alt-GDA) with Duality Gap

In alternating GDA [14], the generator and discriminator networks are trained in alternating steps. It performs exceptionally well for strongly convex-strongly concave (SCSC) minimax problems, reaching a convergence rate that is nearly optimal. This convergence rate is quadratically better than that achieved by GDA and is on par with the rates of the extra-gradient (EG) and optimistic GDA (OGDA) methods. Moreover, when the minimax problem has strong concavity in one player but lacks strong convexity in the other player, Alt-GDA achieves linear convergence, subject to a non-singularity condition on the coupling matrix.

The update rule for Alt-GDA is given by Eq. 4:

$$\begin{bmatrix}\theta_g^{(k+1)}\\ \theta_d^{(k+1)}\end{bmatrix} = \begin{bmatrix}\theta_g^{(k)}\\ \theta_d^{(k)}\end{bmatrix} - \alpha \begin{bmatrix}\nabla_{\theta_g} J_g(\theta_g^{(k)}, \theta_d^{(k)})\\ -\nabla_{\theta_d} J_d(\theta_g^{(k+1)}, \theta_d^{(k)})\end{bmatrix} \tag{4}$$

During each iteration, the generator's update steps stays same as the GDA update steps but the discriminator takes the updated generator's parameter value while calculating the gradient of the objective function.

Algorithm 2 Applying Duality Gap on Alt-GDA (Alt-GDA-DG)

Require: Number of steps T, game objective $M(u, v)$, k

for $t = 1$ to T **do**
  Set $u_{w_0} = u_t$ and $v_{w_0} = v_t$
  for $i = 1$ to k **do**
    # Computing u_{worst} and v_{worst}
    $u_{w_i} \leftarrow u_{w_{i-1}} - \alpha_i \nabla_{u_w} M(u_{w_{i-1}}, v_t)$
    $v_{w_i} \leftarrow v_{w_{i-1}} + \alpha_i \nabla_{v_w} M(u_t, v_{w_{i-1}})$
  end for
  Calculate: $DG(u_t, v_t) = M(u_t, v_{w_k}) - M(u_{w_k}, v_t)$
  $u_{t+1} \leftarrow u_t - \eta_t \nabla_u DG(u_t, v_t)$
  $v_{t+1} \leftarrow v_t - \eta_t \nabla_v DG(u_{t+1}, v_t)$
end for

In the standard GDA with Duality Gap, the optimization updates for u_{t+1} and v_{t+1} are performed in parallel. But this takes the advantage of the sequential computation of iterates to improve optimization process. The algorithm takes advantage of the information obtained during each step to inform the subsequent update. This can potentially lead to better exploitation of the optimization landscape, resulting in improved convergence guarantees and faster convergence rates.

3.3. Extending Optimistic GDA (OGDA) with Duality Gap

Optimistic Gradient Descent Ascent (OGDA) is an extension of gradient descent/ascent that incorporates optimism to improve the optimization process in training Generative Adversarial Networks (GANs) [15]. OGDA introduces a momentum term that enhances exploration in the parameter space and helps avoid getting trapped in local optima. It achieves this by utilizing an estimate of future gradients to update the current parameters.

The update rule for Optimistic GDA is given by Eq. 5:

$$\begin{aligned}\begin{bmatrix}\theta_g^{(k+1)}\\ \theta_d^{(k+1)}\end{bmatrix} = \begin{bmatrix}\theta_g^{(k)}\\ \theta_d^{(k)}\end{bmatrix} &- 2\alpha \begin{bmatrix}\nabla_{\theta_g} J_g(\theta_g^{(k)}, \theta_d^{(k)})\\ -\nabla_{\theta_d} J_d(\theta_g^{(k)}, \theta_d^{(k)})\end{bmatrix}\\ &+ \alpha \begin{bmatrix}\nabla_{\theta_g} J_g(\theta_g^{(k-1)}, \theta_d^{(k-1)})\\ -\nabla_{\theta_d} J_d(\theta_g^{(k-1)}, \theta_d^{(k-1)})\end{bmatrix}\end{aligned} \tag{5}$$

In the OGDA update rules[16], the parameters of the generator and the discriminator are updated by incorporating the current gradient and a weighted contribution from the gradient at the previous iteration. This optimistic update strategy leverages the estimate of future gradients to accelerate convergence and overcome the vanishing gradient problem.

Algorithm 3 Applying Duality Gap on Optimistic GDA (OGDA-DG)

Require: Number of steps T, game objective $M(u, v)$, k

for $t = 1$ to T **do**
  Set $u_{w_0} = u_t$ and $v_{w_0} = v_t$
  for $i = 1$ to k **do**
    # Computing u_{worst} and v_{worst}
    $u_{w_i} \leftarrow u_{w_{i-1}} - \alpha_i \nabla_{u_w} M(u_{w_{i-1}}, v_t)$
    $v_{w_i} \leftarrow v_{w_{i-1}} + \alpha_i \nabla_{v_w} M(u_t, v_{w_{i-1}})$
  end for
  Calculate: $DG(u_t, v_t) = M(u_t, v_{w_k}) - M(u_{w_k}, v_t)$
  $u_{t+1} \leftarrow u_t - 2\eta_t \nabla_u DG(u_t, v_t) + \eta_t \nabla_u M(u_{t-1}, v_{t-1})$
  $v_{t+1} \leftarrow v_t - 2\eta_t \nabla_v DG(u_t, v_t) + \eta_t \nabla_v M(u_{t-1}, v_{t-1})$
end for

In this algorithms, we add the extra momentum terms to the updates. By incorporating optimism and leveraging past information [17], OGDA can be more resilient to getting trapped in suboptimal solutions, enabling better exploration of the parameter space and leading to improved convergence during GAN training.

3.4. Extending ExtraGradient with Duality Gap (ExtraGradient-DG)

Extragradient is an optimization algorithm employed in training Generative Adversarial Networks (GANs) that utilizes a double update scheme [15]. Similar to gradient descent/ascent, Extragradient alternates between generator and discriminator updates. However, it incorporates an additional extrapolation step to increase stability and prevent oscillations during the training process.

The update rule for ExtraGradient is given by Eq. 6:

$$\begin{bmatrix}\theta_g^{(k+1)}\\ \theta_d^{(k+1)}\end{bmatrix} = \begin{bmatrix}\theta_g^{(k)}\\ \theta_d^{(k)}\end{bmatrix} - \alpha \begin{bmatrix}\nabla_{\theta_g} J_g(& \theta_g^{(k)} - \alpha\nabla_{\theta_g} J_g(\theta_g^{(k)}, \theta_d^{(k)}),\\ & \theta_d^{(k)} + \alpha\nabla_{\theta_d} J_d(\theta_g^{(k)}, \theta_d^{(k)}))\\ -\nabla_{\theta_d} J_d(& \theta_g^{(k)} - \alpha\nabla_{\theta_g} J_g(\theta_g^{(k)}, \theta_d^{(k)}),\\ & \theta_d^{(k)} + \alpha\nabla_{\theta_d} J_d(\theta_g^{(k)}, \theta_d^{(k)}))\end{bmatrix} \tag{6}$$

In the Extragradient update rules, the parameters of the generator and discriminator are updated by extrapolating the current update using the next update's gradients. This extrapolation step helps increase stability and prevents oscillations during the training process.

Algorithm 4 Applying Duality Gap on ExtraGradient (ExtraGradient-DG)

Require: Number of steps T, game objective $M(u, v)$, k

for $t = 1$ to T **do**
  Set $u_{w_0} = u_t$ and $v_{w_0} = v_t$
  for $i = 1$ to k **do**
    # Computing u_{worst} and v_{worst}
    $u_{w_i} \leftarrow u_{w_{i-1}} - \alpha_i \nabla_{u_w} M(u_{w_{i-1}}, v_t)$
    $v_{w_i} \leftarrow v_{w_{i-1}} + \alpha_i \nabla_{v_w} M(u_t, v_{w_{i-1}})$
  end for
  Calculate: $DG(u_t, v_t) = M(u_t, v_{w_k}) - M(u_{w_k}, v_t)$
  $u_{temp} \leftarrow u_t - \eta_t \nabla_u M(u_t, v_t)$
  $v_{temp} \leftarrow v_t + \eta_t \nabla_v M(u_t, v_t)$
  $u_{t+1} \leftarrow u_t - \eta_t \nabla_u DG(u_{temp}, v_{temp})$
  $v_{t+1} \leftarrow v_t - \eta_t \nabla_v DG(u_{temp}, v_{temp})$
end for

In this algorithm we incorporate the extrapolation step, which aims to improve the convergence behavior and training stability of GANs. It leverages information from the next update to guide the current update, allowing for more robust and effective parameter updates during GAN training.

3.5. Extending Consensus Optimization (CO) with Duality Gap (CO-DG)

The key idea behind consensus optimization [18] is to enforce agreement between the updates of the generator and discriminator during training. Instead of independently updating the networks in an adversarial manner, consensus optimization combines the updates in a way that promotes convergence towards a shared or consensus solution. This approach ensures that both the generator and discriminator progress together towards a more balanced and stable state by adding a regularization term. The update rule for Consensus Optimization is given by Eq.7:

$$\begin{bmatrix}\theta_g^{(k+1)}\\ \theta_d^{(k+1)}\end{bmatrix} = \begin{bmatrix}\theta_g^{(k)}\\ \theta_d^{(k)}\end{bmatrix} - \alpha \begin{bmatrix}\nabla_{\theta_g} J_g(\theta_g^{(k)}, \theta_d^{(k)})\\ -\nabla_{\theta_d} J_d(\theta_g^{(k)}, \theta_d^{(k)})\end{bmatrix} - \gamma\eta\nabla\|\nabla J(\theta_g^{(k)}, \theta_d^{(k)})\|^2 \tag{7}$$

$\nabla\|\nabla J(\theta_g^{(k)}, \theta_d^{(k)})\|^2$ is the regularization term to enforce the constraint. By leveraging consensus steps [19] and parameter averaging, CO facilitates consistent training across distributed devices, enabling efficient and effective training of large-scale GANs. The algorithm helps overcome scalability challenges and improves the synchronization and agreement among the generator and discriminator networks during distributed training.

$$\begin{aligned}\nabla\|\nabla f(x,y)\|^2 &= 2\,\nabla(\nabla f(x,y)) \cdot \nabla f(x,y)\\ &= 2\,(\mathbf{H}(f(x,y)) \cdot \nabla f(x,y))\end{aligned}$$

- $\nabla\|\nabla f(x,y)\|^2$: This denotes the gradient of the squared norm of the gradient of the function $f(x,y)$. In simpler terms, it represents the derivative of the magnitude of the gradient of $f(x,y)$ with respect to the variables x and y.

- $2\,\nabla(\nabla f(x,y)) \cdot \nabla f(x,y)$: This expression simplifies the first term. Here, $\nabla(\nabla f(x,y))$ refers to the gradient of the gradient of $f(x,y)$. The dot product between this gradient and $\nabla f(x,y)$ is taken, and the result is multiplied by 2 as a result of chain rule of differentiation.

- $2\,\mathbf{H}(f(x,y)) \cdot \nabla f(x,y))$: The final form introduces the Hessian matrix, denoted by $\mathbf{H}(f(x,y))$. The Hessian is the matrix of second-order partial derivatives of $f(x,y)$. The dot product between the Hessian matrix and $\nabla f(x,y)$ is computed, and the entire expression is multiplied by 2.

Algorithm 5 Applying Duality Gap on Consensus Optimization (CO-DG)

Require: Number of steps T, game objective $M(u, v)$, k

for $t = 1$ to T **do**
  Set $u_{w_0} = u_t$ and $v_{w_0} = v_t$
  for $i = 1$ to k **do**
    # Computing u_{worst} and v_{worst}
    $u_{w_i} \leftarrow u_{w_{i-1}} - \alpha_i \nabla_{u_w} M(u_{w_{i-1}}, v_t)$
    $v_{w_i} \leftarrow v_{w_{i-1}} + \alpha_i \nabla_{v_w} M(u_t, v_{w_{i-1}})$
  end for
  Calculate: $DG(u_t, v_t) = M(u_t, v_{w_k}) - M(u_{w_k}, v_t)$
  $reg_x, reg_y \leftarrow \gamma\mathbf{H}(f(x,y)) \cdot \nabla M(u_t, v_t)$
  $u_{t+1} \leftarrow u_t - \eta_t \nabla_u DG(u_t, v_t) - 2reg_x$
  $v_{t+1} \leftarrow v_t - \eta_t \nabla_v DG(u_t, v_t) - 2reg_y$
end for

In this algorithm, we add an extra regularization term $\gamma\mathbf{H}(f(x,y)) \cdot \nabla M(u_t, v_t)$ to foster a more cooperative and synchronized learning process between the generator and discriminator.

3.6. Extending Symplectic Gradient Adjustment with Duality Gap (SGA-DG)

Symplectic Gradient Adjustment (SGA) [20] is an optimization algorithm for training Generative Adversarial Networks (GANs) that draws inspiration from Hamiltonian dynamics. It introduces a symplectic integrator, which enables the update of generator and discriminator parameters while preserving the underlying geometry of the optimization problem. This approach aims to address the issue of convergence to poor local optima by enhancing the exploration of the parameter space.

The update rule for Symplectic Gradient Adjustment is given by Eq. 8:

$$\begin{bmatrix}\theta_g^{(k+1)}\\ \theta_d^{(k+1)}\end{bmatrix} = \begin{bmatrix}\theta_g^{(k)}\\ \theta_d^{(k)}\end{bmatrix} - \alpha \begin{bmatrix}\nabla_{\theta_g} J_g(\theta_g^{(k)}, \theta_d^{(k)})\\ -\nabla_{\theta_d} J_d(\theta_g^{(k)}, \theta_d^{(k)})\end{bmatrix} - \alpha\lambda \begin{bmatrix}\mathbf{H}_{xy}\nabla_{\theta_d} J_g(\theta_g^{(k)}, \theta_d^{(k)})\\ \mathbf{H}_{yx}\nabla_{\theta_g} J_d(\theta_g^{(k)}, \theta_d^{(k)})\end{bmatrix} \tag{8}$$

In SGA, the update rules incorporate symplectic matrices which play a crucial role in preserving the geometry of the underlying optimization problem. By adjusting the gradients with these symplectic operations, SGA aims to enhance exploration in the parameter space, potentially enabling the discovery of better optima and avoiding convergence to poor local optima.

Algorithm 6 Applying Duality Gap on SGA (SGA-DG)

Require: Number of steps T, game objective $M(u, v)$, k

for $t = 1$ to T **do**
  Set $u_{w_0} = u_t$ and $v_{w_0} = v_t$
  for $i = 1$ to k **do**
    # Computing u_{worst} and v_{worst}
    $u_{w_i} \leftarrow u_{w_{i-1}} - \alpha_i \nabla_{u_w} M(u_{w_{i-1}}, v_t)$
    $v_{w_i} \leftarrow v_{w_{i-1}} + \alpha_i \nabla_{v_w} M(u_t, v_{w_{i-1}})$
  end for
  Calculate: $DG(u_t, v_t) = M(u_t, v_{w_k}) - M(u_{w_k}, v_t)$
  $u_{t+1} \leftarrow u_t - \eta_t \nabla_u DG(u_t, v_t) - \eta_t \lambda H_{uv} \nabla_v M(u_t, v_t)$
  $v_{t+1} \leftarrow v_t - \eta_t \nabla_v DG(u_t, v_t) - \eta_t \lambda H_{vu} \nabla_u M(u_t, v_t)$
end for

In this algorithm, we add the adjustment terms to the gradient which are inspired by the principles of symplectic dynamics. This prevents the oscillations that can occur during the traditiona; gradient-based optimization.

3.7. Extending Follow-The-Ridge (FTR) with Duality Gap (FTR-DG)

Follow-the-Ridge[21] is an optimization algorithm that incorporates second-order information in the form of Hessian matrices to enhance the training of Generative Adversarial Networks (GANs). By considering the curvature information of the GAN objective function, Follow-the-Ridge aims to navigate the optimization landscape more efficiently, leading to improved convergence.

The update rule for Follow-The-Ridge is given by Eq. 9:

$$\begin{aligned}\begin{bmatrix}\theta_g^{(k+1)}\\ \theta_d^{(k+1)}\end{bmatrix} = \begin{bmatrix}\theta_g^{(k)}\\ \theta_d^{(k)}\end{bmatrix} &- \alpha \begin{bmatrix}\nabla_{\theta_g} J_g(\theta_g^{(k)}, \theta_d^{(k)})\\ -\nabla_{\theta_d} J_d(\theta_g^{(k)}, \theta_d^{(k)})\end{bmatrix}\\ &+ \alpha \begin{bmatrix}0\\ \mathbf{H}_{yy}^{-1}\mathbf{H}_{yx}\nabla_{\theta_g} J_d(\theta_g^{(k)}, \theta_d^{(k)})\end{bmatrix}\end{aligned} \tag{9}$$

By considering the curvature information through the Hessian matrix, Follow-the-Ridge allows the discriminator to update its parameters in a way that follows the direction of the local ridge in the objective landscape. This enables the algorithm to more effectively explore and converge to better optima during GAN training.

Algorithm 7 Applying Duality Gap on FTR (FTR-DG)

Require: Number of steps T, game objective $M(u, v)$, k

for $t = 1$ to T **do**
  Set $u_{w_0} = u_t$ and $v_{w_0} = v_t$
  for $i = 1$ to k **do**
    # Computing u_{worst} and v_{worst}
    $u_{w_i} \leftarrow u_{w_{i-1}} - \alpha_i \nabla_{u_w} M(u_{w_{i-1}}, v_t)$
    $v_{w_i} \leftarrow v_{w_{i-1}} + \alpha_i \nabla_{v_w} M(u_t, v_{w_{i-1}})$
  end for
  Calculate: $DG(u_t, v_t) = M(u_t, v_{w_k}) - M(u_{w_k}, v_t)$
  $u_{t+1} \leftarrow u_t - \eta_t \nabla_u DG(u_t, v_t)$
  $v_{t+1} \leftarrow v_t - \eta_t \nabla_v DG(u_t, v_t) + \eta_t H_{vv}^{-1} H_{vu} \nabla_u M(u_t, v_t)$
end for

In this algorithm, a ridge penalty term is introduced to the discriminator update step. The purpose of this penalty is to encourage the discriminator to produce smooth gradients, which can help prevent the model from focusing too much on specific modes or details in the data distribution.

It is worth noting that estimating and computing the Hessian matrix can be computationally expensive, especially for high-dimensional problems. Various approximations and techniques can be employed to overcome these challenges and make the Follow-the-Ridge algorithm feasible for GAN optimization.

4. Analysing Updates in Extended Training Algorithms

Let's consider a convex-concave function f(x, y) as

$$f(x, y) = cxy, \quad c \in \{3, 10\} \tag{10}$$

and the corresponding duality gap equation is given as

$$dg(x, y) = c(x \cdot y_w - x_w \cdot y). \tag{11}$$

The SGD updates for the game Equation 10 are:

$$\begin{aligned}\begin{bmatrix}x_{t+1}\\ y_{t+1}\end{bmatrix} &= \begin{bmatrix}x_t\\ y_t\end{bmatrix} - \eta \begin{bmatrix}\nabla_x f(x_t, y_t)\\ -\nabla_y f(x_t, y_t)\end{bmatrix}\\ &= \begin{bmatrix}x_t\\ y_t\end{bmatrix} - \begin{bmatrix}\eta y_t c\\ -\eta x_t c\end{bmatrix}\\ &= \begin{bmatrix}1 & -\eta c\\ \eta c & 1\end{bmatrix}\begin{bmatrix}x_t\\ y_t\end{bmatrix}\end{aligned}$$

To study stability, we need to find the eigenvalues of the above matrix which we denote by $\mathbf{A}$. We set:

$$|\mathbf{A} - \lambda\mathbf{I}| = \left|\begin{bmatrix}1-\lambda & -\eta c\\ \eta c & 1-\lambda\end{bmatrix}\right| = 0$$

The eigenvalues are $\lambda_{1,2} = 1 \pm i\eta c$

In order to get stability, we need the modulus of $\lambda_i < 1$, i.e. $1 + (\eta c)^2 < 1$ which is impossible thus leading to the oscillating or diverging behavior depending on the value of c.

4.1. Analysing Updates in SGD using Duality Gap

Considering the duality gap equation given in Eq.11, the corresponding updates by SGD-DG is given by:

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \end{bmatrix} - \eta \begin{bmatrix} \nabla_x dg(x_t, y_t) \\ -\nabla_y dg(x_t, y_t) \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \end{bmatrix} - \begin{bmatrix} \eta y_w c \\ -\eta x_w c \end{bmatrix}$$

where

$$\begin{bmatrix} x_w \\ y_w \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \end{bmatrix} - \begin{bmatrix} \eta y_t c k \\ -\eta x_t c k \end{bmatrix}$$

Hence, the combined updates are:

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \end{bmatrix} = \begin{bmatrix} x_t - \eta y_t - \eta^2 k c x_t \\ y_t + \eta x_t - \eta^2 k c y_t \end{bmatrix} = \begin{bmatrix} 1 - \eta^2 kc & -\eta \\ \eta & 1 - \eta^2 kc \end{bmatrix} \begin{bmatrix} x_t \\ y_t \end{bmatrix}$$

The eigenvalues of the above matrix can be obtained by:

$$|\mathbf{A} - \lambda \mathbf{I}| = \left| \begin{bmatrix} 1 - \eta^2 kc - \lambda & -\eta \\ \eta & 1 - \eta^2 kc - \lambda \end{bmatrix} \right| = 0$$

The eigenvalues are $\lambda_{1,2} = -\eta^2 kc + 1 \pm i\eta$
In order to get stability, we need the modulus of $\lambda_i < 1$, i.e. $\sqrt{(-\eta^2 kc + 1)^2 + \eta^2} < 1$ which can be obtained for $\eta < \frac{2kc-1}{k^2 c^2}$

4.2. Analysing Updates in Alternating GDA-DG

Considering the duality gap equation given in Eq.11,the corresponding updates by GDA-DG is given by:

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \end{bmatrix} - \eta \begin{bmatrix} \nabla_x dg(x_t, y_t) \\ -\nabla_y dg(x_{t+1}, y_t) \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \end{bmatrix} - \begin{bmatrix} \eta y_w c \\ -\eta x_w c \end{bmatrix}$$

The problem reduces to same as GDA using Duality Gap. Taking any other convex-concave function which have x or y term after finding the gradient of the duality gap function can follow the same procedure.

4.3. Analysing Updates in Optimistic GDA-DG

Considering the duality gap equation given in Eq.11,the corresponding updates by Optimistic GDA-DG is given by: The corresponding updates are given by:

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \end{bmatrix} - 2\eta \begin{bmatrix} \nabla_x dg(x_t, y_t) \\ -\nabla_y dg(x_t, y_t) \end{bmatrix} + \eta \begin{bmatrix} \nabla_x f(x_{t-1}, y_{t-1}) \\ -\nabla_y f(x_{t-1}, y_{t-1}) \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \end{bmatrix} - \begin{bmatrix} 2\eta y_w c \\ -2\eta x_w c \end{bmatrix} + \begin{bmatrix} \eta y_{t-1} c \\ -\eta x_{t-1} c \end{bmatrix}$$

where

$$\begin{bmatrix} x_w \\ y_w \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \end{bmatrix} - \begin{bmatrix} \eta y_t c k \\ -\eta x_t c k \end{bmatrix}$$

$$\begin{bmatrix} x_{t-1} \\ y_{t-1} \end{bmatrix} = \frac{1}{1 + \eta^2 c^2} \begin{bmatrix} 1 & \eta c \\ -\eta c & 1 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \end{bmatrix}$$

Hence, the combined updates are:
let $a = \frac{1}{1+\eta^2 c^2}$

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \end{bmatrix} = \begin{bmatrix} x_t - \eta y_t (2 - a) - \eta^2 c x_t (2k + a) \\ y_t + \eta x_t (2 - a) - \eta^2 c y_t (2k - a) \end{bmatrix} = \begin{bmatrix} 1 - \eta^2 c(2k + a) & -\eta(2 - a) \\ \eta(2 - a) & 1 - \eta^2 c(2k + a) \end{bmatrix} \begin{bmatrix} x_t \\ y_t \end{bmatrix}$$

The eigenvalues of the above matrix can be obtained by:

$$|\mathbf{A} - \lambda \mathbf{I}| = \left| \begin{bmatrix} 1 - \eta^2 c(2k + a) - \lambda & -\eta(2 - a) \\ \eta(2 - a) & 1 - \eta^2 c(2k + a) - \lambda \end{bmatrix} \right| = 0$$

The eigenvalues are $\lambda_{1,2} = -\eta^2 c(2k + a) + 1 \pm i\eta(2 - a)$
In order to get stability, we need the modulus of $\lambda_i < 1$, i.e. $\sqrt{(-\eta^2 c(2k + a) + 1)^2 + \eta^2 (2 - a)^2} < 1$.

4.4. Analysing Updates in Extragradient-DG

Considering the duality gap equation given in Eq.11,the corresponding updates by Extragradient-DG is given by:

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \end{bmatrix} - \eta \begin{bmatrix} \nabla_x dg(x_t - \eta c y_t, y_t + \eta c x_t) \\ -\nabla_y dg(x_t - \eta c y_t, y_t + \eta c x_t) \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \end{bmatrix} - \begin{bmatrix} \eta y_w c \\ -\eta x_w c \end{bmatrix}$$

The problem reduces to same as GDA using Duality Gap. Taking any other convex-concave function which have x or y term after finding the gradient of the duality gap function can follow the same procedure.

4.5. Analysing Updates in CO-DG

Considering the duality gap equation given in Eq.11,the corresponding updates by CO-DG is given by:

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \end{bmatrix} - \eta \begin{bmatrix} \nabla_x dg(x_t, y_t) \\ -\nabla_y dg(x_t, y_t) \end{bmatrix} - 2\gamma \mathbf{H}(f(x_t, y_t)) \cdot \nabla f(x_t, y_t) = \begin{bmatrix} x_t \\ y_t \end{bmatrix} - \begin{bmatrix} \eta y_w c \\ -\eta x_w c \end{bmatrix} - 2\gamma \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} y_t \\ x_t \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \end{bmatrix} - \begin{bmatrix} \eta y_w c - 2\gamma x_t \\ -\eta x_w c - 2\gamma y_t \end{bmatrix}$$

where

$$\begin{bmatrix} x_w \\ y_w \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \end{bmatrix} - \begin{bmatrix} \eta y_t c k \\ -\eta x_t c k \end{bmatrix}$$

Hence, the combined updates are:

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \end{bmatrix} = \begin{bmatrix} (1 + 2\gamma) x_t - \eta y_t - \eta^2 k c x_t \\ (1 + 2\gamma) y_t + \eta x_t - \eta^2 k c y_t \end{bmatrix} = \begin{bmatrix} 1 + 2\gamma - \eta^2 kc & -\eta \\ \eta & 1 + 2\gamma - \eta^2 kc \end{bmatrix} \begin{bmatrix} x_t \\ y_t \end{bmatrix}$$

The eigenvalues of the above matrix can be obtained by:

$$|\mathbf{A} - \lambda \mathbf{I}| = \left| \begin{bmatrix} 1 + 2\gamma - \eta^2 kc - \lambda & -\eta \\ \eta & 1 + 2\gamma - \eta^2 kc - \lambda \end{bmatrix} \right| = 0$$

The eigenvalues are $\lambda_{1,2} = -\eta^2 kc + 2\gamma + 1 \pm i\eta$
In order to get stability, we need the modulus of $\lambda_i < 1$, i.e. $\sqrt{(-\eta^2 kc + 2\gamma + 1)^2 + \eta^2} < 1$

4.6. Analysing Updates in SGA-DG

Considering the duality gap equation given in Eq.11,the corresponding updates by SGA-DG is given by:

$$\begin{aligned}\begin{bmatrix}x_{t+1}\\ y_{t+1}\end{bmatrix} &= \begin{bmatrix}x_t\\ y_t\end{bmatrix} - \eta\begin{bmatrix}\nabla_x dg(x_t,y_t)\\ -\nabla_y dg(x_t,y_t)\end{bmatrix} - \eta\lambda\begin{bmatrix}\mathbf{H}_{xy}\nabla_y f(x_t,y_t)\\ \mathbf{H}_{yx}\nabla_x f(x_t,y_t)\end{bmatrix}\\ &= \begin{bmatrix}x_t\\ y_t\end{bmatrix} - \begin{bmatrix}\eta y_w c\\ -\eta x_w c\end{bmatrix} - \begin{bmatrix}\eta\lambda x_t\\ \eta\lambda y_t\end{bmatrix}\\ &= \begin{bmatrix}x_t\\ y_t\end{bmatrix} - \begin{bmatrix}\eta y_w c - \eta\lambda x_t\\ -\eta x_w c - \eta\lambda y_t\end{bmatrix}\end{aligned}$$

where

$$\begin{bmatrix}x_w\\ y_w\end{bmatrix} = \begin{bmatrix}x_t\\ y_t\end{bmatrix} - \begin{bmatrix}\eta y_t c k\\ -\eta x_t c k\end{bmatrix}$$

Hence, the combined updates are:

$$\begin{aligned}\begin{bmatrix}x_{t+1}\\ y_{t+1}\end{bmatrix} &= \begin{bmatrix}(1+\eta\lambda)x_t - \eta y_t - \eta^2 kc x_t\\ (1+\eta\lambda)y_t + \eta x_t - \eta^2 kc y_t\end{bmatrix}\\ &= \begin{bmatrix}1+\eta\lambda-\eta^2 kc & -\eta\\ \eta & 1+\eta\lambda-\eta^2 kc\end{bmatrix}\begin{bmatrix}x_t\\ y_t\end{bmatrix}\end{aligned}$$

The eigenvalues of the above matrix can be obtained by:

$$|\mathbf{A}-\lambda\mathbf{I}| = \left|\begin{bmatrix}1+\eta\lambda-\eta^2 kc-\lambda & -\eta\\ \eta & 1+\eta\lambda-\eta^2 kc-\lambda\end{bmatrix}\right| = 0$$

The eigenvalues are $\lambda_{1,2} = -\eta^2 kc + \eta\lambda + 1 \pm i\eta$
In order to get stability, we need the modulus of $\lambda_i < 1$, i.e. $\sqrt{(-\eta^2 kc + \eta\lambda + 1)^2 + \eta^2} < 1$

4.7. Analysing Updates in FTR-DG

Considering the duality gap equation given in Eq.11,the corresponding updates by FTR-DG is given by:

$$\begin{aligned}\begin{bmatrix}x_{t+1}\\ y_{t+1}\end{bmatrix} &= \begin{bmatrix}x_t\\ y_t\end{bmatrix} - \eta\begin{bmatrix}\nabla_x dg(x_t,y_t)\\ -\nabla_y dg(x_t,y_t)\end{bmatrix} - \begin{bmatrix}\\ \eta\mathbf{H}_{yy}^{-1}\mathbf{H}_{xy}\nabla_x f(x,y)\end{bmatrix}\\ &= \begin{bmatrix}x_t\\ y_t\end{bmatrix} - \begin{bmatrix}\eta y_w c\\ -\eta x_w c - \eta\mathbf{H}_{yy}^{-1}\mathbf{H}_{xy}\nabla_x f(x,y)\end{bmatrix}\end{aligned}$$

where,

$$\begin{bmatrix}x_w\\ y_w\end{bmatrix} = \begin{bmatrix}x_t\\ y_t\end{bmatrix} - \begin{bmatrix}\eta y_t c k\\ -\eta x_t c k\end{bmatrix}$$

Here, $\mathbf{H}_{yy} = 0$, hence further calculation is not possible. Taking any other convex-concave whose $\mathbf{H}_{yy} \neq 0$, and following the same procedure will give us the parameter analysis of the algorithm.

5. Experimental Setup

The experiments are carried out on the following five toy functions, which are also considered in various experimentation related to GAN training [21] [14] :

(i) $f(x,y) = -3x^2 - y^2 + 4xy$.
(ii) $f(x,y) = 3x^2 + y^2 + 4xy$.
(iii) $f(x,y) = (4x^2 - (y - 3x + 0.05x^3)^2 - 0.1y^4)e^{-0.01(x^2+y^2)}$.
(iv) $f(x,y) = ((0.3x^2 + y)^2 + (0.1y^2 + x)^2) * e^{-0.01(x^2+y^2)}$.
(v) $f(x,y) = \ln(1+e^x) + 3xy - \ln(1+e^y)$.

Table 1: Performance of different training algorithms on function $f(x,y) = -3x^2 - y^2 + 4xy$ with seed at (5, 7).

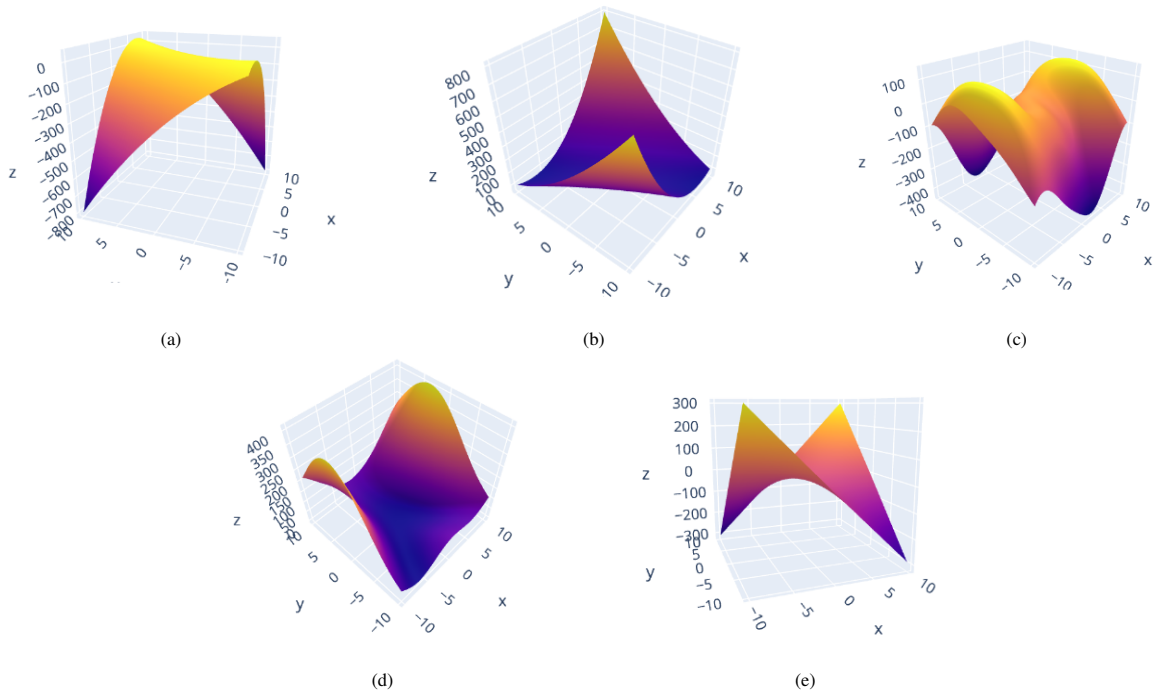
Algorithms	Iterations	Convergence Point	Loss at Convergence point
GDA	N/A	N/A	N/A
GDA-DG	101	(0, 0)	0
Alt-GDA	N/A	N/A	N/A
Alt-GDA-DG	101	(0, 0)	0
OGDA	N/A	N/A	N/A
OGDA-DG	126	(0, 0)	0
ExtraGradient	N/A	N/A	N/A
ExtraGradient-DG	76	(0, 0)	0
CO	>1000	(0, 0)	0
CO-DG	135	(0, 0)	0
SGA	378	(0.0002, 0.0003)	0
SGA-DG	318	(0.0001, 0.0002)	0
FTR	140	(0, 0)	0
FTR-DG	178	(0, 0)	0

The algorithms Gradient Descent Ascent (GDA), Alternating GDA (AGDA), Optimistic GDA (OGDA), ExtraGradient (EG), Consensus Optimization (CO), Symplectic Gradient Adjustment (SGA) and Follow-the-Ridge are experimented for observing the convergence characteristics with and without being extended with Duality Gap (DG). The experiments are conducted using with Python3 kernel in a conda environment. The Autograd library version 1.5 is used to calculate gradients of the objective functions. The generator and discriminator networks are initialized with random weights, and the training process is initiated. A fixed learning rate of 0.05 is used for all the algorithms, and each experiment is run for a maximum of 1000 iterations. For CO, a penalty parameter (γ) of 0.1 is employed, while SGA utilizes a regularization parameter (λ) of 1.0. To ensure the reliability and robustness of the results, each experiment is repeated five times with different random seeds. Throughout the training process, the values of the objective function and the discriminator loss are recorded at each iteration. By conducting these experiments and analyzing the results, this paper aim to gain insights into the convergence behavior of the different optimization algorithms and assess their performance in terms of convergence rate and optimization quality. The utilization of Duality Gap further enhances the evaluation of the algorithms' effectiveness in optimizing the toy functions.

6. Observations & Result Analysis

The experimental results reported indicates that convergence improves in most of the cases with duality gap.

The most contrasting result is reported in Table.1 where, non-converging algorithms like GDA, Alt-GDA, Optimistic GDA, Extragradient are able to converge to the optimal point when extended using duality gap. The contour maps and loss graphs shows the convergence characteristics in detail as provided in Figure.2. It is also observed that CO-DG and SGA-DG shows an improvement in performance while FTR-DG takes more iteration for convergence.



(a) (b) (c) (d) (e)

Figure 1: Surface Plots of the functions (1a) $f(x,y) = -3x^2 - y^2 + 4xy$, (1b) $f(x,y) = 3x^2 + y^2 + 4xy$, (1c) $f(x,y) = (4x^2 - (y - 3x + 0.05x^3)^2 - 0.1y^4)e^{-0.01(x^2+y^2)}$, (1d) $f(x,y) = ((0.3x^2 + y)^2 + (0.1y^2 + x)^2) * e^{-0.01(x^2+y^2)}$, (1e) $f(x,y) = \ln(1 + e^x) + 3xy - \ln(1 + e^y)$

Table 2: Performance of different training algorithms on function $f(x,y) = 3x^2 + y^2 + 4xy$ with seed at (6, 5).

Algorithms	Iterations	Convergence Point	Loss at Convergence point
GDA	153	(0, 0)	0
GDA-DG	312	(-0.0001, 0.0002)	0
Alt-GDA	210	(0, 0)	0
Alt-GDA-DG	312	(-0.0001, 0.0002)	0
OGDA	169	(0, 0)	0
OGDA-DG	124	(0, 0)	0
ExtraGradient	163	(0, 0)	0
ExtraGradient-DG	344	(-0.0001, 0.0002)	0
CO	425	(-0.0002, 0.0003)	0
CO-DG	307	(-0.0001, 0.0002)	0
SGA	361	(0, 0)	0
SGA-DG	293	(0.0002, -0.0003)	0
FTR	N/A	N/A	N/A
FTR-DG	N/A	N/A	N/A

Table 3: Performance of different training algorithms on function $f(x,y) = (4x^2 - (y - 3x + 0.05x^3)^2 - 0.1y^4)e^{-0.01(x^2+y^2)}$ with seed at (7, 5).

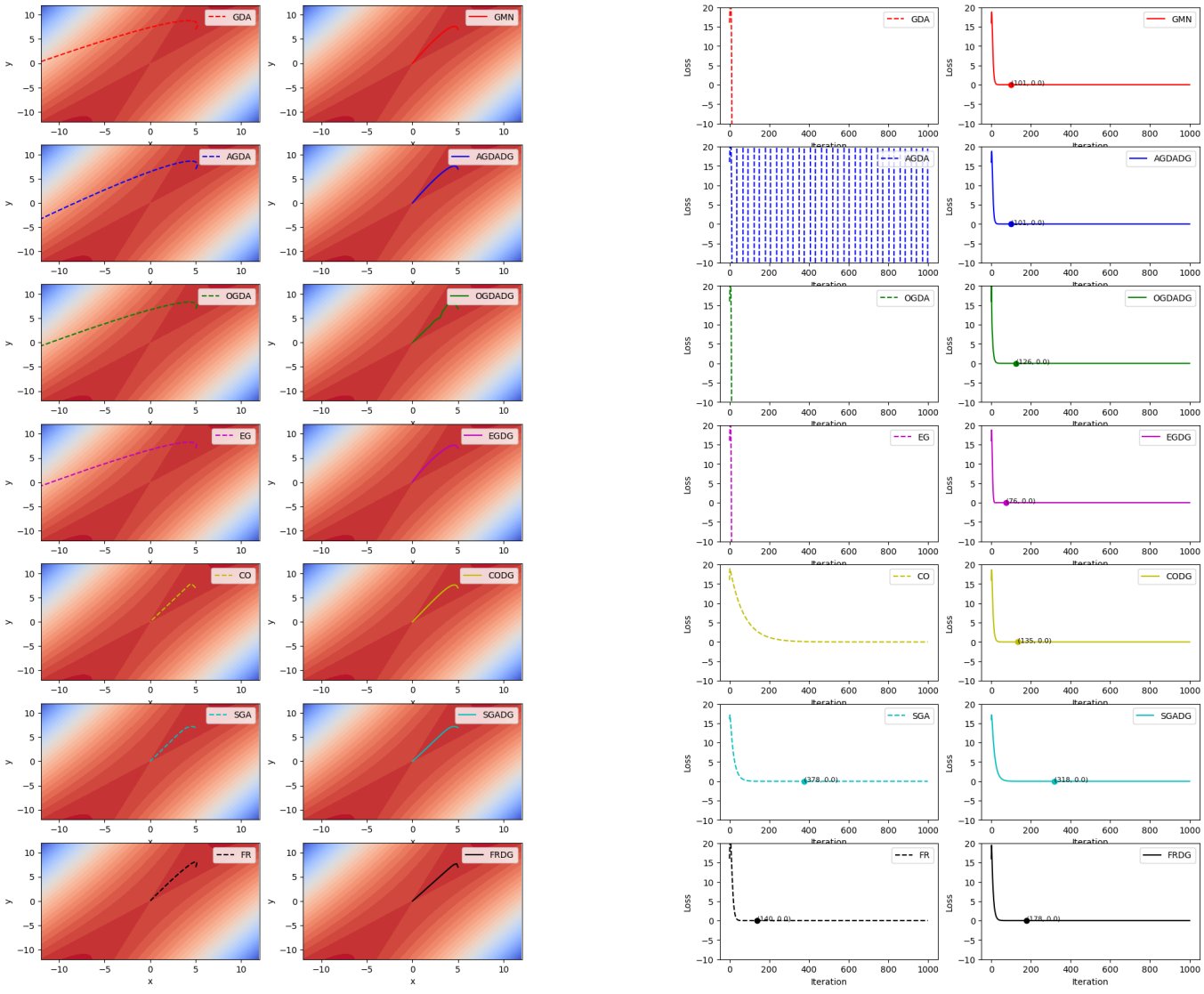
Algorithms	Iterations	Convergence Point	Loss at Convergence point
GDA	N/A	N/A	N/A
GDA-DG	90	(0, 0)	0
Alt-GDA	N/A	N/A	N/A
Alt-GDA-DG	90	(0, 0)	0
OGDA	N/A	N/A	N/A
OGDA-DG	87	(0, 0)	0
ExtraGradient	N/A	N/A	N/A
ExtraGradient-DG	68	(0, 0)	0
CO	N/A	N/A	N/A
CO-DG	95	(0, 0)	0
SGA	169	(0, 0)	0
SGA-DG	134	(0, 0)	0
FTR	130	(0, 0)	0
FTR-DG	101	(0, 0)	0

Table 4: Performance of different training algorithms on function $f(x,y) = ((0.3x^2 + y)^2 + (0.1y^2 + x)^2) * e^{-0.01(x^2+y^2)}$ with seed at (5, 5).

Algorithms	Iterations	Convergence Point	Loss at Convergence point
GDA	87	(-2.5961, 12.4133)	74.59423
GDA-DG	79	(-2.596, 12.4133)	74.59423
Alt-GDA	88	(-2.5961, 12.4133)	74.59423
Alt-GDA-DG	79	(-2.596, 12.4133)	74.59423
OGDA	98	(-2.5961, 12.4133)	74.59423
OGDA-DG	77	(-2.596, 12.4134)	74.59423
ExtraGradient	95	(-2.5961, 12.4133)	74.59423
ExtraGradient-DG	86	(-2.596, 12.4134)	74.59423
CO	56	(-2.596, 12.4133)	74.59423
CO-DG	79	(-2.596, 12.4133)	74.59423
SGA	277	(-2.5962, 12.4133)	74.59423
SGA-DG	261	(-2.5961, 12.4135)	74.59423
FTR	91	(-2.596, 12.4133)	74.59423
FTR-DG	89	(-2.596, 12.4133)	74.59423

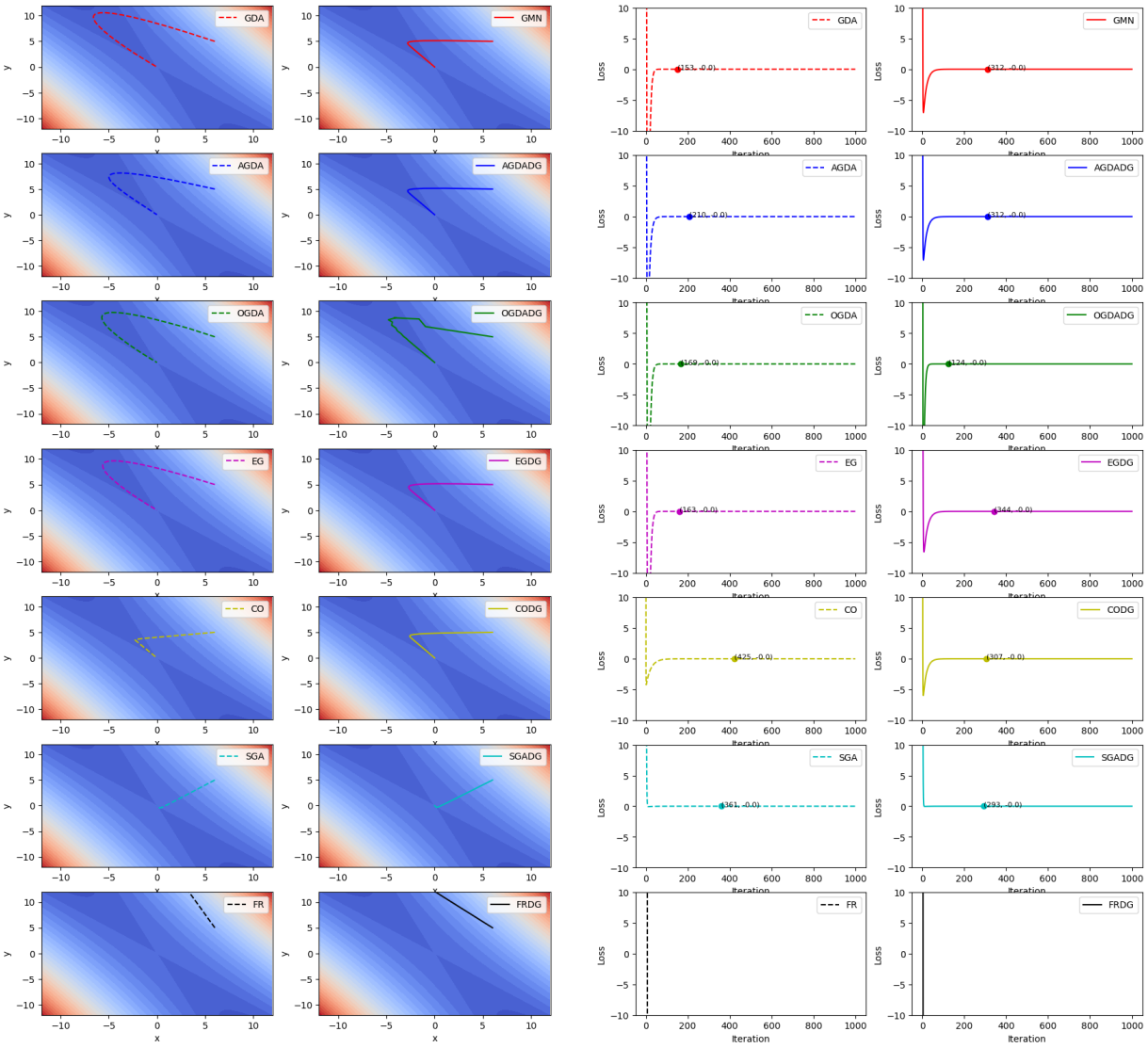
On experimenting with function $f(x,y) = 3x^2 + y^2 + 4xy$, the training algorithms converges at a slow rate as demonstrated in Figure.3 as they approach a non-optimal point (as it is nor a min-max point); however Follow-the-Ridge avoid such point as reported in Table.2. Notably, in function $f(x,y) = (4x^2 - (y - 3x + 0.05x^3)^2 - 0.1y^4)e^{-0.01(x^2+y^2)}$ there is a significant improvement by eliminating the oscillating behavior around the optimal point in GDA, Alternating GDA, Optimistic GDA, Extragradient and CO when extended with duality gap. Also, as shown in Figure.4 and Table.3 rate of convergence slightly improves in SGA-DG and FTR-DG.

In the remaining functions $f(x,y) = ((0.3x^2 + y)^2 + (0.1y^2 + x)^2) * e^{-0.01(x^2+y^2)}$ and $f(x,y) = \ln(1 + e^x) + 3xy - \ln(1 + e^y)$ the results in Table.4 and Table.5 indicates significant improvement in terms of faster convergence rate in GDA-DG, Alt-GDA-DG,



(a) (b)

Figure 2: Convergence plots for function $f(x, y) = -3x^2 - y^2 + 4xy$ (Left) without DG , (Right) with DG (2a) Contour plot showing convergence path. (2b) Training loss and convergence rate.



(a) (b)

Figure 3: Convergence plots for function $f(x, y) = 3x^2 + y^2 + 4xy$ (Left) without DG , (Right) with DG (3a) Contour plot showing convergence path. (3b) Training loss and convergence rate.

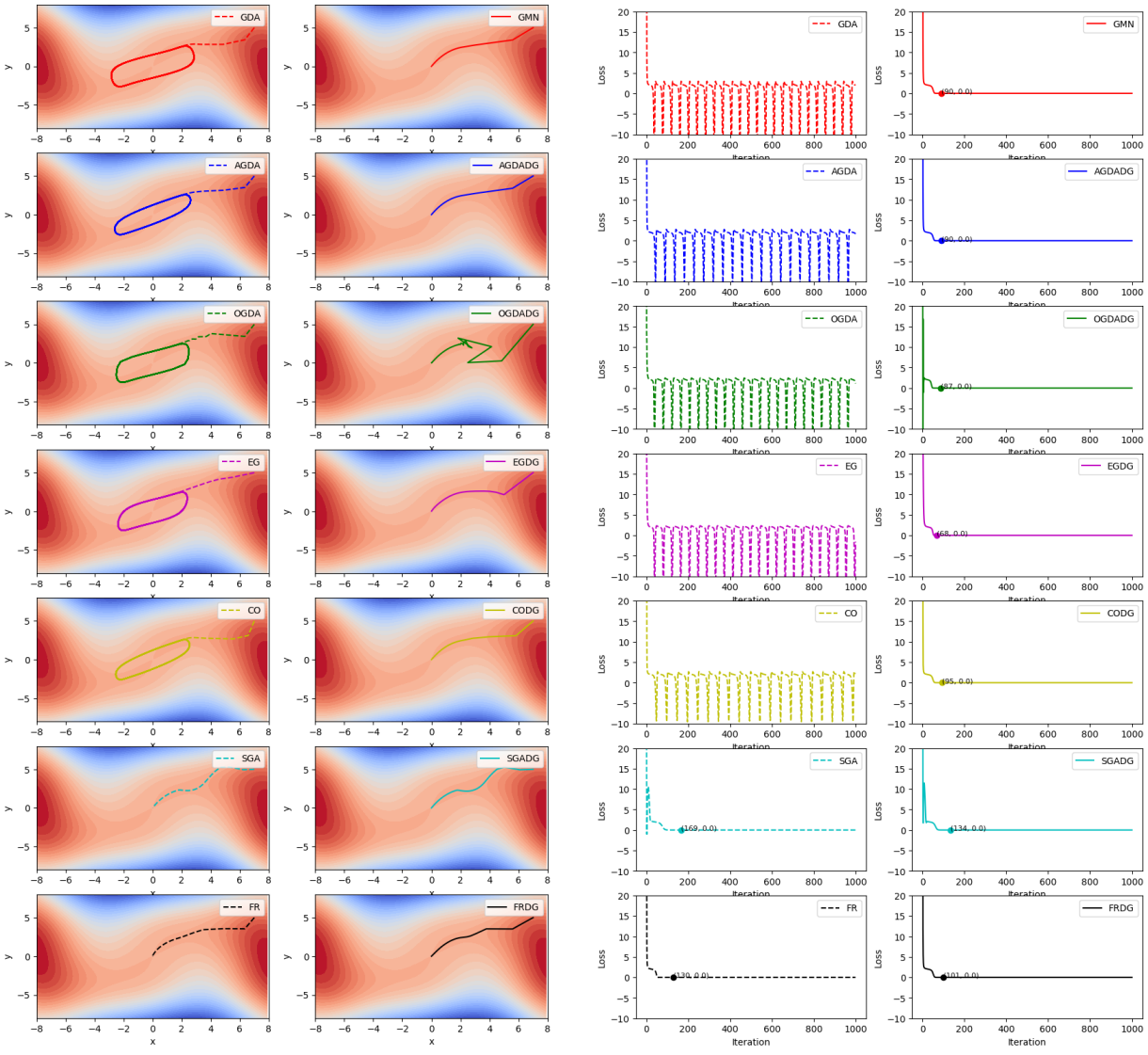
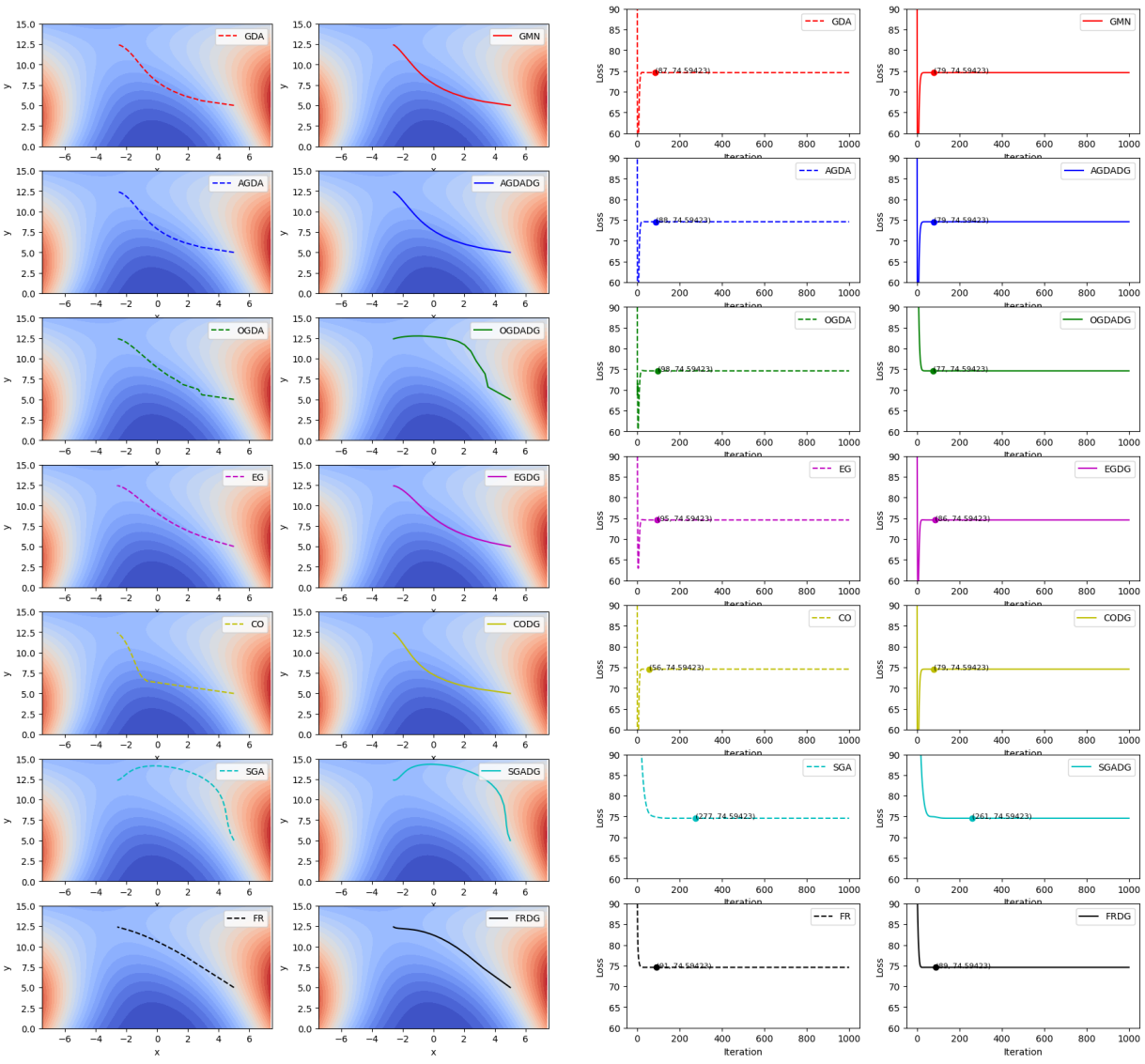
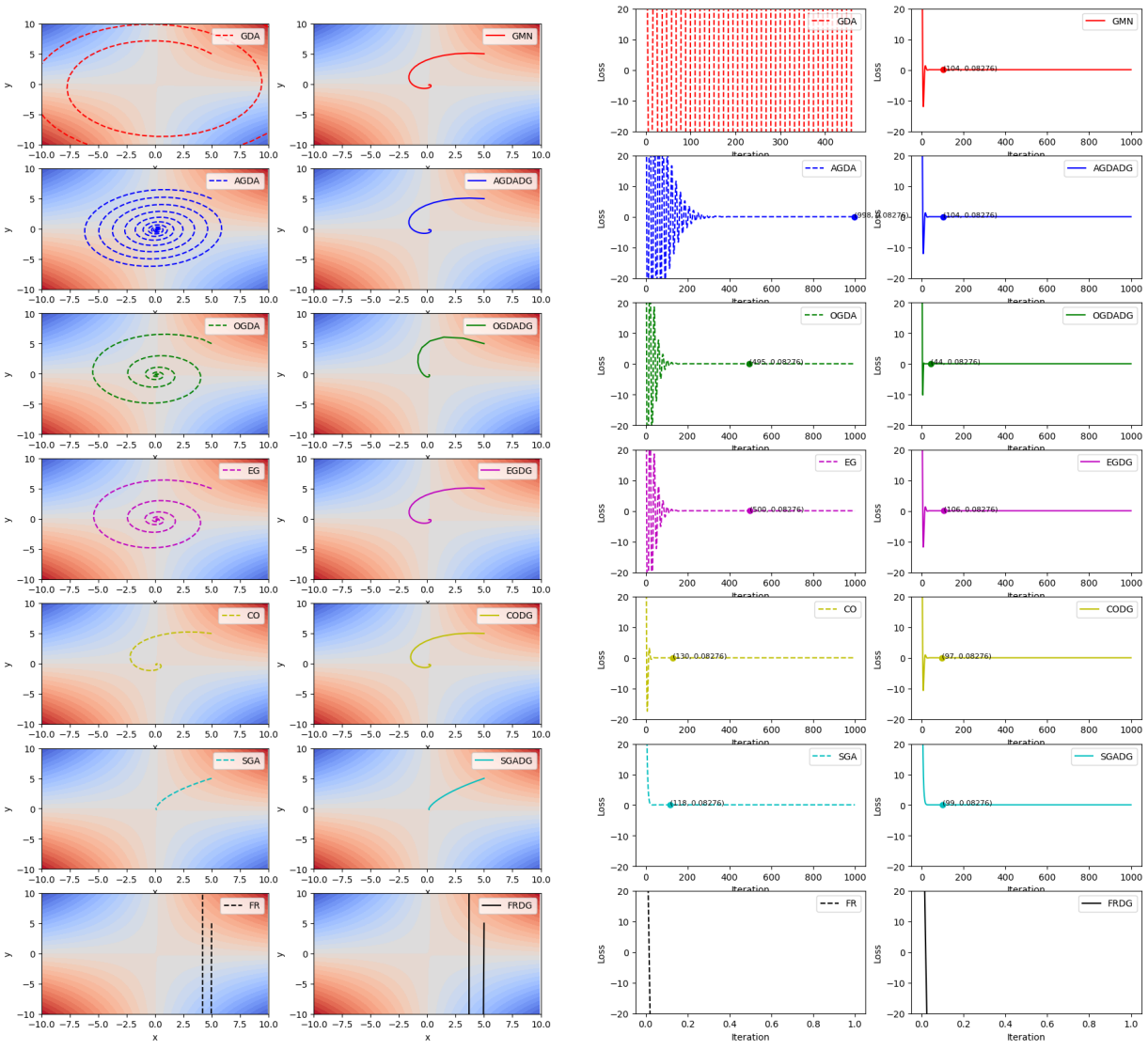


Figure 4: Convergence plots for function $f(x, y) = (4x^2(y - 3x + 0.05x^3)^2 - 0.1y^4)e^{-0.01(x^2+y^2)}$ (Left) without DG , (Right) with DG (4a) Contour plot showing convergence path. (4b) Training loss and convergence rate.



(a) (b)

Figure 5: Convergence plots for function $f(x, y) = ((0.3x^2 + y)^2 + (0.1y^2 + x)^2) * e^{-0.01(x^2+y^2)}$ (Left) without DG , (Right) with DG (5a) Contour plot showing convergence path. (5b) Training loss and convergence rate.



(a) (b)

Figure 6: Convergence plots for function $f(x, y) = \ln(1 + e^x) + 3xy - \ln(1 + e^y)$ (Left) without DG , (Right) with DG (6a) Contour plot showing convergence path. (6a) Training loss and convergence rate.

Table 5: Performance of different training algorithms on function $f(x, y) = \ln(1 + e^x) + 3xy - \ln(1 + e^y)$ with seed at (5, 5).

Algorithms	Iterations	Convergence Point	Loss at Convergence point
GDA	N/A	N/A	N/A
GDA-DG	104	(0.1518,-0.1793)	0.08276
Alt-GDA	998	(0.1518,-0.1792)	0.08276
Alt-GDA-DG	104	(0.1518,-0.1793)	0.08276
OGDA	495	(0.1518,-0.1793)	0.08276
OGDA-DG	44	(0.1518,-0.1793)	0.08276
ExtraGradient	500	(0.1518,-0.1793)	0.08276
ExtraGradient-DG	103	(0.1518,-0.1793)	0.08276
CO	130	(0.1517,-0.1793)	0.08276
CO-DG	97	(0.1517,-0.1793)	0.08276
SGA	118	(0.1517,-0.1793)	0.08276
SGA-DG	99	(0.1518,-0.1793)	0.08276
FTR	N/A	N/A	N/A
FTR-DG	N/A	N/A	N/A

OGDA-DG, ExtraGradient-DG to the optimal point, whereas in case of SGA-DG, FTR-DG and CO-DG the convergence is not so prominent. It also leads to a faster convergence in CO-DG and SGA-DG. It can be observed in the results that in case of FTR-DG the convergence results are not consistent across different functions. One reason is due to the extra term in Follow-the-Ridge algorithm's discriminator update rule that makes the algorithm to diverge.

7. Conclusion

The introduction of the duality gap has shown considerable improvements in various optimization algorithms. It enables non-converging algorithms such as GDA, Alt-GDA, Optimistic GDA, Extragradient, and Consensus Optimization to converge to the optimal point. Additionally, the duality gap improves the performance of Symplectic Gradient Adjustment (SGA) and Follow-The-Ridge (FTR) algorithms, although FTR requires more iterations for convergence compared to other algorithms. Furthermore, the duality gap addresses the issue of slower convergence and eliminates oscillating behavior around the optimal point in algorithms such as GDA, Alternating GDA, Optimistic GDA, Extragradient, and Consensus Optimization. This improvement is particularly evident in equation (iii).

Moreover, the duality gap leads to faster convergence rates in equation (iv) for all the algorithms considered. It also effectively tackles the problem of slow convergence in algorithms like GDA, Alternating GDA, Optimistic GDA, Extragradient, as demonstrated in equation (v). However, it is worth noting that the Follow-the-Ridge algorithm diverges due to the additional term in its discriminator update rule. This highlights a potential limitation or drawback of incorporating the duality gap in certain algorithms.

Future research in this area could explore methods to mitigate the divergence issue in the Follow-the-Ridge algorithm when utilizing the duality gap. Additionally, further investigation could be conducted to understand the underlying mechanisms that enable the duality gap to improve convergence rates and eliminate oscillating behavior. This could lead to the development of enhanced optimization algorithms that leverage the advantages of the duality gap while addressing any associated challenges.

References

[1] I.J. Goodfellow, J.P. Abadie, M. Mirza, B. Xu, D.W. Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

[2] F. Schafer and A. Anandkumar. Competitive gradient descent. *arXiv preprint arXiv:1905.12103*, 2019.

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

[4] F. Farnia and D. Tse. A convex duality framework. *arXiv preprint arXiv:1810.11740*, 2018.

[5] S. Sidheekh, A. Aimen, V. Madan, and N.C. Krishnan. On duality gap as a measure for monitoring gan training. *arXiv preprint arXiv:2012.06723*, 2020.

[6] P. Grnarova, Y. Kilcher, K.Y. Levy, A. Lucchi, and T. Hofmann. Generative minimisation network: Training gans without competition. *arXiv preprint arXiv:2103.12685*, 2021.

[7] S. Sidheekh, A. Aimen, and N.C. Krishnan. On characterizing gan convergence through proximal duality gap. *arXiv preprint arXiv:2105.04801*, 2021.

[8] E. Mazumdar, M.I. Jordan, and S.S. Sastry. On finding local nash equilibria (and only local nash equilibria) in zero-sum continuous games. *arXiv preprint arXiv:1901.00838*, 2019.

[9] X. Chan, J. Wang, and H. Ge. Training gans via primal-dual subgradient methods: A lagrangian perspective on gan. *arXiv preprint arXiv:1802.01765*, 2018.

[10] M. Arjovsky, S. Chintala, and L. Botton. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[11] N. Kodali, J. Abernethy, J. Hays, and Z. Kira. On convergence and stability of gans. *arXiv preprint arXiv:1705.07215*, 2017.

[12] K. Roth, A. Lucchi, S. Nowozin, and T. Hofmann. Stabilizing training of gans through regularization. *arXiv preprint arXiv:1705.09367*, 2017.

[13] P. Grnarova, K. Y. Levy, A. Lucchi, N. Perraudin, I. Goodfellow, T. Hofmann, and A. Krause. A domain agnostic measure for monitoring and evaluating gans. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[14] G. Zhang, Y. Wang, L. Lessard, and R. Grosse. Near-optimal local convergence of alternating gradient descent-ascent for min-max optimization. *arXiv preprint arXiv:2102.09468*, 2021.

[15] A. Mokhtari, A. Ozdaglar, and S. Pattathil. A unified analysis of extragradient and optimistic methods for saddle point problems: Proximal point approach. *arXiv preprint arXiv:1901.08511*, 2019.

[16] C. Daskalis, A. Ilyas, V. Syrgkanis, and H. Zeng. Training gans with optimism. *arXiv preprint arXiv:1711.00141*, 2018.

[17] T. Salimans, H. Zhang, A. Radford, and D. Metaxas. Improving gans using optimal transport. *arXiv preprint arXiv:1803.05573*, 2018.

[18] L. Mescheder, S. Nowozin, and A. Geiger. The numerics of gans. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[19] J. Aberethy, K.A. Lai, and A. Wibisono. Last-iterate convergence rates for min-max optimization. *arXiv preprint arXiv:1906.02027*, 2019.

[20] D. Balduzzi, S. Racaniere, J. Martens, J. Foerster, K. Tuyls, and T. Graepel. The mechanics of n-player differentiable games. *arXiv preprint arXiv:1802.05642*, 2018.

[21] Y. Wang, G. Zhang, and J. Ba. On solving min-max optimization locally: A follow-the-ridge approach. *arXiv preprint arXiv:1910.07512*, 2019.