DEMONSTRATING THE BASIC PRINCIPLES OF GENERATIVE MODELS FOR (FCNN) GAUSSIAN AND UNIFORM MAPPINGS

Maria DaRocha

VUW, AIML425: Neural Networks and Deep Learning

1. INTRODUCTION

The following is my own work and supported by code available on GitHub and Google Collab. Please allow for some lenience on paper length, given the inclusion of three separate experiments.

Generative models are neural networks that identify patterns in existing data to generate new data that adheres to the patterns found during learning[1]. This project aimed to apply the basic principles of generative models to solutions for a defined problem set: (1) Train a fully connected neural network (f_1) to convert a 2D standard normal (Gaussian) distribution into a 2D uniform density Y, (2) Train a second network (f_2) that performs the inverse map, converting a 2D Y into a Gaussian 2D, (3) Train (f_1) with different levels of L1 and L2 regularisation on the weights; show the effect on the distribution of weights, discussing the effect of regularisation for this problem and more complicated problems, and finally, (4) Train a third network (f_3) that converts a 1D uniform Y into a 2D Gaussian Z.

2. THEORY

This paper defers to *Demonstration of a Simple Fully Connected Feed-Forward Neural Network (FCNN) Mapping 3D Vectors to a Sphere* for background on probability theory, stochastic gradient descent (SGD), backpropagation, and the chain rule (for vectors and tensors)[2].

Further requisite theory for this context is objective functions, regularisation (L2, L1, and dropout), integral probability metrics (MMD, EMD, or 1-Watterstein), and kernels (Gaussian/RBF). The primary reference for this section was AIML425 lecture and course notes[3].

Objective functions, often referred to as loss functions, ascribe a mathematical expression to the goal of an optimasation problem. The minimisation (e.g., MSE) or maximization (e.g., log likelihood) of this expression provides the fitness of a model in the form of a scalar output.

Complimentary to objective functions, regularisation can be used to reduce the freedom of a network and achieve better generalisation. This can be done either by applying a penalty term in the objective function or by modifying the network's operation. The regularisation techniques focused on for this discussion are L2, L1, and dropout.

L2 regularisation applies a penalty term defined by the L2-norm squared of the flattened weight vector. The inclusion of squaring has the effect of penalising outliers in the weights, resulting in a comparatively dense weight distribution. Conversely, L1 regularisation applies a penalty term defined by the L1-norm of the flattened vector (i.e., the sum of the absolute value of the weights). This type of regularisation does not penalise outliers and tends to result in sparsity, causing dominant weights to direct the behaviour of a network. Dropouts modify the network by dropping neurons or connections at random during training according to a specified percentage of the units of a layer (e.g., 50%). In practice, this can make a network robust and insensitive to overfitting.

Integral probability metrics such as maximum mean discrepancy (MMD) are highly useful for comparing empirical densities (as opposed to divergences like Kullblack Leibler and Jensen-Shannon, which are used to compare distributions in their analytical forms). MMD requires a kernel (commonly Gaussian/RBF), whereby a kernel is loosely defined as the generalisation of a matrix. When MMD is used as an objective function in a generative setting, it is sometimes called the critic or discriminator function. In this context, the goal is to minimise the MMD – that is, to make the distance between two distributions p_X of X and p_Y of Y (two datasets) as small as possible for a set of samples. Mathematically the MMD can be expressed as:

$$MMD\left(\{x^{(i)}\}_{i=1,\dots,m}, \{y^{(j)}\}_{j=1,\dots,n}\right) = \frac{1}{m(m-1)}\sum_{i\neq j} k\left(x^{(i)}, x^{(j)}\right) + \frac{1}{n(n-1)}\sum_{i\neq j} k\left(y^{(i)}, y^{(j)}\right) - 2\frac{1}{mn}\sum_{i=j} k\left(x^{(i)}, y^{(j)}\right)$$

For a Gaussian kernel,

$$k(x,y) = \exp\left(-\frac{||x-y||^2}{r}\right), \quad r = med\left(\{d_{ij} | 1 \le i \le j \le n\}\right)$$

where r is found using the median heuristic[4]. To attain r, we compute the pairwise (squared Euclidean) distances

between all data point pairs, flatten the upper triangle of the distance matrix (excl. the diagonal) to create a vector of all pairwise distances, and then compute the median of the flattened distance vector.

3. EXPERIMENTS

The experiments were set up in Jupyter Lab and used Python to create three instances of feed-forward fully connected neural networks (FCNN's), f_1, f_2, and f_3. For clarity the discussion will describe each network's operation, rather than responding directly to the numbering of the problem set (though all questions have been answered). Some design choices were applicable to all or more than one FCNN:

1. All networks were FCNN's that made use of training, testing, validation data, and mini batching.
2. All network and layers widths were tested and selected in magnitudes to introduce complex learning.
3. The most performant regularisation type tended to be L2, and for each network, the regularisation strength was also attained through magnitude testing [0.001, 0.01, 0.1].
4. Networks f_1 and f_2 trained on manual implementations of SGD optimsers, as specified by the problem set.
5. Networks f_1 and f_2 used Sigmoidal activation functions because of their differentiability and smooth gradient. Further, this helped normalize (bound) the values between (0,1), keeping them within a consistent scale while preventing vanishing or exploding gradients.
6. Minimum MMD was used as the objective function for model performance, given the comparison between two empirical distributions p_X of X and p_Y of Y.
7. Further, the MMD loss was computed using a Gaussian kernel and median heuristic; the justification for this was that the Gaussian kernel is ubiquitous, and the median heuristic was simple to employ, adaptable to data scale and distribution, and medians are often more stable than means in empirical contexts (i.e., less sensitive to outliers).

3.1. f_1 Design, Results, and Discussion

Table 1, f_1 ***Final Configuration***

Input ----------	Gaussian
Output ----------	Uniform
Width ----------	[2, 125, 125, 2]
Activation(s) ---------	Sigmoid
Learning Rate --------	0.01
Regularisation (Str.) -	L2 (0.01)
Batch Size / Epochs –	100 / 1,000

The goal of the first FCNN is articulated by introduction points (1) and (3); *Figure 1 (Appendix, 5.1)* is provided for visual reference. The final unique aspects of f_1's configuration (i.e., in addition to 1-5 above) are represented by *Table 1*. The regularisation type and strength were selected through experimentation. As discussed, we would not have expected L1 to reach the final configuration for f_1 due to the introduction of weight sparsity and dominance. Evidence of the impact of L1 regularisation applied to weights can be seen by comparing *Figure 2* and *Fingure 3 (Appendix).* The results of this model can be seen in *Figures 2-4 (Appendix, 5.1)*. Despite model performance scoring a supposed 0.087 test loss and its predictions outperforming other attempts' qualitative assessments, the output distribution did not successfully scale to a 2D square. Many other attempts were made, including hyperparameter tuning and testing different activation functions, different kernels, and even applying dropout. It remains that *Table 1* reflects the attempt with the lowest MMD loss and highest qualitative evaluation.

3.2. f_2 Design, Results, and Discussion

Table 2, f_2 ***Final Configuration***

Input ----------	Uniform
Output ----------	Gaussian
Width ----------	[2, 125, 125, 2]
Activation(s) ---------	Sigmoid
Learning Rate --------	0.01
Regularisation (Str.) -	L2 (0.01)
Batch Size / Epochs –	100 / 1,600

The goal of the second FCNN is articulated by introduction point (2); *Figure 5 (Appendix, 5.2)* is provided for visual reference. The final unique aspects of f_2's configuration (i.e., in addition to 1-5 above) are represented by *Table 2*. This model presented the same scaling issues as f_1, and performed well without significant modification (0.097 MMD loss). Similar testing, tuning, and debugging methods were used for f_2 and f_1 but unfortunately, they remained unsuccessful after many more hours and attempts. The quantitative and qualitative results of this model can be seen in *Figures 6-8 (Appendix, 5.2)*.

3.3. f_3 Design, Results, and Discussion

Table 3, f_3 ***Final Configuration***

Input ----------	1D Uniform
Output ----------	2D Gaussian
Width ----------	[1, 50, 50, 1] (x2)
Activation(s) ---------	Tanh
Learning Rate --------	0.001
Regularisation (Str.) -	L2 (0.001)
Batch Size / Epochs –	100 / 300

The goal of the final FCNN is articulated by introduction point (4). The final unique aspects of f_3's configuration (i.e., in addition to any applicable points from 1-5 above) are represented by *Table 3*. This solution was designed to handle mapping a 1D uniform input to a 2D Gaussian output by first

separating each coordinate of a Gaussian pair $G(x)G(y)$. The model transformed the first 1D distribution according to one coordinate of a Gaussian pair, thus creating a normal distribution in one direction and a uniform distribution in another. Then, the second coordinate was prepared using the first model's predictions. For the plotted mapping from the 1D input to the predicted 2D Gaussian output, see *Figures 9-11 (Appendix, 5.3)*. Another less complex approach could have been to add Gaussian noise to the transformation. The model performed moderately well in the way of purported loss (0.0287), but still didn't quite achieve the desired spiral projection.

4. CONCLUSION

While not all attempts were equally performant, this set of experiments successfully applied some of basic principles used in generative modelling, including:

- SGD optimisation and backpropagation,
- Objective functions (MMD loss),
- Regularisation (L2/L1/Dropout),
- Integral probability metrics (MMD), and
- Kernels (Gaussian/RBF)

These methods and techniques demonstrated the intricacy of FCNN-optimisation. Despite the challenges described, the experiments still succeeded in demonstrating the usefulness of MMD for comparing empirical distributions, the sensitivity of hyper-parameter tuning, and the importance of regularization and kernel selection for a network.

If allowed more time, my aspiration would be for all models to attain a minimal MMD loss whilst observing sound qualitative (plotted prediction) feedback. Future work could focus on further tuning of hyperparameters or exploring alternative architectures to achieve more precise transformations between distributions.

5. APPENDIX

5.1. Model f_1

Figure 1, f_1 Inputs and Aspired Outputs, Respectively

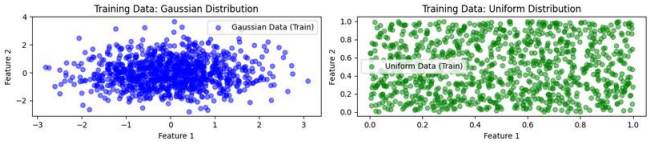


Figure 2, L1 (0.01) Weight Regularisation

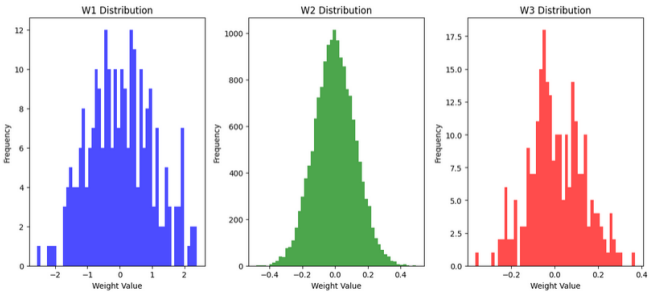


Figure 3, L2 (0.01) Weight Regularisation

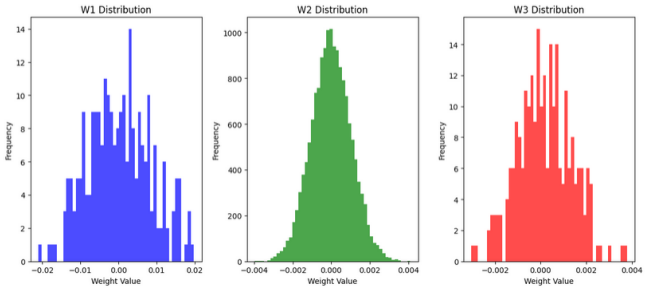
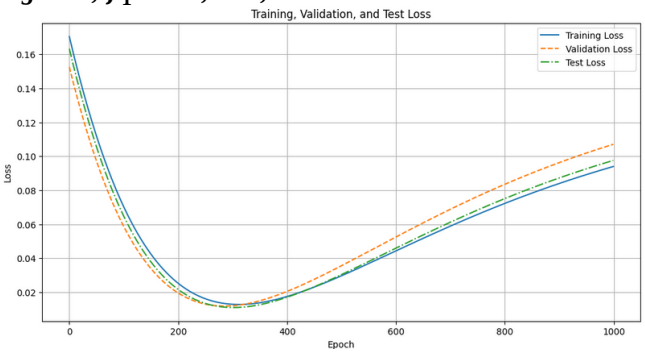
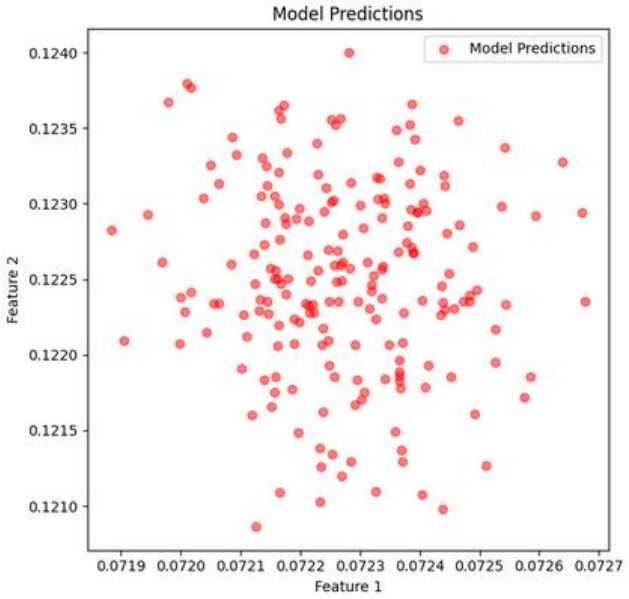


Figure 4, f_1 Train, Test, and Validation Loss



	Epoch	Train Loss	Validation Loss	Test Loss
0	0.0	0.1704	0.1525	0.1633
1	100.0	0.0702	0.0586	0.0647
2	200.0	0.0252	0.0195	0.0216
3	300.0	0.0129	0.0120	0.0110
4	400.0	0.0175	0.0205	0.0170
5	500.0	0.0297	0.0356	0.0304
6	600.0	0.0443	0.0525	0.0459
7	700.0	0.0589	0.0688	0.0612
8	800.0	0.0723	0.0835	0.0751
9	900.0	0.0840	0.0963	0.0873

Figure 4, f_1 Model Predictions for Qualitative Assessment



5.2. Model f_2

Figure 5, f_2 Inputs and Aspired Outputs, Respectively

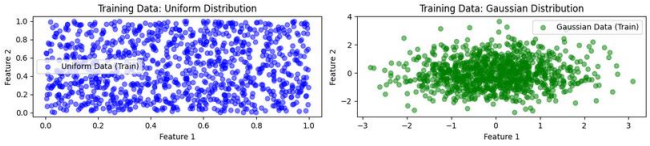
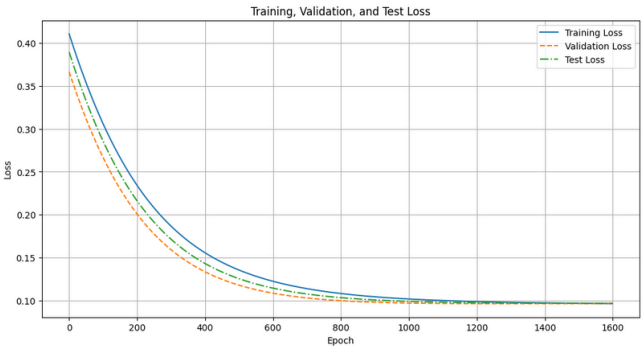


Figure 6, f_2 Train, Test, and Validation Loss



	Epoch	Train Loss	Validation Loss	Test Loss
0	0.0	0.4107	0.3895	0.3895
1	100.0	0.3059	0.2856	0.2856
2	200.0	0.2341	0.2161	0.2161
3	300.0	0.1866	0.1714	0.1714
4	400.0	0.1556	0.1431	0.1431
5	500.0	0.1355	0.1254	0.1254
6	600.0	0.1224	0.1144	0.1144
7	700.0	0.1138	0.1076	0.1076
8	800.0	0.1081	0.1033	0.1033
9	900.0	0.1043	0.1007	0.1007
10	1000.0	0.1018	0.0991	0.0991
11	1100.0	0.1000	0.0980	0.0980
12	1200.0	0.0988	0.0974	0.0974
13	1300.0	0.0979	0.0970	0.0970
14	1400.0	0.0973	0.0968	0.0968
15	1500.0	0.0968	0.0967	0.0967

Figure 7, f_2 Input, Test, and Model Predictions, Respectively

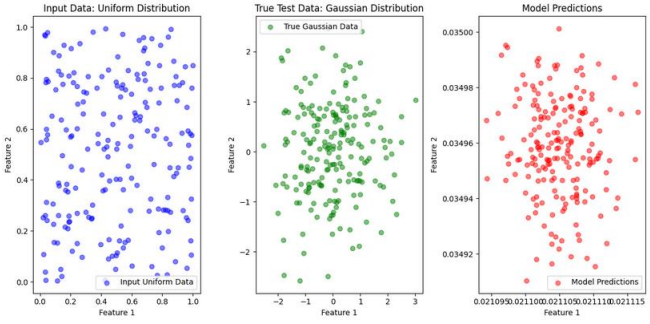
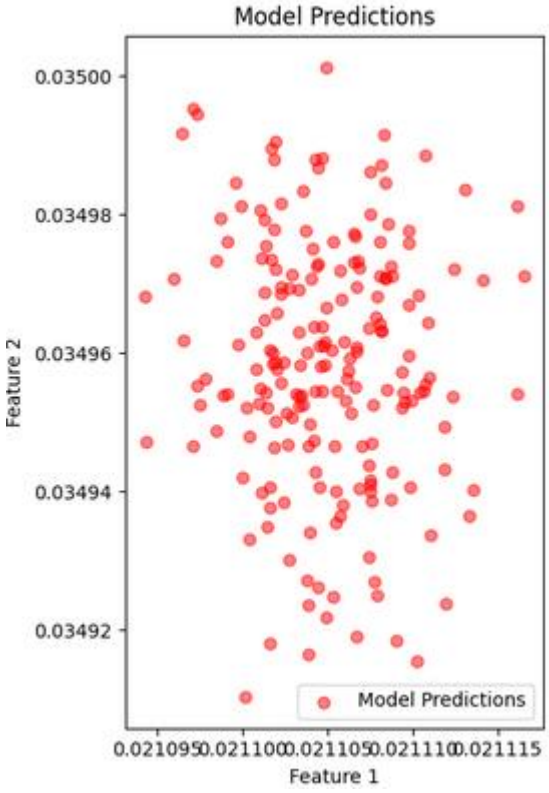


Figure 8, f_2 Model Predictions Only



5.3. Model f_3

Figure 9, *f_3 2D Gaussian Mapped from 1D Uniform*

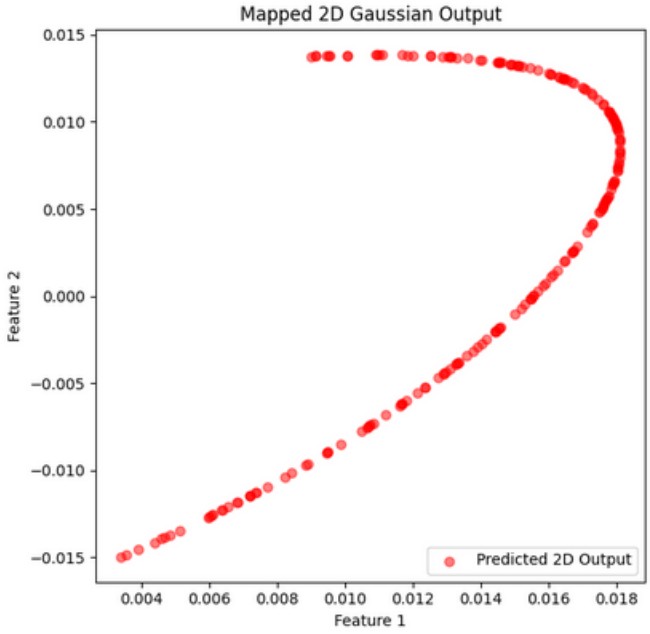
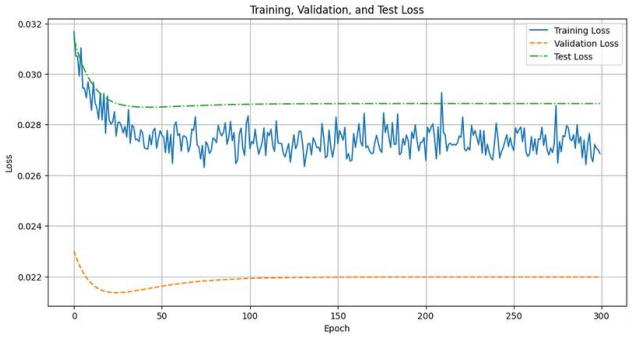
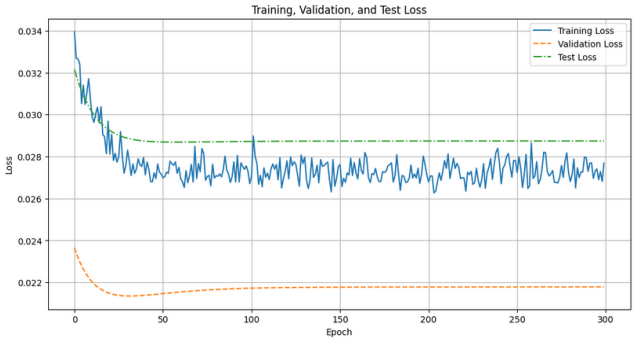


Figure 10, *f_3 Train, Test, and Validation Losses 1&2*



	Epoch	Train Loss	Validation Loss	Test Loss
0	0.0	0.0317	0.0230	0.0314
1	100.0	0.0271	0.0219	0.0288
2	200.0	0.0266	0.0220	0.0288



	Epoch	Train Loss	Validation Loss	Test Loss
0	0.0	0.0339	0.0236	0.0322
1	100.0	0.0271	0.0217	0.0287
2	200.0	0.0268	0.0218	0.0287

6. REFERENCES

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Generative Models*. MIT Press, 2016. [Online]. Available: https://www.deeplearningbook.org/

[2] M. DaRocha, "Demonstration of a Simple Fully Connected Feed-Forward Neural Network (FCNN) Mapping 3D Vectors to a Sphere," *VUW, AIML425: Neural Networks and Deep Learning*, Jul. 2024, Accessed: Aug. 11, 2024. [Online]. Available: https://github.com/Marianette/A1-AIML425/blob/main/A1-AIML425.pdf

[3] B. Kleijn, "Basic Deep Learning Regressor & Gradient Descent and Context: AIML425 [Lecture 3 & 4 Notes]," *Master of Artificial Intelligence Year One, Victoria University of Wellington*, Aug. 2024, Accessed: Jul. 29, 2024. [Online]. Available: https://ecs.wgtn.ac.nz/foswiki/pub/Courses/AIML425_2024T2/LectureSchedule/DLintro.pdf

[4] B. Schölkopf and A. J. Smola, *Smola, A.: Learning with Kernels - Support Vector Machines, Regularization, Optimization and Beyond. MIT Press, Cambridge, MA*, vol. 98. 2001.