# Applying Dual-Averaging Sub-gradient method to saddle point problem of Generative adversarial networks

**Amol Damare**
Department of Computer Science
Stony Brook University
SBU Id:107914028
adamare@cs.stonybrook.edu

**Arjun Krishna**
Department of Computer Science
Stony Brook University
SBU Id:111753053
arjkrishna@cs.stonybrook.edu

## Abstract

Training of Generative Adversarial Networks is hard. It is one of the open research problem in the field of deep learning. The objective of GAN is modeled as saddle point problem. In this paper we have applied already existing algorithms for saddle point problem from optimization theory to GAN. Specifically we have implemented SDA method proposed by (**?**). Code for this project can be found at `https://github.com/amoldamare/CSE-592-Convex-Optimization-Project`

## 1 Introduction

"Generative Adversarial Networks " proposed by **?** is generative model that uses 2 feed-forward neural networks which compete with each other to produce a generative model that approximates the real distribution. The results produced by GAN's are realistic and very sharp. Hence they have gained success as a Generative model. But training of GAN is very unstable and is still a open research problem.
For our class project we will try to apply one of the convex optimization method used to solve saddle point problems to GAN training. This report will present the work we have done so far on this problem. In section 2 we will present the overview of literature survey we did for this project so far. The section 2 is same as what we submitted in project progress report. In section 3 we will present the algorithm we used and the assumptions we made to implement this algorithm. We will present our results in section 4, followed by conclusion and future work.

## 2 Literature survey

**?** first presented the idea of Generative Adversarial Networks, in which the 2 feed forward neural networks "Discriminator" and "Generator" compete with each other by trying to fool one another. Both discriminator and generator are trained simultaneously using loss function:

$$L(\theta_D, \theta_G) = \max_{\theta_D} \min_{\theta_G} \mathbb{E}_{x\sim P_d}[\log(D_{\theta_D}(x)] + \mathbb{E}_{z\sim P_z}[\log(1 - D_{\theta_D}(G_{\theta_G}(z))] \tag{1}$$

$$= \max_{\theta_D} \min_{\theta_G} \frac{1}{m} \sum_{i=1}^{m} (\log(D_{\theta_D}(x_i) + \log(1 - D_{\theta_D}(G_{\theta_G}(z_i))) \tag{2}$$

Where $D$ i the discriminator parameterized by $\theta_D$ and $G$ is the generator network parameterized by $\theta_G$. $P_d$ is the probability distribution of the samples and $P_z$ is some prior distribution; **?** used a Gaussian prior. Stochastic gradient descent is used to find the optimal parameter values. Detailed algorithm in **?** is given in algorithm **??** There are number of loss functions that are used in practice

**Algorithm 1** Algorithm for GAN

For training iteration T:
For k iterations:
Train Discriminator using S.G.D. by using $\nabla_{\theta_D} L(\theta_D, \theta_G)$
End For
Train Generator using S.G.D. by using $\nabla_{\theta_G} L(\theta_D, \theta_G)$
End For
return $\theta_D, \theta_G$

giving rise to a different flavor of GAN **???**. But all of them use variation basic algorithm **??** to find optimal parameters.
An interesting approach has been presented in **?**. GAN loss function was modeled after Lagrangian dual function and a variation of algorithm **??** was proposed that will optimize this loss function. Although its an interesting idea, we found that formulation of Lagrangian dual function was not correct and algorithm proposed a different updates for generator network which were not derived from loss function, which we think is a mistake.

In optimization theory various methods have been proposed to solve saddle point problem of the form

$$\min_u \max_v f(u, v) \tag{3}$$

Dual-averaging sub gradient method **?**, stochastic accelerated primal dual method **?** are some of the methods that we looked at. For this project we will focus on dual-averaging method proposed in **?**

# 3 Our approach

## 3.1 Optimization Algorithm

We are going to implement dual averaging sub-gradient method proposed in **?** to solve GAN optimization problem. This method finds an optimum solution for problem **??** within given bounds.

For this algorithm we assume $u \in U$ , $U$ is convex compact set, $v \in V$ , $V$ is also a convex compact set. We also assume $f(u, .)$ is a convex problem in $u$ and $f(., v)$ is a concave problem in $v$. Let $d_u$ and $d_v$ be convex, continuous prox functions defined on $U$ and $V$ respectively.We also have following sub-gradients

$$g_u \in \partial_u f(u, v)$$
$$g_v \in \partial_v f(u, v)$$

we also have

$$g = (g_u, -g_v)$$

Let optimal solution to problem **??** exists

$$x^* = (u^*, v^*)$$

where $x^* \in UXV = Q$. We also define, for some $\alpha \in 0, 1$

$$d(x) = \alpha d_u(u) + (1 - \alpha) d_v(v)$$

$$||x|| = (\alpha \sigma_u ||u||_2^2 + (1 - \alpha)\sigma_v ||v||_2^2)^{\frac{1}{2}}$$

Where are $\sigma_u$ and $\sigma_v$ are convexity constants of $d_u$ and $d_v$ respectively. We also define a projection function as follows:

$$\pi_G(s) = \underset{x \in Q}{\text{argmin}} \{- < s, x > + G d(x)\}$$

We define

$$x_0 = \underset{x \in Q}{\text{argmin}} \{d(x)\}$$

Now we can obtain optimal solution to problem **??** by using algorithm **??**. For convergence analysis

**Algorithm 2** Algorithm for Dual-Averaging for Saddle point problem

1: $S_0 = 0$
2: $\hat{\beta}_0, \hat{\beta}_1 = 1$
3: Choose $\gamma > 0$
4: For k=0..T
5: Compute

$$g_k = (g_{u_k}, -g_{v_k})$$

6: Compute

$$S_{k+1} = S_k + g_k$$

7: Compute

$$\hat{\beta_{k+1}} = \sum_{i=0}^{k} \frac{1}{\hat{\beta}_i}$$

8: Compute

$$\beta_{k+1} = \gamma \hat{\beta_{k+1}}$$

9: Compute

$$x_{k+1} = \pi_{\beta_{k+1}}(-S_{k+1})$$

10: End For
11: return $x_T$

of algorithm **??** we refer you to **?**.
Now to implement algorithm **??** we made some assumptions. We assumed following:

$$d_u(x) = \frac{1}{2}||x||_2^2 \tag{4}$$

and

$$d_v(x) = \frac{1}{2}||x||_2^2 \tag{5}$$

These prox functions will give us following results

$$\begin{aligned} x_0 &= \underset{x \in Q}{\text{argmin}}\{d(x)\} \\ &= \underset{u \in U, v \in V}{\text{argmin}} \{\alpha d_u(u) + (1-\alpha) d_v(v)\} \\ &= \underset{u \in U, v \in V}{\text{argmin}} \{\frac{\alpha}{2}||u||_2^2 + \frac{(1-\alpha)}{2}||v||_2^2\} \end{aligned}$$

This gives us

$$x_0 = (u_0, v_0) = (0, 0)$$

Now we compute the step 9 of algorithm **??**

$$\begin{aligned} x_{k+1} &= \pi_{\beta_{k+1}}(-S_{k+1}) \\ &= \underset{z \in Q}{\text{argmin}}\{- < -S_{k+1}, z > + \beta_{k+1} d(z)\} \end{aligned}$$

Now let's consider following function

$$g(z) = - < -S_{k+1}, z > + \beta_{k+1} d(z) \tag{6}$$

Let's simplify it further

$$\begin{aligned} g(z) &= - < -S_{k+1}, z > + \beta_{k+1} d(z) \\ &= - < \sum_{i=1}^{k} g_{u,i} - \sum_{i=1}^{k} g_{v,i}), (u, v) > + \beta_{k+1}(\frac{\alpha}{2}||u||_2^2 + \frac{(1-\alpha)}{2}||v||_2^2) \end{aligned}$$

Taking gradients with respect to u and v of above equation we get solution to step 9 of algorithm **??**

$$u_{k+1} = \frac{-\sum_{i=1}^{k} g_{u,i}}{\alpha\beta_{k+1}} \tag{7}$$

$$v_{k+1} = \frac{\sum_{i=1}^{k} g_{v,i}}{(1-\alpha)\beta_{k+1}} \tag{8}$$

Equation **??** and equation **??** gives us the update rules for our algorithm.

### 3.2 GAN Implementation

For our implementation we have used Wasserstein GAN whose objective function is given by

$$L(\theta_D, \theta_G) = \max_{\theta_D} \min_{\theta_G} \frac{1}{m} \sum_{i=1}^{m} (D_{\theta_D}(x_i) - D_{\theta_D}(G_{\theta_G}(z_i))) \tag{9}$$

The deep neural network models that we used are given in figure **??** and in figure **??**

```
G = torch.nn.Sequential(
    torch.nn.Linear(z_dim, h_dim),
    torch.nn.ReLU(),
    torch.nn.Linear(h_dim, X_dim),
    torch.nn.Sigmoid()
)
```

Figure 1: Generator Model

```
D = torch.nn.Sequential(
    torch.nn.Linear(X_dim, h_dim),
    torch.nn.ReLU(),
    torch.nn.Linear(h_dim, 1),
)
```

Figure 2: Descriminator Model

## 4 Experiment and Results

We have implemented the Wasserstein GAN and optimizer given in algorithm **??** in pyTorch. We have conducted our experiments on MNIST dataset. For our optimizer we have set $\gamma = 1e-4$ and $\alpha = 0.9$. We also done comparison with RMSProp optimizer for which we used learning rate of $1e-2$ and momentum=$0.9$. Figure **??** shows the generated images using out implementation of optimizer, figure **??** shows generated images using RMSProp. Figure **??** and figure **??** give the loss plot of both optimizers.

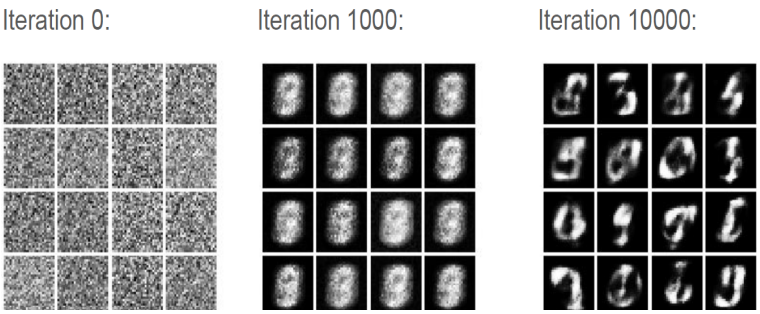


Figure 3: Generated Images using our optimizer

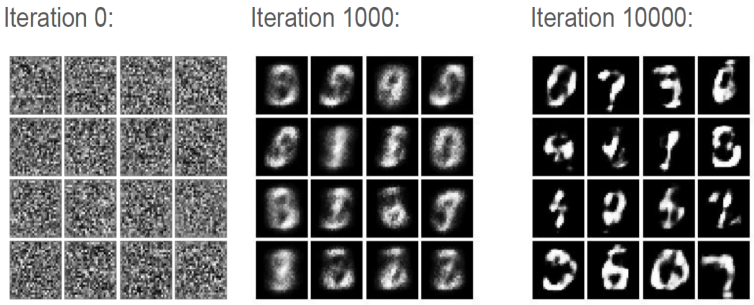


Figure 4: Generated images using RMSProp optimizer

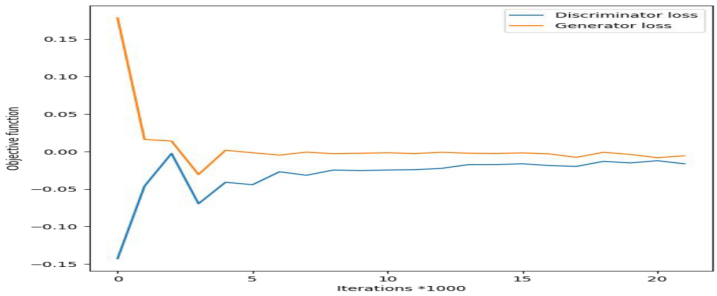


Figure 5: Loss plot of our optimizer

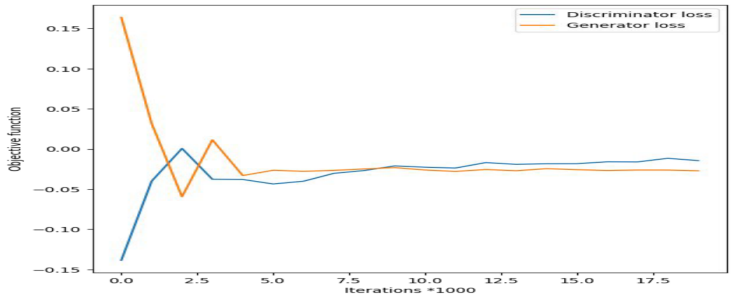


Figure 6: Loss plot of RMSProp optimizer

## 5 Conclusion

We were able to successfully apply an existing optimization algorithm for Saddle point problem to Wasserstein GAN with relatively simple model( 1 or 2 layer deep). But we were not able to replicate same for cross-entropy GAN objective or a more complex GAN model like DC-GAN. When compared with existing optimization techniques used to solve GAN we found that they perform very similarly.