GENERATIVE MODELING OF GAUSSIAN AND UNIFORM DISTRIBUTIONS

Swati Swati

Victoria University of Wellington, NZ

1. INTRODUCTION

Generative models are a class of machine learning algorithms designed to learn and generate new data samples that follow the same distribution as a given dataset. These models are pivotal in various applications, such as image generation, text synthesis, and data augmentation [1].

A fundamental approach to generative modeling involves training a pair of neural networks: an encoder and a decoder. The encoder maps data points into a latent space, while the decoder reconstructs the original data from the latent representation.

In this study, we explore foundational concepts of generative models by transforming one distribution into another. This approach offers insights into the challenges and potential solutions for learning complex data distributions.

2. THEORY

In this section, we focus on the theoretical background relevant to the implemented system.

2.1. Gaussian Distribution

A Gaussian distribution, or normal distribution [2], is a continuous probability distribution characterized by a bell-shaped curve symmetric around its mean. It is defined by two parameters, first is the mean (μ), which is the central value of the distribution, secondly, standard deviation (σ), which is the measure of the spread or dispersion of the data points around the mean. The probability density function (PDF) of a Gaussian distribution for variable x is given by:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{1}$$

2.2. Uniform Distribution

A uniform distribution [7] is a type of probability distribution where all outcomes are equally likely within a certain interval. If a random variable X follows a uniform distribution over an interval $[a, b]$, then every value within this interval has the same probability, and values outside this interval have a probability of zero. Mathematically, the probability density function (PDF) of a continuous uniform distribution is:

$$f(x) = \frac{1}{b-a} \quad \text{for } a \leq x \leq b \tag{2}$$

This means that the distribution is constant across the interval $[a, b]$, and the total area under the curve is equal to 1.

3. EXPERIMENTS

In this section, we train three networks using principle of generative modelling, model *f1* to convert a 2D Gaussian to a 2D uniform distribution, *f2* to perform the inverse mapping, and *f3* to convert a 1D uniform distribution to a 2D Gaussian distribution. We also investigate the effect of L1 and L2 regularization on the distribution of the weights in *f1*.

3.1. Experimental Setup

Two neural network models were employed. Model *f1*, designed to map a 2D Gaussian distribution to a 2D uniform distribution, featured a three-layer fully connected architecture with 64 neurons per hidden layer using *ReLU* activation. The output layer used a *Sigmoid* activation to constrain outputs between *0 and 1*, aligning with the uniform distribution's range. Model *f2*, which performed the inverse mapping, had a similar architecture but used *ReLU* activations throughout, omitting the final *Sigmoid* layer.

Model *f3*, tasked with mapping a 1D uniform distribution to a 2D Gaussian distribution, also had a similar architecture but with added complexity. It included an input layer for the 1D data, followed by hidden layers with 128 and 64 neurons. The output layer was divided into two parts: one to predict the mean of the 2D Gaussian and another for the log-variance, allowing for sampling from the learned distribution.

3.2. Training Process

Both models, *f1* and *f2*, were trained on 5000 samples of 2D Gaussian and 2D Uniform distributions using stochastic gradient descent, with *f1* employing a learning rate of 0.001 and *f2* using Adam optimization with the same rate.

Model *f1* was trained over 1000 epochs, while *f2* was trained for 5000 epochs, with data shuffled in each epoch. Mini-batches of 64 for *f1* and 32 for *f2* were used to improve

generalization, and model parameters were updated based on calculated gradients.

The loss function for both models was *Maximum Mean Discrepancy* (MMD) [5], which measures the difference between two probability distributions by comparing their means in a reproducing kernel Hilbert space (RKHS), embedding distributions into a higher-dimensional space and computing the distance between their means and is defined as:

$$\text{MMD}(X, Y) = \left\| \frac{1}{n} \sum_{i=1}^{n} \phi(x_i) - \frac{1}{m} \sum_{j=1}^{m} \phi(y_j) \right\|_H^2 \quad (3)$$

, where X and Y are the two distributions, and ϕ is the feature mapping function.

Finally the output from both the models, as shown in Fig. 1, were visualized along with their losses.

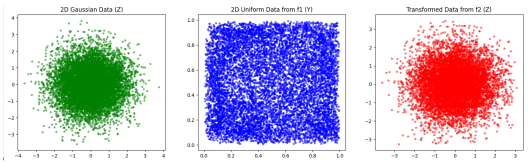

Fig. 1. Output of 2D Gaussian to 2D Uniform (model *f1*) and inverse mapping *(model f2)*.

The model f1 was then trained with different levels of L1 (Lasso), refer Eq. 4, and L2 (Ridge), refer Eq. 5, regularization [6] which are the techniques for preventing overfitting in machine learning models by adding a penalty term to the loss function. The effects of these techniques on weight is shown in Fig. 2.

$$\text{Loss} = \text{Original Loss} + \lambda \sum_i |w_i| \quad (4)$$

$$\text{Loss} = \text{Original Loss} + \lambda \sum_i w_i^2 \quad (5)$$

,where λ is the regularization strength.

The loss curve was plotted for each combination of regularization, as shown in Fig. 3, and best combination was found depending on early convergence, lower loss and balancing between underfitting and overfitting, as mentioned in Table 1.

Finally, model *f3* was trained using 1000 1D Uniform samples over 1000 epochs. Multiple attempts were made to generate 2D Gaussian data from 1D by increasing the number of epochs to 5000, adding more neural network layers, and switching to Kullback-Leibler divergence loss, but none succeeded. Ultimately, the successful approach was employing the principle of Variational Autoencoders (VAEs) [4], a type of generative model that learns a probabilistic representation of the input data through a layer of probabilistic reasoning. VAEs [3] use an encoder to predict the mean and log-variance of a probability distribution in latent space, and the reparameterization trick to sample points in this space. The output of model *f3* is displayed in Fig. 4.

The trained network maps input Y values to 2D Gaussian distributions by extracting features from Y using hidden layers, then predicting mean and log-variance vectors for each Y. By sampling points from these Gaussian distributions, the network effectively transformed a 1D uniform distribution into a complex 2D Gaussian distribution, visualized as a cloud of points shifting with varying Y values.

3.3. Results

As illustrated in Fig. 5, model *f1* exhibited a steep initial loss, indicating rapid learning at the outset. However, the loss curve plateaued at a higher MMD value than anticipated, suggesting that the model encountered difficulties in capturing finer details. In contrast, model *f2* showed a more gradual decrease in MMD, reflecting a cautious learning process likely due to its more complex architecture.

During the regularization of model *f1*, it was observed that increasing the *L1* regularization parameter caused more weights to trend toward zero, resulting in a sparser weight distribution. Similarly, increasing the *L2* regularization parameter led to a more concentrated weight distribution around zero, with fewer extreme values. This effect is evident in the narrower and taller peaks of the distributions as the *L2* regularization strength increases.

For model *f3*, although it converged slowly, as seen in Fig 6, it achieved a lower final MMD, indicating a better fit to the target distribution.

4. CONCLUSION

Both models, *f1* and *f2*, effectively mapped 2D Gaussian to 2D Uniform data and vice versa, despite encountering some losses. In the regularization of complex problems, L1 regularization proves advantageous in high-dimensional spaces by performing feature selection, thereby enhancing model interpretability and preventing overfitting. Meanwhile, L2 regularization, being more straightforward, successfully mitigates overfitting across a broader range of scenarios. Lastly, although model *f3* demonstrated slow convergence, it ultimately succeeded in learning the intricate mapping from 1D Uniform to 2D Gaussian data through the application of Variational Autoencoders (VAE).

5. STATEMENT OF TOOLS USED

The tools used in this study include PyTorch for neural network implementation, Plotly for graph visualization, and Google Colab for executing the code. The code was written by me, and I verify that it is my original work. The code can be accessed at the following link: *Final-Code*.

6. REFERENCES

[1] Yilun Du and Leslie Kaelbling. Compositional Generative Modeling: A Single Model is Not All You Need, 2024.

[2] Christopher M. Bishop F.R.Eng. *Pattern Recognization and Machine Learning: Chapter 2*. Springer, 2006.

[3] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. Foundations and Trends R in Machine Learning: Vol. xx, No.xx, pp 1–18. DOI: 10.1561/XXXXXXXXX, 2019.

[4] Prof Bastiaan Kleijn. Elbo application: Vae. Victoria University Lecture in lect4ObjF.pdf, 2024.

[5] Prof Bastiaan Kleijn. Maximum mean discrepancy (mmd): the theory. Victoria University Lecture in lect4ObjF.pdf, 2024.

[6] Andrew Y. Ng. Feature selection, l1 vs. l2 regularization, and rotational invariance. Association for Computing Machinery New York, NY, United States, 2004.

[7] Hongyi Zhang, Jan Bosch, and Helena Holmstrom Olsson. Federated learning systems: Architecture alternatives, 2020.

7. APPENDIX

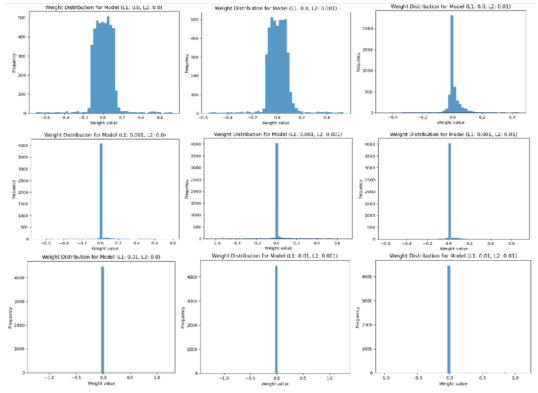

Fig. 2. Effects of L1 and L2 regularization on weights.

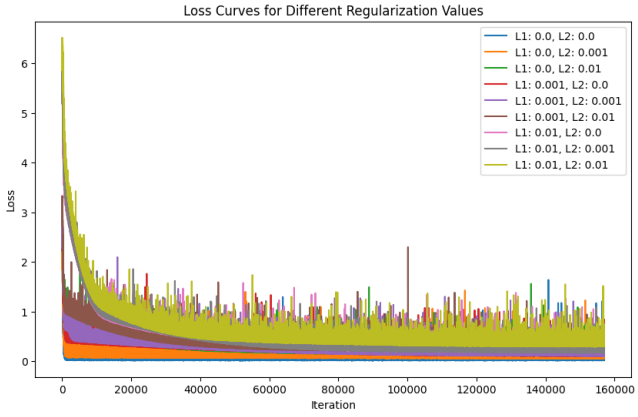

Fig. 3. Loss curve for combinations L1 and L2 regularization.

Table 1. L1 and L2 Lambda parameters optimization.

Regularization	Values	Best Value
L1_lambda	0.0, 0.001, 0.01	0.001
L2_lambda	0.0, 0.001, 0.01	0.001

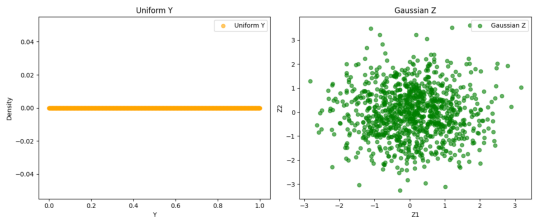

Fig. 4. Mapping of 1D Uniform data to 2D Gaussian data (model *f3*).

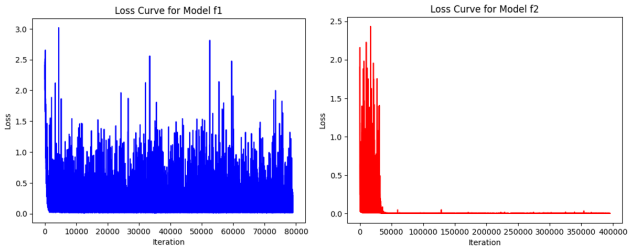

Fig. 5. Loss curves for 2D Gaussian to 2D Uniform mapping (model *f1*) and vice versa (model *f2*).

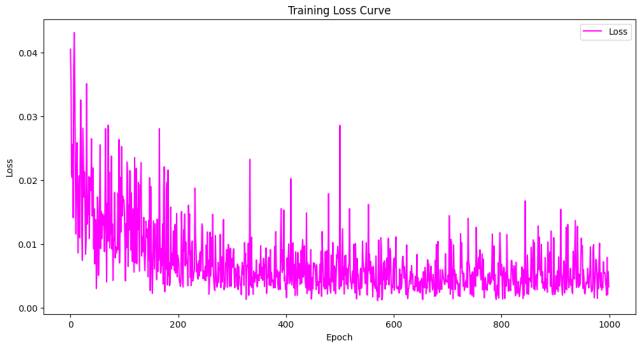

Fig. 6. Loss curve for 1D Uniform to 2D Gaussian data (model *f3*).