Analysis of Maximum Mean Discrepancy Generative Adversarial Networks (MMD GAN)

Temirlan Kaiyrbekov

Supervisor: Rustem Takhanov

Second Reader: Adilet Otemissov

Department of Mathematics, Nazarbayev University

28.04.2024

Abstract

Deep neural networks can be used to generate new data by sampling from the data distribution without explicitly defining the distribution. These nets heavy rely on optimization for efficient learning, and hence, they need mathematical guarantees for feasibility of learning. Generative Adversarial Networks (GAN) were proposed to generate images by the use of a mini-max objective function that is "played" among two agents - a generator and a discriminator network. Later, Generative Moment Matching Networks (GMMN) were proposed to use a two-sample test instead of a discriminator network. GMMN uses Maximum Mean Discrepancy metric for distinguishing between real and generated images, but it only trains the generator network, and was implemented inefficiently. Lastly, Maximum Mean Discrepancy Generative Adversarial Networks (MMD GAN) were introduced that use adversarial kernel learning that has a mini-max objective function, efficient learning and mathematical guarantees that justify its improved performance. In this work, the mathematical reasoning behind the idea of MMD GAN was analyzed and experiments were made to tweak the parameters of the network. The loss function of MMD GAN is said to enjoy a weak topology - that MMD should tend to zero as two probability distributions converge to each other - and it will be shown empirically. Also, since the network has a loss function that is locally Lipschitz and continuous everywhere, and almost everywhere differentiable, the network was able to learn efficiently. Finally, MMD GAN with changed bandwidth parameters will be introduced that showed improved convergence with less MMD loss during training, although the loss was less smooth over epochs.

Keywords: generator, discriminator, kernel, bandwidth, maximum mean discrepancy, loss function.

1 Introduction

1.1 The Learning Paradigm

The idea of statistical learning is based on error minimization. The learnability of common modern deep learning networks are defined through a "loss function". Learning is done through minimizing a loss, and the parameters or weights of a network are tuned by taking a step over the gradient. Formally, we want to find such $w_1, w_2, ..., w_n$ that minimize the loss function. The weight update rule is done through computing gradients. The gradient is a list of partial derivatives of the loss functions relative to each weight component:

$$\nabla f(w) = \left(\frac{\partial f(w)}{\partial w_1}, \ldots, \frac{\partial f(w)}{\partial w_n} \right)$$

[6] For a simple minimization task, a learner needs to do gradient descent over the loss landscape towards a minimum by minimizing loss.

Toy examples of optimization problems usually have continuous, differentiable, and convex loss functions. Yet, with real data and real problems, the loss landscapes of deep learning models are more complex - they are highly-dimensional and are not convex, they can have saddle points, or may contain flatter minima that can be more desirable due to robustness, or minima can have "asymmetric" directions due to high-dimensionality [3]. At the very least, there should be guarantees for continuity and differentiability for feasibility of learning when a new mode is proposed. This paper is aimed at revealing how MMD GAN's efficiency is justified mathematically, and at introducing experiments with discussions.

1.2 Generative Adversarial Networks

The generative adversarial nets (GAN) were proposed for generating images using a two-agent deep neural network by sampling from the data distribution without explicitly defining it. It consists of a generator and discriminator part which are designed to learn in a two-player mini-max regime [2]. The generator takes noise input data and tries to generate images, and the discriminator tries to distinguish if an image is from the dataset or is it generated by the network. The generator is learnt to better fool the discriminator by generating images that are similar to those from a given dataset, while the discriminator learns to better predict if an image is generated or not.

The objective function for the network was defined to be

$$\min_G \max_D V(D,G) = E_{x \sim p_{\text{data}}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

, where

- p_g represents the probability distribution of the generator.
- D denotes the discriminator network with parameters θ_d.
- $D(x)$ represents the probability that x is from the dataset rather than p_g.
- G denotes the generator function with parameters θ_g.

- $G(z)$ represents the mapping from noise space to data space.
- $p_z(z)$ represents the generator's prior noise distribution.

[2]

The learning is done through gradient ascent for the discriminator network so that it gets better at differentiating between real and fake images, and gradient descent for the generator to produce images that result in as less loss as possible in a discriminator network.

1.3 Generative Moment Matching Networks (GMMN) and the Maximum Mean Discrepancy (MMD) metric

The main idea of GMMN is to replace the discriminator network in the original GAN architecture with a two-sample test metric. Intuition of using the Maximum Mean Discrepancy metric lies in the comparison of the moments of probability distributions of real data and generated data: if the corresponding moments of given two datasets are similar, then they are likely to be from the same distribution [5].

The MMD metric can be defined as follows:

for given two samples $X = \{x_1, x_2, \ldots, x_N\}$ and $Y = \{y_1, y_2, \ldots, y_M\}$ the mean squared differences of statistics using MMD is

$$L_{\mathrm{MMD}^2} = \left\| \frac{1}{N} \sum_{i=1}^{N} \phi(x_i) - \frac{1}{M} \sum_{j=1}^{M} \phi(y_j) \right\|^2 \tag{1}$$

$$\begin{aligned} &= \frac{1}{N^2} \sum_{i=1}^{N} \sum_{i'=1}^{N} \phi(x_i)^T \phi(x_{i'}) \\ &\quad - \frac{2}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} \phi(x_i)^T \phi(y_j) \\ &\quad + \frac{1}{M^2} \sum_{j=1}^{M} \sum_{j'=1}^{M} \phi(y_j)^T \phi(y_{j'}) \end{aligned} \tag{2}$$

One can choose ϕ for comparing a specific moment (e.g., choosing ϕ as an identity function would compare the first moments - the means of distributions).

The form (2) includes only inner products of ϕ which means that a kernel trick, that will be introduced in the following subsection, can be applied.

$$\begin{aligned} L_{\mathrm{MMD}^2} &= \frac{1}{N^2} \sum_{i=1}^{N} \sum_{i'=1}^{N} k(x_i, x_{i'}) \\ &\quad - \frac{2}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} k(x_i, y_j) \\ &\quad + \frac{1}{M^2} \sum_{j=1}^{M} \sum_{j'=1}^{M} k(y_j, y_{j'}) \end{aligned} \tag{3}$$

Regarding the objective function, GMMN network relies on a fixed "discriminator" that is the MMD metric with a fixed kernel, so its objective function is now relaxed to be

$$\min_{\theta} M_k(P_X, P_\theta)$$

, where k is a Gaussian kernel, X is the real data sample, θ are parameters of the generator network, and P_θ is the distribution of generator's output data [4]. Later it will be discussed why this simpler objective function is one of the disadvantages of the GMMN network.

1.4 Kernel trick and its application to the MMD metric

A kernel is a similarity measure between two entities, and they can be represented as an inner product in some Hilbert space. According to [6], "A Hilbert space is a vector space with an inner product, which is also complete. A space is complete if all Cauchy sequences in the space converge." A "kernel trick" is a method that allows to learn efficiently without having to handle higher-dimensional representation of vectors.

Kernel tricks are often used in machine learning. A trivial example is a linear separation task, where data can be initially non-separable, but applying a kernel trick and raising the data space into a higher dimension would make it separable.

For example, the Gaussian kernel is defined as

$$K(x, x') = e^{-\frac{\|x-x'\|^2}{2}}$$

The general case would be

$$K(x, x') = e^{-\frac{\|x-x'\|^2}{2\sigma}}$$

, where σ is a bandwidth parameter.

To interpret, the similarity between two vectors that are far away would be 0 with this kernel function, and it equals 1 for vectors that are close to each other.

When this kernel function is applied to the MMD metric with a Taylor expansion, one can get a feature map representation ϕ that contains infinite terms that correspond to all the higher (infinite) moments of distributions [5].

Having this kind of feature expansion allows us to minimize the MMD loss by minimizing differences between all the moments of distributions.

Previous research on this topic has shown that when the data space is raised to a reproducing kernel Hilbert space (RKHS) (if k is a characteristic kernel), the MMD metric is equal to 0 if and only if distribution of data are the same: $P_1 = P_2$ which is a desired characteristic for learning how to sample from a distribution [5].

1.5 Maximum Mean Discrepancy GAN (MMD GAN)

This paper introduced adversarial kernel learning which results in a mini-max objective function similar to that of the original GAN network. In other words, it allows to have such an MMD metric that will be learnt to distinguish between different data better. While GMMN had a fixed kernel, MMD GAN has an adversarially learned kernel

$$\arg\max_{k \in K} M_k(P_X, P_\theta)$$

[4]

The new objective function is

$$\min_{\theta} \max_{k \in K} M_k(P_X, P_\theta)$$

, where k is a kernel from the set K of characteristic kernels, X is the real data sample, θ are parameters of the generator network, and P_θ is the distribution of generator's output data.

The authors also suggest that g (the mapping from noise input to data space) is locally Lipschitz, and that the objective function of maximizing the MMD metric is continuous, and differentiable almost everywhere. In such cases, the network can be learnt via gradient descent and backpropagation [4]. Thus, the mathematical criteria for feasibility of learning were met for the introduced adversarial kernel learning method.

Also, it was shown in the paper that MMD GAN benefits from a weak topology - here, convergence in distribution implies decreasing MMD distance, and the converse statement is true. This matches with the experiments done for this project.

2 Experimentation and Discussion

2.1 Experimental setup

The training of this network requires a machine with an up-to-date GPU and the software part. The setup was the following: Ubuntu 22.04, NVIDIA GeForce GTX 1650, Python 3.10.12, NVIDIA Driver Version 550.54.15, CUDA 12.1, CUDNN 8.9.2.26, and PyTorch 2.3.0+cu121.

2.2 Experiments

There are several results from tweaking the parameters of the network during this project. The bandwidth parameters of the Gaussian kernel were changed. The network is firstly trained on MNIST dataset (one channel handwritten digits dataset) with $\sigma \in$ [1, 2, 4, 8, 16] as in the MMD GAN paper, then it is trained separately on the same dataset with $\sigma \in$ [2, 4, 8, 16, 32]. The value 32 of the parameter was chosen since it is the next power of two.

The first result - presence of rare distorted images with larger bandwidth parameters

There is one major difference between generation of images by these two sets of bandwidths. In some batches, the network with changed bandwidths [2, 4, 8, 16, 32] produce noisy images that were not expected to be observed, and that are more noisy even compared to images at early stages of training (Figure 1). A possible reason for that is that this networks uses the Gaussian kernel variation with σ^2 in the denominator of the exponent, so 32 gets squared which may affect computation heavier than squaring lower powers of two.

Figure 1: A few batches of rare erroneous generated images from a modified network

The second result - the quality of images is roughly similar, with images produced by changing bandwidths being slightly better

Apart from these, there are no major noticeable differences in the quality of generated images, except net trained with [2, 4, 8, 16, 32] bandwidths might have more noise during the training and images produced by it might have a better quality; the full results will be uploaded on Github alongside with the code. For comparison, images from the dataset and generated images by the networks are introduced (Figure 2; Figure 3). Figure 2 is from the first epoch of training, and Figure 3 is from the last one. To be more conclusive on the difference of qualities of images, additional test can be done (e.g., Inception score - comparing classification scores by using a pre-trained convolutional network).

Figure 2: Left: samples from MNIST dataset; middle: generated by MMD GAN with bandwidth parameters $\sigma \in [1, 2, 4, 8, 16]$ at first epoch; right: generated by MMD GAN with bandwidth parameters $\sigma \in [2, 4, 8, 16, 32]$ at first epoch

Figure 3: Left: samples from MNIST dataset; middle: generated by MMD GAN with bandwidth parameters $\sigma \in [1, 2, 4, 8, 16]$ at final epoch; right: generated by MMD GAN with bandwidth parameters $\sigma \in [2, 4, 8, 16, 32]$ at final epoch

The third result - MMD GAN with larger bandwidths results in less smooth MMD measurements during training, but has lower MMD metric and seems to converge more rapidly

From the logs of trainings, dynamics of MMD distance were plotted to see if the graph matches with the weak topology proposed for both variations of the model. The moving averages were calculated to increase smoothness and decrease the influence of variance created by mini-batch stochastic gradient descent.

As can be seen from Figure 4, the improvement of quality corresponds to decreasing MMD distance. Also, the training consists of 1000 epochs, and after the first half of training, quality of images slowed in improvement which corresponds with the observation that the MMD metric does not decrease sufficiently.

Comparing the two plots, the original MMD GAN starts at higher loss values, and gradually converges, while the changed MMD GAN starts at lower loss values, maintains lower values during training, but is less smooth - it has a lot of sharp

fluctuations that might correspond to the unexpected distorted images from Figure 1. Such fluctuations can imply that using larger bandwidths may be used in a "high risk, high reward" fashion for less smooth, but more efficient learning.

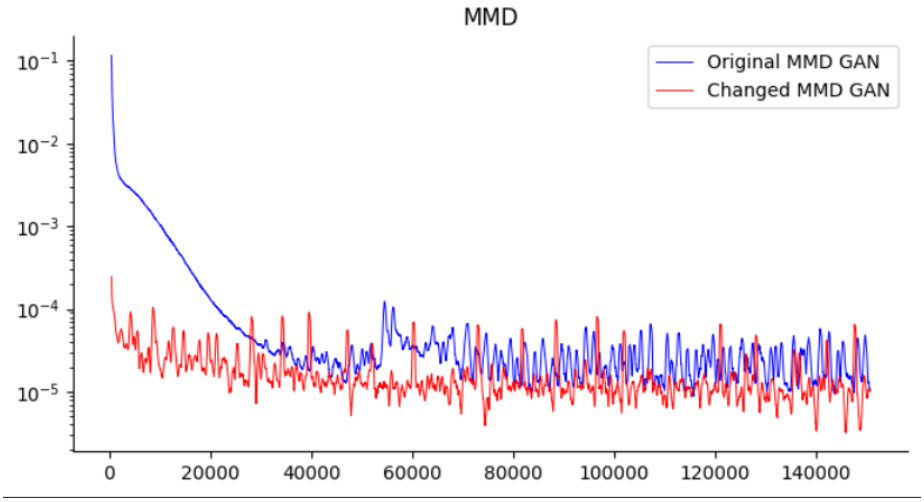

Figure 4: MMD distance of Original MMD GAN ($\sigma \in [1, 2, 4, 8, 16]$) and Changed MMD GAN ($\sigma \in [2, 4, 8, 16, 32]$) over generator iterations (max. 1000 epochs)

2.3 Future work

The GMMN and MMD GAN papers use Gaussian kernel. These networks can be implemented with other characteristic kernels too. According to [1], Laplacian kernel is a characteristic one. It is similar to the Gaussian and can be defined as

$$K(x, x') = e^{-\frac{\|x-x'\|}{\sigma}}$$

The code for the Laplacian kernel was made and commented in the repository for this project, but not yet implemented. One can expect less smooth results and faster convergence relative to using the Gaussian kernel.

A lot of experiments in terms of changing bandwidth parameters can be made too as was shown above.

2.4 Code

The original code is from the authors of MMD GAN paper and it can be found here: https://github.com/OctoberChang/MMD-GAN .

Since the paper was published in 2017, some functions of the code were deprecated and libraries have changed. The debugged code and accompanying work done for this project can be found here: https://github.com/Memirlan/MMD-GAN .

3 Conclusion

Although currently there are many better generative deep learning networks, GAN, GMMN and MMD GAN had a significant impact on the development of machine learning. They added novelty to existent methods and the mathematical analysis that lies behind the idea of these networks is useful for general understanding of learning.

MMD GAN outperformed GMMN, but it still can be improved, since the choice of kernel functions and selection of bandwidths remains an open problem.

Such models cannot outperform current state-of-the-art models, but they can shed light on interpretability and transparency of "black-box" deep learning models by using mathematical analysis, and they can re-gain their popularity later if theoretical machine learning and practical deep learning will eventually converge.

References

[1] Fukumizu, K., Gretton, A., Schölkopf, B., & Sriperumbudur, B. K. (2008). Characteristic kernels on groups and semigroups. Advances in Neural Information Processing Systems (NeurIPS), 21.
https://proceedings.neurips.cc/paper_files/paper/2008/file/d07e70efcfab08731a97e7b91be644de-Paper.pdf

[2] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative Adversarial Networks.
https://doi.org/10.48550/ARXIV.1406.2661

[3] He, H., Huang, G., & Yuan, Y. (2019). Asymmetric valleys: Beyond sharp and flat local minima. In arXiv [cs.LG].
http://arxiv.org/abs/1902.00744

[4] Li, C.-L., Chang, W.-C., Cheng, Y., Yang, Y., & Póczos, B. (2017). MMD GAN: Towards deeper understanding of moment matching network. In arXiv [cs.LG]. *http://arxiv.org/abs/1705.08584*

[5] Li, Y., Swersky, K., & Zemel, R. (2015). Generative Moment Matching Networks. In arXiv [cs.LG].
http://arxiv.org/abs/1502.02761

[6] Shalev-Shwartz, S., & Ben-David, S. (2014). Understanding machine learning: From theory to algorithms. Cambridge University Press.