Training of Generative Adversarial Networks

Raphael Ruschel, Devendra Jangid

June 09, 2020

1 Introduction:

Generative Adversarial Networks (GANs), first proposed by Ian Goodfellow[3], have attracted attention to the machine learning research community due to their impressive results. However, despite their success, training GANs has been really difficult due to their fundamental nature of optimization problem, which was investigated theoretically by Martin Arjoskvy[1]. In this report, our objective is to understand fundamental problems in training of GAN networks and solutions developed by researchers to overcome these problems. We discuss about general network architecture of GAN, fundamental training algorithm, difficulties with original training algorithm and solutions to these problems.

2 Generative Adversarial Network:

Generative Adversarial Network consists of one generator network and one discriminator network as shown in Figure 1. The generator tries to learn the distribution of training samples from a random noise vector, while the discriminator differentiates between generated samples and training samples and gives a confidence value of 1 for training samples and 0 otherwise. Both networks work in an adversarial manner, therefore, it's called an adversarial network.

2.1 Generator Network:

The generator network takes a random noise vector z as a input and generates $x = G(z)$ samples which have $P_g(x)$ distribution. The objective of generator is to minimize divergence between the distribution of real (training) data $P_{data}(x)$ and generated data $P_g(x)$.

$$G^* = argmin_G Div(P_g, P_{data}) \tag{1}$$

2.2 Discriminator Network:

The discriminator network acts as a classifier, which classifies between generated samples and real samples. It gives a value of 1 for real samples and 0 for generated samples. The objective function $V(G, D)$, which is also called binary cross entropy (BCE), for discriminator is

$$V(G, D) = E_{x \sim P_{data}}[\log D(x)] + E_{x \sim P_g}[\log(1 - D(x))] \tag{2}$$

Discriminator tries to maximizes V(G,D):

$$D^* = argmax_D V(G, D) \tag{3}$$

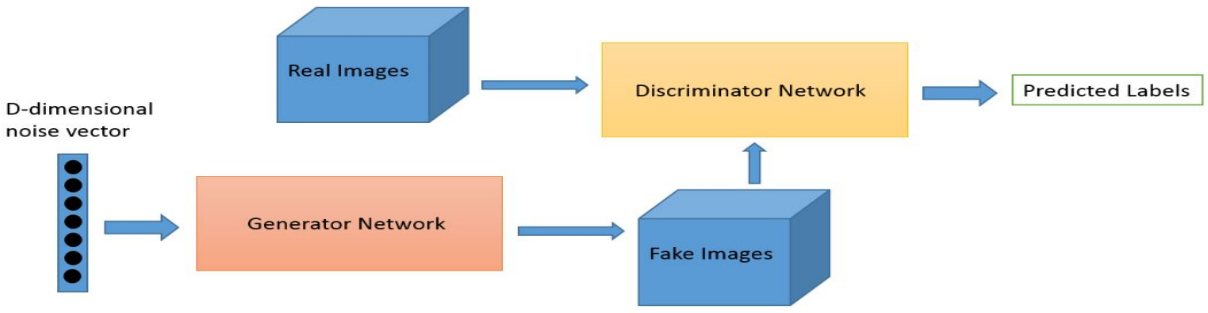

Figure 1: Architecture of GAN

2.3 Computing Divergences:

There are many divergences existing in the literature. However, the most common divergences, which are used in generative models, are the KL-Divergence and JS-Divergence. KL-Divergence and JS-Divergence are defined as follows, respectively:

$$D_{KL}(P||Q) = \sum_{x=1}^{N} P(x) \log \frac{P(x)}{Q(x)} \tag{4}$$

$$D_{JS}(P||Q) = \frac{1}{2} D_{KL}(P||\frac{P+Q}{2}) + \frac{1}{2} D_{KL}(Q||\frac{P+Q}{2}) \tag{5}$$

where P and Q are any discrete distribution.

2.4 Training of GAN:

In this section, we discuss the first training algorithm for GAN network. A general approach to train a GAN is to first reach an optimum on the discriminator before updating parameters of generator. Therefore, we update weights of discriminator multiple times before updating weights of generator.

Algorithm: General algorithm for GAN network is as follows:

1. Initialize parameters w for Discriminator (D) and θ for Generator (G)
2. Train discriminator:
   - Sample m examples $\{x^1, x^2, ..., x^m\}$ from data distribution $P_{data}(x)$
   - sample m noise samples $\{ z^1, z^2, ..., z^m \}$ from the prior distribution $P_{prior}(z)$
   - Obtaining generated data $\{ \tilde{x}^1, \tilde{x}^2, ..., \tilde{x}^m\}$, $\tilde{x}^i = g_\theta(z^i)$
   - Update discriminator parameter w to maximize
     - $\tilde{V} = \frac{1}{m} \sum_{i=1}^{m} \log D(x^i) + \frac{1}{m} \sum_{i=1}^{m} \log (1 - D(\tilde{x}^i))$
     - $w \longleftarrow w + \eta \nabla_w \tilde{V}(w)$
3. Update discriminator multiples times
4. Train Generator
   - Sample another m noise samples $\{z^1, z^2, ..., z^m\}$ from the prior $P_{prior}(z)$
   - update generator parameters θ to minimize
     - $\tilde{V} = \frac{1}{m} \sum_{i=1}^{m} \log (1 - D(g_\theta(z^i))$
     - $\theta \longleftarrow \theta - \eta \nabla_\theta \tilde{V}(\theta)$
5. Update generator once

3 Problem with Training:

In previous section, we have seen the training algorithm for GAN network, which is difficult to converge because of perfect discriminator theorem as described in [1]

3.1 Perfect Discriminator Theorem

The Perfect Discriminator Theorem states that the discriminator will have at most cost $2 \log 2 - 2JSD(P_{data}||P_g)$ in theory. However, in practice, if we train D till convergence, its error will go to **zero** due to discontinuous distributions or disjoint support of them. Zero cost means that discriminator can perfectly differentiate between real and generated samples.

Proof: Discriminator has at most cost $2 \log 2 - 2JSD(P_{data}||P_g)$

- Given G, what is optimal D^* maximizing

$$V = E_{x\sim P_{data}}[log(D(x))] + E_{x\sim P_g}[log(1 - D(x))] \tag{6}$$

$$V = \int_x P_{data}(x)\log(D(x))dx + \int_x P_g(x)\log(1 - D(x))dx \tag{7}$$

$$V = \int_x [P_{data}(x)\log(D(x)) + P_g(x)\log(1 - D(x))]dx \tag{8}$$

- Given x, the optimal D^* maximizing

$$P_{data}(x)\log(D(x)) + P_g(x)\log(1 - D(x)) \tag{9}$$

- D can be any function, therefore we assume it as constant for a given x and after differentiating equation (9), we get optimal D^* as

$$D^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_g(x)} \tag{10}$$

- Putting value of D^* back into equation (7), we get a maximum value of V

$$V_{max} = -2\log 2 + KL\left(P_{data}||\frac{P_{data} + P_g}{2}\right) + KL\left(P_g||\frac{P_{data} + P_g}{2}\right) \tag{11}$$

$$V_{max} = -2\log 2 + 2JSD\left(P_{data}||P_g\right) \tag{12}$$

- Maximum Cost of discriminator will be the negative of equation (12), which is $2\log 2 - 2JSD\left(P_{data}||P_g\right)$

3.2 Vanishing Gradient:

The discriminator has maximum cost zero due to perfect discriminator theorem and it gives vanishing gradient problem as can be seen in Figure 2. Let's assume p and q be the distributions of real and generated samples respectively. If p and q are equal, the divergence is zero but as the mean of q increases, the divergences increases but after a certain point its gradient will be close to zero, making the algorithm learns nothing. In practice, GAN can optimize the discriminator easier than the generator.

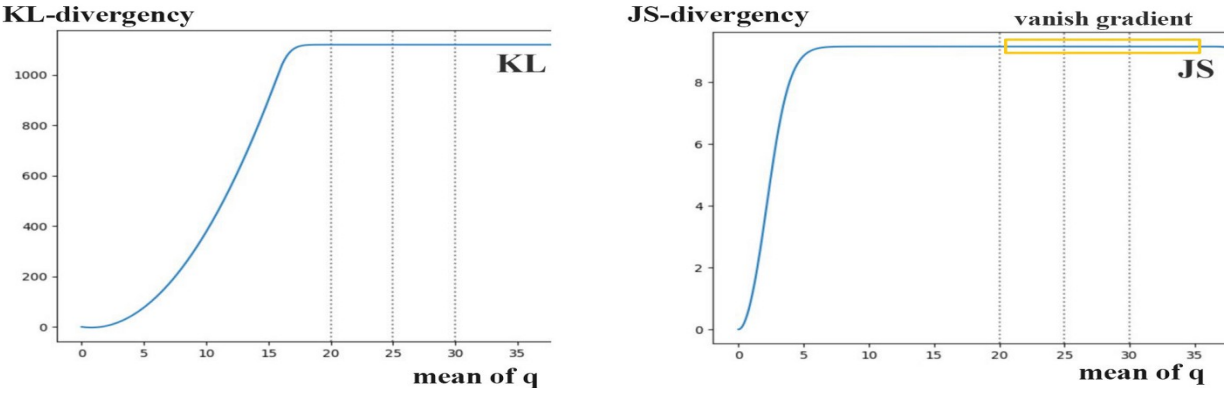

Figure 2: KL divergence (in left), JS divergence(in right)

4 Wasserstein GAN (WGAN):

4.1 WGAN:

To solve problem of vanishing gradient in training of GAN networks, Martin Arjovsky[2] proposes a new cost function using Wasserstein distance that has a smoother gradient everywhere. WGAN is able to learn no matter the performance of the generator. The original formulation of Wasserstein distance is usually intractable. However, using the Kantorovich-Rubinstein duality, it can be simplifies from equation (13) to equation (14)

$$W(P_r, P_g) = \inf_{\gamma\in\Pi(P_r,P_g)} E_{(x,y)\sim\gamma}[\| x - y \|] \tag{13}$$

$$W(P_r, P_g) = \sup_{||f||_l<1} E_{x\sim P_r}[f(x)] - E_{x\sim P_g}[f(x)] \tag{14}$$

Where *sup* is the least upper bound and we need to find f which is a 1-Lipschitz function that follows the constraint

$$|f(x_1) - f(x_2)| \leq |x_1 - x_2| \tag{15}$$

and maximizes

$$E_{x \sim P_r}[f(x)] - E_{x \sim P_g}[f(x)] \tag{16}$$

To enforce the Lipschitz constraint, WGAN applies a very single clipping to restrict the maximum weight value in f, i.e, the weights of the discriminator must be within a certain range controlled by the hyperparameters c.

$$w \leftarrow w + \eta RMSProp(w, g_w) \tag{17}$$

$$w \leftarrow clip(w, -c, c) \tag{18}$$

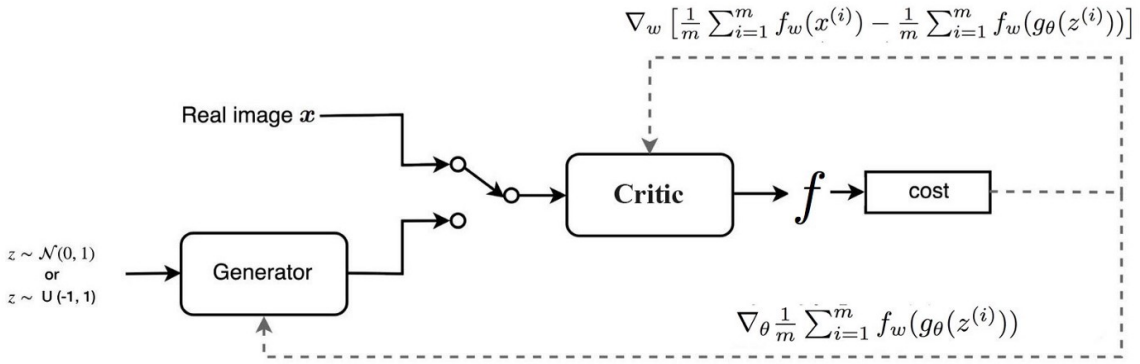

Figure 3: WGAN Architecture

In Figure 3, we can see that f is used instead of D and we enforce Lipschitz constraint on it using weight clipping. From Figure 4 (left), we can see that wasserstein loss function is continuous everywhere, while GAN discrimnator have vanishing gradient problem.

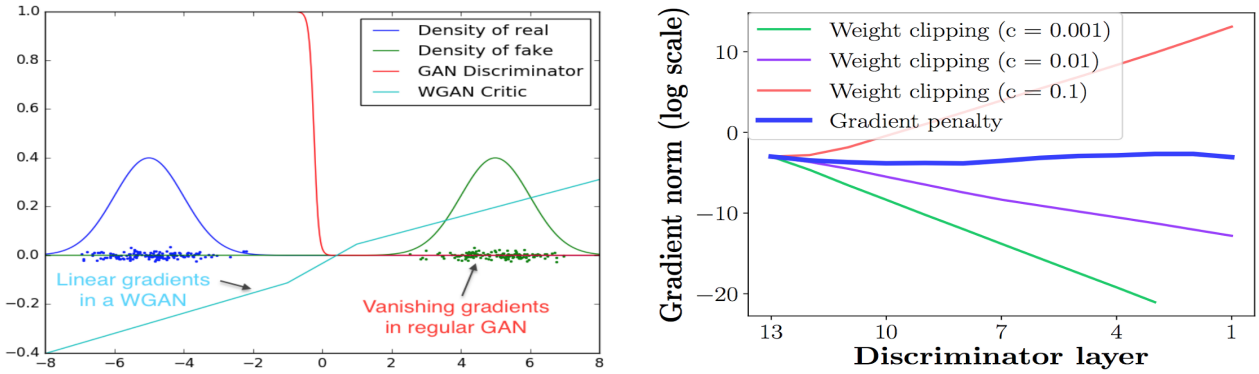

Figure 4: WGAN

4.2 Kantorovich-Rubinstein Duality:

In mathematical optimization theory, duality or the duality principle is the principle that optimization problems may be viewed from either of two perspectives, the primal problem or the dual problem. The solution to the dual problem provides a lower bound to the solution of the primal (minimization) problem. Applying this in finding the Wasserstein Distance, the primal and dual problem will in the form of:

primal form :				**dual form :**			
minimize	z	$=$	$\mathbf{c}^T\mathbf{x}$,	maximize	$\tilde{z}$	$=$	$\mathbf{b}^T\mathbf{y}$,
so that	$\mathbf{Ax}$	$=$	$\mathbf{b}$	so that	$\mathbf{A}^T\mathbf{y}$	$\leq$	$\mathbf{c}$
and	$\mathbf{x}$	$\geq$	$\mathbf{0}$				

The primal is $min_x z = c^T x$ subject to $Ax = b$, where x is the flattened version of $\gamma(i, j)$, c is the flattened version of the distance matrix D between i and j, and b is a vector containing P_r and P_g concatenated.

But this is a hard to solve problem, so we proceed to look at the dual instead, which is $max_y z = b^T y$ subject to $A^T y \leq c$. The objective is maximizing $f^T P_r + g^T P_g$ while maintaining the constraint $f(x_i) + g(x_j) \leq D_{ij}$. If $i = j$, this simplifies to $g(x_i) \leq -f(x_i)$.

Sinec P_r and P_g are non-negative, it makes sense to pick the highest possible value for g that maximizes the objective, in essence, pick $g = -f$. With this results, our constraint becomes $|f(x_i) - f(x_j)| \leq D_{ij}$. Therefore, to solve the dual problem, we need f to be a Lipschitz constraint.

4.3 Training Algorithm:

WGAN Algorithm: Algorithm with wasserstein loss function for GAN network is as follows:

1. Initialize w for Discriminator (D) and θ for Generator (G)
2. Train discriminator:
   - Sample m examples $\{x^i\}_{i=1}^m$ from data distribution $P_{data}(x)$
   - sample m noise samples $\{z^i\}_{i=1}^m$ from the prior distribution $P_{prior}(z)$
   - Obtaining generated data $\{\ \tilde{x}^i\}_{i=1}^m$, $\tilde{x}^i = g_\theta(z^{(i)})$
   - Update discriminator parameter w to maximize
     - $g_w \leftarrow \nabla_w[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(\tilde{x}^{(i)})]$
     - $w \longleftarrow w + \eta RMSProp(w, g_w))$
     - $w \leftarrow clip(w, -c, c)$
3. Update discriminator multiples times
4. Train Generator
   - Sample another m noise samples $\{z^i\}_{i=1}^m$ from the prior $P_{prior}(z)$
   - update generator parameters θ to minimize
     - $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(\tilde{x}^i)$
     - $\theta \longleftarrow \theta - \eta RMSProp(\theta, g_\theta)$
5. Update generator once

There is a problem with weight clipping method. As from Figure 4 (right), we can see that c is a very sensitive parameter and gradient norm can explode by just changing value of c from 0.01 to 0.1. It can be solved using Wasserstein GAN with a gradient penalty [4].

5 Conclusion:

In this report, we have studied problems in training of GAN network and available solutions in the literature. The problem of vanishing gradient comes in a typical GAN network due to a perfect discriminator and it is solved using a WGAN network. To avoid sensitivity of parameter c in weight clipping, we use Wasserstein GAN with a gradient penalty. More theoretical and empirical details are given in [1], [2] and [4].

References

[1] Martin Arjovsky and Léon Bottou. *Towards Principled Methods for Training Generative Adversarial Networks*. 2017. arXiv: `1701.04862 [stat.ML]`.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. *Wasserstein GAN*. 2017. arXiv: `1701.07875 [stat.ML]`.

[3] Ian Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., 2014, pp. 2672–2680. URL: `http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf`.

[4] Ishaan Gulrajani et al. "Improved training of wasserstein gans". In: *Advances in neural information processing systems*. 2017, pp. 5767–5777.