DT8122 Project Assignment — Normalizing Flows

Qamar F S Alfalouji

Septemebr 2022

1 Introduction

This report shows the results and implementation details of the Normalising Flows assignment as part of the ProbAI 2022 summer school. All the implementations are done using PyTorch, and based on what was discussed during the summer school lectures. Code is available at: `https://github.com/Qamar-93/dt8122_Submission`

2 Task 1: Normalizing Flows (NFs)

2.1 Description

Normalizing flows are used to estimate a complex distribution by starting from a simple distribution. This transformation happens by mapping the points from the initial distribution to the target complex one by using a set of invertible functions. So if f is a smooth invertible function, where $f{:}R^d \to R^d$, with inverse $f^{-1} = g$, so that the composition $g \bigcirc f(z) = z$. If this mapping happens over the variable z with distribution $q(z)$, then the resulting random variable density (pdf) can be seen from the change of variables formula as follows:

$$q\left(z'\right) = q\left(z\right)\left|det\frac{\partial f^{-1}}{\partial z'}\right| = q\left(z\right)\left|det\frac{\partial f}{\partial z'}\right|^{-1} \tag{1}$$

If we use a stack of k invertible transformations sequentially, then the density of the last variable can be estimated as follows:

$$z_k = f_k \bigcirc ... \bigcirc f_2 \bigcirc f_1(z_0) \tag{2}$$

$$ln\, q_k(z_k) = ln\, q_0(z_0) - \sum_{k=1}^{K} ln\, \left|det\, \frac{\partial f_k}{\partial z_{k-1}}\right| \tag{3}$$

The word *flow* presents the path traversed by z which is shown as $f_k(z_{k-1})$ with initial distribution $q_0(z_0)$, where the path formed by the successive distributions q_k is a *normalizing flow.* The main issue in this computation is the determinant term in the density formula, which has a cubic cost in either the dimension of z, or the number of hidden units. To do so, there are several types of normalizing flows that apply different approaches including Planar Flows and Real-valued Non-Volume Preserving.

2.2 Planar Flows

2.2.1 Definition

The Planar Flows introduce the following invertible transformation:

$$f(\mathbf{z}) = \mathbf{z} + \mathbf{u}h(\mathbf{w}^T\mathbf{z} + b) \tag{4}$$

where $\mathbf{u}, \mathbf{w}, \mathbf{b}$ are the network parameters. Accordingly, the absolute value of the determinant of the Jacobian is given by:

$$\left| det \frac{\partial f(\mathbf{z})}{\partial \mathbf{z}} \right| = \left| 1 + h'(\mathbf{w}^T \mathbf{z} + b) \mathbf{u}^T \mathbf{w} \right| \tag{5}$$

The flow is constructed by stacking multiple layers of these transformations over each other. The training process of the Planar flow includes sampling from a defined target distribution, $x\ p(x)$, in the forward function of the neural network, in order to derive the latent space variable z. The final loss, which is in this case the log Jacobian, is minimized over the training iterations.

2.2.2 Implementaion and Results

Since the Target distribution is not defined, and we can not assume that we know it in advance, two possible approaches were tried.

- Estimating the target distribution: By using the variational inference, the target distribution could be estimated using Variational Auto Encoder (VAE), and placing the Planar Flows network in the middle of an encoder and a decoder. The encoder in the first step is used to estimate the variance and the standard deviation of the distribution given the input samples. This distribution is treated as the target distribution that the Planar Flows need to calculate the loss.
- Using analytical solutions to calculate the inverse and find the roots of the function using Newton Raphson method or the bisection method.

However, None of these methods was able to give good results. And it was not possible to plot the inverse of the functions or sampling from the flow due to the absence of the inverse part.

2.3 Real-valued Non-Volume Preserving (Real NVP)

2.3.1 Definition

The main concept of this transformation is using the affine coupling layers. The affine coupling includes the process of dividing variables of the sample into two chunks. Then a neural network is used to estimate a scale in the forward function of the network. This scale is then used to shift the other chunk of the samples so that:

$$z_{1:d} = x_{1:d}, \;\; z_{1+d:D} = x_{1+d:D} \odot \exp\left(s(x_{1:d})\right) + t(x_{1:d}) \tag{6}$$

where D is the sample dimension, and hence the inverse is:

$$x_{1:d} = z_{1:d}, \;\; x_{1+d:D} = z_{1+d:D} - t(y_{1:d}) \odot \exp\left(-s(y_{1:d})\right) \tag{7}$$

Same as Planar Flows, a neural network is used to train this type of flow, by stacking layers of coupling (forward and inverse) sequentially. The Jacobian determinant is used as the loss function to be minimized over the training iterations.

2.3.2 Implementations and Results

Datasets: The three dataset are split and shuffled as a preprocessing step by using 80% for the training and the rest 20% for testing. The training is performed over batches of size = 32 at each iteration.

Network and Flow: A Multi-Layer-Perceptron (MLP) with the following hyperparameters is used as the network of the flow:

- Number of layers = 3.
- Number of hidden units = 32.
- Activation function = GELU.
- Learning rate = 1e-3.

- Number of epochs = 200.

All the flows were generated using 4 layers of couplings and a last inverse coupling for the output.
Results Results of training and testing the Real-NVP flows on the three different given datasets are shown below:

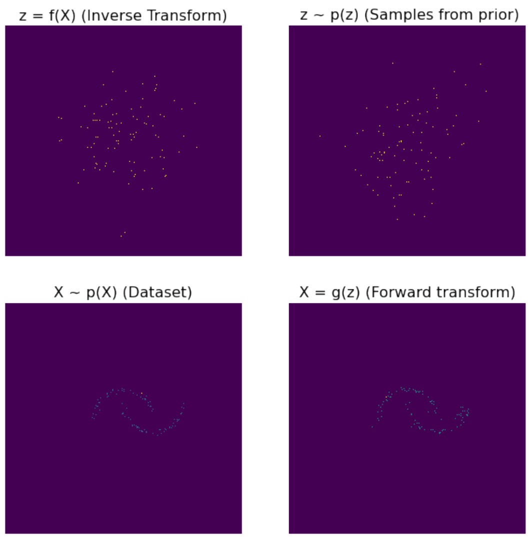

Figure 1: The results of Real-NVP for dataset two moons.

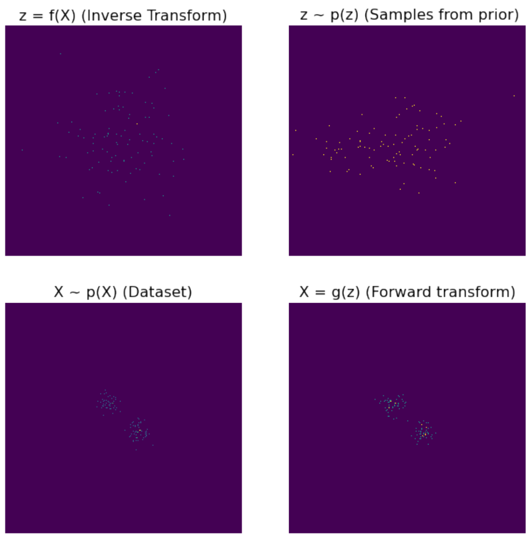

Figure 2: The results of Real-NVP for dataset two blobs.

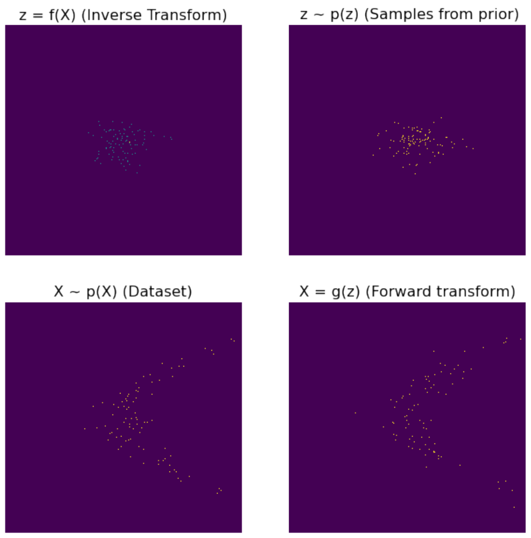

Figure 3: The results of Real-NVP for dataset boomerang.

2.4 Difference between Planar Flows and Real-NVP

The main advantage of Real-NVP over Planar flows is the coupling feature. Stacking multiple layers of coupling allows for estimating complex distributions while preserving all the details of the samples. Additionally, using the inverse process allows sampling from the flow easily and going back to the initial distribution. The absence of this feature in the Planar Flows made it not possible to solve such a problem without using analytical solutions.

3 Task 2: Continuous Normalizing Flows (CNFs)

3.1 Description

As mentioned in section 2.1 the main problem in solving equation 1 is the term of the determinant of jacobian, which has a high computational cost. Continuous-Time Flows shows that moving from a discrete set of layers to a continuous transformation simplifies the computation of the change of the variable formula. According to Theorem 1 [1], we can use a trace operation to compute the log determinant. Additionally, f, in this case, does not need to be a bijective function since the uniqueness is satisfied, then the entire transformation is bijective. If we reflect this on the planar flows, then the change of variable formula becomes:

$$\frac{\partial \mathbf{z}(t))}{\partial t} = uh(w^T\mathbf{z}(t) + b), \quad \frac{\partial\, log\, p(\mathbf{z}(t))}{\partial t} = -u^T \frac{\partial h}{\partial \mathbf{z}(t)} \tag{8}$$

And hence, using the initial distribution $p(\mathbf{z}(0))$, we can sample from $p(\mathbf{z}(t))$ and evaluates the density by solving the combined ODE in terms of the time.

3.1.1 Implementation and Results

Dataset Same three datasets are used with a random shuffle and 80% of training data.
Network and Flow The implementation of the Continuous-Time Flows is done using Neural ODEs, the *torchdiffeq* library, and a hyper-network that allows the function f to change over time. The change over time happens by adapting the weights in the forward function. The network learning rate is 3e-3 with Adam optimizer. The training was performed for 100 epochs, using a batch size of 32, hidden dimension of 32 and network width of 64.

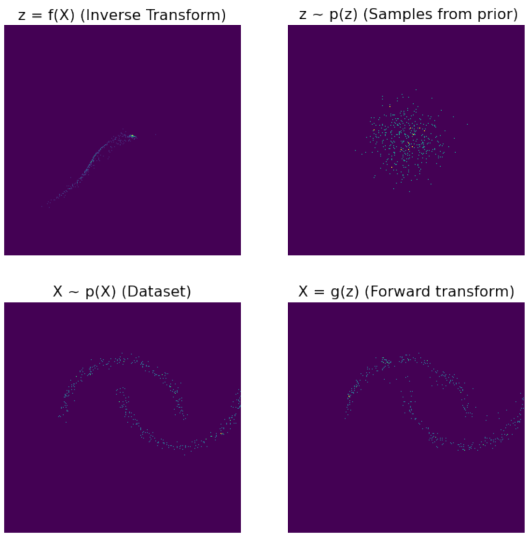

Figure 4: The results of CTF for dataset two moons.

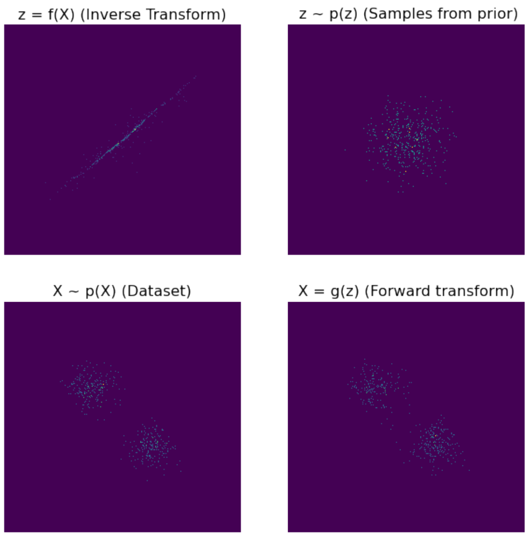

Figure 5: The results of CTF for dataset two blobs.

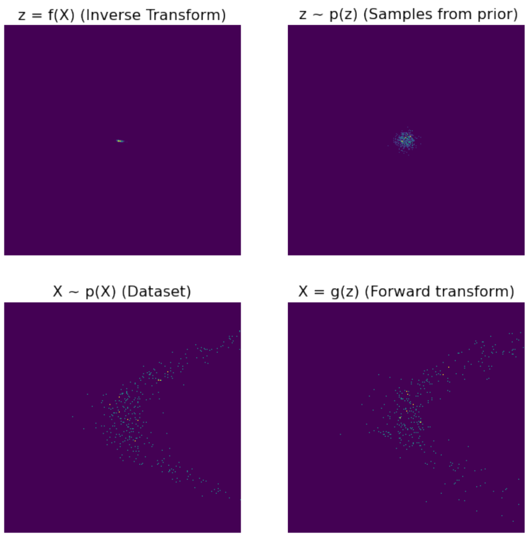

Figure 6: The results of CTF for dataset boomerang.

3.2 Difference between CTFs and Coupling Flows

- Using the trace operation helps om evaluating the model with many hidden neurons using a low computational cost compared to the coupling flows. In contrast to the determinant term of the normalizing flows, the trace function is linear, so that if the system is represented by a sum of functions then the differential equation is also a sum as described in equation 10 **empty citation**
- Due to the possibly infinite number of transformations that can be done between every two states, CTFs can be seen as more general than NFs.
- CTFs are more theoretically grounded, in the sense that they are guaranteed to approach a target distribution asymptotically.
- In CTFs, the estimated function f does not need to be bijective.
- In terms of the results, since CTFs are more expressive than NFs, they took a lower number of epochs to capture the details of the estimated distribution.
- Considering the time makes the transition from one step to the other smoother, compared to the deterministic point-wise solutions from the Real-NVP.
- On the other hand, due to the high number of transformations, training a CTF needs more time compared to Real-NVP for example it needs 20.76 minutes for the two moons dataset while the Real-NVP needs only 1.15 minutes.

References

[1] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, *Neural ordinary differential equations*, 2018. DOI: 10.48550/ARXIV.1806.07366. [Online]. Available: https://arxiv.org/abs/1806.07366.

[2] L. Dinh, J. Sohl-Dickstein, and S. Bengio, *Density estimation using real nvp*, 2016. DOI: 10.48550/ARXIV.1605.08803. [Online]. Available: https://arxiv.org/abs/1605.08803.

[3] D. J. Rezende and S. Mohamed, *Variational inference with normalizing flows*, 2015. DOI: 10.48550/ARXIV.1505.05770. [Online]. Available: https://arxiv.org/abs/1505.05770.

[4] C. Chen, C. Li, L. Chen, W. Wang, Y. Pu, and L. Carin, *Continuous-time flows for efficient inference and density estimation*, 2017. DOI: 10.48550/ARXIV.1709.01179. [Online]. Available: https://arxiv.org/abs/1709.01179.