Unsupervised Learning using Bayesian GAN

A Project Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of

Bachelor of Technology

by

Sriharsha Gaddipati
(111601005)

under the guidance of

Mrinal Kanti Das

INDIAN INSTITUTE
OF TECHNOLOGY
PALAKKAD

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

*This is to certify that the work contained in this thesis entitled "**Unsupervised Learning using Bayesian GAN**" is a bonafide work of **Sriharsha Gaddipati** (**Roll No. 111601005**), carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Palakkad under my supervision and that it has not been submitted elsewhere for a degree.*

Mrinal Kanti Das

Assistant/Associate Professor

Department of Computer Science & Engineering

Indian Institute of Technology Palakkad

Acknowledgements

I would like to thank **Prof. Mrinal Kanti Das** for mentoring and guiding me in doing this project.

Abstract

With the invent of GAN by IAN GOODFELLOW in 2014, it has opened up good research in Deep learning. Today many researchers in the world are researching in GAN and coming up with applications in generating new data by learning the distributions of training data set. Some of the GAN applications are creating new animated characters, pose guided person image generation, creating super resolution images from lower resolution, Image inpainting etc.

Although GAN is very powerful in generating new data, it has got many limitations. One of its limitation is the mode collapse in which it tries to learn multiple modes in the data and ends up learning one mode i.e. GAN collapse to one mode. In this thesis, We are trying to solve this problem and come up with the state of the art model which can learn multiple modes in the data, enable to cluster the data and generate data in those respective clusters using Bayesian GAN.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Definition

Aim of this project is to design and implement new GAN model to do unsupervised learning. Solve the problem of model collapse and learn different types of cluster distributions in data and generate data in those clusters.

1.2 Adversial Machine Learning

Adversarial machine learning is a technique employed in the field of machine learning which attempts to fool models through malicious input. This technique can be applied for a variety of reasons, the most common being to attack or cause a malfunction in standard machine learning models.

1.3 Generative Modeling

Generative modeling is the use of artificial intelligence (AI), statistics and probability in applications to produce a representation or abstraction of observed phenomena or target variables that can be calculated from observations.

Chapter 2

Literature Review

2.1 Gaussian Mixture Modeling(GMM)

A Gaussian mixture model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. One can think of mixture models as generalizing k-means clustering to incorporate information about the covariance structure of the data as well as the centers of the latent Gaussians.

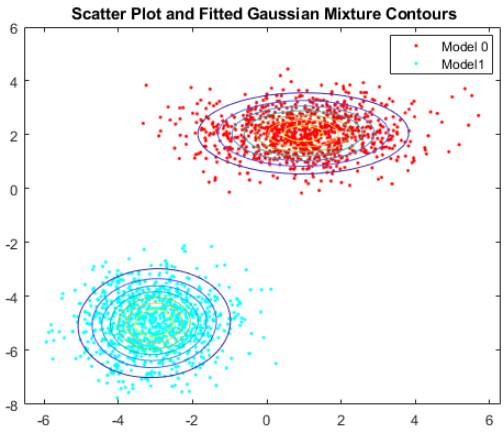

Fig. 2.1 Gaussian Mixture Modeling

2.1.1 Expectation Maximization Algorithm

Given a set of Data X={ $x_1, x_2, ..., x_n$} drawn from an unknown distribution (probable a GMM), estimate the parameters Θ of the GMM that fits the data.

Solution: Maximize the likelihood P(X/Θ) of the data with regard to the model parameters. One of approaches to maximize the likelihood is Expectation Maximization Algorithm.

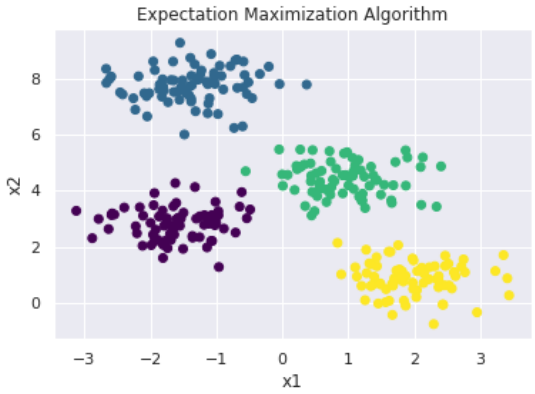

Fig. 2.2 Expectation Maximization Algorithm

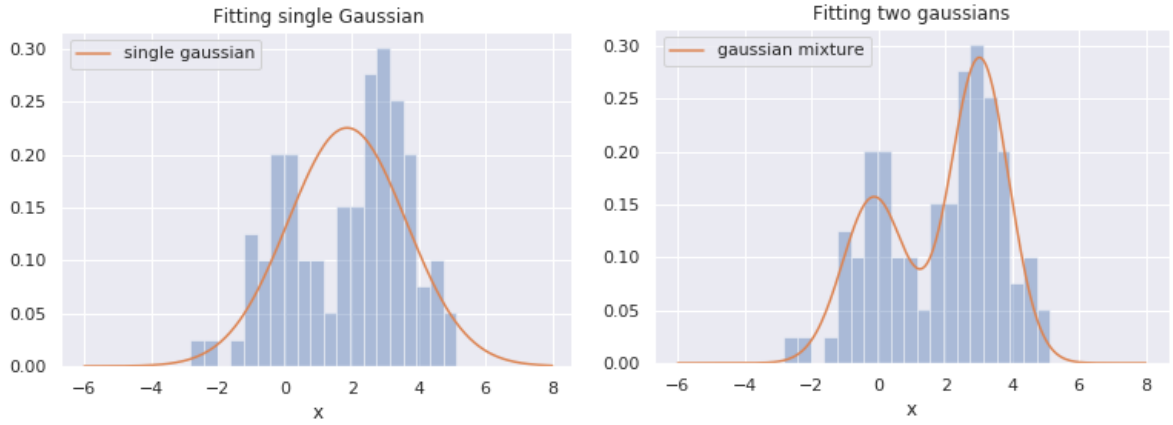

Fig. 2.3 Gaussian Mixture Model

2.2 Justification for using Bayesian GAN

The problem with Gaussian Mixture Model is we need to identify the number of clusters prior and then we try to fit k gaussians to the data. Now we want to automate this process using Bayesian GAN by identifying the number of clusters in the data and try to learn those many modes in the data.

2.3 Generative Adversial Network(GAN)

A generative adversarial network (GAN) is a class of machine learning systems. Two neural networks contest with each other in a game (in the sense of game theory, often but not always in the form of a zero-sum game). Given a training set, this technique learns to generate new data with the same statistics as the training set. For example, a GAN trained on photographs can generate new photographs that look at least superficially authentic to human observers, having many realistic characteristics.

In GAN, the generative model is pitted against an adversary: a discriminative model that learns to determine whether a sample is from the model distribution or the data distribution. The generative model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency. Competition in this game drives both teams to improve their methods until the counterfeits are indistiguishable from the genuine articles.

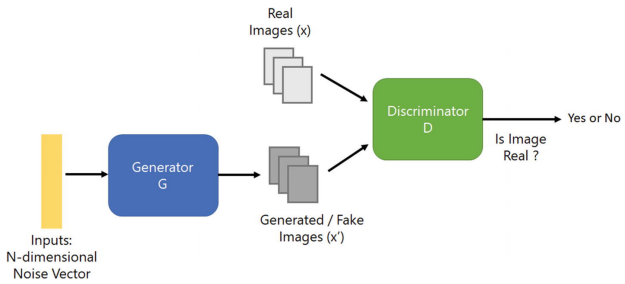

Fig. 2.4 Generative Adversial Network

2.3.1 Mini-max game

To learn the generators distribution p_g over data x, we define a prior on input noise variables $p_z(z)$, then represent a mapping to data space as G(z; θ_g), where G is a differentiable function represented by a multilayer perceptron with parameters θ_g. We also define a second multilayer perceptron D(x; θ_d) that outputs a single scalar. D(x) represents the probability that x came from the data rather than pg. We train D to maximize the probability of assigning the correct label to both training examples and samples from G. We simultaneously train G to minimize $log(1 - D(G(z)))$. In other words, D and G play the following two-player minimax game with value function V(G, D):

$$\min_G \max_D V(G, D) = E_{x \sim p_{data}(x)} \log(D(x)) + E_{z \sim p(z)} \log(1 - D(z)) \tag{2.1}$$

2.3.2 Mode Collapse

Mode collapse, also known as the Helvetica scenario, is a problem that occurs when the generator learns to map several different input z values to the same output point. In practice, complete mode collapse is rare, but partial mode collapse is common. Partial mode collapse refers to scenarios in which the generator makes multiple images that contain the same color or texture themes, or multiple images containing different views of the same dog.

Fig. 2.5 Mode Collapse

Chapter 3

Bayesian GAN

3.1 Handling Mode collapse of GAN

The problem of mode collapse can be dealt using Bayesian GAN. By exploring an expressive posterior over the parameters of the generator, the Bayesian GAN avoids mode-collapse, produces interpretable and diverse candidate samples, and provides state-of-the-art quantitative results for semi-supervised learning

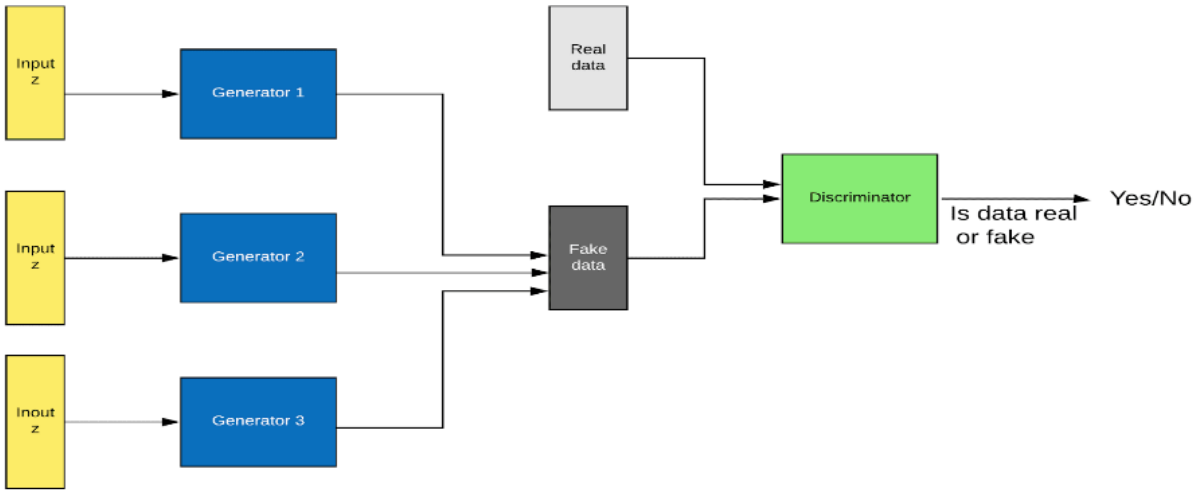

Fig. 3.1 Bayesian GAN

To infer posteriors over θ_g, θ_d, we can iteratively sample from the following conditional posteriors:

$$p(\theta_g|z, \theta_d) \propto (\Pi_{i=1}^{n_d} D(G(z^{(i)}; \theta_g); \theta_d)) p(\theta_g|\alpha_g) \tag{3.1}$$

$$p(\theta_d|z, X, \theta_g) \propto \Pi_{i=1}^{n_d} D(x^{(i)}; \theta_d) \times \Pi_{i=1}^{n_g} (1 - D(G(z^{(i)}; \theta_g); \theta_d)) \times p(\theta_d|\alpha_d) \tag{3.2}$$

$p(\theta_g|\alpha_g)$ and $p(\theta_d|\alpha_d)$ are priors over the parameters of the generator and discriminator with hyperparameters α_g and α_d

Suppose we were to sample weights θ_g from the prior $p(\theta_g|\alpha_g)$, and then condition on this sample of the weights to form a particular generative neural network. We then sample white noise z from p(z), and transform this noise through the network $G(z; \alpha_g)$ to generate candidate data samples. The discriminator, conditioned on its weights θ_d, outputs a probability that these candidate samples came from the data distribution. Eq. 3.1 says that if the discriminator outputs high probabilities, then the posterior $p(\theta_g|z, \theta_d)$ will increase in a neighbourhood of the sampled setting of θ_g (and hence decrease for other settings). For the posterior over the discriminator weights θ_d, the first two terms of Eq. 3.2 form a discriminative classification likelihood, labelling samples from the actual data versus the generator as belonging to separate classes. And the last term is the prior on θ_d.

3.2 Conclusion

In this chapter, we discussed the approach of Bayesian GAN to solve the problem of Mode collapse.

Chapter 4

Experiments

We carried out experiments on the bayesian gan for different types of datasets and different number of clusters.

All the generators and discriminators are 2 layered fully connected neural networks with relu as activation function.

For each cluster type we experimented with vanilla(simple) gan and bayesian gan with different number of generators.

After fine tuning the parameters we fixed some variables which are giving best results.
Batch size = 64
Learning rate = 1e-2
Learning decay rate = 3.
Alpha = 0.01
Z dimension = 2

4.1 Datasets

We generate D-dimensional synthetic data as follows:

$$z \sim N(0, 10 * I_d) \tag{4.1}$$

$$A \sim N(0, I_{D*d}) \tag{4.2}$$
$$x = Az + \epsilon \tag{4.3}$$
$$\epsilon \sim N(0, 0.01 * I_D) \tag{4.4}$$

4.2 2 Clusters

Parameters:

clusters = 2

x_dimension = 2

num_generators = 1

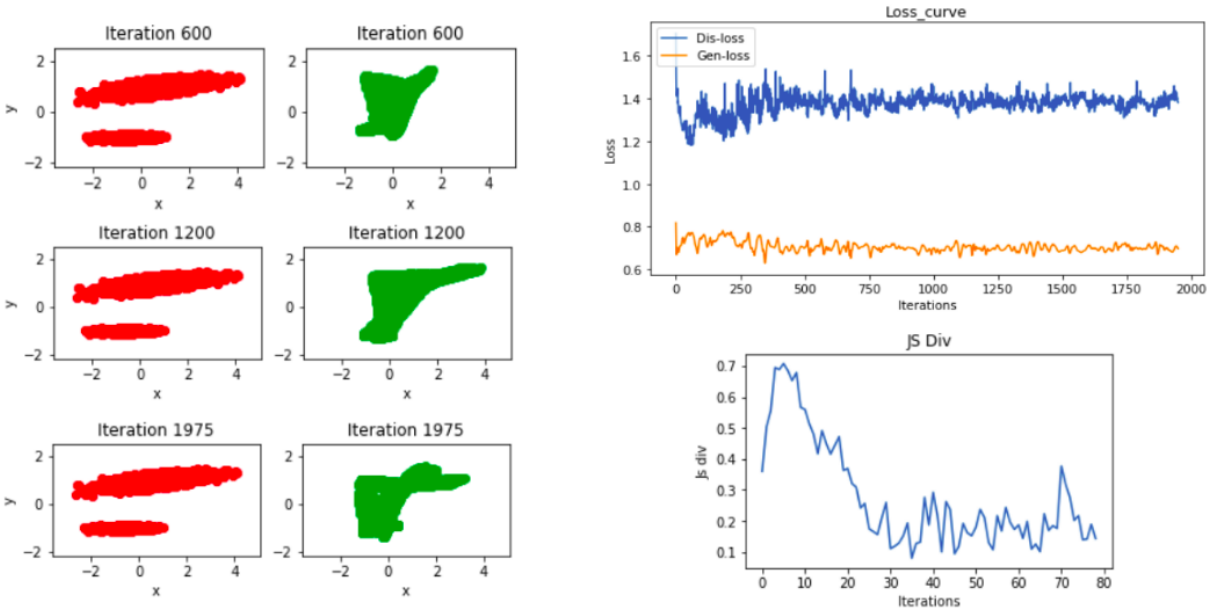

Fig. 4.1 2 Clusters, Vanilla(Simple) GAN with 1 generator

Vanilla(Simple) gan does not have the ability to learn the different distributions exactly. Discriminator losss function, generator loss function and JS divergence are continously fluctating with out converging. It is learning the region enclosed by the 2 clusters.

Parameters:

clusters = 2

x_dimension = 2

num_generators = 2

mcmc_samples = 50

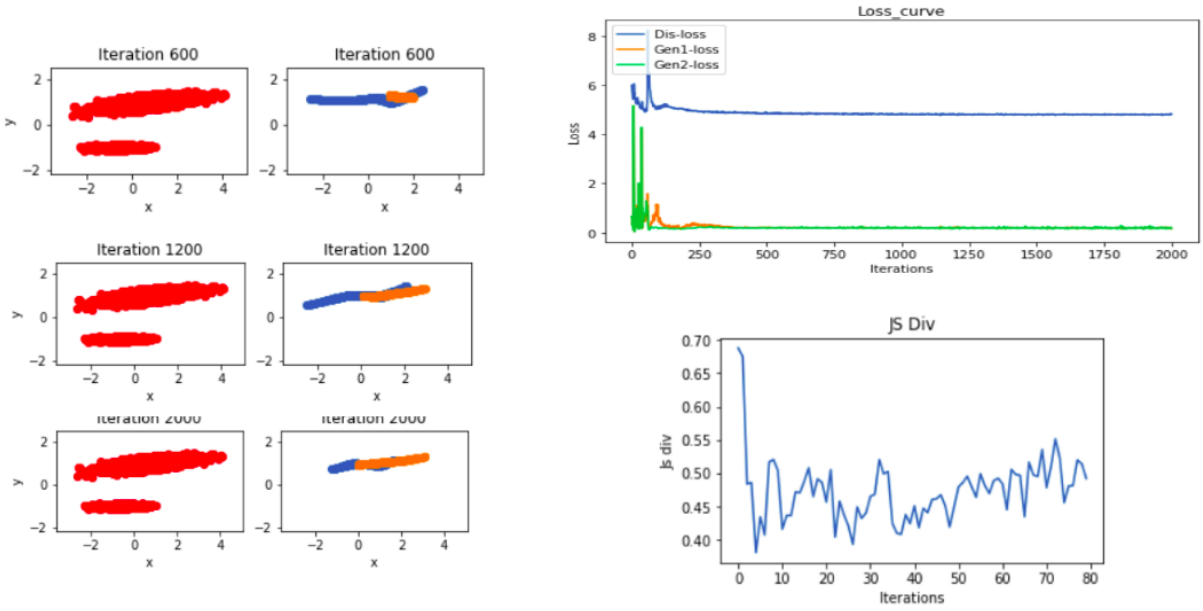

Fig. 4.2 2 Clusters, Bayesian GAN 2 Generators

With the help of 2 generators, bayesian gan model has only learnt the single cluster and it could not learn the second generator.

Parameters:

clusters = 2

x_dimension = 2

num_generators = 3

mcmc_samples = 100

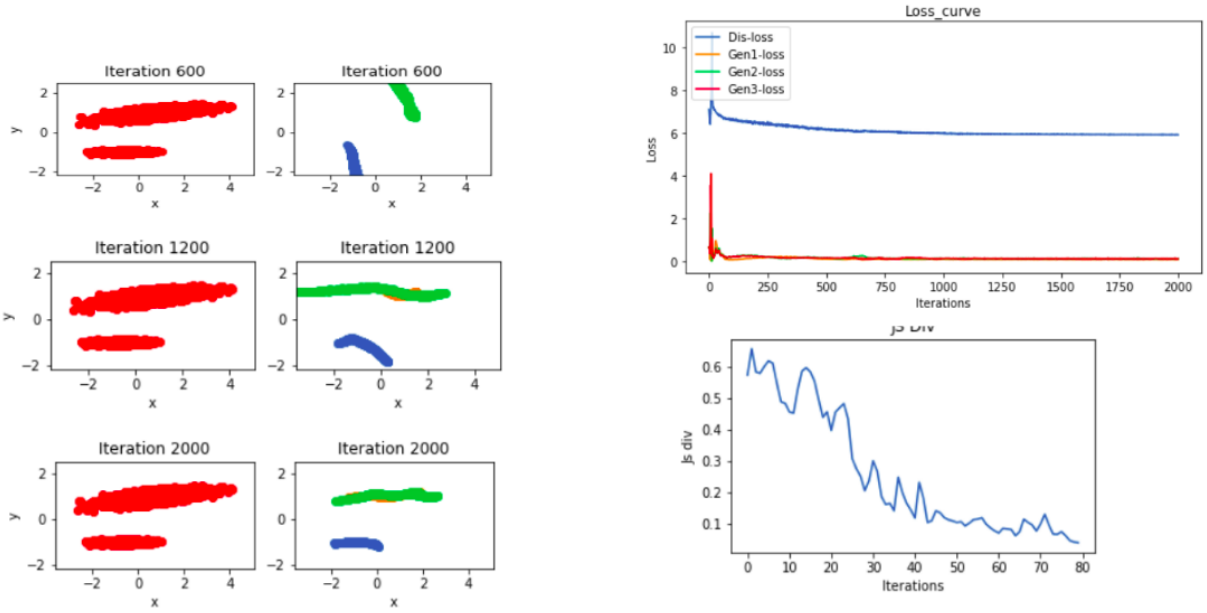

Fig. 4.3 2 Clusters, Bayesian GAN 3 Generators

Now we added the extra generator to learn the other cluster. Finally with 3 generators and mcmc samples = we got the best result. And also the jenson shanon divergernce(Js div) is continously minimizing with out any fluctuations and converging to its miniumum.

Parameters:

clusters = 2

x_dimension = 2

num_generators = 4

mcmc_samples = 50

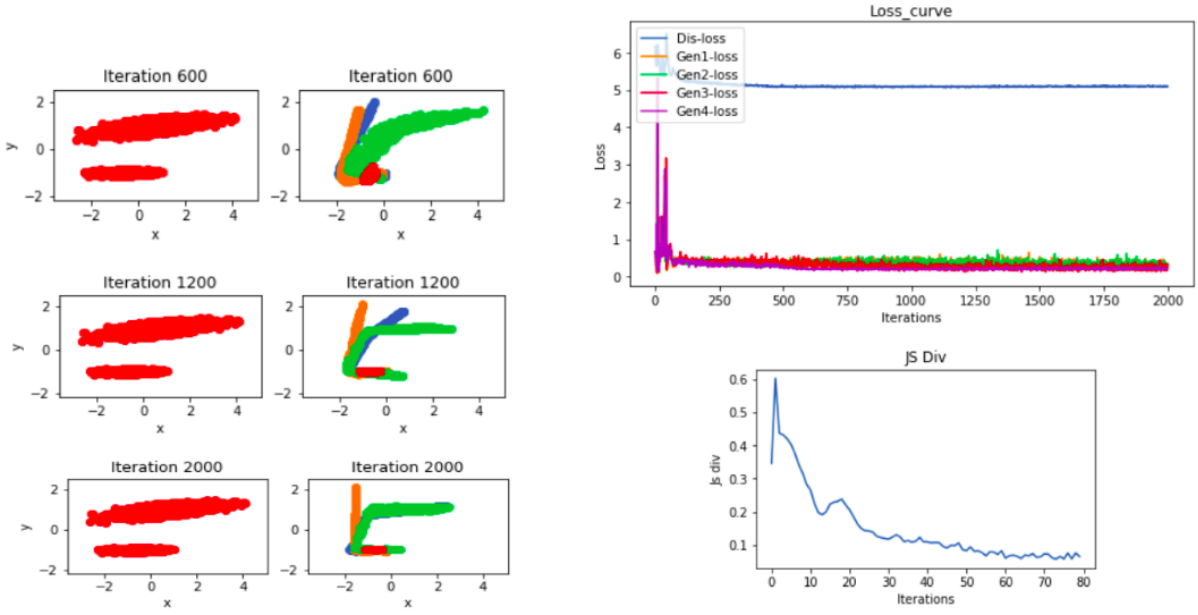

Fig. 4.4 2 Clusters, Bayesian GAN 4 Generators

If the number to generators further increased to 4 and 5, the extra generators are creating noise by joining the nearest clusters. You can see that the clusters are joined in the above figure.

Parameters:

clusters = 2

x_dimension = 2

num_generators = 5

mcmc_samples = 50

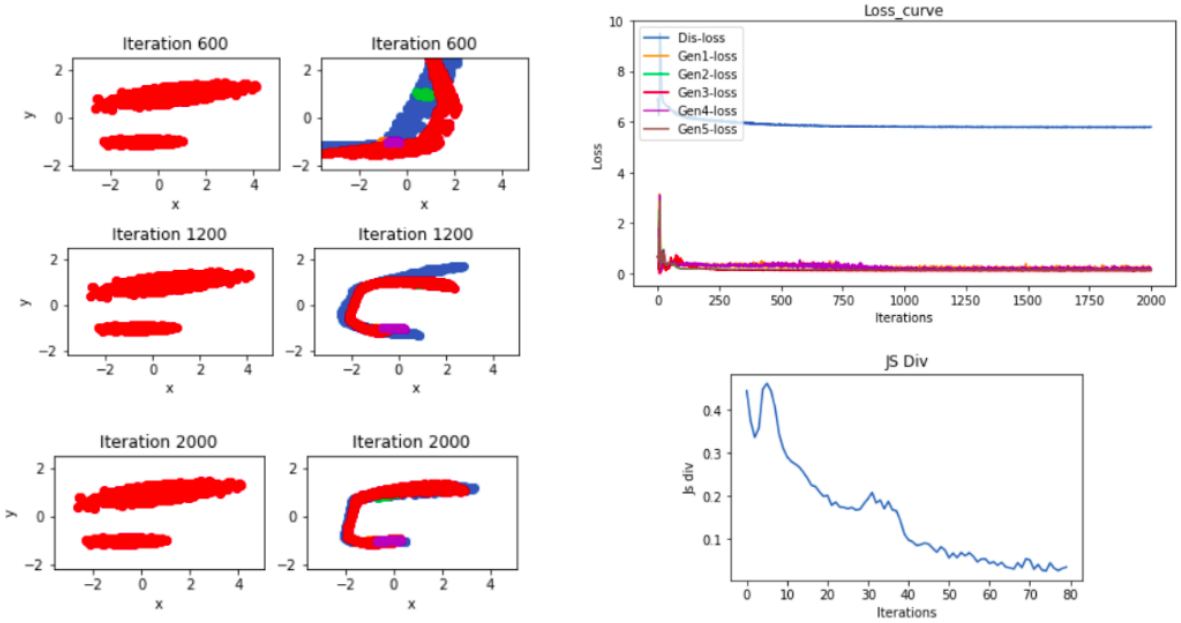

Fig. 4.5 2 Clusters, Bayesian GAN 5 Generators

4.3 3 Clusters

Parameters:

clusters = 3

x_dimension = 2

num_generators = 1

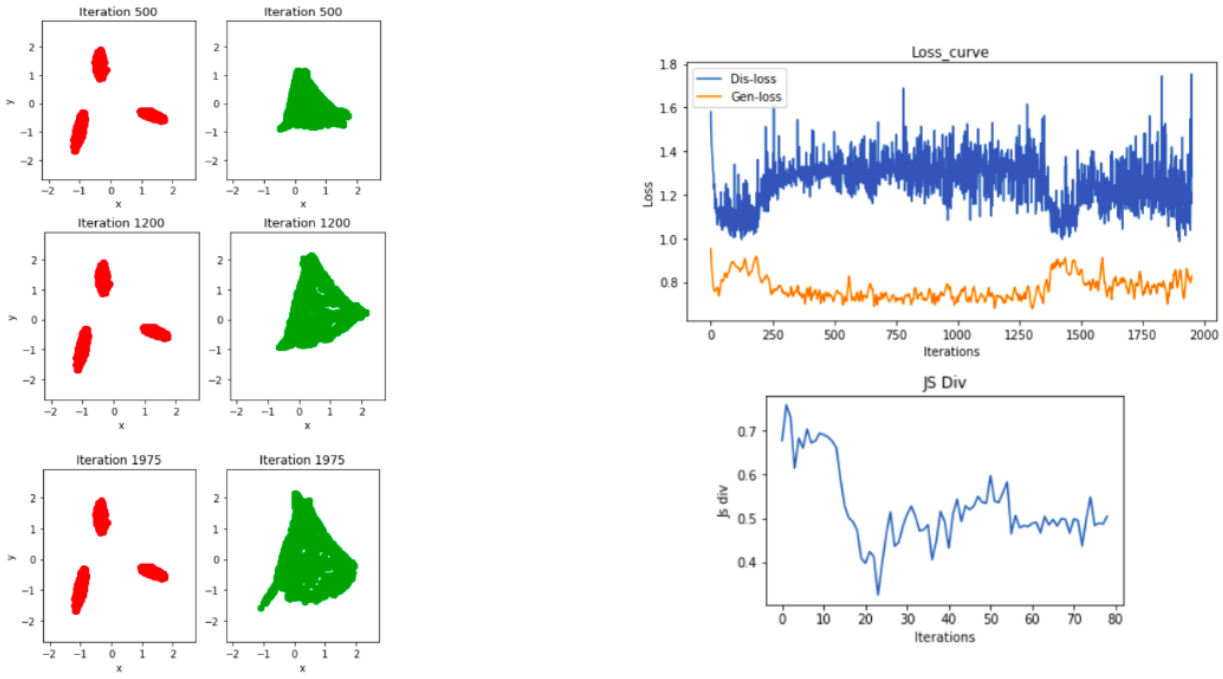

Fig. 4.6 3 Clusters, Vanilla(Simple) GAN with 1 generator

Parameters:

clusters = 3

x_dimension = 2

num_generators = 3

mcmc_samples = 30

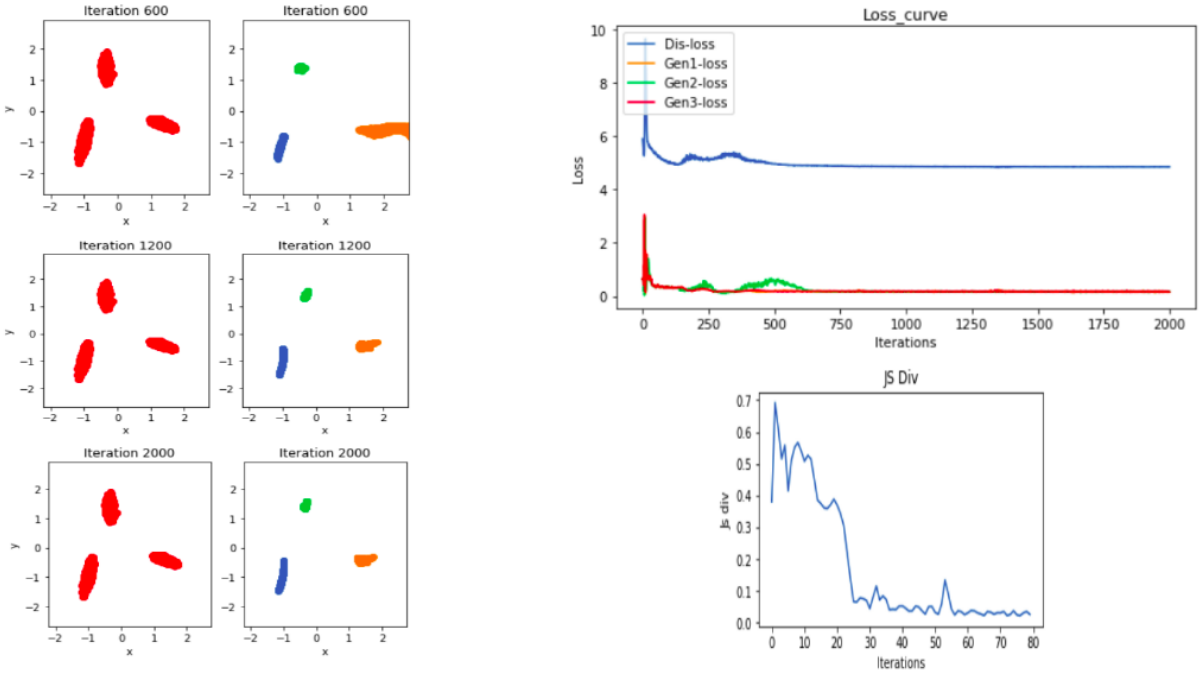

Fig. 4.7 3 Clusters, Bayesian GAN 3 Generators

Each generator is learning one distribution and the model is able to learn 3 clusters.

Parameters:

clusters = 3

x_dimension = 2

num_generators = 4

mcmc_samples = 50

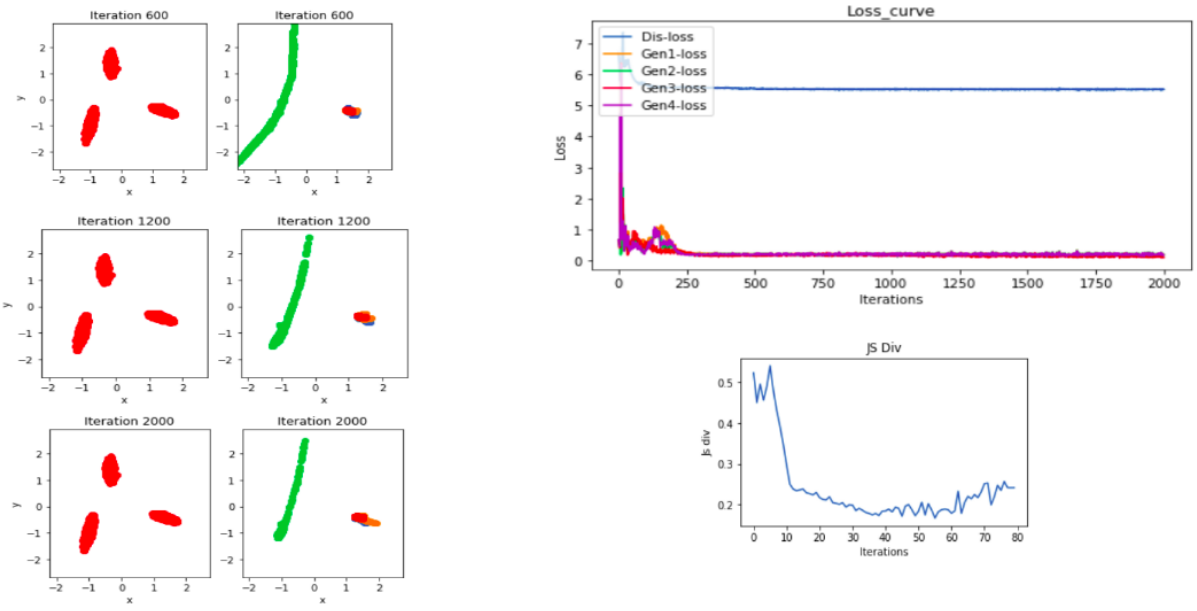

Fig. 4.8 3 Clusters, Bayesian GAN 4 Generators

With 4 generators, 2 clusters are learnt by 1 generator by joining them and remaining 3 generators learnt only one cluster. Increasing the number of generators will definitely lead to joining the clusters.

4.4 4 Clusters

Parameters:

clusters = 4

x_dimension = 2

num_generators = 1

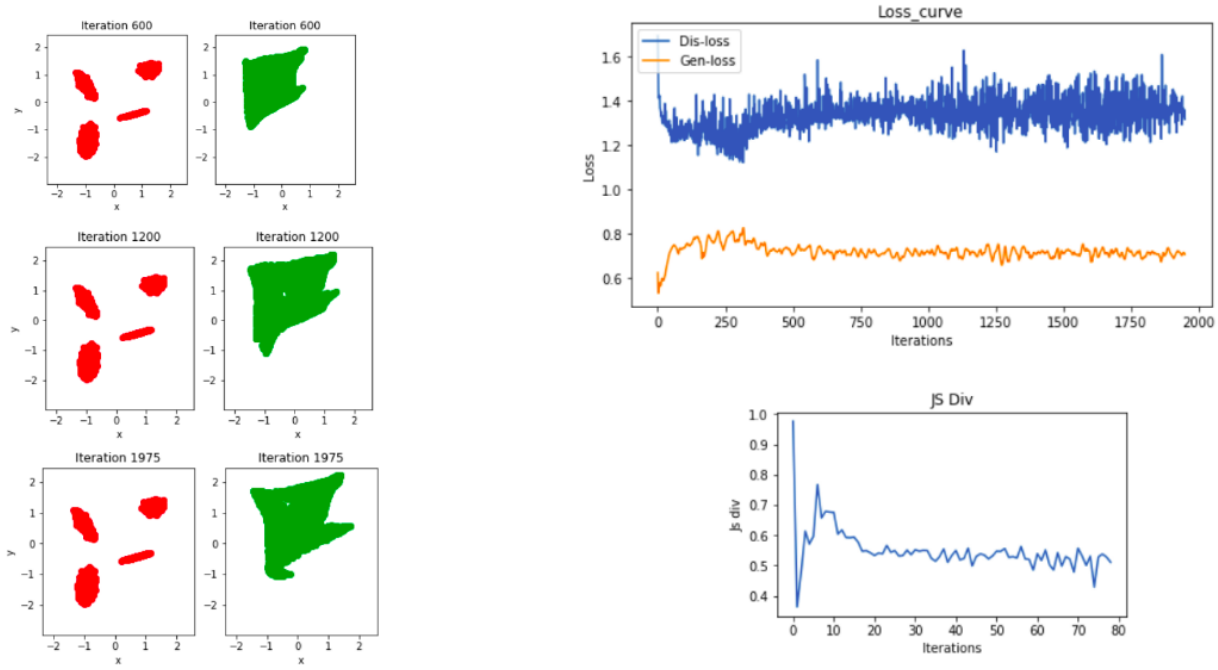

Fig. 4.9 4 Clusters, Vanilla(Simple) GAN with 1 generator

Parameters:

clusters = 4

x_dimension = 2

num_generators = 4

mcmc_samples = 50

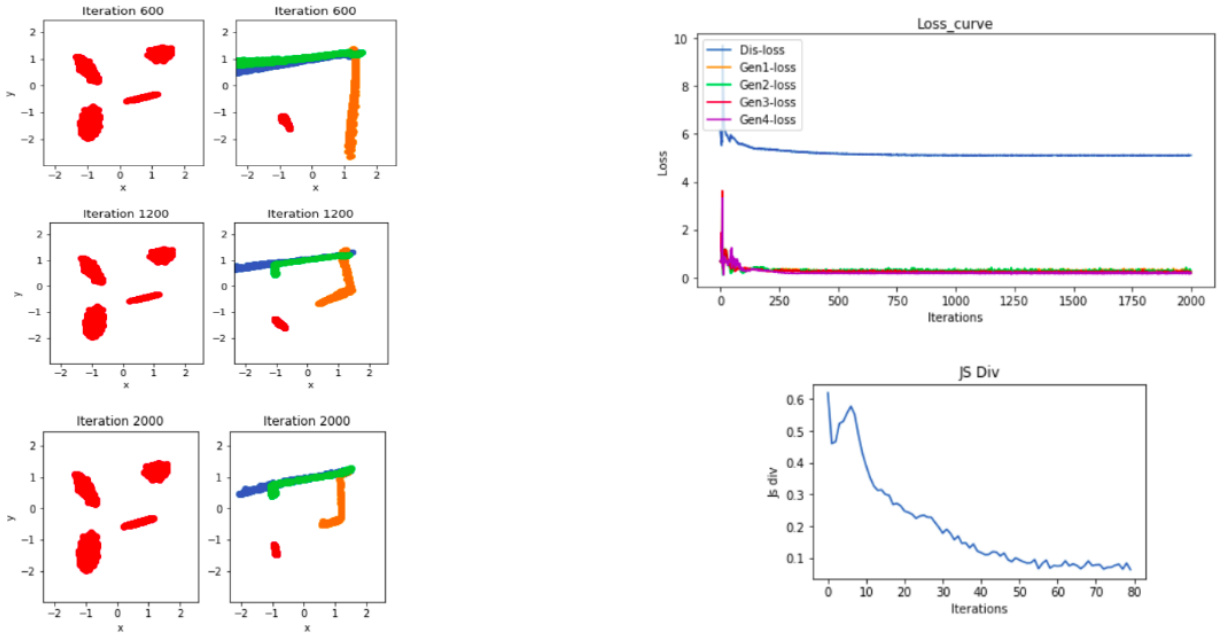

Fig. 4.10 4 Clusters, Bayesian GAN 4 Generators

In this model, all the clusters were learnt nearly by the 4 generators but were not seperated properly.

Parameters:

clusters = 4

x_dimension = 2

num_generators = 5

mcmc_samples = 50

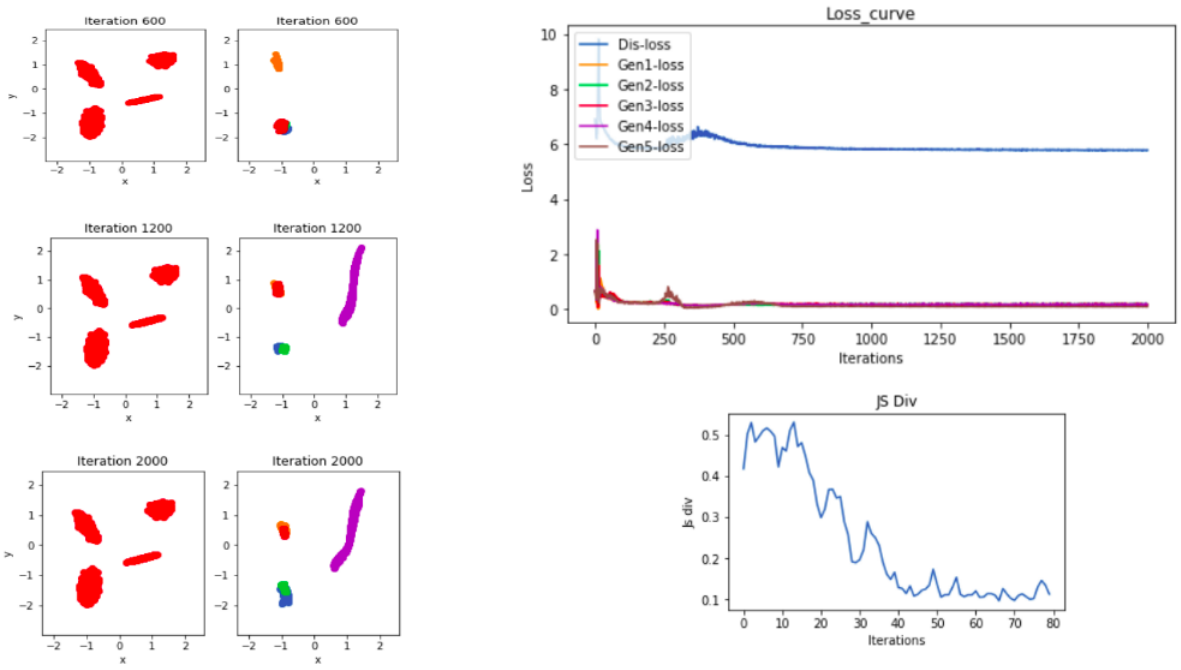

Fig. 4.11 4 Clusters, Bayesian GAN 5 Generators

2 generators learnt one cluster and 2 other generators learnt another cluster. And the remaining generator joined the 2 clusters.

4.5 5 Clusters

For 5 clusters, we experimented with very high dimensional data(100) and then we transformed it to 2 dimensions using principal component analysis for visualizing.

Parameters:

clusters = 5

x_dimension = 100

num_generators = 1

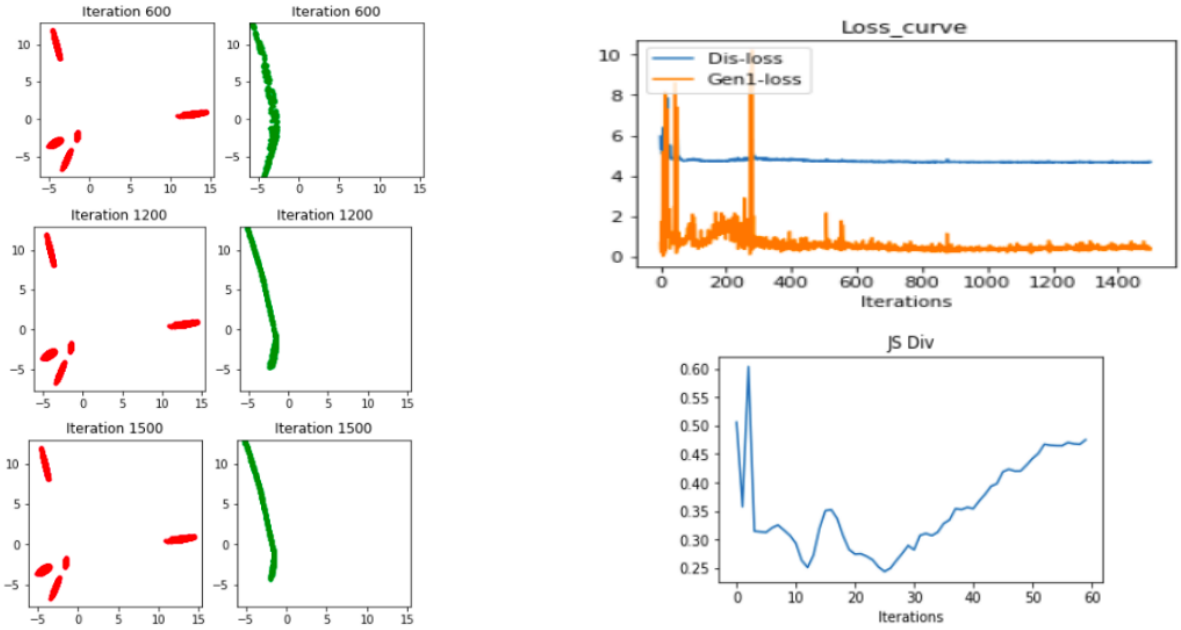

Fig. 4.12 5 Clusters, Vanilla(Simple) GAN with 1 generator

Parameters:

clusters = 5

x_dimension = 100

num_generators = 5

mcmc samples = 50

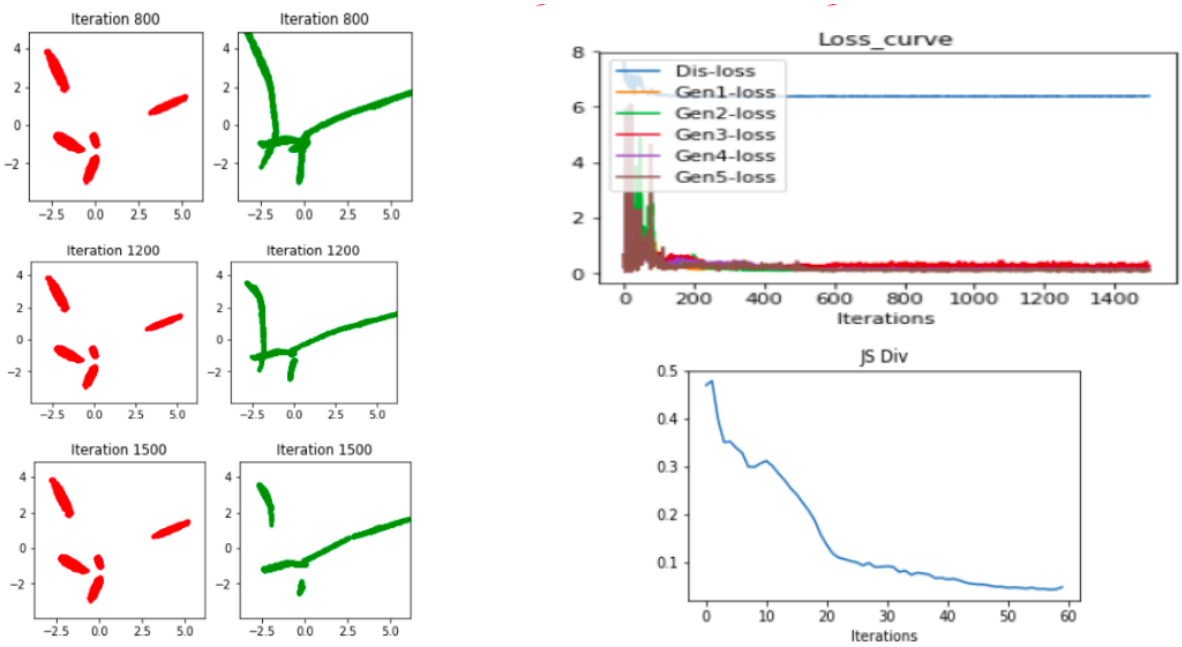

Fig. 4.13 5 Clusters, Bayesian GAN 5 Generators

4.6 Conclusion

We experimented the bayesian gan in all possible ways and the best results we got for each cluster type are listed below.

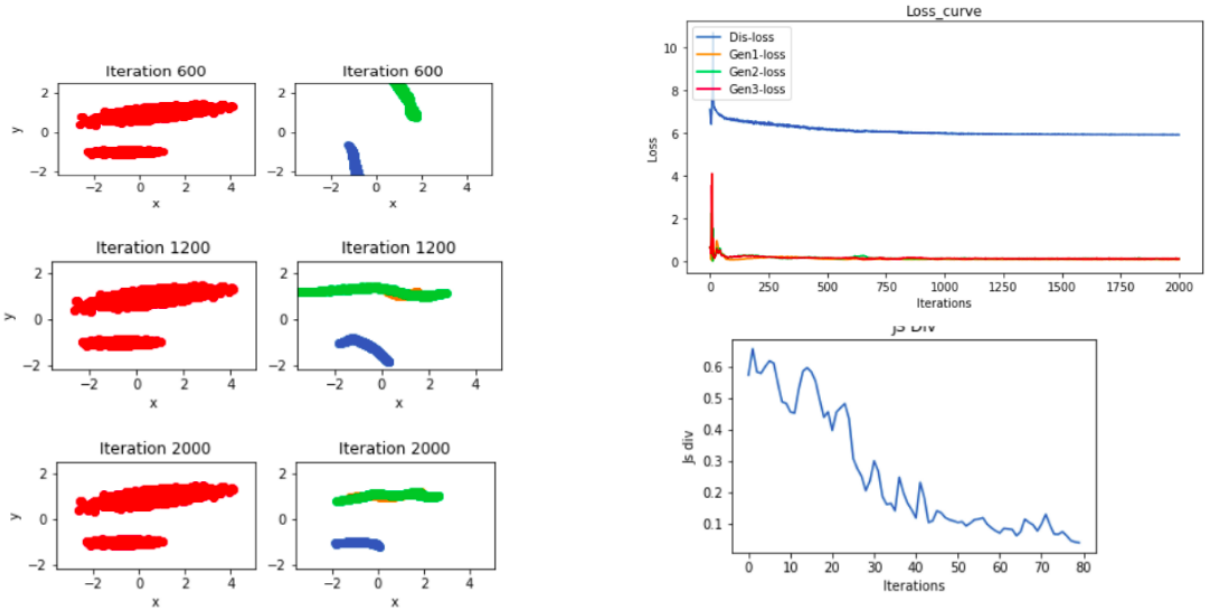

Fig. 4.14 2 Clusters, Bayesian GAN 3 Generators

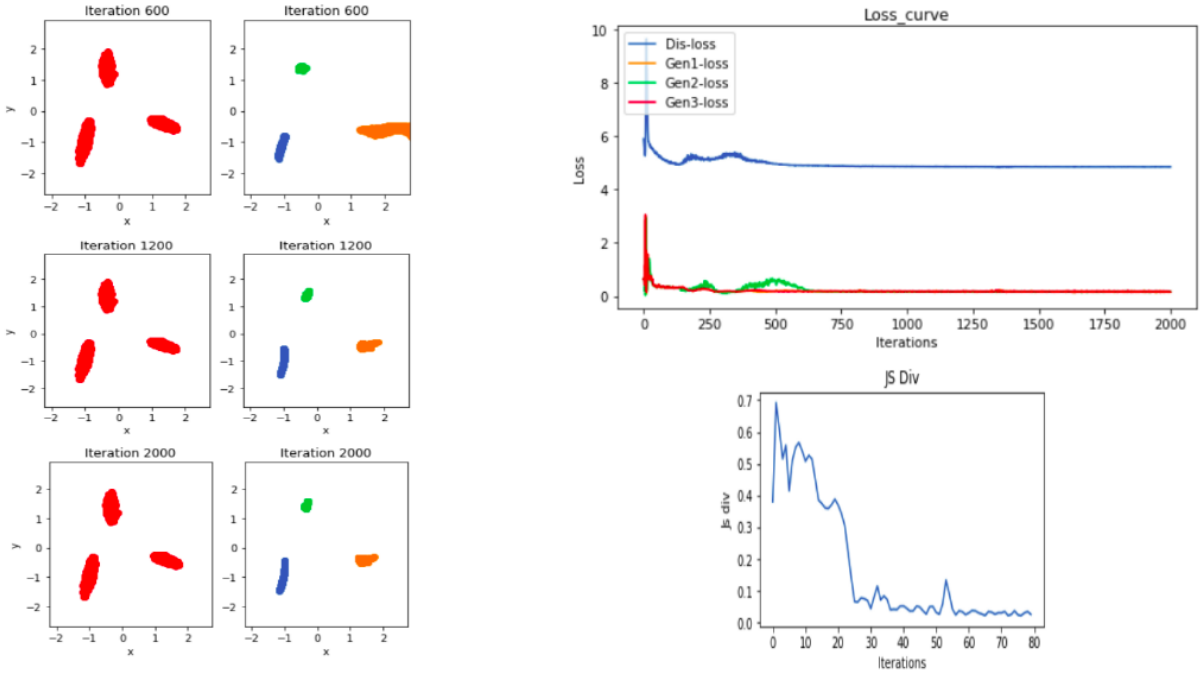

Fig. 4.15 3 Clusters, Bayesian GAN 3 Generators

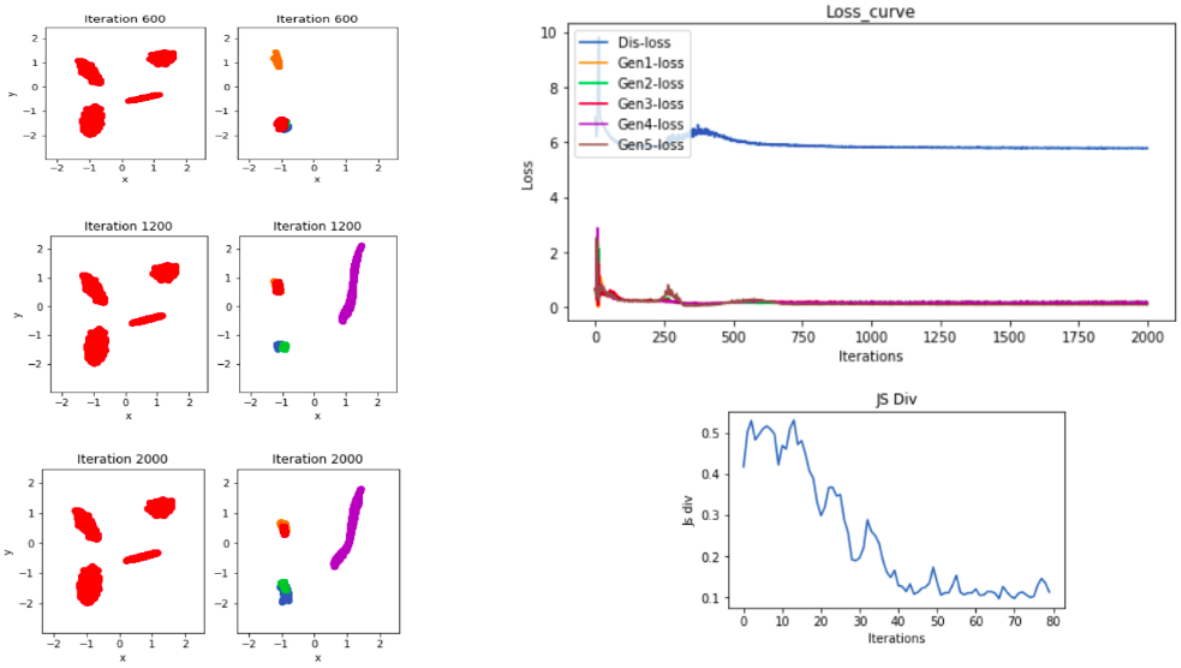

Fig. 4.16 4 Clusters, Bayesian GAN 5 Generators

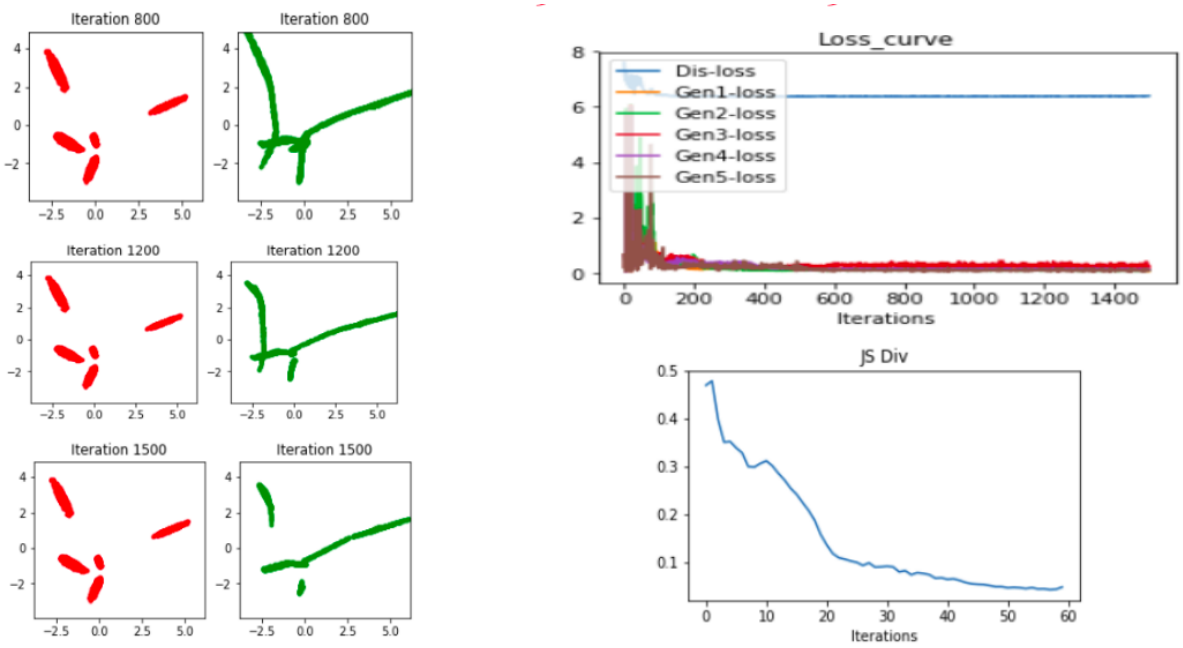

Fig. 4.17 5 Clusters, Bayesian GAN 5 Generators

Chapter 5

Project progress

5.1 Work done before mid sem

- Learned Tensorflow
- Studied few papers on GAN
- Implemented GAN model in Tensorflow and Keras for MNIST data
- Implemented GMM and fitted two gaussians to data

5.2 Work done before mid sem

- Studied Bayesian Gan paper
- Implemented Bayesian gan
- Experimented with different datasets and clusters on bayesian gan

5.3 Topics learnt

- Monte carlo Integration
- Jenson shannon divergence, kl divergence, pca
- Adam optimizer
- Kernel density estimator
- Gaussian Mixture modeling
- SGHMC
- Minibatch Gradient Descent
- Exponential Weighted Moving Averages
- Gradient Descent with Momemtum
- RMS Prop
- Adam Optimizer
- Learning rate decay
- Batch Normalization

5.4 Challenges faced

- Nearly 150 versions of bayesian code has been tried out
- Debugging the code
- Understanding bayesian concepts

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Latent space(z) has never been explored in bayesian gan. Have to experiment out on how we can use the latent space in getting good models. Currently we have tried only on multiple generators, we can also try out having multiple discriminators. In the above experiments we found that most of the times generators are trying to join the clusters and learning them. This problem needs to be addressed. Have to read few other new research that has been done in bayesian gan and explore the capabilties of gan with few other ideas to design a new model to do unsupervised learning.

References

[1] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio *Generative Adversarial Nets.*

[2] Ian Goodfellow, OpenAI *NIPS 2016 Tutorial: Generative Adversarial Networks.*

[3] Yunus Saatchi, Andrew Gordon Wilson *Bayesian GAN*