Extending Convergence Studies of GAN Training Algoritims 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, Optimization algorithms, Convergence rate, Comparative Analysis

1. Introduction

Generative Adversarial Networks (GANs) have become increasingly popular as advanced models for creating realistic and high-quality synthetic data [1]. The fundamental concept behind GANs is to train a generator network with the objective of producing synthetic data samples that takes random noise as input that are virtually indistinguishable from real 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. GANs often encounter mode collapse, which occurs when the generator produces a limited variety of samples, failing to capture the full diversity of 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.

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.

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. The fundamental concept underlying GANs involves training two neural networks, namely the generator and the discriminator, to engage in a competitive two-player min-max game.

In this game, the generator network receives a random noise vector as input and generates synthetic data samples. On the other hand, the discriminator network receives both real and fake data samples as input and outputs a probability value indicating the authenticity of the input (whether it is real or fake). The generator's objective is to generate data samples that are convincingly similar to real data, while the discriminator aims to accurately discriminate between real and fake samples.

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. 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. Another issue with GANs is instability during training. This instability arises when the generator and discriminator, due to oscillatory behavior, struggle to find the optimal balance. 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 [1]. 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 [3].

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[9] [10]. 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] [8] 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.[8] 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]. The update rule for Duality Gap is given by:

$$\theta_g^{(k+1)} = \theta_g^{(k)} - \alpha \nabla_{\theta_g} DG(\theta_g^{(k)}, \theta_d^{(k)}) \tag{1}$$

$$\theta_d^{(k+1)} = \theta_d^{(k)} + \alpha \nabla_{\theta_d} DG(\theta_g^{(k)}, \theta_d^{(k)}) \tag{2}$$

where,

- θ_g represents the weight of the generator network.
- θ_d represents the weight of the discriminator network.
- α denotes the learning rate, which determines the step size for parameter updates.
- $DG(\theta_g(k), \theta_d(k))$ is the dual objective function, which is calculated as the difference between the objective function J evaluated at $\theta_g(k)$, $\theta_{d_w}(t)$ and the objective function J evaluated at $\theta_{g_w}(t)$, $\theta_d(k)$.
- J represents the game objective, which is the overall objective function of the GAN.
- $\theta_{g_w}(t)$, $\theta_{d_w}(t)$ represents the worst weights after t iterations, which are used in the dual objective function calculation.

Eq.1 updates the weight of the generator (θ_g) at iteration ($k+1$) by subtracting the learning rate (α) multiplied by the gradient of the dual objective function (DG) with respect to θ_g at the current iteration (k). Eq.2 updates the weight of the discriminator (θ_d) at iteration ($k+1$) by adding the learning rate (α) multiplied by the gradient of the dual objective function (DG) with respect to θ_d at the current iteration (k).

3. Extending Training Algorithms with Duality Gap

3.1. Gradient Descent Ascent (GDA)

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:

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

$$\theta_d^{(k+1)} = \theta_d^{(k)} + \alpha \nabla_{\theta_d} J_d(\theta_g^{(k)}, \theta_d^{(k)}) \tag{4}$$

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, aiming to enhance its ability to distinguish between real and fake samples. This alternating update process continues until convergence is achieved, or a predefined stopping criterion is met.

While GDA is a commonly used optimization algorithm for GANs, it can encounter challenges such as vanishing gradients, where the gradients become extremely small and hinder effective learning, and mode collapse, where the generator fails to capture the full diversity of the real data distribution. Researchers employ various techniques, such as architectural modifications, regularization methods, and advanced optimization algorithms, to mitigate these issues and improve the training stability and quality of GANs.

Algorithm 1 Applying Duality Gap on GDA

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

3.2. Alternating Gradient Descent Ascent (Alt-GDA)

In alternating GDA, the generator and discriminator networks are trained in alternating steps. During each step, the generator updates its weights based on the gradient of the generator loss, and the discriminator updates its weights based on the gradient of the discriminator loss.

The update rule for GDA is given by:

$$\theta_g^{(k+1)} = \theta_g^{(k)} - \alpha \nabla_{\theta_g} J_g(\theta_g^{(k)}, \theta_d^{(k)}) \tag{5}$$

$$\theta_d^{(k+1)} = \theta_d^{(k)} + \alpha \nabla_{\theta_d} J_d(\theta_g^{(k+1)}, \theta_d^{(k)}) \tag{6}$$

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.

Alternating GDA (Alt-GDA) achieves a near-optimal local convergence rate for strongly convex-strongly concave (SCSC) minimax problems, which is quadratically better than GDA and matches extra-gradient (EG) and optimistic GDA (OGDA). Alt-GDA attains linear convergence when the minimax problem has only strong concavity in one player but no strong convexity in the other player, under a non-singularity condition on the coupling matrix.

3.3. Optimistic GDA (OGDA)

Optimistic Gradient Descent Ascent (OGDA) is an extension of gradient descent/ascent that incorporates optimism to

Algorithm 2 Applying Duality Gap on Alt-GDA

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

improve the optimization process in training Generative Adversarial Networks (GANs) [17]. 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:

$$\theta_g^{(k+1)} = \theta_g^{(k)} - 2\alpha \nabla_{\theta_g} J_g(\theta_g^{(k)}, \theta_d^{(k)}) + \alpha \nabla_{\theta_g} J_g(\theta_g^{(k-1)}, \theta_d^{(k-1)}) \tag{7}$$

$$\theta_d^{(k+1)} = \theta_d^{(k)} + 2\alpha \nabla_{\theta_d} J_d(\theta_d^{(k)}, \theta_g^{(k)}) - \alpha \nabla_{\theta_d} J_d(\theta_g^{(k-1)}, \theta_d^{(k-1)}) \tag{8}$$

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

By incorporating optimism and utilizing information from previous iterations, OGDA enhances the exploration of the parameter space, which can lead to improved convergence and better avoidance of local optima during GAN training.

Algorithm 3 Applying Duality Gap on Optimistic GDA

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

3.4. ExtraGradient

Extragradient is an optimization algorithm employed in training Generative Adversarial Networks (GANs) that utilizes a double update scheme [17]. 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:

$$\theta_g^{(k+1)} = \theta_g^{(k)} - \alpha \nabla_{\theta_g} J_g(\theta_g^{(k)} - \alpha \nabla_{\theta_g} J_g(\theta_g^{(k)}, \theta_d^{(k)}), \theta_d^{(k)} + \alpha \nabla_{\theta_d} J_d(\theta_g^{(k)}, \theta_d^{(k)})) \tag{9}$$

$$\theta_d^{(k+1)} = \theta_d^{(k)} + \alpha \nabla_{\theta_d} J_d(\theta_g^{(k)} - \alpha \nabla_{\theta_g} J_g(\theta_g^{(k)}, \theta_d^{(k)}), \theta_d^{(k)} + \alpha \nabla_{\theta_d} J_d(\theta_g^{(k)}, \theta_d^{(k)})) \tag{10}$$

In the Extragradient update rules, the generator's parameters are updated by extrapolating the current update using the next update's gradients. Similarly, the discriminator's parameters 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. By incorporating the extrapolation step, Extragradient 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.

Algorithm 4 Applying Duality Gap on ExtraGradient

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

3.5. Consensus Optimization (CO)

Consensus Optimization (CO) is a distributed optimization algorithm specifically designed for training large-scale Generative Adversarial Networks (GANs) on multiple machines or devices [14]. It enables parallel training of multiple copies of the generator and discriminator networks, allowing for efficient utilization of computational resources. CO involves periodically exchanging the parameters of these networks to ensure consistent training across devices. The key idea behind CO is to enforce agreement among the networks through parameter

averaging during the parameter exchange steps, known as consensus steps. Each copy of the network computes an average of the parameter values from the other copies, helping to synchronize the network parameters and ensure consistent learning across devices.

The update rule for Consensus Optimization is given by Eq.11 and Eq.12:

$$\theta_g^{(k+1)} = \theta_g^{(k)} - \alpha\nabla_{\theta_g} J_g(\theta_g^{(k)}, \theta_d^{(k)}) - \gamma\eta\nabla\|\nabla J(\theta_g^{(k)}, \theta_d^{(k)})\|^2 \tag{11}$$

$$\theta_d^{(k+1)} = \theta_d^{(k)} + \alpha\nabla_{\theta_d} J_d(\theta_g^{(k)}, \theta_d^{(k)}) - \gamma\eta\nabla\|\nabla J(\theta_g^{(k)}, \theta_d^{(k)})\|^2 \tag{12}$$

$\nabla\|\nabla J(\theta_g^{(k)}, \theta_d^{(k)})\|^2$ is the regularization term to enforce the constraint. By leveraging consensus steps 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.

3.6. Symplectic Gradient Adjustment (SGA)

Symplectic Gradient Adjustment (SGA) 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.13 and Eq.14:

$$\theta_g^{(k+1)} = \theta_g^{(k)} - \alpha\nabla_{\theta_g} J_g(\theta_g^{(k)}, \theta_d^{(k)}) - \alpha\lambda\mathbf{H}_{xy}\nabla_{\theta_d} J_g(\theta_g^{(k)}, \theta_d^{(k)}) \tag{13}$$

Algorithm 5 Applying Duality Gap on Consensus Optimization

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: $DG(u_t, v_t) = M(u_t, v_{w_k}) - M(u_{w_k}, v_t)$
  $u_{t+1} \leftarrow u_t - \eta_t\nabla_u DG(u_t, v_t) - 2\gamma\mathbf{H}(f(x,y))\cdot\nabla M(u_t, v_t)$
  $v_{t+1} \leftarrow v_t - \eta_t\nabla_v DG(u_t, v_t) - 2\gamma\mathbf{H}(f(x,y))\cdot\nabla M(u_t, v_t)$
end for

$$\theta_d^{(k+1)} = \theta_d^{(k)} + \alpha\nabla_{\theta_d} J_d(\theta_g^{(k)}, \theta_d^{(k)}) - \alpha\lambda\mathbf{H}_{yx}\nabla_{\theta_g} J_d(\theta_g^{(k)}, \theta_d^{(k)}) \tag{14}$$

In SGA, the update rules incorporate symplectic matrices which play a crucial role in preserving the volume and shape of the underlying optimization problem. By adjusting the gradients with these symplectic operations, SGA aims to enhance exploration in the parameter space, thereby improving the overall convergence behavior of GANs.

The use of symplectic integrators helps maintain the geometry and conservation properties of the optimization problem, which can lead to more robust and efficient training. By preserving the volume and shape, SGA facilitates the exploration of different regions of the parameter space, potentially enabling the discovery of better optima and avoiding convergence to poor local optima.

Algorithm 6 Applying Duality Gap on SGA

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

3.7. Follow-The-Ridge (FTR)

Follow-the-Ridge[20] 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.15 and Eq.16:

$$\theta_g^{(k+1)} = \theta_g^{(k)} - \alpha_x \nabla_{\theta_g} J_g(\theta_g^{(k)}, \theta_d^{(k)}) \tag{15}$$

$$\begin{aligned}\theta_d^{(k+1)} = \theta_d^{(k)} &+ \alpha \nabla_{\theta_d} J_d(\theta_g^{(k)}, \theta_d^{(k)}) \\ &+ \alpha_y \mathbf{H}_{yy}^{-1} \mathbf{H}_{yx} \nabla_{\theta_g} J_d(\theta_g^{(k)}, \theta_d^{(k)})\end{aligned} \tag{16}$$

In Follow-the-Ridge algorithm, the generator's parameters are updated according to the standard gradient descent rule. However, for the discriminator, an additional term is included in the update rule, which involves the inverse of the Hessian matrix (H_{yy}), the Hessian matrix between the generator and discriminator (H_{yx}), and the gradient of the generator objective function ($\nabla_{\theta_g} J_g$).

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.

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.

Algorithm 7 Applying Duality Gap on FTR

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

4. Experimental Setup

The experiments carried in this paper are carried out in following five toy equations considered in various experimentation related to GAN training [20] [21] :

(i) $-3x^2 - y^2 + 4xy$ [20]
(ii) $3x^2 + y^2 + 4xy$ [20]
(iii) $(4x^2 - (y - 3x + 0.05x^3)^2 - 0.1y^4)e^{-0.01(x^2+y^2)}$ [20]
(iv) $((0.3x^2 + y)^2 + (0.1y^2 + x)^2) * e^{-0.01(x^2+y^2)}$ [21]
(v) $\ln(1 + e^x) + 3xy - \ln(1 + e^y)$ [21]

The algorithms under investigation are Gradient Descent Ascent (GDA), Alternating GDA (AGDA), Optimistic GDA (OGDA), ExtraGradient (EG), Consensus Optimization (CO), Symplectic Gradient Adjustment (SGA), Follow-the-Ridge, and the incorporation of Duality Gap (DG) to each algorithm. The experiments are carried out using Python and the Autograd library. 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 equations.

Table 1: Performance of different training algorithms on equation (i) with seed at (5, 7). First row of each algorithm represents the performance without Duality Gap and the second represents with Duality Gap

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

5. Observations & Result Analysis

The improvement made by duality gap can be observed in equation (i) that non-converging algorithms like GDA, Alt-GDA, Optimistic GDA, Extragradient are able to converge to the optimal point. Consensus Optimization and Symplectic Gradient Adjustment shows an improvement over the performance while Follow-The-Ridge takes more iteration for convergence when extended with Duality Gap.

Equation (ii) demonstrates a slower rate of convergence as they approach a non-optimal point (as it is nor a min-max point); however Follow-the-Ridge avoid such point. Notably, equation (iii) presents a significant improvement by eliminating the oscillating behavior around the optimal point in GDA, Alternating GDA, Optimistic GDA, Extragradient, CO. Introduction of Duality Gap also enables the faster rate of convergence in SGA and FTR.

Equation (iv) displays an improvement in terms of a faster convergence rate to the optimal point in all the algorithms except CO with the application of duality gap. Furthermore, equation (v) exhibits a substantial enhancement, as it effectively ad-

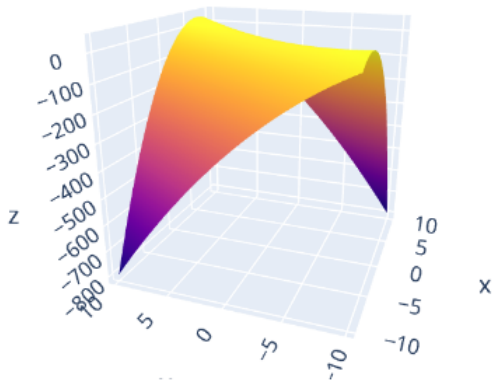

Figure 1: Surface Map of Equation (i)

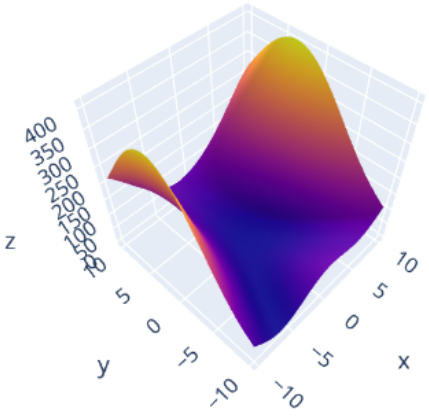

Figure 4: Surface Map of Equation (iv)

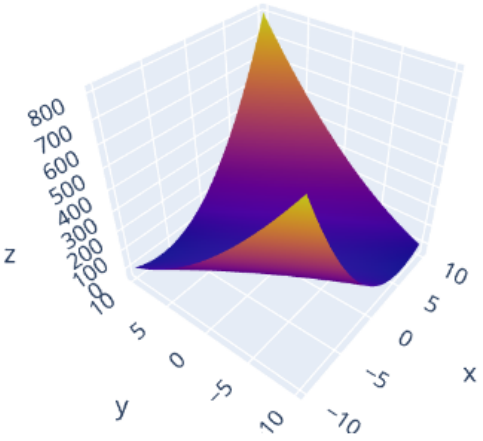

Figure 2: Surface Map of Equation (ii)

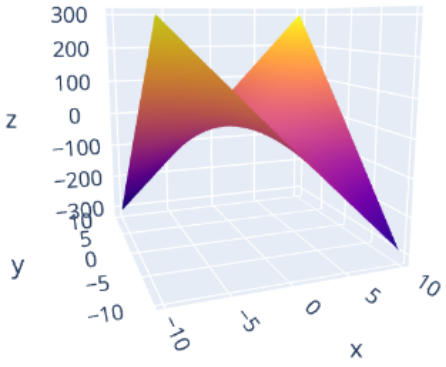

Figure 5: Surface Map of Equation (v)

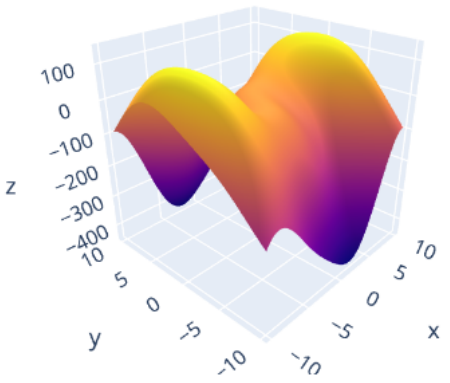

Figure 3: Surface Map of Equation (iii)

Table 2: Performance of different training algorithms on equation (ii) with seed at (6, 5). First row of each algorithm represents the performance without Duality Gap and the second represents with Duality Gap

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

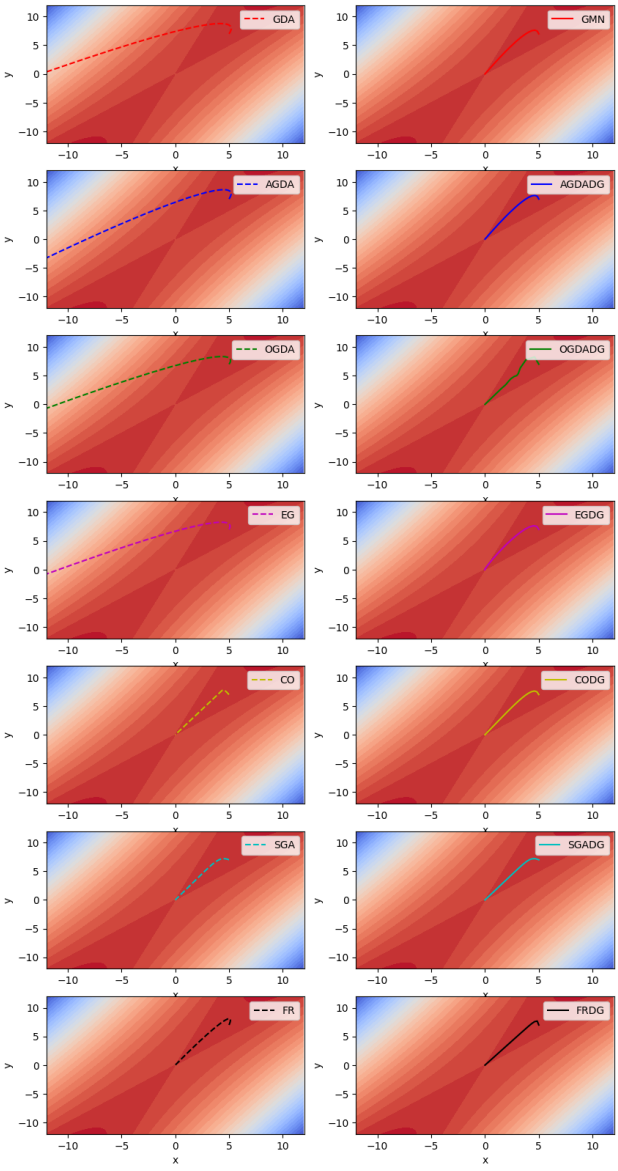

Figure 6: Paths taken on the Contour Map of Equation (i) with seed at (5,7); Left represents the rates without Duality Gap and right represents graph with Duality Gap

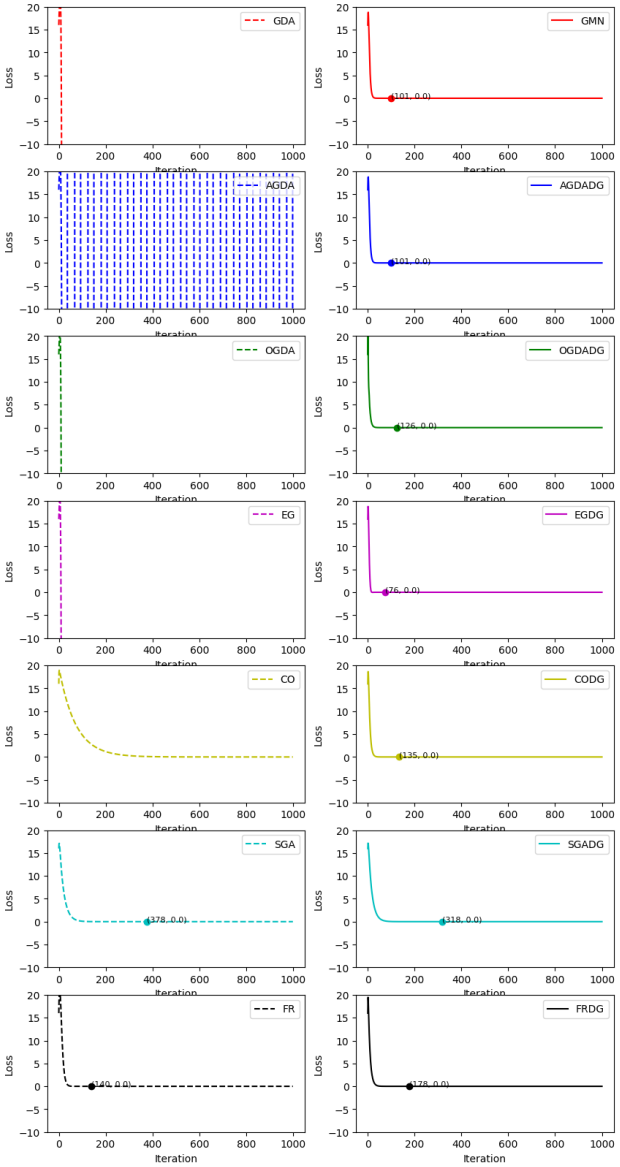

Figure 7: The convergence rate of Equation (i) with seed at (5,7); Left represents the rates without Duality Gap and right represents graph with Duality Gap

Table 3: Performance of different training algorithms on equation (iii) with seed at (7, 5). First row of each algorithm represents the performance without Duality Gap and the second represents with Duality Gap

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

Table 4: Performance of different training algorithms on equation (iv) with seed at (5, 5). First row of each algorithm represents the performance without Duality Gap and the second represents with Duality Gap

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

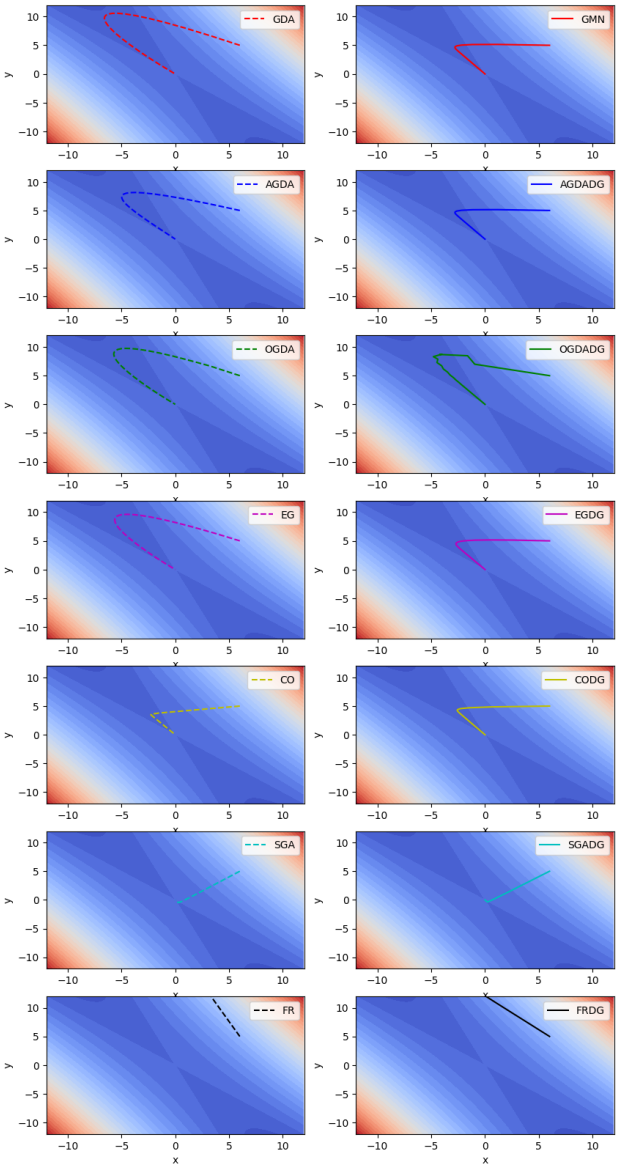

Figure 8: Paths taken on the Contour Map of Equation (ii) with seed at (6, 5); Left represents the rates without Duality Gap and right represents graph with Duality Gap

Table 5: Performance of different training algorithms on equation (v) with seed at (5, 5). First row of each algorithm represents the performance without Duality Gap and the second represents with Duality Gap

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

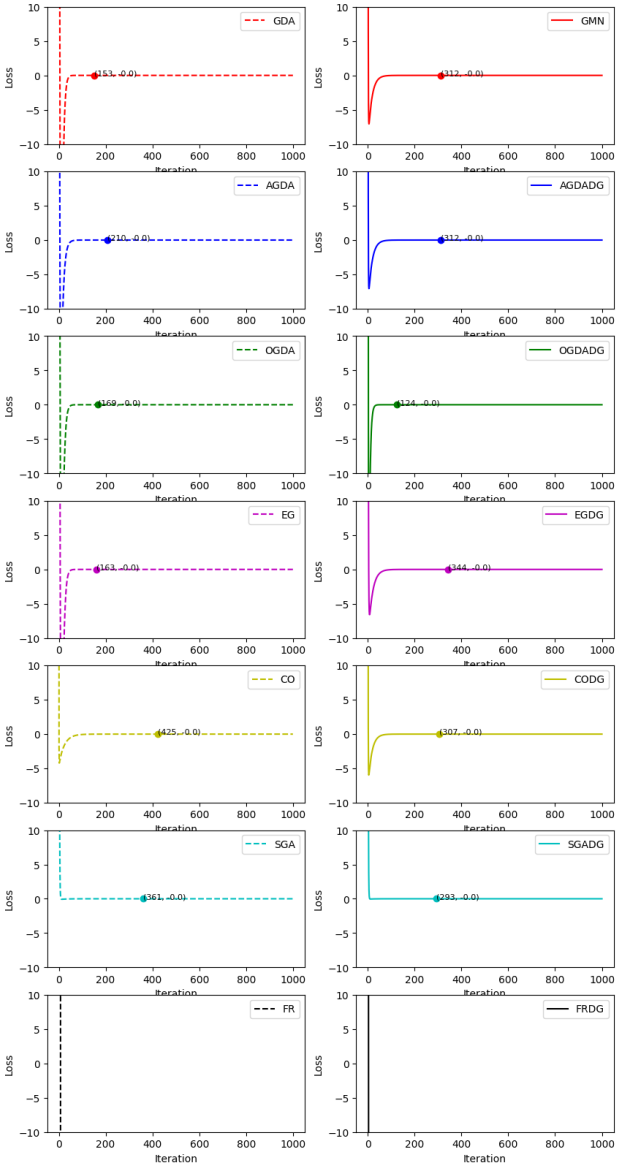

Figure 9: The convergence rate of Equation (ii) with seed at (6, 5); Left represents the rates without Duality Gap and right represents graph with Duality Gap

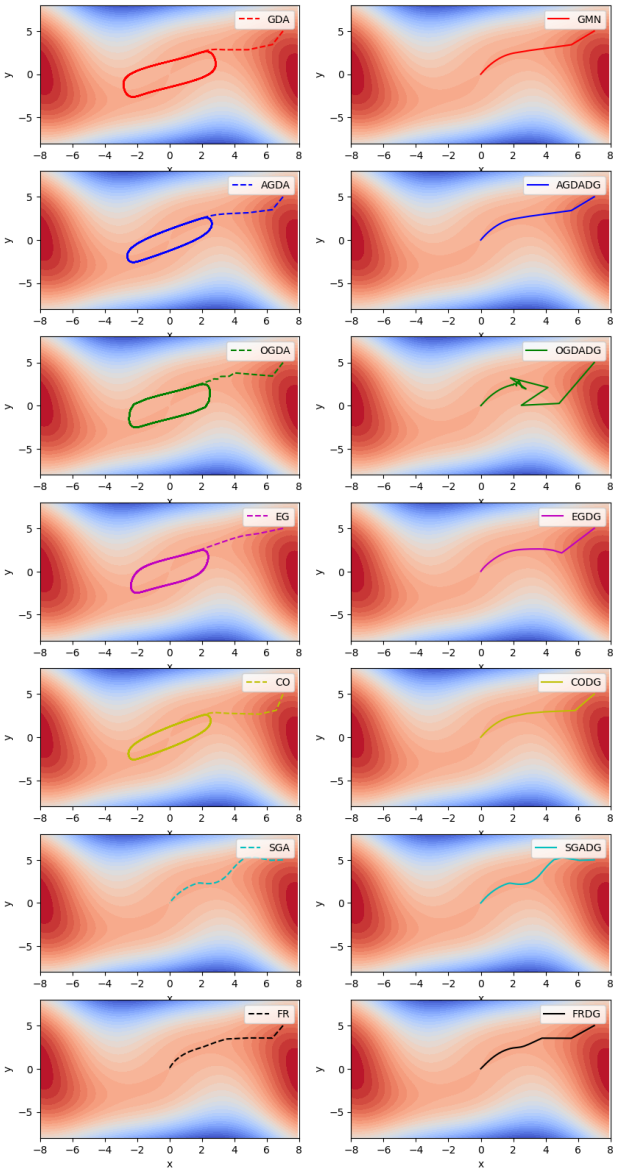

Figure 10: Paths taken on the Contour Map of Equation (iii) with seed at (7, 5); Left represents the rates without Duality Gap and right represents graph with Duality Gap

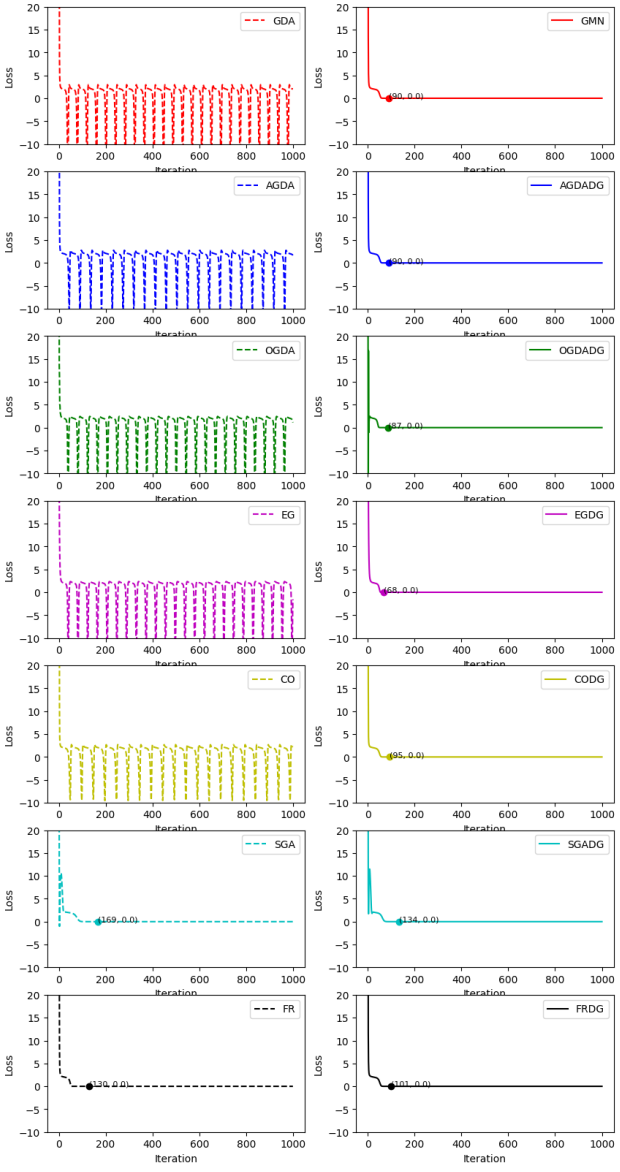

Figure 11: The convergence rate of Equation (iii) with seed at (7, 5); Left represents the rates without Duality Gap and right represents graph with Duality Gap

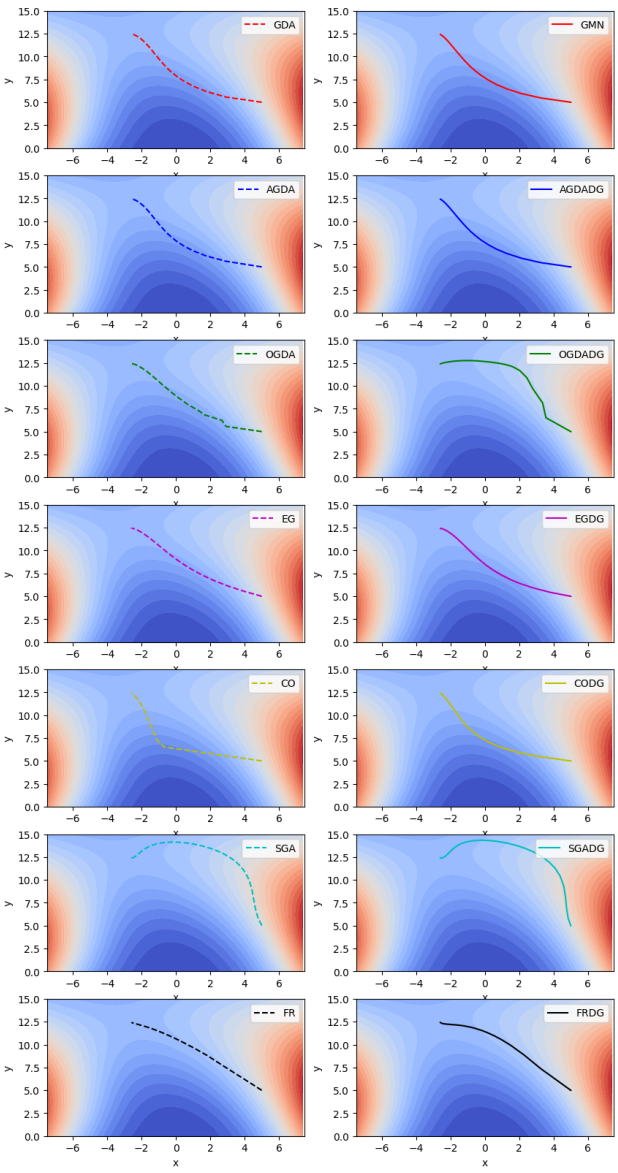

Figure 12: Paths taken on the Contour Map of Equation (iv) with seed at (5, 5); Left represents the rates without Duality Gap and right represents graph with Duality Gap

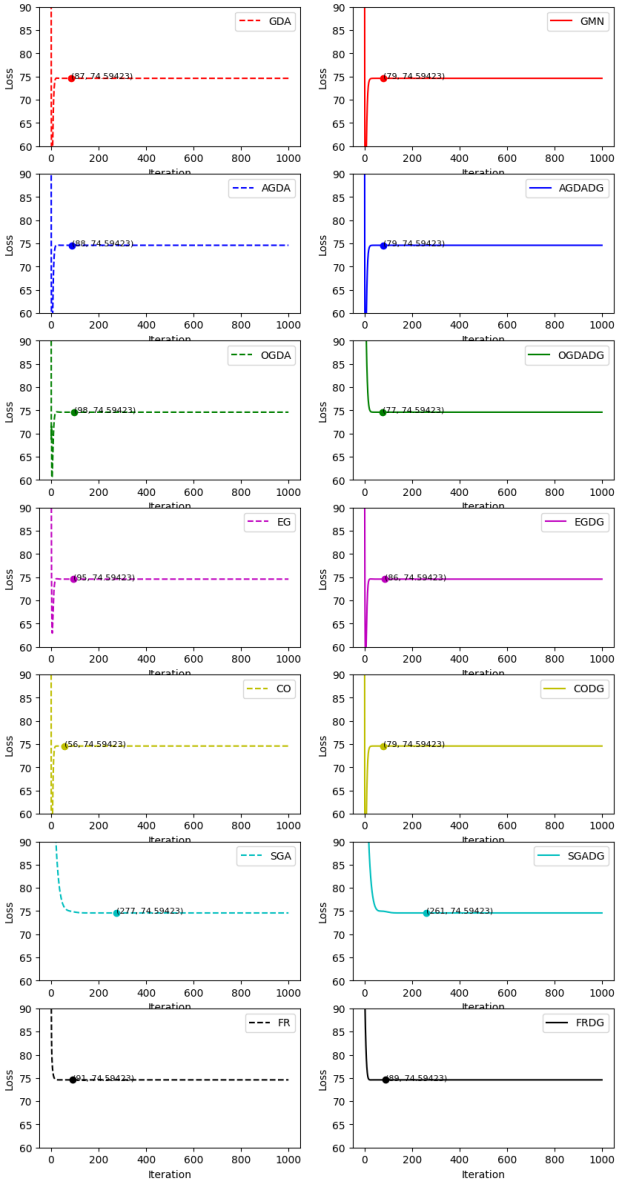

Figure 13: The convergence rate of Equation (iv) with seed at (5, 5); Left represents the rates without Duality Gap and right represents graph with Duality Gap

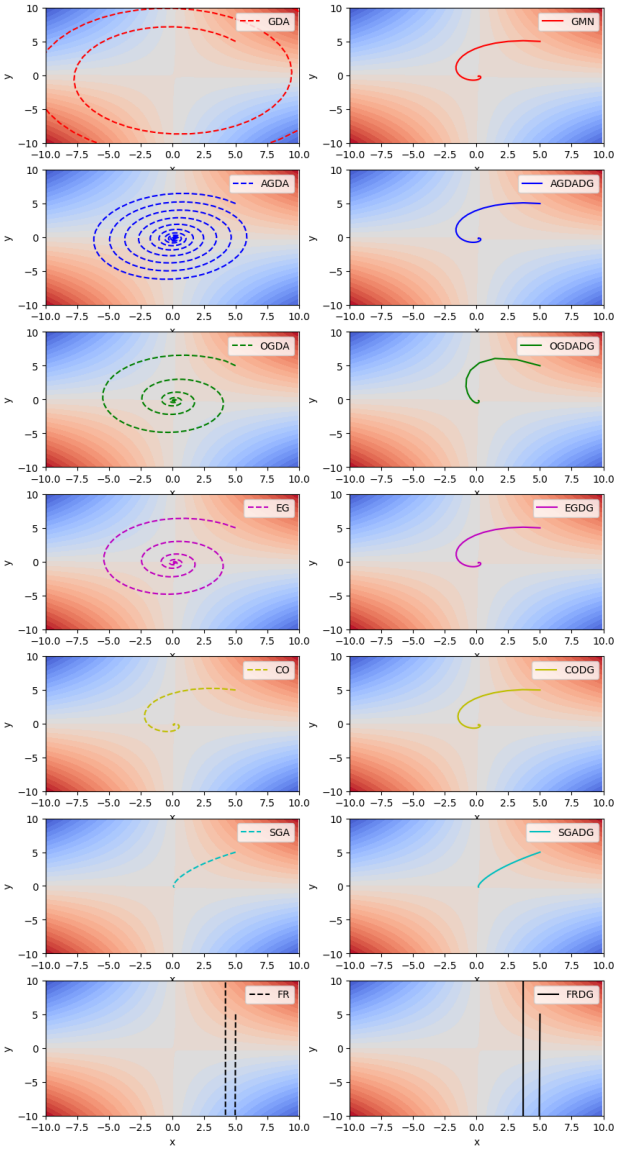

Figure 14: Paths taken on the Contour Map of Equation (v) with seed at (5, 5); Left represents the rates without Duality Gap and right represents graph with Duality Gap

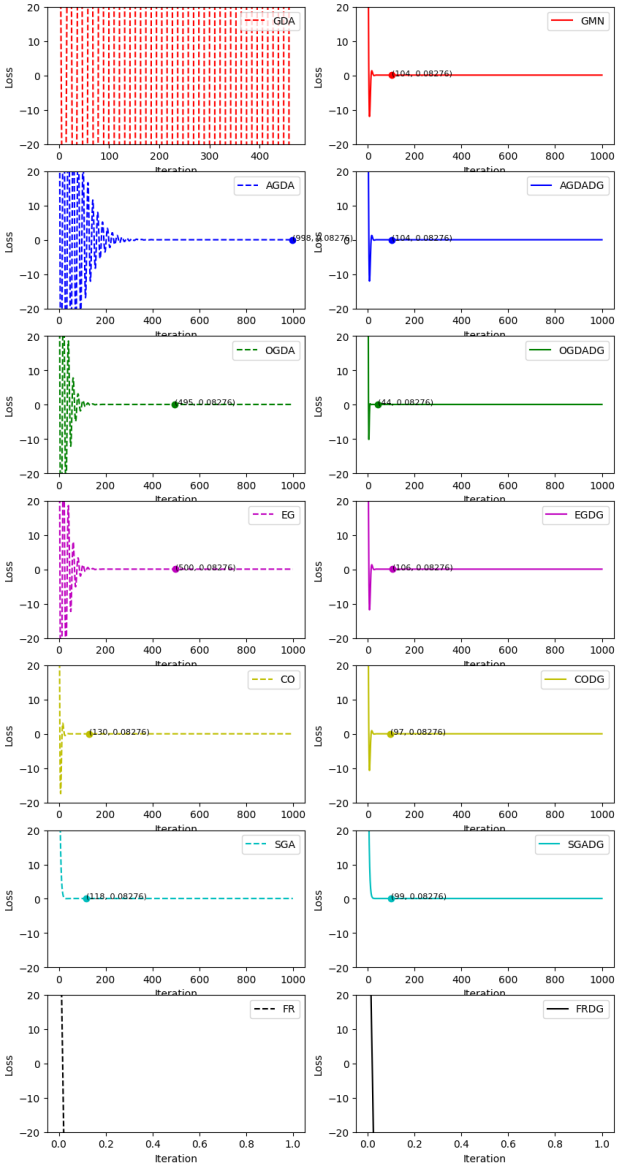

Figure 15: The convergence rate of Equation (v) with seed at (5, 5); Left represents the rates without Duality Gap and right represents graph with Duality Gap

dresses the issue of slow convergence in algorithms like GDA, Alternating GDA, Optimistic GDA, Extragradient. It also leads to a faster convergence in CO and SGA. The extra term in Follow-the-Ridge algorithm's discriminator update rule makes the algorithm to diverge.

6. Conclusion

In conclusion, the introduction of the duality gap has shown significant 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.

Acknowledgements

We are profoundly grateful to DR. MAINAK BANDYOPADHYAY of KIIT University for his expert guidance and continuous encouragement throughout.

References

[1] I.J. Goodfellow, J.P. Abadie, M. Mirza, B. Xu, D.W. Farley, S. Ozair, A. Courville, Y. Bengio. Generative Adversarial Networks. arXivpreprint arXiv:1406.2661, 2014.

[2] L. Metz, B. Poole, D. Pfau, and J. Sohl Dickstein. Unrolled Generative Adversarial Networks. arXivpreprint arXiv:1611.02163, 2016.

[3] M. Arjovsky, S. Chintala, L. Botton. Wasserstein GAN. arXivpreprint arXiv:1701.07875, 2017.

[4] F. Farnia, D. Tse. A Convex Duality Framework. arXivpreprint arXiv:1810.11740, 2018.

[5] S. Sidheekh, A. Aimen, V. Madan, N.C. Krishnan. On Duality Gap as a Measure for Monitoring GAN Training. arXivpreprint arXiv:2012.06723, 2020.

[6] P. Grnarova, Y. Kilcher, K.Y. Levy, A. Lucchi, T. Hofmann. Generative Minimisation Network: Training GANs Without Competition. arXivpreprint arXiv:2103.12685, 2021.

[7] S. Sidheekh, A. Aimen, N.C. Krishnan. On Characterizing GAN Convergence Through Proximal Duality Gap. arXivpreprint arXiv:2105.04801, 2021.

[8] Grnarova, P., Levy, K. Y., Lucchi, A., Perraudin, N., Goodfellow, I., Hofmann, T., Krause, A. (2019). A domain agnostic measure for monitoring and evaluating GANs. Advances in Neural Information Processing Systems, 32.

[9] N. Kodali, J. Abernethy, J. Hays, Z. Kira. On Convergence and Stability of GANs. arXivpreprint arXiv:1705.07215, 2017.

[10] K. Roth, A. Lucchi, S. Nowozin, T. Hofmann. Stabilizing Training of GANs through Regularization. arXivpreprint arXiv:1705.09367, 2017.

[11] C. Daskalis, A. Ilyas, V. Syrgkanis, H. Zeng. Training GANs With Optimism. arXivpreprint arXiv:1711.00141, 2018.

[12] X. Chan, J. Wang, H. Ge. Training GANs via Primal-Dual Subgradient Methods: A Lagrangian Perspective on GAN. arXivpreprint arXiv:1802.01765, 2018.

[13] D. Balduzzi, S. Racaniere, J. Martens, J. Foerster, K. Tuyls, T. Graepel. The Mechanics of n-player Differentiable Games. arXivpreprint arXiv:1802.05642, 2018.

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

[15] T. Salimans, H. Zhang, A. Radford, D. Metaxas. Improving GANs Using Optimal Transport. arXivpreprint arXiv:1803.05573, 2018.

[16] E. Mazumdar, M.I. Jordan, S.S. Sastry. On Finding Local Nash Equilibria (and Only Local Nash Equilibria) in Zero-Sum Continous Games. arXivpreprint arXiv:1901.00838, 2019.

[17] A. Mokhtari, A. Ozdaglar, S. Pattathil. A Unified Analysis of Extra-Gradient and Optimistic Methods for Saddle Point Problems: Proximal Point Approach. arXivpreprint arXiv:1901.08511, 2019.

[18] F. schafer, A. Anandkumar. Competitive Gradient Descent. arXivpreprint arXiv:1905.12103, 2019.

[19] J. Aberethy, K.A. Lai, A. Wibisono. Last-iterate Convergence Rates for Min-Max Optimisation. arXivpreprint arXiv:1906.02027, 2019.

[20] Y. Wang, G. Zhang, J. Ba. On Solving Min-max Optimisation Locally: A Follow-The-Ridge Approach. arXivpreprint arXiv:1910.07512, 2019.

[21] G. Zhang, Y. Wang, L. Lessard, R. Grosse. Near-optimal Local Convergence of Alternating Gradient Descent-Ascent for Min-max Optimization. arXivpreprint arXiv:2102.09468, 2022.

Appendix A. Analysing Updates in SGD

Taking a Convex-concave function like

$$f(x, y) = cxy, \quad c \in \{3, 10\} \tag{A.1}$$

The SGD updates for the game Equation A.1 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.

Appendix B. Analysing Updates in SGD using Duality Gap

The duality gap function can be defined as $dg(x, y) = c(x \cdot y_w - x_w \cdot y)$ and the corresponding updates are:

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

where

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

Hence, the combined updates are:

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

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 $\eta < \frac{2kc-1}{k^2c^2}$

Appendix C. Analysing Updates in Alternating GDA using Duality Gap

The duality gap function can be defined as $dg(x, y) = c(x \cdot y_w - x_w \cdot y)$ and the corresponding updates are:

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

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.

Appendix D. Analysing Updates in Optimistic GDA using Duality Gap

The duality gap function can be defined as $dg(x, y) = c(x \cdot y_w - x_w \cdot y)$ and the corresponding updates are:

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

where

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

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

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

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

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. $\sqrt{(-\eta^2 c(2k + a) + 1)^2 + \eta^2 (2 - a)^2} < 1$.

Appendix E. Analysing Updates in Extragradient using Duality Gap

The duality gap function can be defined as $dg(x, y) = c(x \cdot y_w - x_w \cdot y)$ and the corresponding updates are:

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

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.

Appendix F. Analysing Updates in CO using Duality Gap

The duality gap function can be defined as $dg(x, y) = c(x \cdot y_w - x_w \cdot y)$ and the corresponding updates are:

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

where

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

Hence, the combined updates are:

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

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. $\sqrt{(-\eta^2 kc + 2\gamma + 1)^2 + \eta^2} < 1$

Appendix G. Analysing Updates in SGA using Duality Gap

The duality gap function can be defined as $dg(x, y) = c(x \cdot y_w - x_w \cdot y)$ and the corresponding updates are:

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

where

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

Hence, the combined updates are:

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

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. $\sqrt{(-\eta^2 kc + \eta\lambda + 1)^2 + \eta^2} < 1$

Appendix H. Analysing Updates in FTR using Duality Gap

The duality gap function can be defined as $dg(x, y) = c(x \cdot y_w - x_w \cdot y)$ and the corresponding updates are:

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

where

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

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.