Regularization of GANs

Project-I and II report submitted to

Indian Institute of Technology Kharagpur

in partial fulfilment for the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

Abhishek Panigrahi

(14CS10001)

Under the supervision of

Professor Pabitra Mitra

Department of Computer Science and Technology

Indian Institute of Technology Kharagpur

Spring semester, 2017-18

May 07, 2018

DECLARATION

I certify that

(a) The work contained in this report has been done by me under the guidance of my supervisor.

(b) The work has not been submitted to any other Institute for any degree or diploma.

(c) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

(d) Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Date: May 07, 2018 (Abhishek Panigrahi)
Place: Kharagpur (14CS10001)

DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

KHARAGPUR - 721302, INDIA

CERTIFICATE

This is to certify that the project report entitled "**Regularization of GANs**" submitted by **Abhishek Panigrahi** (Roll No. 14CS10001) to Indian Institute of Technology Kharagpur towards partial fulfilment of requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering is a record of bona fide work carried out by him under my supervision and guidance during Spring semester, 2017-18.

Professor Pabitra Mitra
Department of Computer Science and Technology
Indian Institute of Technology Kharagpur
Kharagpur - 721302, India

Date: May 07, 2018

Place: Kharagpur

Abstract

Name of the student: **Abhishek Panigrahi** Roll No: **14CS10001**

Degree for which submitted: **Bachelor of Technology**

Department: **Department of Computer Science and Technology**

Thesis title: **Regularization of GANs**

Thesis supervisor: **Professor Pabitra Mitra**

Month and year of thesis submission: **May 07, 2018**

Generative Adversarial Networks (GANs) tend to generate data corresponding to a few dominant modes in the data manifold. Two reasons, that are responsible for this imperfect training, are the imbalance between generator and discriminator and the dependence of loss function on individual data sample, rather than an entire batch of samples. For solving mode collapse problem, researchers use distance metrics like wasserstein distance and maximum mean discrepancy to train GANs. However, using finite sample estimates of these distance measures doesn't always converge to the correct manifold. Hence, we propose novel techniques based on boosting and tangent space alignment to solve this problem, known as the multi-manifold problem. We show that adding weights to outliers in the loss function results in better convergence to the original manifold.

Acknowledgements

I would first like to thank my thesis advisor Prof. Pabitra Mitra. The door to Prof. Mitra office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this paper to be my own work, but steered me in the right direction whenever he thought I needed it.

Finally, I must express my very profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Abhishek Panigrahi

Contents

List of Figures

Part I

Semester 7

Chapter 1

Generative Adversarial networks

1.1 Introduction

There are two kinds of models used in current machine learning algorithms, a discriminative model and a generative model. A discriminative model learns a function that maps the input data (x) to some desired output class label (y). In probabilistic terms, they directly learn the conditional distribution P(y|x). A generative model tries to learn the joint probability of the input data and labels simultaneously, i.e. P(x,y).

Generative Adversarial networks (GANs) Goodfellow et al. (2014) constitute a combination of the above two models. The main idea behind a GAN is to have two competing neural network models. One takes noise as input and generates samples (and so is called the generator). The other model (called the discriminator) receives samples from both the generator and the training data, and has to be able to distinguish between the two sources. These two networks play a continuous game, where the generator is learning to produce more and more realistic samples, and the discriminator is learning to get better and better at distinguishing generated data from real data.

Let the function of generator be denoted by G and it's set of parameters are denoted by θ. Also, let the function of discriminator be denoted by D and it's set of parameters are denoted by ϕ. Let p_{data} denote the probability distribution of real

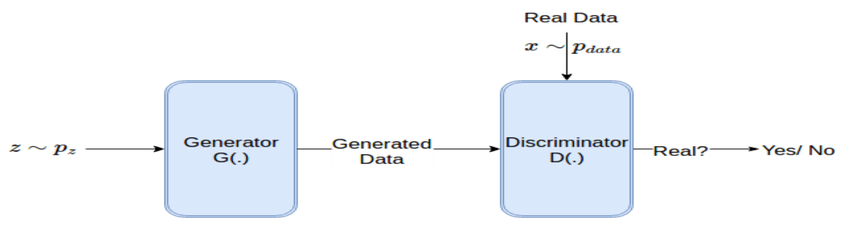


FIGURE 1.1: A simple logical figure of a GAN

data and p_z denote the prior probability distribution of noise. Then, GAN's training procedure can be formulated as a minimax game:

$$min_\theta \ max_\phi E_{x \sim p_{data}} log D(x;\phi) + E_{z \sim p_z}(1 - log(1 - D(G(z;\theta);\phi)))$$

1.2 Major Problems in GAN training

1.2.1 Imbalance between generator and discriminator

It is necessary to balance the two players to prevent one from overpowering the other. For example, one of the most common situations faced during GAN training is that the gradient to train the generator starts vanishing when the discriminator becomes too accurate. A common failure pattern for training GANs is the vanishing gradient problem, in which the discriminator D perfectly classifies real and fake examples, such that around the fake examples, D is nearly zero. In such cases, the generator will receive no gradient to improve itself.

1.2.2 Non convergence of stochastic gradient descent of games

GANs require finding the equilibrium to a game with two players Goodfellow (2016). Even if each player successfully moves downhill on that player's update, the same update might move the other player uphill. Sometimes the two players eventually reach an equilibrium, but in other scenarios they repeatedly undo each others' progress without arriving anywhere useful. Hence, GANs tend to oscillate a lot, meaning that they progress from generating one kind of sample to generating another kind of sample without eventually reaching an equilibrium.

1.2.2.1 Mode Collapse

Mode collapse is a problem that occurs when the generator learns to map several different input z values to the same output point Goodfellow (2016). In mode collapse, GANs may enter an irrecoverable failure state where the generator incorrectly assigns all its probability mass to a small region in space. This arises because a globally optimal strategy for the generator is to push all its mass towards the single point that the discriminator is the most confident about being a real data point.Che et al. (2016) states that since the discriminator's output is nearly 0 and 1 on fake and real data respectively, the generator is not penalized for missing modes. Goodfellow (2016) states that when the generator and discriminator are trained simultaneously using gradient descent, the minimax game may become a maximin game. They state that when the max and min component in GAN training are exchanged,

$$max_{\phi} min_{\theta} V(G, D)$$

the minimization with respect to the generator now lies in the inner loop of the optimization procedure. The generator is thus asked to map every z value to the single x coordinate that the discriminator believes is most likely to be real rather than fake.

There is also a theory that the mode collapse problem is because learning the optimal G minimizes the Jensen-Shannon divergence, which is more similar to minimizing D_{KL} ($p_{model} \parallel p_{data}$), often referred to as the reverse KL divergence, than to minimizing the KL divergence D_{KL} ($p_{data} \parallel p_{model}$). A model trained with minimizing the reverse KL divergence would prefer to generate samples that are around only a few modes of the training distribution while ignoring other modes.

1.3 Problem Statement

The main aim of my B.Tech thesis is to improve the training of GANs and to search for a regularizer to solve mode collapse phenomenon observed during training of GANs.

1.4 Literature Survey

Many GAN variants have been developed to solve the non-convergence and mode collapse problems. The main approach is to yield better gradient signal. Zhao et al. (2016) view the discriminator as an energy function that attributes low energies to the regions near the data manifold and higher energies to other regions. The EBGAN framework uses an auto-encoder architecture, with the energy being the reconstruction error, in place of the discriminator. Unrolled GANs Metz et al. (2016) attempts to address the defect of training the generator and discriminator simultaneously by including several optimization steps of the discriminator into the computational graph. Bayesian GAN Saatchi and Wilson (2017) induces distributions over an uncountably infinite space of generators and discriminators, corresponding to every possible setting of these weight vectors. By exploring rich multimodal distributions over the weight parameters of the generator, the Bayesian GAN can capture a diverse set of complementary and interpretable representations of data. Arjovsky et al. (2017) introduced Wasserstein GANs. These GANs use Earth-mover(EM) distance to compare two probability distributions. The reason behind this is that EM distance is continuous, compared to other measures like KL divergence, JS divergence and total variation distance. However, the technique to make the function learned by the generator lipschtiz continuous still remains an open problem. Banijamali et al. (2017) introduced a novel architecture called generative mixture of networks, consisting of multiple GANs.The dataset was broken into K clusters randomly and each GAN was trained to learn a single cluster. Now, the GANs were trained together in an Expectation maximization algorithm to update the clusters of the data.

I haven't mentioned here those papers that I have experimented with. I discuss about them in details as I discuss about my experiments.

Chapter 2

Progress and Results

I explored two directions of approach to solve mode collapse.

- K-means
- Maximum mean discrepancy

2.1 K-means approach

2.1.1 Motivation

Mode collapse is caused because the generator is not able to generate data points corresponding to different sections on the data manifold. See Fig (2.1).

Tolstikhin et al. (2017) recently introduced AdaGAN. AdaGAN learned a new GAN at each iteration, trained on the dataset with changed weights/importance of each sample in the data. The data generated by the group of GANs learned were mixed in a particular weighted manner. The weights of the samples in each iteration varied based on the performance of the mixture of GANs on the current dataset. The portion of the data manifold where the mixture of GAN performed badly, were given higher weight, compared to other portions of the data manifold. Hence, this

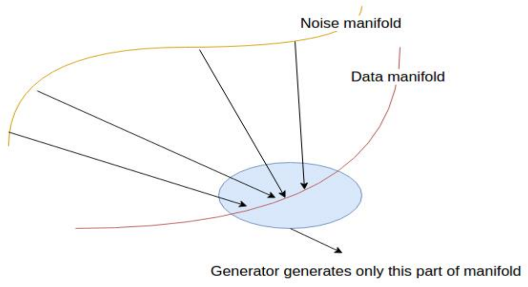


FIGURE 2.1: Mode Collapse

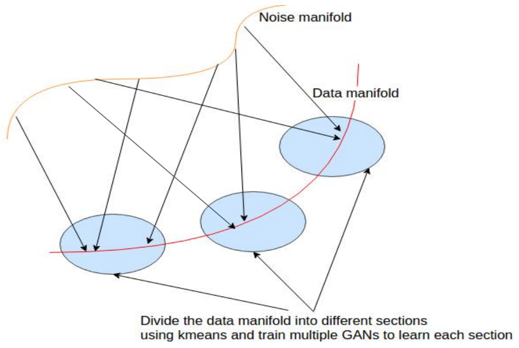

Noise manifold
Data manifold
Divide the data manifold into different sections using kmeans and train multiple GANs to learn each section

FIGURE 2.2: Kmeans approach

can be seen as a method to learn a bag of GANs to learn different portions of the manifold.

Our idea was to divide the manifold of the original data into different sections using kmeans on the data samples and train multiple generators, where each generator generates each cluster. See Fig(2.2).

In one of the first experiments, where we had manually divided the original MNIST dataset into different classes of digits (0-9) and trained a different GAN to learn each class, we observed that each GAN was able to generate the class of dataset on which it was trained.(The observations are shown in Fig. (2.32.42.5)) That showed that changing the probability distribution of the dataset on which a GAN is trained also changed the data generated by a generator.

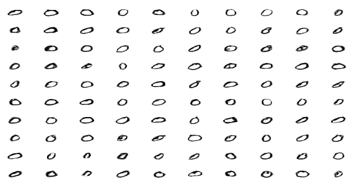


FIGURE 2.3: GAN trained on digit class 0

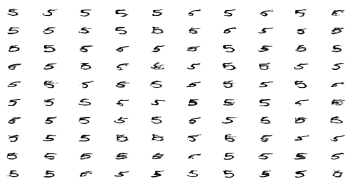


FIGURE 2.4: GAN trained on digit class 5

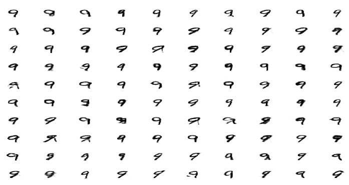


FIGURE 2.5: GAN trained on digit class 9

2.1.2 Experiments and Observations

- **Multiple Generators** We used kmeans to divide the dataset into k number of clusters and trained k number of generators with a single discriminator, where each generator was to learn a single cluster. We varied the value of k as 5, 10 and 20. Observations in Figs (2.6 2.7 2.8), that show the results generated for k=5. The observations showed that the clustering algorithm doesn't divide the original dataset into well defined clusters. Hence, the data generated by different generators seen quite intermixed. A possible explanation can be the high variance of the cluster that is to be learned by the generator.

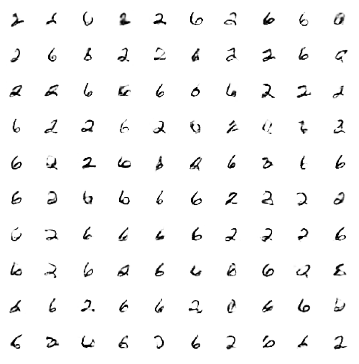


FIGURE 2.6: Data generated by generator trained on cluster number 0, in multi generator experiment with 5 generators

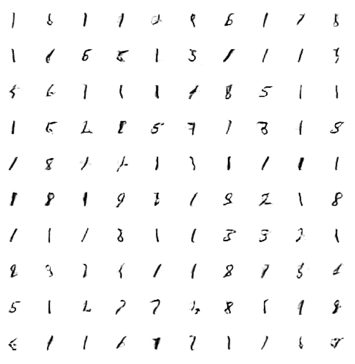


FIGURE 2.7: Data generated by generator trained on cluster number 2, in multi generator experiment with 5 generators

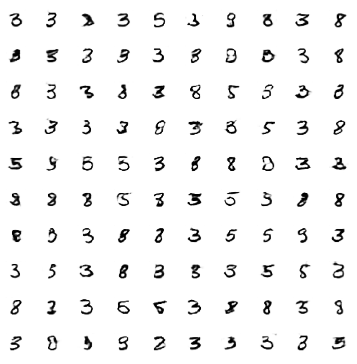


FIGURE 2.8: Data generated by generator trained on cluster number 4, in multi generator experiment with 5 generators

Observations in Figs (2.9 2.10 2.11 2.12 2.13) show the results generated for k=10. We again observe non-disjoint clusters that are learned by the generators. Also, some digits appear with greater frequency, compared to other digits.

- **Multiple GANs** We did a similar experiment, as the previous one. Here, we trained different GANs to learn each cluster. The motivation behind this was that perhaps a single discriminator wasn't enough to compete with multiple generators. However, we observed similar observations, as the previous experiment.

- **Multiple Discriminators** We tried to learn a model, where a single generator was to compete with multiple discriminators and each discriminator discriminated based on whether the generator generated the class of data represented by the discriminator. We manually divided the MNIST dataset into multiple classes and trained multiple discriminators to learn a class each. We

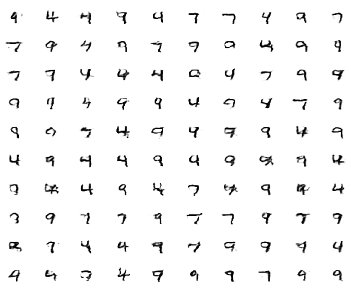


FIGURE 2.9: Data generated by generator trained on cluster number 0, in multi-generator experiment with 10 generators

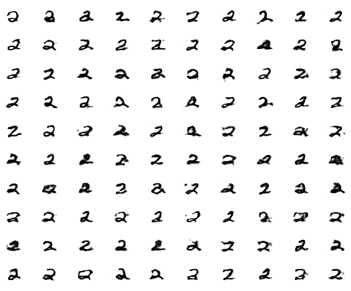


FIGURE 2.10: Data generated by generator trained on cluster number 3, in multi-generator experiment with 10 generators

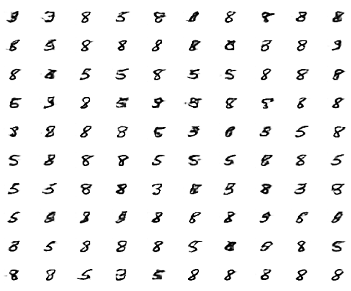


FIGURE 2.11: Data generated by generator trained on cluster number 5, in multi-generator experiment with 10 generators

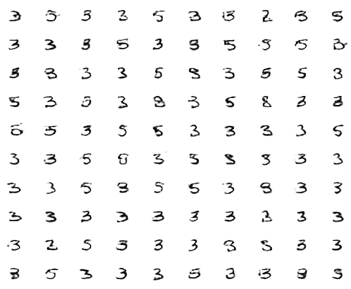


FIGURE 2.12: Data generated by generator trained on cluster number 7, in multi-generator experiment with 10 generators

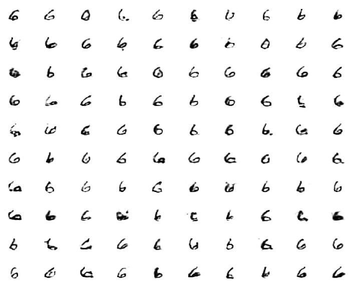


FIGURE 2.13: Data generated by generator trained on cluster number 9, in multi-generator experiment with 10 generators

observed a huge amount of oscillation in the training of the model and the images produced by the single generator were not of high quality. 2.14

- **Train an extra classifier** Chen et al. (2016) introduced a model called InfoGAN, where they trained a generator to accept a structured latent code, c, in addition to the random noise, z and a classifier was trained on top of the features learned by discriminator to predict the categorical distribution of the latent code used.

We implemented a similar model, that was given by Premachandran and Yuille (2016). The minimax game includes K generators G_k, a classifier C and a

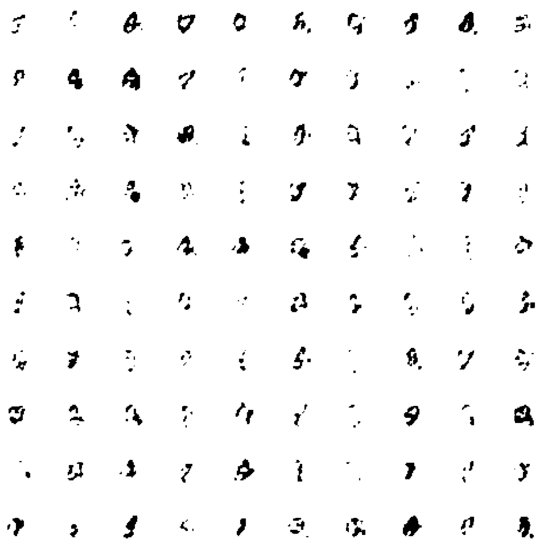


FIGURE 2.14: Images generated by generator when trained against multiple discriminators

discriminator D. C estimates the probability that a sample came from each of the generators and D estimates the probability that a sample came from the training data rather than from Gs. The model used is shown in Fig (2.15).

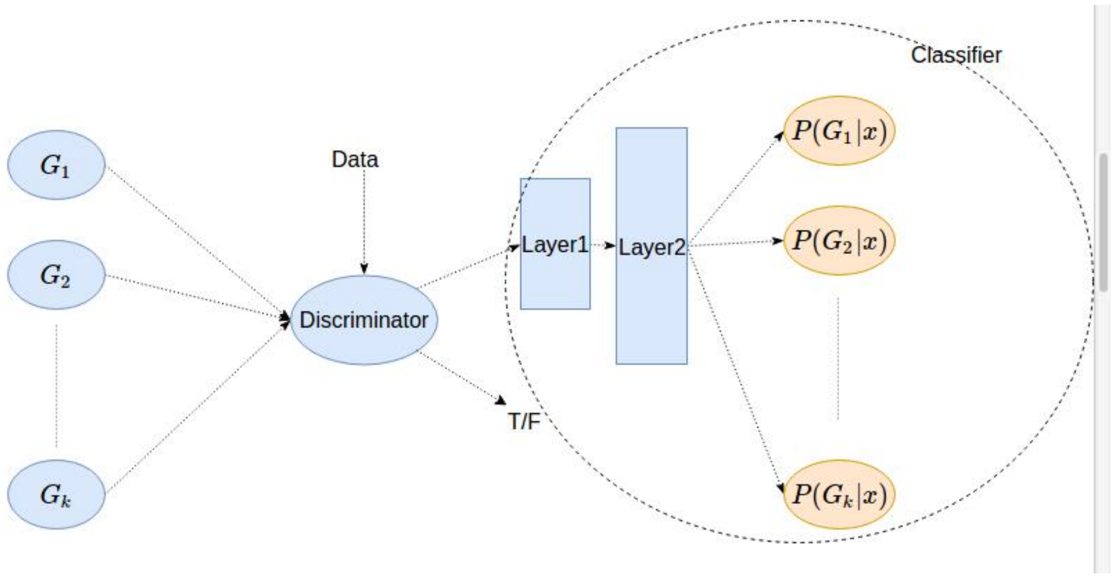


FIGURE 2.15: Model used

The loss functions used was

$$min_{G,C}\ max_D\ E_{x\sim p_{data}} logD(x) + E_{z\sim p_z}(1 - log(1 - D(G(z)))) - \frac{\lambda}{k}\sum_{i=1}^{k} E_{x\sim p_{G_i}} log(C_i(x))$$

where p_{G_i} denotes the probability distribution of data generated by generator G_i.

We experimented with models of different k values on MNIST. Although we observed differences in the data generated by the different generators, the quality of data generated was too low. The observations for k = 5 are shown in fig (2.16 2.17 2.18 2.19 2.20).

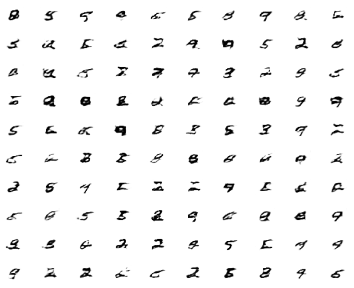


FIGURE 2.16: Data generated by generator number 0, in multi-generator experiment with classifier

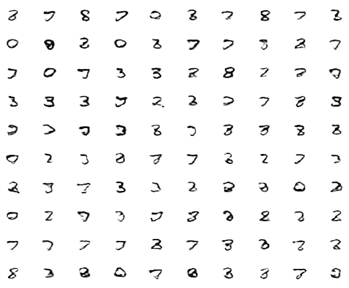


FIGURE 2.17: Data generated by generator number 1, in multi-generator experiment with classifier

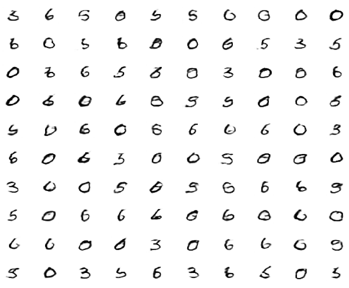


FIGURE 2.18: Data generated by generator number 2, in multi-generator experiment with classifier

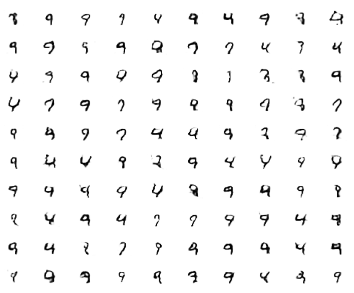


FIGURE 2.19: Data generated by generator number 3, in multi-generator experiment with classifier

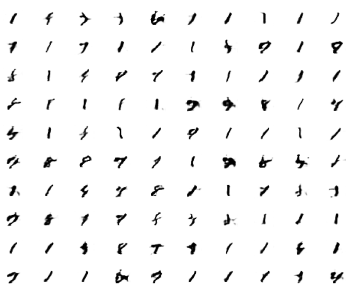


FIGURE 2.20: Data generated by generator number 4, in multi-generator experiment with classifier

2.1.3 Conclusion

Applying kmeans directly to a dataset and training multiple models to learn each cluster doesn't seem to clearly learn a restrictive set of classes. Kmeans clusters data based on style, contour, shape, etc. Hence, it doesn't guarantee an exclusive set of clusters covering all the classes in the dataset. For learning an exclusive set of classes, we need features learned by a trained network. However, this would require a labeled dataset, which won't be available for all kinds of data. Hence, kmeans doesn't provide a valuable solution to the current problem.

Also, adding a classifier to the model to force the different generators to give different data results, doesn't seem to perform well. This is because there is a mismatch between the power of discriminator and generator. The generator, as shown by the loss function of generator in fig (2.21), seems to perform well in terms of decrease in loss but still the quality of data generated by the generators degrades.

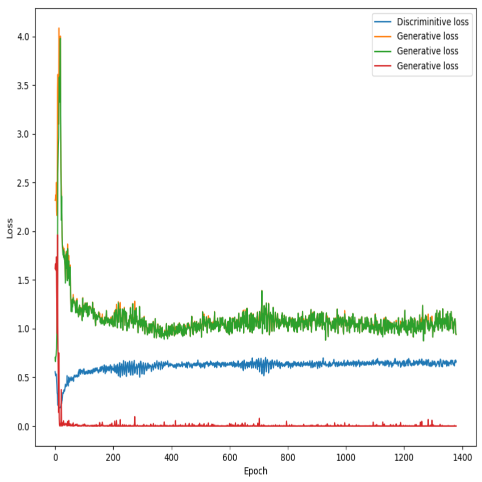


FIGURE 2.21: Loss functions of a generator(green) and discriminator(blue), in multi-generator experiment with classifier

2.2 Maximum mean discrepancy

2.2.1 Motivation

The above techniques aimed to change the structure of the model to incorporate learning of various sections of the manifold. The original GAN tries to match the generator probability distribution with the real distribution by a minimax game on the samples generated by generator. As a result, the generator can focus on only some sections of the manifold to fool it's adversary. However, if we try to match the distributions by **statistical moment matching**, the generator will have to generate samples from the entire distribution for decreasing it's loss function.

2.2.2 Theory

Let us go through some definitions. We refer to Gretton (2013)Gretton et al. (2012a) for more details and proofs of the given theorems.

- **Inner product** Let H be a vector space over $\mathbb{R}$. A function $\langle .,. \rangle_H$: H x H $\rightarrow$ $\mathbb{R}$ is an inner product on H if
  - Linear:
    $$\langle \alpha_1 f_1 + \alpha_2 f_2, g \rangle_H = \alpha_1 \langle f_1, g \rangle_H + \alpha_2 \langle f_2, g \rangle_H$$
  - Symmetric:
    $$\langle f, g \rangle_H = \langle g, f \rangle_H$$
  -
    $$\langle f, f \rangle_H \geq 0 \text{ and } \langle f, f \rangle_H = 0 \text{ if and only if } f = 0$$
- **Hilbert Space** A Hilbert space H is a real or complex inner product space that is also a complete metric space with respect to the distance function induced by the inner product. A complete metric space is a space that has cauchy sequence limits i.e. each infinite sequence $\{f\}_{n=1}^{\infty}$ has limit in the space itself.

- **Kernel** Let X be a non-empty set. A function k : X x X → ℝ is a kernel if there exists an ℝ-Hilbert space and a feature map ϕ : X → H such that ∀ x, x' ∈ X,
$$k(x, x') = \langle \phi(x), \phi(x') \rangle_H$$

- **Reproducing Kernel Hilbert space (RKHS)** Let H be a Hilbert space of ℝ-valued functions on non-empty set X. A function k : X x X → ℝ is a reproducing kernel of H, and H is a reproducing kernel Hilbert space, if
$$\forall x \in X, k(., x) \in H,$$
$$\forall x \in X, \forall f \in H, \langle f, k(., x) \rangle_H = f(x) \text{ (the reproducing property) .}$$
In particular, for any x, y ∈ X, k(x, y) = ⟨ k (., x), k (., y) $\rangle_H$.

- **Maximum mean discrepancy** Let F be a class of functions f : A → ℝ. Let x and y denote two random variables from the topological space A and let X and Y denote the set of observations for x and y respectively. Also, let the probability distribution of x and y be denoted by p and q respectively. Maximum mean discrepancy (MMD) is defined as
$$MMD(F, p, q) := sup_{f \in F}(E_x f(x) - E_y f(y))$$
Hence, given two distributions P and Q, and a kernel k, the square of MMD distance is defined as ($\mu_p = E_{x \sim p}$ k(x, .), $\mu_y = E_{y \sim q}$ k(., y) and ||f||$_H \leq 1$)
$$MMD_k^2(P, Q) = ||\mu_p - \mu_q||^2 = E_{x,x' \sim p} k(x, x') - 2\ E_{x \sim p, y \sim q} k(x, y) + E_{y,y' \sim q} k(y, y'). \tag{2.1}$$

Next, I will just mention the important theorems that will be used for the training of GANs.

- **Theorem 1** The MMD is a metric only when the RKHS H considered is a universal kernel. For universality, the kernel k(., .) needs to be continuous. Also, if X is the topological space from which data is drawn and C(X) denotes the set of continuous bounded functions over X, then H should be dense in

C(X) w.r.t. the L_∞ norm, i.e. for every function f ∈ C(X) and any given $\epsilon > 0$, ∃ g ∈ C(X) such that

$$\|f - g\|_\infty \leq \epsilon$$

- **Theorem 2** Let F be a unit ball in a universal RKHS H, defined on the compact metric space X, with associated continuous kernel k(., .). Then MMD(F, p, q) = 0 if and only if p = q.

Steinwart (2001) proved that the **Gaussian** and **Laplace** RKHSs are universal.

2.2.3 Experiments with existing models and ongoing research

- Li et al. (2017a) is currently working on distributional GANs. The GAN model is shown in Fig. (2.22).

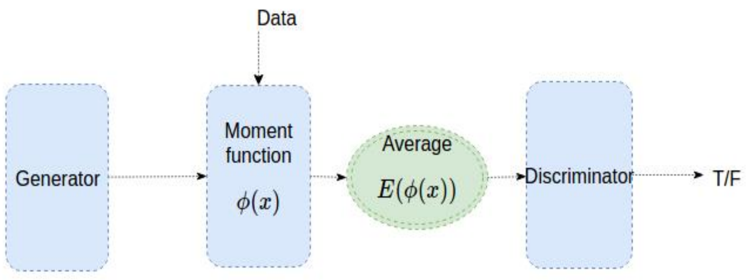


FIGURE 2.22: Distributional GAN logical view

The authors introduce a neural network in the original GAN that acts as a moment generating function $\phi(x)$. The discriminator, rather than discriminating real samples from the generated samples, works on an entire batch of sample. The discriminator discriminates between fake and real image distribution by working on the moments, calculated by $\phi(x)$. Thus, the learning function becomes,

$$max_G \; min_D \; log(D(E_{x \sim p_{data}} \; \phi(x))) + log(1 - D(E_{z \sim p_z} \; \phi(G(z)))),$$

Problem: Having an explicit discriminator again may result in oscillations, due to imbalance in power of discriminator and generator.

Experiments: We experimented on the distributional GAN with the the given loss function and on another model with explicit L_2 regularization. We trained both the models on a toy problem, where the models needed to learn k gaussian distributions placed in a 1-Dimensional plane. The real data distribution is shown in Fig. (2.23.

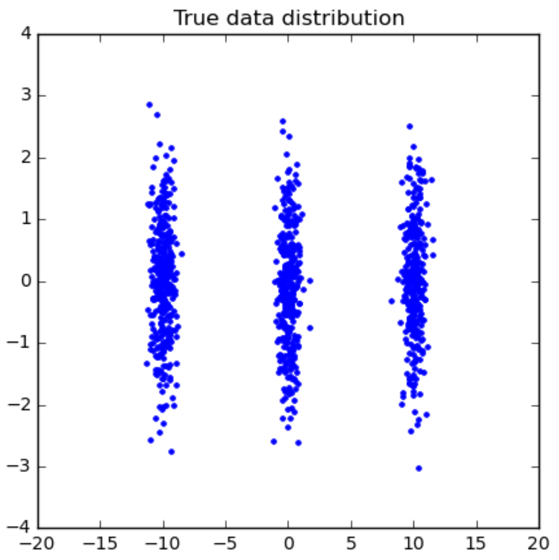


FIGURE 2.23: Real data distribution of 3 gaussians, taken into consideration

There were 4 layers in the generator, 2 layers in the moment generating neural network and 2 layers in the discriminator in both the models. In the 2nd model, the L_2 regularization was applied on the discriminator.

- The original distributional GAN failed to learn the 3 gaussian distributions present in the training set. 2.24
- Adding a simple L_2 regularization on the discriminator weights resulted in the generator producing meaningful results.2.25

- As has been used in generative moment networks Li et al. (2015), we directly apply a gaussian kernel on the data generated by generator and the real data distribution, rather than training an explicit adversary.

  We experiment on a four-layered generator, which is trained on a toy dataset containing k gaussians in a 1-dimensional plane. The generator had to minimize the maximum mean discrepancy, computed between samples of the generator and the real dataset. Using an unbiased estimate of Eqn((2.1)) for a

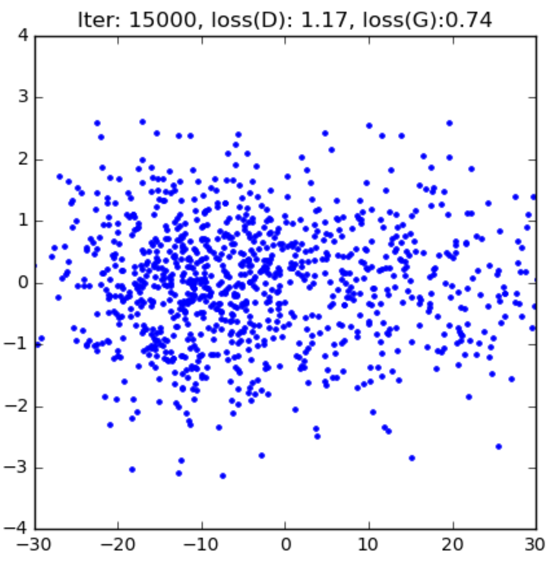


FIGURE 2.24: Data generated by generator without L_2 regularization on discriminator weights, in distributional GAN

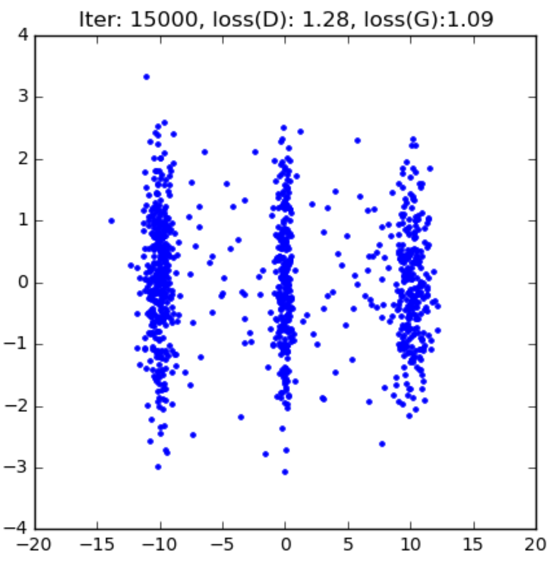


FIGURE 2.25: Data generated by generator with L_2 regularization on discriminator weights, in distributional GAN

batch of samples,

$$MMD^2(p_{data}, p_G) = \frac{1}{n^2}\sum_{i=1}^{N}\sum_{j=1}^{N} K(x_i, x_j) + \frac{1}{MN}\sum_{i=1}^{N}\sum_{j=1}^{M} K(x_i, y_j) + \frac{1}{M^2}\sum_{i=1}^{M}\sum_{j=1}^{M} K(y_i, y_j) \tag{2.2}$$

where k(x,y) = $\exp(-\|x-y\|^2)$ for a gaussian kernel, p_{data}, p_G denote probability distribution of real data and generator data respectively, M and N denote the number of samples in a batch of real data and generator data respectively.

The model was able to learn the toy problem, with both 3 gaussians dataset, as shown in Fig(2.27), and 5 gaussians dataset, as shown in Fig(2.29).

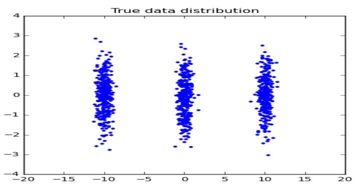


Figure 2.26: Real data distribution containing 3 gaussians in 1-Dimensional plane

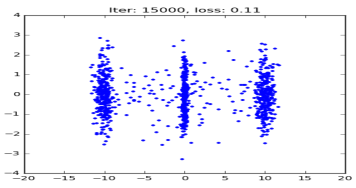


Figure 2.27: Data generated by generator on 3-gaussians dataset, in MMD GANs

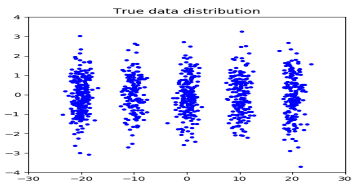


Figure 2.28: Real data distribution containing 5 gaussians in 1-Dimensional plane

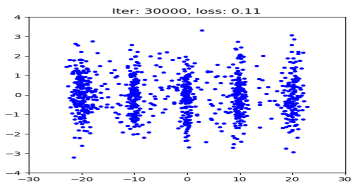


Figure 2.29: Data generated by generator on 5-gaussians dataset, in MMD GANs

2.2.4 Disadvantages of using maximum mean discrepancy

- Using the loss function 2.2 directly is computationally expensive, since it has a time complexity of order O(N^2), where N is the number of samples in a batch.
- Sutherland et al. (2016) showed that for mean discrepancy test to be reliable, the number of samples should grow linearly with the dimensions. Thus, this makes the method unsuitable for capturing very complex distances in high dimensional sample spaces such as natural images.

2.2.5 Recent works to tackle the above problems and experiments

Li et al. (2017b) used the fact that an injective function can be used to reduce the dimension of the data. Because, suppose f is an injective function and C is a characteristic kernel, then f $\circ$ C is also characteristicGretton et al. (2012b),Gretton et al. (2012a). For an injective function f , there exist an function f^{-1} such that $f^{-1}(f(x))$ = x, $\forall$ x $\in$ X and $f^{-1}(f(G(z)))$ = G(z), $\forall$ z $\in$ Z, which is approximated by an auto-encoder. Let the encoder of the auto-encoder be denoted by f_{enc} and the decoder be denoted by f_{dec}. Then, the loss function used was

$$min_G \ max_{f_{enc}} \ E_{x \sim p_{data}} E_{z \sim p_z} MMD^2(f(x), f(G(z))) - \lambda \ E_{x \sim data \cup G(z)} \|x - f_{dec}(f_{enc}(x))\|^2$$

Hence, the loss function also maximizes over the encoder function. This can be viewed as replacing the fixed kernel k with an adversarially learned kernel to have stronger signal where P $\neq P_\theta$ to train G.

Problems

– However, this again means to use an adversarial network and a minimax game, which may lead to non convergence. We trained the model on a toy dataset, containing 5 gaussians in a 1-dimensional plane. We observed a really slow learning in case of the generator, even though we used a simple auto-encoder with only 4 layers.2.30

– Again, the optimal number of dimensions of the latent code, learned by the encoder, that can capture optimal feature information becomes a hyper-parameter.

2.3 Future work

Theorem 3: Moore-Aronszajn. Every positive definite kernel k is associated with a unique RKHS H.

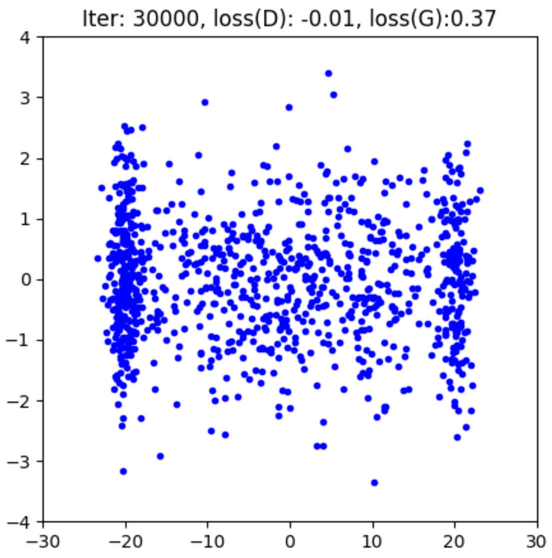


FIGURE 2.30: Data generated by generator trained along with an auto-encoder after 30000 iterations

My future work will be to capture this theorem and a characteristic kernel with the help of a neural network. My aim is to learn the feature map ϕ that will map the data from input space to RKHS. Doing so, we can capture the mean of the samples in the feature space directly and just compute MMD as the L2 norm difference between the means of the real and generated data distributions. A logical view of my idea is shown in Fig(2.31).

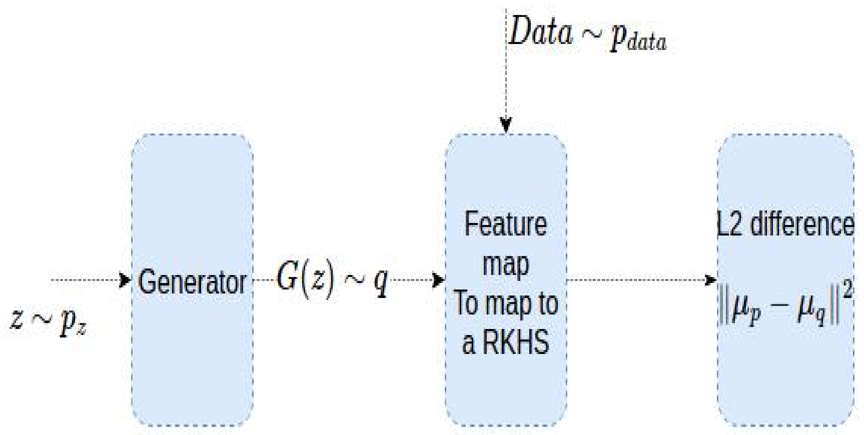


FIGURE 2.31: My predicted model

Part II

Semester 8

Chapter 3

Baseline model results

3.1 Baseline results to be compared with

In this section, I will give the results of the baseline model, with which I will compare the quality of the results produced in the chapters 3, 4, 5 and 6. For comparing the results, we use two datasets, one is a toy dataset and the other is MNIST dataset. The toy dataset is given in fig 3.1.

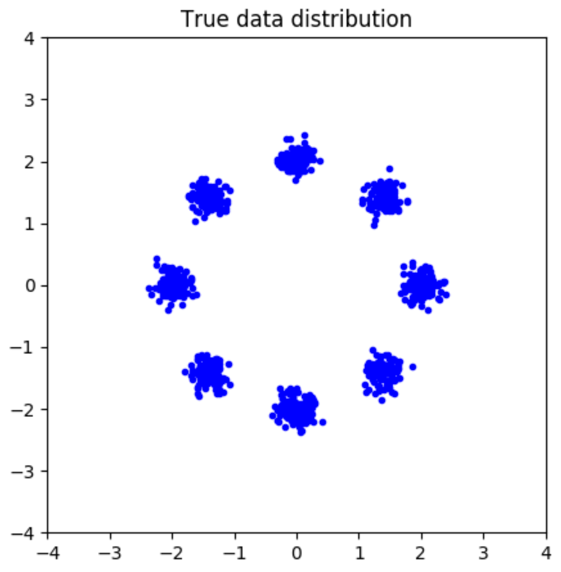


FIGURE 3.1: Real toy data

The performance of MMD GAN on the two dataset is given in fig. 3.4 and 3.5.

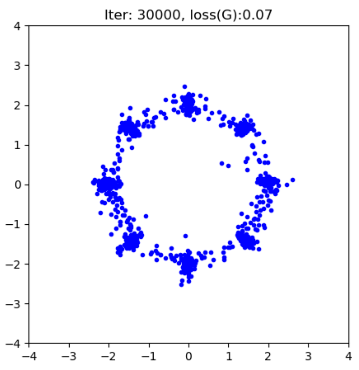


FIGURE 3.2: Toy generated data by generator trained with MMD, step 30000

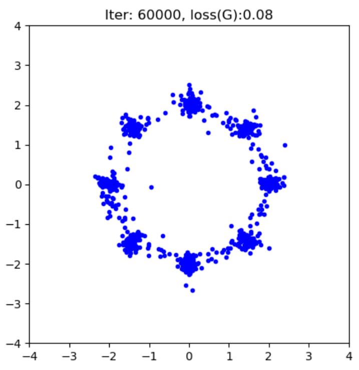


FIGURE 3.3: Toy generated data by generator trained with MMD, step 60000

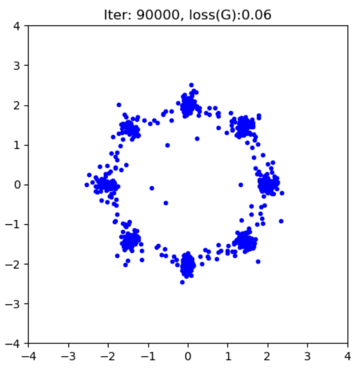


FIGURE 3.4: Toy generated data by generator trained with MMD, step 90000

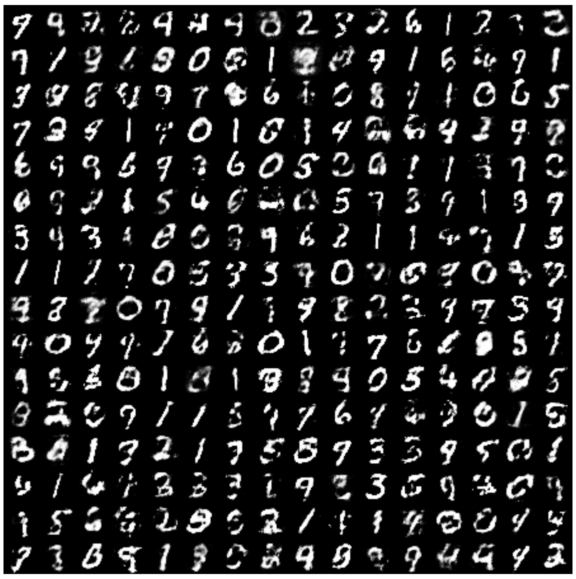


FIGURE 3.5: MNIST Generated data by generator trained with MMD

Chapter 4

Approaches to reduce complexity of MMD calculation

4.1 Introduction

In this chapter, I am going to describe techniques that I used to calculate the MMD loss with a reduced time complexity.

4.2 Regularized model to learn bi-lipschitz function

Definition 4.1. Bilipschitz function f is defined as a function that has upper and lower bound on the first order derivative i.e. $\exists$ two constants c_1 and c_2 such that $\forall$ a,b $\in$ Domain(f),

$$c_1\|a-b\| \leq \|f(a)-f(b)\| \leq c_2\|a-b\|$$

By definition, bi-lipschitz functions are bijections.

Theorem 4.2. *Li et al. (2017b) If k is a characteristic kernel and f is an injective function, then the resulted kernel $\hat{k} = k \circ f$ is also characteristic.*

Theorem 4.3. *Ramdas et al. (2014) The power of the two sample test using MMD with gaussian kernel is proportional to $O(\sqrt{\frac{m}{d}})$, where m is the available number of available samples and d is the number of dimension of input vectors.*

That means the above theorem states that the number of samples needed for the convergence of the power of 2 sample test using MMD to go to 1 must be higher than the dimension of input vector. That means, if the discriminator used learns a d-dimensional vector from n-dimensional input and the mapping is a bijection, we can use MMD directly on the d-dimensional output vectors. And the number of samples needed will be O(d), rather than O(n), leading to improved time complexity of training the neural network.

4.2.1 Regularized formulation

We explicitly constrain the discriminator function to a bilipschitz function with a regularization term in the loss function. A similar kind of approach was followed in Petzka et al. (2017) to constrain the function learned by the discriminator to a lipschitz function. The discriminator(f) loss function becomes

$$E_{x\sim P}f(x)-E_{y\sim Q}f(y)+\lambda_1\max(0,\frac{\|f(x)-f(y)\|}{\|x-y\|}-1)^2+\lambda_2\max(0,\epsilon-\frac{\|f(x)-f(y)\|}{\|x-y\|})^2$$

i.e. for each possible pair x,y we apply constraints ϵ and 1.0 as c_1 and c_2 in definition 4.1. Here, f is the function learned by the discriminator. The first term in the regularization term constrains the rate of change of function w.r.t domain to be less than 1. The second term in the regularization term constrains the rate of change of function w.r.t domain to be greater than a small constant ϵ. Now using theorem 4.2, we can say that k(f(x), f(y)) is a characteristic kernel, provided that k is a characteristic kernel. Thus, the new formulation of MMD becomes

$$\text{MMD loss} = E_{x_1,x_2\sim P}k(f(x_1),f(x_2))+E_{y_1,y_2\sim Q}k(f(y_1),f(y_2))-2E_{x\sim P,y\sim Q}k(f(x),f(y))$$

We train the generator using this new MMD formulation.

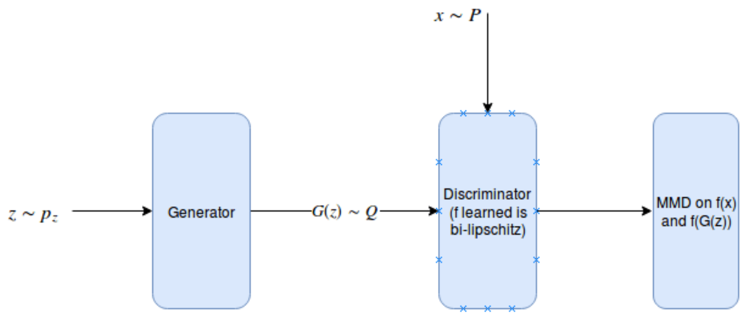


FIGURE 4.1: Pictorial representation of the regularized formulation to learn bilipschitz function

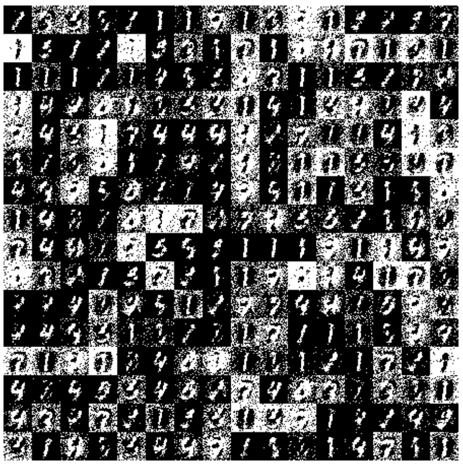


FIGURE 4.2: Generated data with regularized formulation of discriminator to a bilipschitz function and MMD applied on output of discriminator

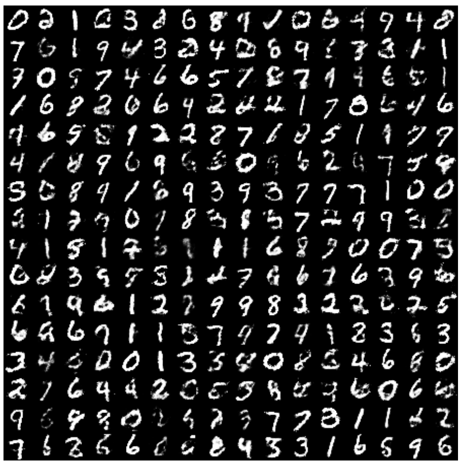


FIGURE 4.3: Generated data with regularized formulation of discriminator to a lipschitz function and MMD applied on output of discriminator

4.2.2 Observations

Fig. 4.2 and 4.3 show that the model with both the regularization terms doesn't perform well compared to the model, where the discriminator is trained only to learn a lipschitz function Petzka et al. (2017). One of the possible reasons for this is that we are trying to give an explicit lower bound on the rate of change of the function. This may not always be correct to do so, since the lower bound can be infinitesimally small for a bilipschitz function.

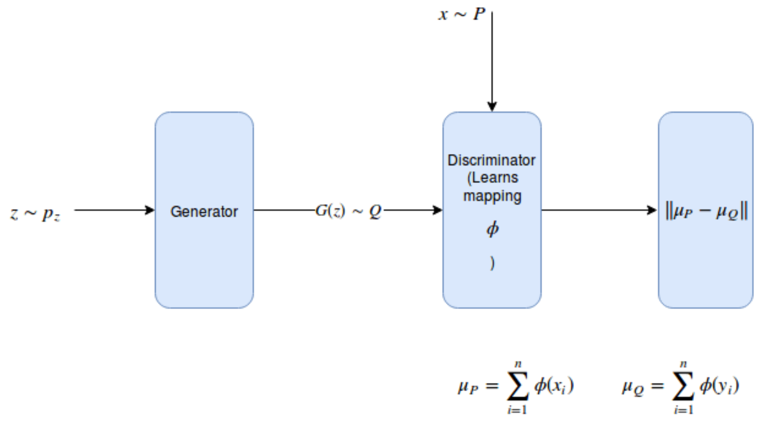

$x \sim P$
$z \sim p_z$
Generator
$G(z) \sim Q$
Discriminator (Learns mapping ϕ)
$\|\mu_P - \mu_Q\|$
$\mu_P = \sum_{i=1}^{n} \phi(x_i)$
$\mu_Q = \sum_{i=1}^{n} \phi(y_i)$

FIGURE 4.4: Pictorial representation of the model that learns gaussian kernel

4.3 Random weights to explicitly learn Gaussian Kernel

The following formulation of gaussian kernel has been referred from Mairal et al. (2014).

Lemma 4.4. *For all x and y in R^m and $\sigma > 0$*

$$\exp(-\frac{\|x-y\|^2}{2\sigma^2}) = (\frac{2}{\pi\sigma^2})^{m/2} \int_{w \sim R^m} \exp(-\frac{\|x-w\|^2}{\sigma^2}) \exp(-\frac{\|y-w\|^2}{\sigma^2}) dw$$

So, we can try to approximate the gaussian kernel on training data using importance weights $\eta = [\eta_l]_{l=1}^p$ in R_+^p and sampling weights W $= [w_l]_{l=1}^p$ with each w_i in R^m such that

$$\exp(-\frac{\|x-y\|^2}{2\sigma^2}) \approx \sum_{l=1}^{p} \eta_l \exp(-\frac{\|x-w_l\|^2}{\sigma^2}) \exp(-\frac{\|y-w_l\|^2}{\sigma^2})$$

Hence, we can train a neural network with weights W and η with the loss function

$$E_{(x,y)} \exp(-\frac{\|x-y\|^2}{2\sigma^2}) - \sum_{l=1}^{p} \eta_l \exp(-\frac{\|x-w_l\|^2}{\sigma^2}) \exp(-\frac{\|y-w_l\|^2}{\sigma^2})$$

The expectation is taken w.r.t. n training samples $(x_i, y_i)_{i=1}^n$.

After learning, a new unit norm point x $\sim R^m$ is mapped to the vector $[\sqrt{\eta_l}\exp(-\frac{\|x-w_l\|^2}{\sigma^2})]_{l=1}^p$ in R^p. Now this representation in R^p can be taken as the embedding of the point x in the RKHS corresponding to the gaussian kernel. If we denote this mapping by ϕ, MMD loss can be reformulated as

$$\text{MMD} = \|E_{x\sim P}\phi(x) - E_{y\sim Q}\phi(y)\|$$

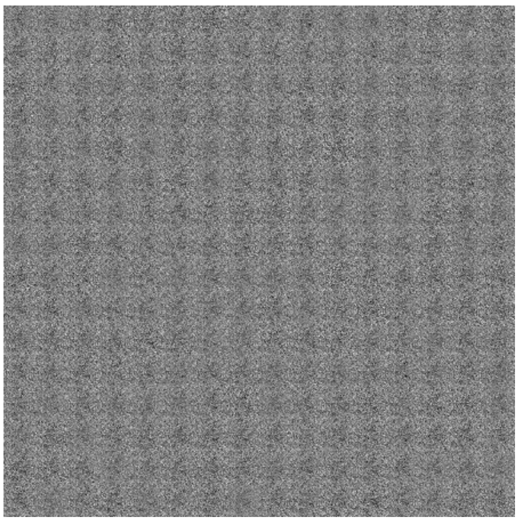


FIGURE 4.5: Generated data with random weights in discriminator to learn gaussian kernel

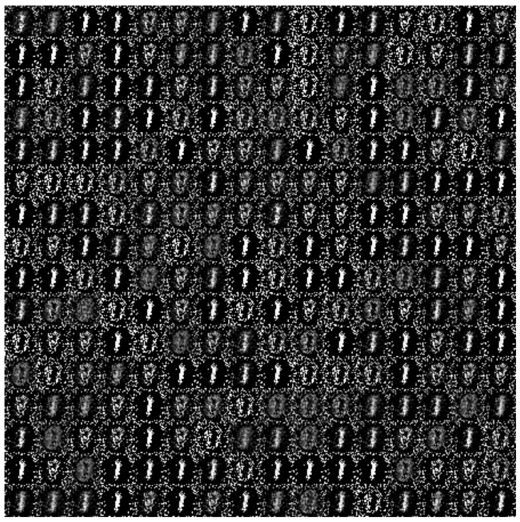


FIGURE 4.6: Generated data with discriminator trained to learn gaussian kernel

4.3.1 Observations

Figures 4.5 and 4.6 show that the model with weights trained to explicitly learn about gaussian kernel produces some results compared to the model with random weights. However, the overall quality of the generated data is not good at all. One of the possible reasons for the failure is that we are trying to approximate gaussian kernel by a model, which will have some loss and we use the same model to calculate MMD loss. Hence, the losses incurred in both the phases get added up.

4.4 Dimension reduction using local PCA

The real data and generated data vectors were passed through PCA to obtain low dimensional projections, on which MMD was applied.

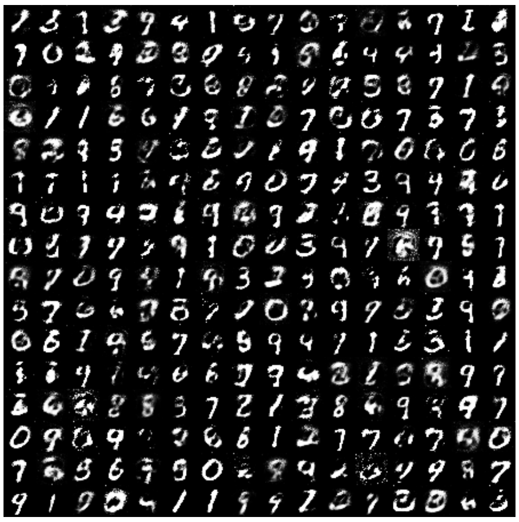


FIGURE 4.7: MMD applied on MNIST Data reduced to 100 dimension by PCA

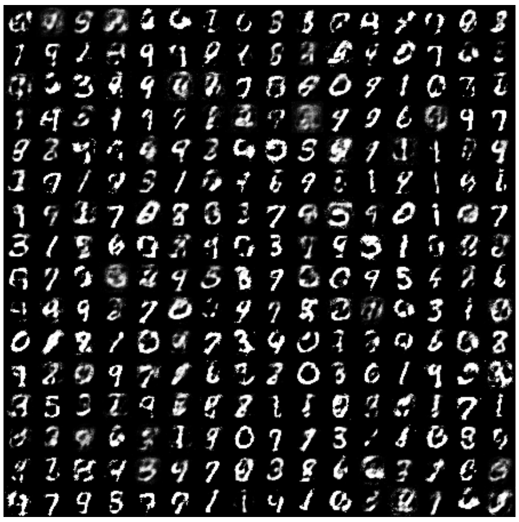


FIGURE 4.8: MMD applied on MNIST Data reduced to 200 dimension by PCA

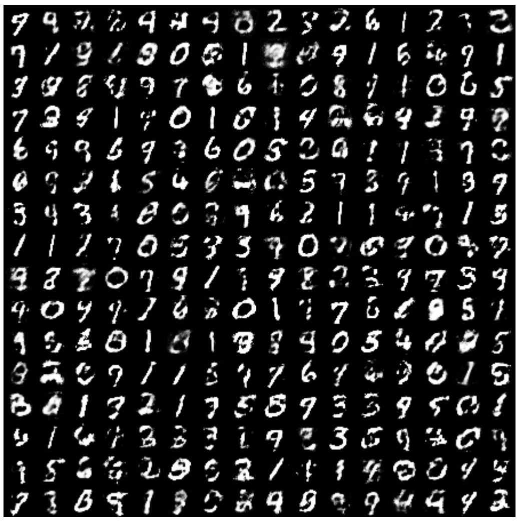


FIGURE 4.9: MMD applied on MNIST Data reduced to 400 dimension by PCA

4.4.1 Observations

Figures 4.7, 4.8, 4.9 show that reducing dimension using PCA doesn't improve results. The quality of mnist data produced by the model increases with the increase in the number of dimensions to which PCA reduces to. One of the possible reasons for this is that PCA learns a linear mapping from the input vectors to the output vectors. Suppose the manifold on which the data exists has support in R^k and the data points are in R^n. This mapping from $R^n \to R^k$ may not be a linear mapping. That implies, we need to learn non linear low dimensional mappings, rather than a linear one. That's one of the reasons why Li et al. (2017b)'s model, that comprises of an auto encoder to learn a low dimensional mapping, works better.

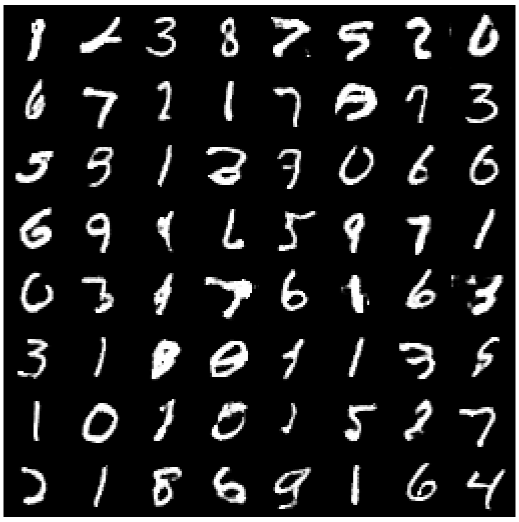


FIGURE 4.10: Results by MMD GAN model of Li et al. (2017b) that uses an autoencoder to learn low dimensional mapping

Chapter 5

Tangent Space Alignment

5.1 Introduction

In this chapter, I am going to discuss the theory and algorithm of a novel technique used to align manifolds of generated data and real data.

5.2 Reason of failure of KL divergence in GANs

The Kullback-Leibler (KL) divergence between unknown data distribution P_r and generated distribution P_g is given as

$$KL(P_r \| P_g) = \int_X P_r(x) \log \frac{P_r(x)}{P_g(x)} dx \tag{5.1}$$

This cost function has a unique minimum at $P_g = P_r$.

Generative adversarial networks have been shown to optimize the Jensen-shannon divergence

$$JSD(P_r||P_g) = 0.5KL(P_r||P_A) + 0.5KL(P_g||P_A)$$

where $P_A = 0.5(P_r + P_g)$.

5.2.1 Sources of instability

The given theorems in this section have been referred from Arjovsky and Bottou (2017). I discuss some of the proofs, since I will use them to develop my theory.

Theorem 5.1. *If two distributions P_r and P_g have support contained on two disjoint compact subsets M and P respectively, then there is a smooth optimal discriminator D^*: $X \to [0,1]$ that has accuracy 1 and $\nabla_x D^*(x) = 0 \; \forall \; x \in M \cup P$.*

Proof. The discriminator is trained to maximize

$$E_{x \sim P_r} log(D(x)) + E_{x \sim P_g} log(1 - D(x))$$

Since M and P are compact and disjoint, $0 < \delta =$ d(P, M) the distance between both sets. Define

$$\hat{M} = x : d(x, M) \leq \delta/3$$

$$\hat{P} = x : d(x, P) \leq \delta/3$$

$\hat{M}$ and $\hat{P}$ are disjoint compact sets. Construct a smooth function D^*: X $\to$ [0,1] such $D^*(x) = 1$ if x $\in \hat{M}$ and 0 if x $\in \hat{P}$. Since, the discriminator gives 0 in the support of P_g and 1 in the support of P_r, the discriminator is optimal and has accuracy 1. Furthermore, let x be in M $\cup$ P. If x $\in$ M, then $\exists$ an open ball B = B(x, $\delta/3$) such on which D is constant. Similar argument holds for x $\in$ P. Thus, $\nabla_x D^*(x) = 0 \; \forall$ x $\in$ M $\cup$ P. □

Definition 5.2. Let F be the space in which our two regular boundary free manifolds M and P are embedded. Let x be a point of intersection of the two manifold M and P. Then M and P are said to intersect transversely in x if T_xM + T_xP = T_xF, where T_xM denotes the tangent space of M around x.

Definition 5.3. Two manifolds without boundary M and P are said to perfectly align if there is an x $\in$ M $\cup$ P such that M and P don't intersect transversely at x.

Basic intuition If F is the real space R^n, then T_xF will contain a basis of n linearly independent vectors. If the two manifolds M and P are each of dimension less than n, then any point on M and P will have a set of less than n linearly independent vectors as basis of its tangent space. If M and P align at a point x, the T_xM = T_xP and so the basis vectors of M and P in the tangent space will be common at x. Hence, the number of basis vectors in M $\cup$ P cannot be equal to n, implying T_xP + T_xM $\neq$ T_xF.

Lemma 5.4. ***General position lemma*** *Let M and P be two regular sub manifolds of R^d that don't have full dimension. Let η, η' be arbitrary independent continuous random variables. The perturbed manifolds $\hat{M}$ and $\hat{P}$ are given as $\hat{M} = M + \eta$ and $\hat{P} = P + \eta'$. Then*

$$Pr_{\eta,\eta'}(\hat{M} \textit{ doesn't perfectly align with } \hat{P}) = 1$$

Lemma 5.5. *Let M and P be two regular sub manifolds of R^d that don't perfectly align and don't have full dimension. Let L = M $\cap$ P . If M and P don't have the boundary, then L is also a manifold and has strictly lower dimension than both of M and P. If they have boundary, L is a union of at most 4 strictly lower dimensional manifolds. In both cases, L has measure 0 in both M and P.*

Theorem 5.6. *Let P_r and P_g be two distributions that have support contained in two closed manifolds M and P that don't perfectly align and don't have full dimension. Also, assume that P_r and P_g are continuous in their respective manifolds, meaning that if there is a set A with measure 0 in M, then P_r(A) = 0. Then, there exists an optimal discriminator D^*: X $\rightarrow$ [0,1] that has accuracy 1 and for almost any x in M $\cup$ P, D^* is smooth in a neighborhood of x and $\nabla_x D^*(x) = 0$. Hence,*

$$JSD(P_r||P_g) = \log 2$$

$$KL(P_g||P_r) = \infty$$

$$KL(P_r||P_g) = \infty$$

Proof. By Lemma 4.5, we have that L=M $\cap$ P has measure 0. That implies P_r(L) = 0 and P_g(L) = 0 by continuity assumption. Let x $\in$ M $\setminus$ L. Then x $\in$ P^c, which is an open set. Also, there will exist a radius ϵ_x such that B(x, ϵ_x) $\cap$ P = ϕ. Hence, define $\hat{M}$ as

$$\hat{M} = \cup_{x \in M \ P} B(x, \epsilon_x)$$

Similarly define $\hat{P}$. Define a smooth function D^* such that D^*(x) = 1 if x $\in \hat{M}$ and 0 if x $\in \hat{P}$. This function D^* differentiates between the two manifolds and hence, has accuracy 1. □

5.2.2 Significance of theorems 5.6 and lemmas 5.4 and 5.5 to GAN training

Let say at any point of time, the distribution learned by generator is Q and it's support lies on the manifold M_Q. As has been previously used, P denotes the real data distribution and M_P is the manifold on which the data lies. With probability 1, lemma 5.4 says that M_P and M_Q will intersect transversely, i.e. they won't align at any intersection point. Since, we have assumed that the manifolds are continuous, this will imply that the intersection points L = M∩P will form a set of measure 0 and so, the probability of their occurrence will be 0. Now, this case will act as case of theorem 5.1, where we assumed that M_P and M_Q are disjoint. Once, we get the case of disjointedness, we can always find a function that separates the two sets M_P and set M_Q. This argument shows that, no matter what we do, KL divergence will always stay ∞, thus making the gradient 0. This will lead to zero learning in the case of the generator.

5.3 Distance metrics for probability distribution

Theorem 5.6 and lemma 5.4, 5.5 show that with probability 1, there will always exist a discriminator with accuracy 1 for a generator trained with KL divergence loss function. This was the reason why researchers started using probability distance metrics to train GANs. Let X be a compact metric space endowed with Borel σ algebra Σ. Let supp(μ) denotes its support, i.e. the manifold on which the distribution μ lies. Let P and Q be two probability distributions on X, with manifolds M_P and M_Q respectively. The manifolds are k dimensional, embedded in n dimensional space. The wasserstein p-distance W_p is defined as (Villani (2008)).

$$W_p(P,Q) = (\inf_{\gamma \in \prod(P,Q)} \int_{x,y} (\|x-y\|)^p d\gamma(x,y))^{1/p}$$

The basic intuition behind wasserstein distance is that given two set of probability distributions, we create a map between the supports of the two distributions such that the distance between the domain and the image is the smallest among all possible mappings. If $T_*(P)$ = Q means a Borel map T from P to Q i.e. $\int_{T^{-1}(B)} p$ = $\int_B$q for any Borel set B $\subset M_P$, then

$$W_p(P,Q) = \inf_{T_*(P)=Q} (\mathop{\mathbb{E}}_{x\sim P} \|x - T(x)\|^p)^{1/p}$$

The kantorovich duality theorem transforms the formulation of wasserstein-1 distance to its dual form

$$W_1(P,Q) = \inf_{f_L} E_{x\sim P} f_L(x) - E_{x\sim Q} f_L(x)$$

where f_L is a L-lipschitz function i.e. $\forall$ a,b $\in$ domain of f_L

$$\|f_L(a) - f_L(b)\| \leq L\|a - b\|$$

Maximum mean discrepancy(MMD) Gretton et al. (2012a) distance has a similar formulation, where we use the function space F = f: $\|f\|_H \leq 1$ and H denotes a reproducing kernel hilbert space. Since, both of the metrics are distance measures, the theorems applied to one measure can also be applied to the other measure.

5.3.1 Problems with wasserstein distance

The definitions and theorems have been referred from Lui et al. (2017).

Definition 5.7. (Realistic samples) If M_P positively aligns with M_Q, then samples from Q are realistic w.r.t. P.

Basic intuition Given a point x on M_P, if we add small error ϵ to x, then $\hat{x}$ = x+ϵ $\in M_P$ if $\hat{x}$ travels along T_x.

We need two assumptions for the following theorem.

Assumption 1. P and Q are compactly supported on M_P and M_Q i.e. L^k (M_P), L^k $(M_Q) > 0$

i.e. since by assumption the manifolds are k-dimensional, the lebesgue measure L^k defined as the volume of the set on which the manifold lies must be greater than 0.

Assumption 2. P and Q are continuous i.e. for any set B $\subset R^n$ if P(B) = Q(B) = 0 then L^k (B) = 0.

Theorem 5.8. *Let $\epsilon > 0$. Let $\Gamma = \{ Q: W_p(P, Q) < \epsilon \}$ and $\Gamma_1 = \{ Q: W_p(P, Q) < \epsilon; L^k(M_P \cap M_Q) > 0 \}$. Under Assumption 1, Γ - Γ_1 is dense in Γ.*

Proof. Let Q be the current distribution lying on a manifold M_Q, with distance W_p(P,Q) from M_P. Let $\delta = (\epsilon$ - W_p(P,Q))/10. By general position lemma, $M_Q + \epsilon$ intersects transversely with M_P almost with probability 1. By triangle inequality,

$$W_p(P, Q + \delta) < W_p(P, Q) + \delta < \epsilon$$

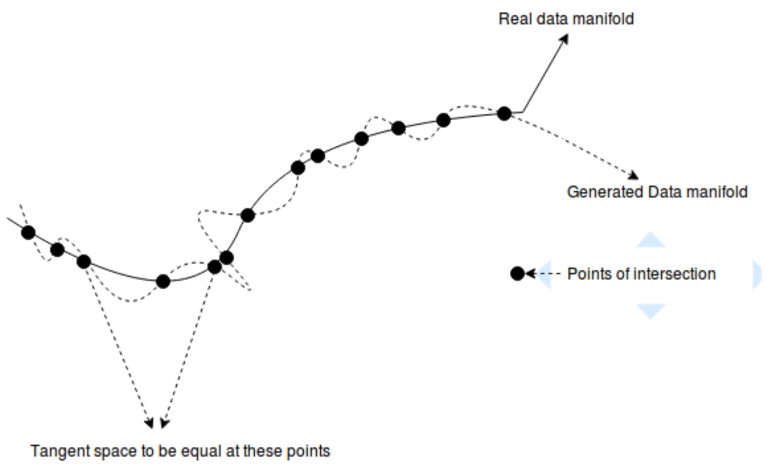
Since, we can make δ arbitrarily small, the number of possible manifolds M_Q with W_p distance less than ϵ stays infinite. $\square$

So the above theorem states that there are infinitely many manifolds that are at a distance $\leq \epsilon$ from the real data manifold. Hence, with decreasing loss function, we can't say that the generated data distribution is converging to the real manifold. Thus, we need to ensure through regularization terms that the generated data distribution converges, with the rate of decrease in loss being an indicator of the rate of convergence.

5.4 Aim of the approach

Fig. 3.4 shows the performance of MMD on toy dataset. The results show that MMD can't remove the outliers generated between any 2 clusters and also the outliers present at the centre of the circular manifold. We match this phenomenon with the fact that infinitely many manifolds can be generated by the model trained with MMD with loss ϵ. Since theorem 5.8 states that the number of candidate manifolds is infinite when the W_p distance is less than ϵ, the resulting generated manifold can be anyone of the candidate manifolds. Thus, we need to prune out some of them on the basis of some characteristic of the generated manifold, so that we can also remove the outlier points present in the generated data.

Lemma 5.9. *By lemma 5.7 we need to find manifolds that align with the real data manifold. Hence, the generated manifold must have the same tangent space as that of the real manifold at the points of intersection.*



5.4.1 MMD with tangent space

The kernel used in the MMD formulation was changed to a new kernel, that included the tangent space information at each point. For computing tangent directions, I choose a small neighborhood of radius ϵ around each point. This neighborhood contains all information about the local manifold at the point. Calculating the major axes of variance in this neighborhood gives the basis that span the tangent space at the point. I use local PCA to calculate the major axes of variance. Once I get the tangent space at a point x, I can project the point onto the tangent space to get the projected embedding Q_x. This projected embedding contains the information of tangent space. The new kernel to be used in MMD is

$$\hat{k}(x, y) = \exp(\frac{-\|x - y\|^2}{\sigma^2}) \exp(\frac{-\|Q_x - Q_y\|^2}{\eta^2})$$

This kernel matches the tangent directions of points x and y, if x and y are closer to each other. A similar kind of idea was used in Arias-Castro et al. (2013) where the authors used tangent space information to differentiate intersecting clusters.

Theorem 5.10. *Steinwart (2001) Let X be a bounded domain, i.e. if $x \in X$, $\|x\| < r$. Let f be a function and let it's taylor expansion in 0 be given as follows.*

$$f(x) = \sum_{n=0}^{\infty} a_n x^n$$

If all a_n's > 0, then $f(\langle x, y\rangle)$ is a universal kernel on every compact subset of X. Also, if k(x, y) is a universal kernel, then

$$\frac{k(x,y)}{\sqrt{k(x,x)k(y,y)}}$$

is also a universal kernel.

Lemma 5.11. *The new kernel $\hat{k}$ is a universal kernel.*

Proof. Q_x and Q_y are functions of B(x, ϵ) and B(y, ϵ) respectively, where B(x, ϵ) represents a ball of radius ϵ around x. Thus, comparing Q_x and Q_y implies pairwise comparison of a and b for all a $\in$ B(x, ϵ) and b *in* B(y, ϵ). This can be written as

$$k(Q_x, Q_y) = \prod_{a \in B(x,\epsilon), b \in B(x,\epsilon)} k(a,b)$$

Each a and b are functions of x and y respectively i.e.

$$a = x + \epsilon_1, b = y + \epsilon_2$$

where ϵ_1, $\epsilon_2 < \epsilon$. k(a, b) is simply k(f(x), f(y)) where f is a linear function. Thus,

$$k(Q_x, Q_y) = \prod_f k(f(x), f(y))$$

where each f is a linear function. Now, since k is a gaussian kernel, which is universal kernel by definition, $\hat{k}$, which is a product of k's, is also universal. This comes from the theorems 4.2 and 5.10. By theorem 4.2, we can say that k(f(x), f(y)) where f is a linear function (bijection) will also be a universal kernel, given that k is a universal kernel. Also, since each kernel function has positive coefficients in it's taylor expansion, a product of the functions will also have positive coefficients in it's taylor expansion. Thus, $\hat{k}$ will also be a universal kernel, which can be used in the MMD formulation. $\square$

Algorithm 1 Algorithm to compute new kernel $\hat{k}$(x, y) to include tangent space information

```
 1: procedure KERNEL($x_1, x_2, ..., x_n$, $y_1, y_2, ..., y_n$, $\epsilon$, $\sigma$, $\eta$, k)    ▷ $x_i$‘$s$ and $y_j$‘$s$ are
    real data samples and generated data samples respectively.
 2:     for z ∈ ($x_1, ..., x_n$) do                ▷ Find the tangent space at each $x_i$‘
 3:         Find all $x_i$‘s such that $\|x_i - z\| < \epsilon$. Let $S_z$ be the set.
 4:         Use local PCA to find k major axes of the tangent space at z using $S_z$.
 5:         Project z onto the axes obtained to get $Q_z$.
 6:     end for
 7:     for z ∈ ($y_1, ..., y_n$) do                ▷ Find the tangent space at each $y_i$‘
 8:         Find all $y_i$‘s such that $\|y_i - z\| < \epsilon$. Let $S_z$ be the set.
 9:         Use local PCA to find k major axes of the tangent space at z using $S_z$.
10:         Project z onto the axes obtained to get $Q_z$.
11:     end for
12:     Let S = $\{x_i\} \cup \{y_i\}$              ▷ Union of real data and generated data
13:     for ($z_1, z_2$) ∈ SxS do ▷ Find the new kernel $\hat{k}$ for each pair ($z_1, z_2$) in SxS
14:         $\hat{k}(z_1, z_2) = \exp(\frac{-\|z_1-z_2\|^2}{\sigma^2}) \exp(\frac{-\|Q_{z_1}-Q_{z_2}\|^2}{\eta^2})$
15:     end for
16:     Return $\hat{k}$
17: end procedure
```

Algorithm 2 Algorithm to compute new MMD formulation

```
1: procedure TANGENT MMD($x_1, x_2, ..., x_n$, $y_1, y_2, ..., y_n$, $\epsilon$, $\sigma$, $\eta$, k)    ▷ $x_i$‘$s$ and
   $y_j$‘$s$ are real data samples and generated data samples respectively.
2:     $\hat{k}$ ← KERNEL($x_1, x_2, ..., x_n$, $y_1, y_2, ..., y_n$, $\epsilon$, $\sigma$, $\eta$, k)
3:     $M\hat{M}D \leftarrow \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\hat{k}(x_i, x_j) + \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\hat{k}(y_i, y_j)$ -
   $\frac{2}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\hat{k}(x_i, y_j)$
4:     Return $M\hat{M}D$
5: end procedure
```

5.4.2 Observations

The parameter η is a hyper parameter to be tuned. For the toy dataset, Fig 5.1, 5.2, 5.3 show that on increasing η from 0.1 to 1.0, the quality decreases substantially. Also, the difference between training generator with MMD loss (fig. 5.1) and MMD loss with tangent space information (fig. 3.4) is not that substantial. The quality of

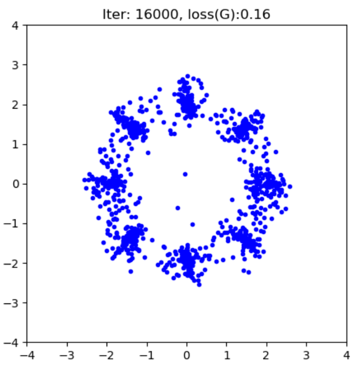


FIGURE 5.1: Generated data using MMD with tangent space η=0.1

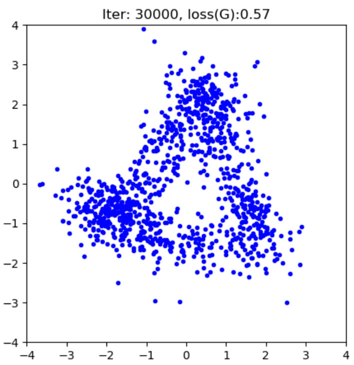


FIGURE 5.2: Generated data using MMD with tangent space η=0.5

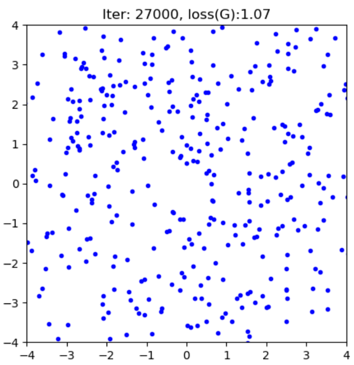


FIGURE 5.3: Generated data using MMD with tangent space η=1.0

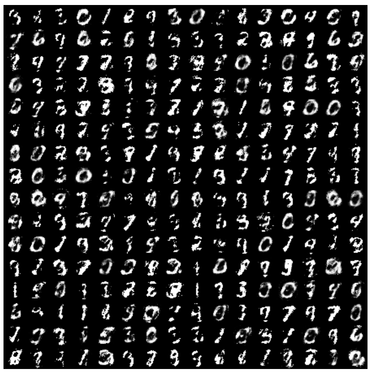


FIGURE 5.4: Data generated by mmd with tangent space at step 30 on mnist data

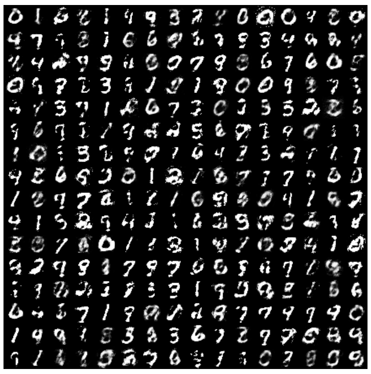


FIGURE 5.5: Data generated by mmd with tangent space at step 60 on mnist data

mnist images produced with MMD loss function having tangent space information (see Fig 5.4, 5.5, 5.6) is also not good enough compared to the quality of images produced by using simple MMD function (fig. 3.5). Hence, we can't say that the method improves the quality of generated data. One of the possible reasons is that I included PCA in the stochastic gradient descent process, which involves calculating the gradient of eigenvalues. The gradient involves computing the inverse of difference of eigenvalues. As a result, the gradient can explode, leading to divergence in training.

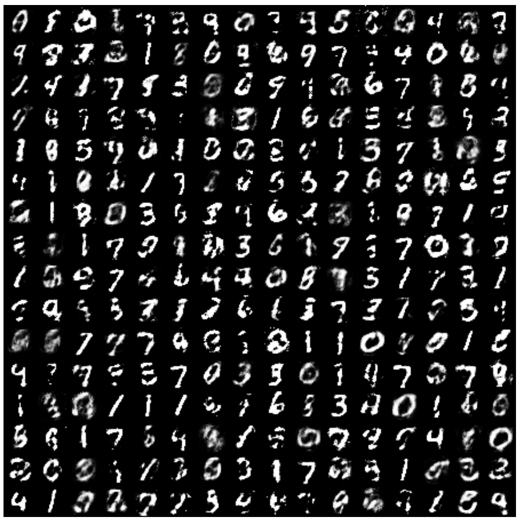


FIGURE 5.6: Data generated by mmd with tangent space at step 100 on mnist data

5.4.3 Regularized Wasserstein GAN

We use the same technique as above. However, rather than changing the wasserstein loss function used in wasserstein GAN Arjovsky et al. (2017) explicitly, I added a regularization term. The regularization constraint looped over each point in real data. For each point x $\in$ real data, we compared the neighborhoods around x in real data manifold and in generated data manifold. We computed the projections Q_x, as described in previous subsection, for each point x in real data and generated data. Now, the regularization term R(P, Q) becomes

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\exp(\frac{-\|x_i - y_j\|^2}{\sigma^2})\exp(\frac{-\|Q_{x_i} - Q_{y_j}\|^2}{\eta^2})$$

where $x_1, ..., x_n$ are real data samples generated by probability distribution P and $y_1, ..., y_n$ are generated data samples, generated by probability distribution Q.

Thus the new wasserstein loss function becomes

$$E_{x \sim P} f(x) - E_{x \sim Q} f(x) + \lambda R(P, Q)$$

where f is a lipschitz function.

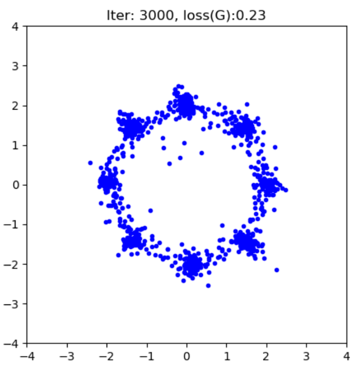


FIGURE 5.7: Data generated by regularized wasserstein GAN with $\lambda = 1.0$

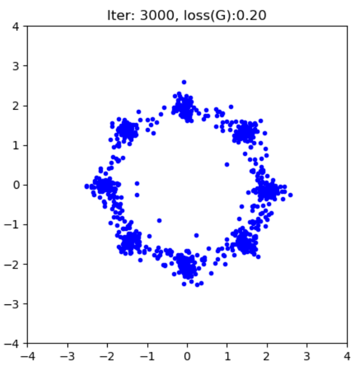


FIGURE 5.8: Data generated by regularized wasserstein GAN with $\lambda = 0.5$

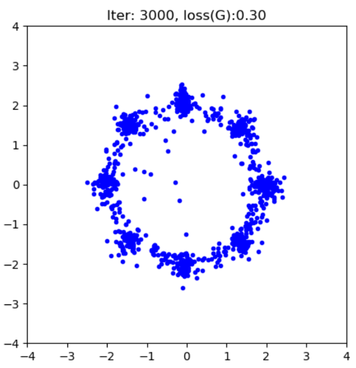


FIGURE 5.9: Data generated by regularized wasserstein GAN with $\lambda = 0.25$

5.4.4 Observations

The method doesn't seem to improve results produced by unregularized GANs. The figures 5.7, 5.8, 5.9 don't show any remarkable improvement over the results produced by unregularized wasserstein GANs (fig. 3.5). Hence, again we can't say that the results improved upon using the current method.

Chapter 6

Boosting

6.1 Introduction

In this chapter, I am going to discuss the theory and algorithm of a boosting technique used to remove outliers.

6.2 Algorithm

Basic Intuition Suppose the current generated samples are from probability distribution Q and it's support is M_Q. The real data distribution is P and it's support is M_P. The MMD formulation gives an equal weight to all the data points, while calculating the mean in the hilbert space. As a result, the outlier points in the generated set, which form a small fraction in M_Q are not given importance during training.

If the importance or weight of each outlier is increased, the model will try to converge the outliers to the real data distribution. This is the basic intuition of my model, where we boost each outlier point for faster convergence. The weight of a generated point, denoted by w, must be proportional to the distance of the point from the real data manifold. We can approximate this distance by the distance of the point from the closest point in the real data samples i.e. weight of a generated sample y is

$$w_y \propto \min_i \|x_i - y\|$$

where x_i‘s are real data samples. We use the differentiable function

$$w_y = \exp(\min_i \|x_i - y\|)$$

as the weight function. Next, we normalize the weights so that the sum of all weights becomes 1.

Lemma 6.1. *The weighted MMD formulation with equal weights given to the real data samples and generated data samples y having weight w_y, as derived above, is*

$$\textit{MMD loss} = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} k(x_i, x_j) + \sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j k(y_i, y_j) - \frac{2}{n}\sum_{i=1}^{n}\sum_{j=1}^{n} w_j k(x_i, y_j) \tag{6.1}$$

Proof. Let P and Q be the probability distributions over R^n i.e. the samples $x_1, ..., x_n$ (real data) and the samples $y_1, ..., y_n$ (generated data) are in R^n. Let f be the mapping from R^n to a reproducing kernel hilbert space (RKHS) as given by the universal kernel k in MMD formulation. Mean μ_P in RKHS is given by

$$\mu_P = \frac{1}{n}\sum_{i=1}^{n} f(x_i)$$

Mean μ_Q in RKHS is given by

$$\mu_Q = \sum_{i=1}^{n} w_{y_i} f(y_i)$$

μ_Q is just weighted mean of y_i‘s with weights w_{y_i} and $\sum_{i=1}^{n} w_{y_i} = 1$.

By the definition of MMD,

$$\text{Weighted MMD} = \|\mu_P - \mu_Q\|$$

Squaring both the sides, we get

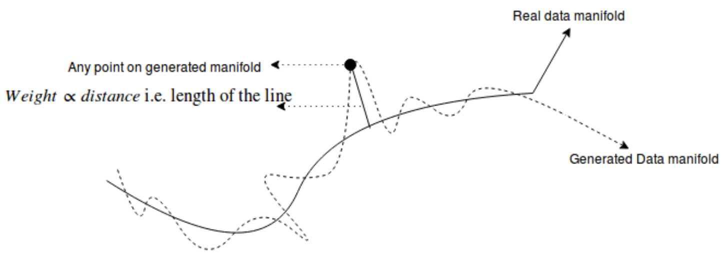
$$loss^2 = \|\mu_P - \mu_Q\|^2 = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} f(x_i)f(x_j) + \sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j f(y_i)f(y_j) - \frac{2}{n}\sum_{i=1}^{n}\sum_{j=1}^{n} w_j f(x_i)f(y_j)$$

By the kernel trick, we can replace f(x)f(y) by k(x,y) to get the required formulation.

□

Algorithm 3 Boosted MMD GAN

```
1: procedure BOOSTED MMD GAN($x_1, x_2, ..., x_n$, $\sigma$)    ▷ $X_i$'s are real data samples, $\sigma$ is a parameter to be tuned
2:     while No. of Iterations < Some global constant do
3:         $y_1, y_2, ..., y_n$ = Generator()
4:         for z ∈ ($y_1, y_2, ..., y_n$) do
5:             $dist_z = \min_i \|z - x_i\|$
6:             $w_z = \exp(\sigma * dist_z)$
7:         end for
8:         Normalize the weights $w_z$ such that $\sum_i w_{y_i} = 1$
9:         Use Eq 6.1 to calculate the boosted MMD loss.
10:        Train Generator to reduce the boosted MMD loss function.
11:    end while
12: end procedure
```


Real data manifold
Any point on generated manifold
Weight ∝ *distance* i.e. length of the line
Generated Data manifold

For defining the parameter σ, we used a number of heuristics. They are

- Constant value of σ, that is tuned by hyperparameter tuning on validation set. The results with $\sigma = 1.0$ on toy dataset are shown in Fig. 6.1, 6.2, 6.3.
- $\sigma \propto \frac{1}{\text{MMD loss}}$. That is, the parameter must increase with decrease in loss value. The results with $\sigma = 1.0/\text{loss}$ on toy dataset are shown in Fig. 6.4, 6.5, 6.6.

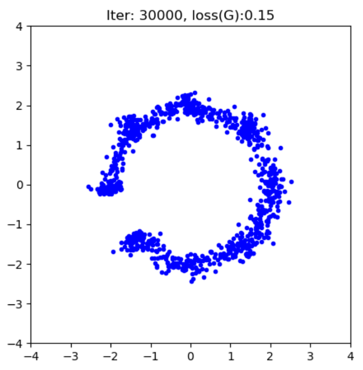


FIGURE 6.1: Data generated by weighted mmd with $\sigma = 1.0$ at step 30000

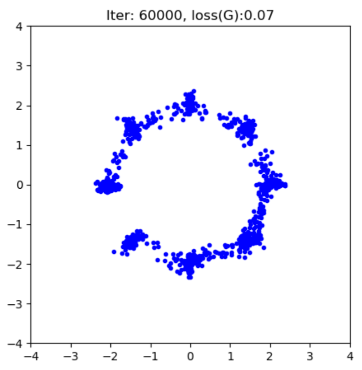


FIGURE 6.2: Data generated by weighted mmd with $\sigma = 1.0$ at step 60000

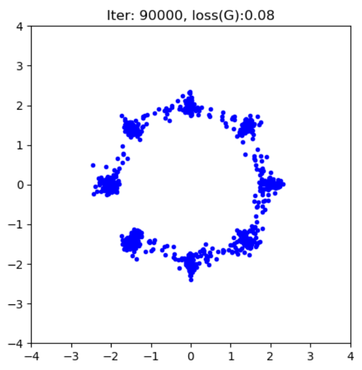


FIGURE 6.3: Data generated by weighted mmd with $\sigma = 1.0$ at step 90000

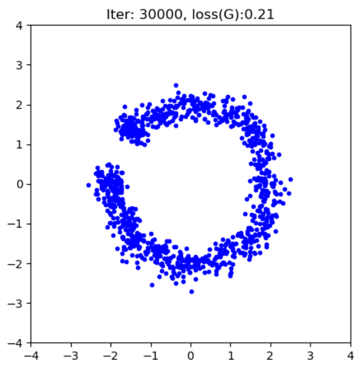


FIGURE 6.4: Data generated by weighted mmd with $\sigma = 1.0/\text{loss}$ at step 30000

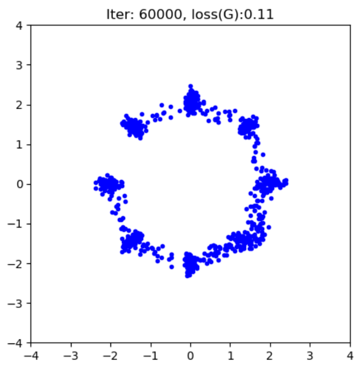


FIGURE 6.5: Data generated by weighted mmd with $\sigma = 1.0/\text{loss}$ at step 60000

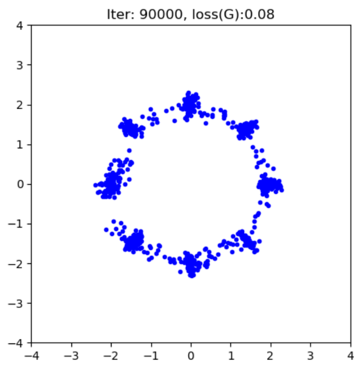


FIGURE 6.6: Data generated by weighted mmd with $\sigma = 1.0/\text{loss}$ at step 90000

- $\sigma \propto \frac{1}{\epsilon}$ where ϵ is the fraction of generated data samples with distance > 0 from real data manifold. That is, the parameter must increase with the decrease in the fraction of mistake set. The results with $\sigma = 1.0/\epsilon$ on toy dataset are shown in Fig. 6.7, 6.8, 6.9.

6.2.1 Observations

Figures 6.3, 6.6, 6.9 show that the outlier points that were present in the centre of the circular manifolds have been removed. Also, two or three clusters have been

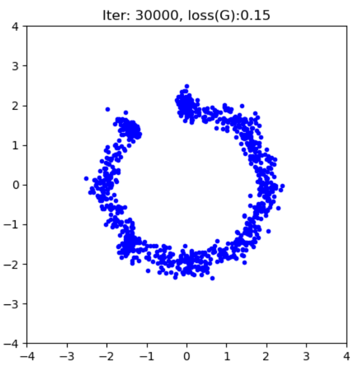


FIGURE 6.7: Data generated by weighted mmd with $\sigma = 1.0/\epsilon$ at step 30000

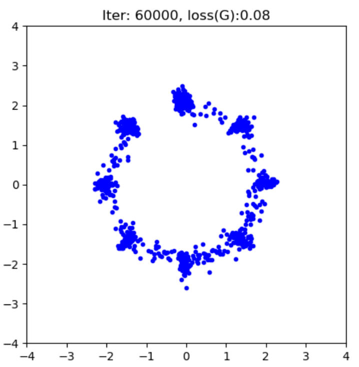


FIGURE 6.8: Data generated by weighted mmd with $\sigma = 1.0/\epsilon$ at step 60000

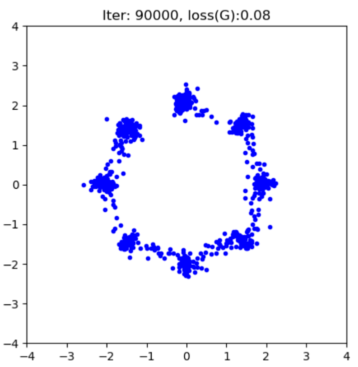


FIGURE 6.9: Data generated by weighted mmd with $\sigma = 1.0/\epsilon$ at step 90000

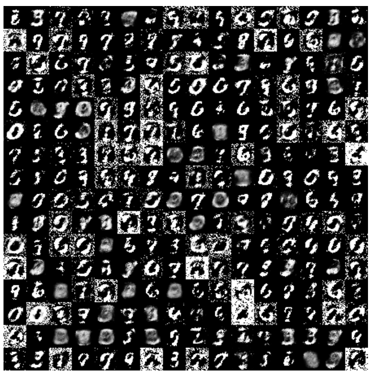


FIGURE 6.10: Data generated by weighted mmd with $\sigma = 0.1$ at step 30 on mnist data

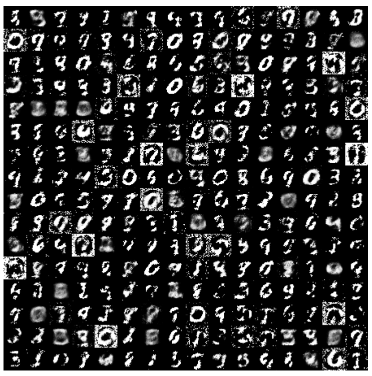


FIGURE 6.11: Data generated by weighted mmd with $\sigma = 0.1$ at step 60 on mnist data

separated out, giving a partial success.

The plots 6.16 and 6.17 show the behavior of MMD loss in both the models, first one being trained by simple MMD loss function and the second one being trained by boosted MMD loss function. Although, there are spikes in the convergence rate of weighted MMD trained model, the generated outliers that are at a large distance from the original manifold have been removed as compared to the model trained by simple MMD loss. This shows that weighted MMD function provides a scope to remove the outlier points and improve the result of generated data.

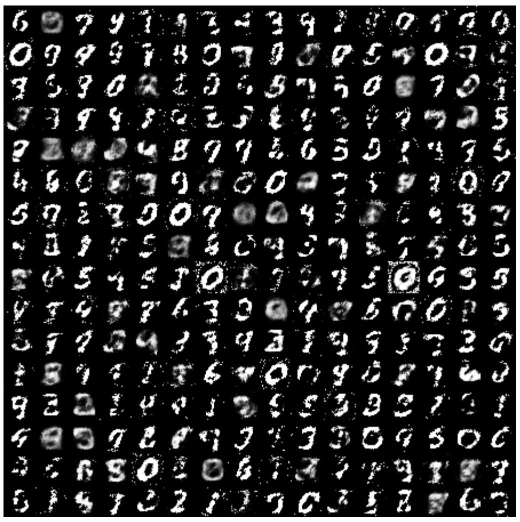


FIGURE 6.12: Data generated by weighted mmd with $\sigma = 0.1$ at step 100 on mnist data

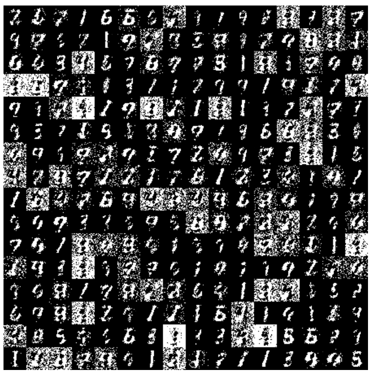


FIGURE 6.13: Data generated by weighted mmd with $\sigma = 1.0$/loss at step 30 on mnist data

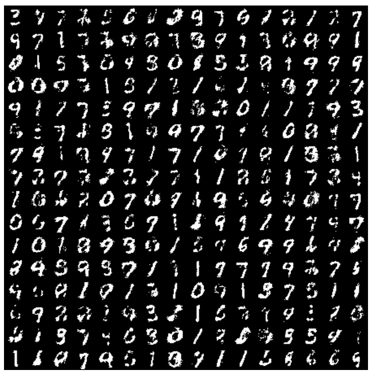


FIGURE 6.14: Data generated by weighted mmd with $\sigma = 1.0$/loss at step 60 on mnist data

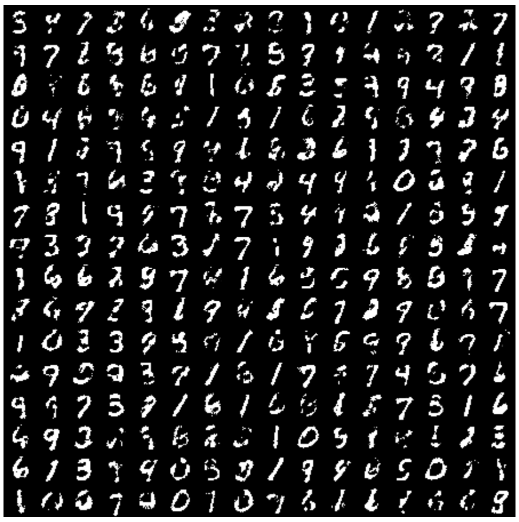


FIGURE 6.15: Data generated by weighted mmd with $\sigma = 1.0$/loss at step 100 on mnist data

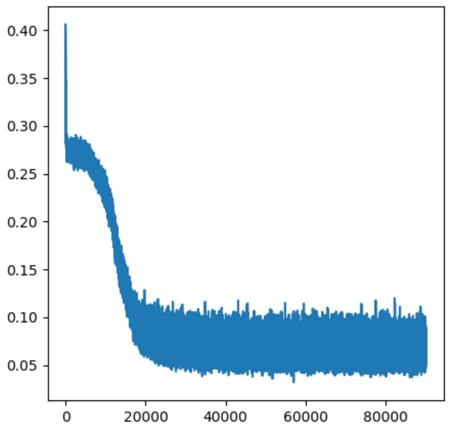


FIGURE 6.16: MMD Loss as a function of time for a model trained with MMD Loss

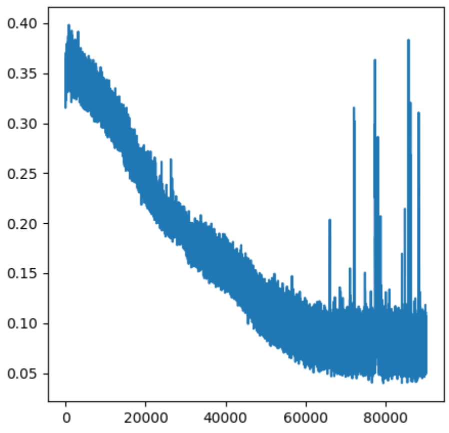


FIGURE 6.17: MMD Loss as a function of time for a model trained with weighted MMD Loss

Chapter 7

List of other approaches

7.1 Introduction

Here, I am going to list down some approaches with the theory behind them that I had implemented but failed.

7.2 Regularized MMD

The power of a hypothesis testing is defined as 1 - β where β is defined as the probability of rejecting the alternate hypothesis H_1 given that the alternate hypothesis H_1 was true. If we map our problem to hypothesis testing, we can say that if P is the real data distribution and Q is the generated data distribution,

$$H_0 : P = q$$

$$H_1 : P \neq Q$$

If MMD is used as the tool for the hypothesis testing, we assume that the power of the test is 1 - β. Danafar et al. (2013) show that the power of the test can be improved by using a regularized version of MMD (RMMD) i.e.

$$RMMD = MMD(P, Q)^2 - k_P \|\mu_P\|^2 - k_Q \|\mu_Q\|^2$$

where k_P and k_Q are variables in [0, 1].

Let the real data samples be $x_1, ..., x_n$ and generated data samples be $y_1, ..., y_n$. Let $z_i = (x_i, y_i)$. Denote $h(z_i, z_j) = k(x_i, x_j) + k(y_i, y_j)$ - $k(x_i, y_j)$ - $k(y_i, x_j)) + h'(z_i, z_j)$. $h'(z_i, z_j) = k_P\ k(x_i, y_j) + k_Q\ k(y_i, x_j)$. Let $k_P = k_Q =$ k. Replacing each term in RMMD by its sample estimate gives the sample estimate of RMMD, denoted by $R\hat{M}MD$.

Theorem 7.1. *If $E(h^2) < \infty$, then the asymptotic convergence of $R\hat{M}MD$ is given by*

$$\frac{1}{\sqrt{n}}(R\hat{M}MD - RMMD) \rightarrow \mathcal{N}(0, \sigma^2)$$

where, $\sigma^2 = E_{z_1}(E_{z_2}h(z_1, z_2)^2)$ - $(E_{z_1}E_{z_2}h(z_1, z_2))^2$.

Theorem 7.2. *The maximum power of RMMD is achieved at k=1.*

Proof. Let A = $k(x_i, x_j) + k(y_i, y_j)$ and B = $k(x_i, y_j) + k(y_i, x_j)$. Then

$$\begin{aligned}\sigma^2 &= E_{z_1}(E_{z_2}h(z_1, z_2)^2) - (E_{z_1}E_{z_2}h(z_1, z_2))^2\\ &= E(A + (1-k)B)^2 + E^2(A + (1-k)B)\\ &= E(A^2) + (1-k)^2E(B^2) + 2(1-k)E(AB) - E^2(A) - (1-k)^2E^2(B) - 2(1-k)E(A)E(B)\\ &= Var(A) + (1-k)^2E(B^2)\end{aligned}$$

In the derivation above, we have assumed that E(AB)=E(A)E(B), i.e. A and B are independent of each other. σ^2 is minimum at k=1 (by inspection). □

Since, MMD is RMMD with k=0 and theorem 7.2 shows that maximum power is achieved at k=1, RMMD with k=1 has higher power than MMD. Since, the method RMMD is of higher power, more mistake manifolds can be pruned out easily by RMMD w.r.t. MMD.

7.2.1 Observation

The RMMD process applied to the generator to learn toy dataset could learn only 1 cluster out of the 8 clusters present on the real data. Thus, it failed in practice compared to the MMD approach.

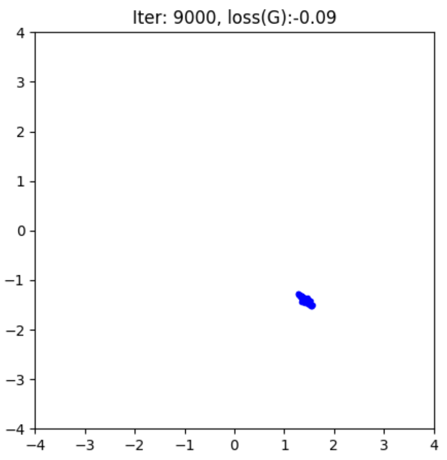


FIGURE 7.1: Generated data with RMMD at step 9000 on toy dataset

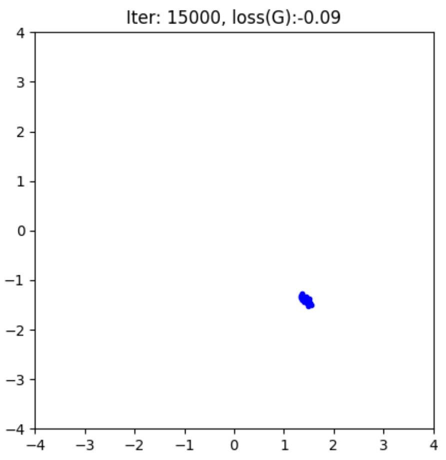


FIGURE 7.2: Generated data with RMMD at step 15 000 on toy dataset

7.3 Using polynomial kernel in place of gaussian kernel

Since polynomial kernel (f(x) = $(1-x)^(-\alpha)$, $\alpha > 0$) can be written as

$$(1-x)^{-\alpha} = \sum_{n=0}^{\infty} \binom{-\alpha}{n} (-1)^n x^n$$

where $\binom{-\alpha}{n}(-1)^n > 0 \ \forall$ n, the theorem 5.10 implies that polynomial kernels are universal kernels and so can be used in MMD formulation.

Basic intuition The reason we want to use polynomial kernels in place of gaussian kernels is that gaussian function has exponentially degrading coefficients in taylor expansion, while polynomial kernels have coefficients that increase with degree.

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

The maximum mean discrepancy compares two probability distributions by comparing all statistical moments. That's the reason why theorem 5.10 specifies a function as universal if it has all components in it's infinite taylor expansion. The weight given to a statistical moment of order n is proportional to the coefficient a_n in the expansion of function f used as kernel. Since gaussian kernels have exponentially degrading coefficients, they will give more importance to low degree statistical moments to compare two probability distributions. However polynomial kernels, whose coefficients increase with n, will give importance to all statistical moments to compare two probability distributions.

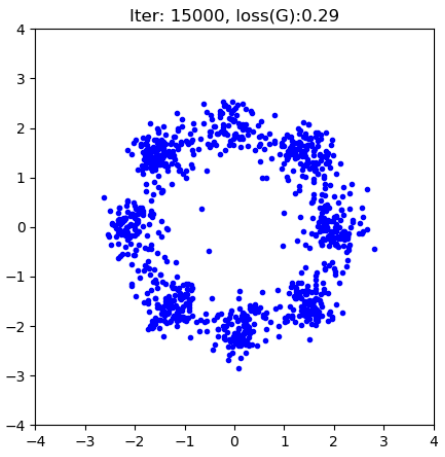


FIGURE 7.3: Generated data with polynomial kernel at step 15000 on toy dataset

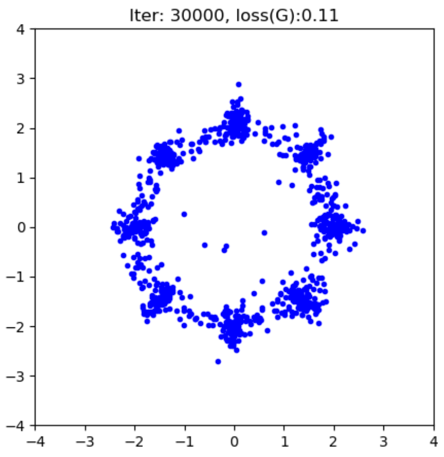


FIGURE 7.4: Generated data with polynomial kernel at step 30000 on toy dataset

7.3.1 Observations

Polynomial kernels perform similarly compared to gaussian kernels, when they are used in the MMD formulation. Hence, my claim that outliers will be removed by use of polynomial kernels, since they use higher order statistical moments, doesn't seem to work in this case.

Chapter 8

Conclusion

Boosting outliers to have higher weights at each iteration provides a scope for improving the GAN model. This method successfully removed the outliers that are present at a higher distance from the original manifold and also the outliers present in between two or three clusters in the original toy manifold. The only parameter that needs to tuned for boosting is η, that computes the bandwidth of weights based on distance. This parameter needs to be tuned to get optimal performance of GANs.

Future work will involve finding a closed form solution of the parameter η, as has been derived in Adaboost (Freund et al. (1999)). Optimal parameter η will result in pruning of more outliers present in the generated data, that will lead to improvement in quality of data generated by GANs.

Bibliography

Arias-Castro, E., Lerman, G., and Zhang, T. (2013). Spectral clustering based on local pca. *arXiv preprint arXiv:1301.2007*.

Arjovsky, M. and Bottou, L. (2017). Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

Banijamali, E., Ghodsi, A., and Poupart, P. (2017). Generative mixture of networks. *arXiv preprint arXiv:1702.03307*.

Che, T., Li, Y., Jacob, A. P., Bengio, Y., and Li, W. (2016). Mode regularized generative adversarial networks. *arXiv preprint arXiv:1612.02136*.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. (2016). Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180.

Danafar, S., Rancoita, P., Glasmachers, T., Whittingstall, K., and Schmidhuber, J. (2013). Testing hypotheses by regularized maximum mean discrepancy. *arXiv preprint arXiv:1305.0423*.

Freund, Y., Schapire, R., and Abe, N. (1999). A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612.

Goodfellow, I. (2016). Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Gretton, A. (2013). Introduction to rkhs, and some simple kernel algorithms. *Adv. Top. Mach. Learn. Lecture Conducted from University College London.*

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012a). A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773.

Gretton, A., Sejdinovic, D., Strathmann, H., Balakrishnan, S., Pontil, M., Fukumizu, K., and Sriperumbudur, B. K. (2012b). Optimal kernel choice for large-scale two-sample tests. In *Advances in neural information processing systems*, pages 1205–1213.

Li, C., Alvarez-Melis, D., Xu, K., Jegelka, S., and Sra, S. (2017a). Distributional adversarial networks. *arXiv preprint arXiv:1706.09549.*

Li, C.-L., Chang, W.-C., Cheng, Y., Yang, Y., and Póczos, B. (2017b). Mmd gan: Towards deeper understanding of moment matching network. *arXiv preprint arXiv:1705.08584.*

Li, Y., Swersky, K., and Zemel, R. (2015). Generative moment matching networks. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1718–1727.

Lui, K. Y. C., Cao, Y., Gazeau, M., and Zhang, K. S. (2017). Implicit manifold learning on generative adversarial networks. *arXiv preprint arXiv:1710.11260.*

Mairal, J., Koniusz, P., Harchaoui, Z., and Schmid, C. (2014). Convolutional kernel networks. In *Advances in neural information processing systems*, pages 2627–2635.

Metz, L., Poole, B., Pfau, D., and Sohl-Dickstein, J. (2016). Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163.*

Petzka, H., Fischer, A., and Lukovnicov, D. (2017). On the regularization of wasserstein gans. *arXiv preprint arXiv:1709.08894.*

Premachandran, V. and Yuille, A. L. (2016). Unsupervised learning using generative adversarial training and clustering.

Ramdas, A., Reddi, S. J., Poczos, B., Singh, A., and Wasserman, L. (2014). On the high-dimensional power of linear-time kernel two-sample testing under mean-difference alternatives. *arXiv preprint arXiv:1411.6314*.

Saatchi, Y. and Wilson, A. G. (2017). Bayesian gan. *arXiv preprint arXiv:1705.09558*.

Steinwart, I. (2001). On the influence of the kernel on the consistency of support vector machines. *Journal of machine learning research*, 2(Nov):67–93.

Sutherland, D. J., Tung, H.-Y., Strathmann, H., De, S., Ramdas, A., Smola, A., and Gretton, A. (2016). Generative models and model criticism via optimized maximum mean discrepancy. *arXiv preprint arXiv:1611.04488*.

Tolstikhin, I., Gelly, S., Bousquet, O., Simon-Gabriel, C.-J., and Schölkopf, B. (2017). Adagan: Boosting generative models. *arXiv preprint arXiv:1701.02386*.

Villani, C. (2008). *Optimal transport: old and new*, volume 338. Springer Science & Business Media.

Zhao, J., Mathieu, M., and LeCun, Y. (2016). Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*.