Rössler attractor - continuous deterministic chaos
Imitating a dynamical system

Gladys Roch
CentraleSupélec
Deep Learning in Practice
gladys.roch@supelec.fr

Abstract

The task is to imitate a given dynamical system, namely, Rössler attractor. This is a dynamical system of the form $w_{t+1} = f(w_t)$ where w = (x,y,z) are 3D coordinates and where f is a function (from R^3 to R^3) which does not depend on t. Given a location w_t at time t, this tells where the new location at $t+1 : w_{t+1} = f(w_t)$. The goal is then to find this function f (i.e., to infer it from data samples). The estimation of f is build using a RNN of input in R^3 and outputs in R^3.

1. Introduction

The Rössler attractor is a chaotic system, which means that any small change in the initial point, $w + dw$ will lead to a completely different trajectory after some time. The state vector $w = (x, y, z)$ is defined according to a set of constant parameters (a, b, c), by Eq.1

$$\begin{cases} \dot{x} = -y - z, \\ \dot{y} = x + ay, \\ \dot{z} = b + z(x - c) \end{cases}$$

The speed at which trajectories of close initial states diverge from each other is given by the Lyapunov exponent (λ). For two very close initial points w_0 and w'_0, $||w_t - w'_t|| \sim e^{\lambda t}||w_0 - w'_0||$ (when $t \longrightarrow +\infty$).

To imitate the Rössler attractor I chose to use the continuous formulation. With this approach, the task is to induce the function g, where g defines the system such as $\frac{dw}{dt} = g(w_t)$. This task can be completed using a supervised method of input w and output $\frac{dw}{dt}$. g can be learnt by a simple forward neural network.

To train the network, the experimental data is generated by the RosslerMap simulator with the set of parameters $a = 0.2, b = 0.2, c = 5.7$. The derivative at each point is then estimated by $\frac{dw}{dt} \sim \frac{w_{t+delta\_t} - w_t}{delta\_t}$ where $delta\_t$ is the sampling time-delta. $delta\_t$ can be set as parameter of the simulator. In order to have a good approximation of the derivative for training the model $delta\_t$ should be as small as possible I choose $delta\_t = 10^{-2}$.

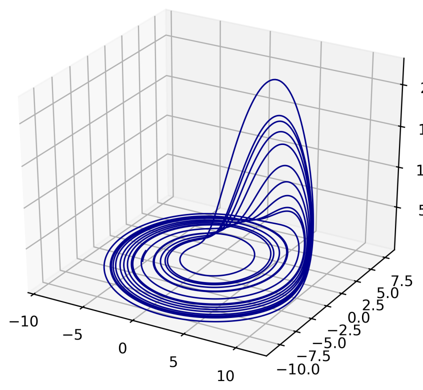

Figure 1. Rössler Attractor with $a = 0.2, b = 0.2, c = 5.7$ and 100s (1000 data points)

2. Implementation

2.1. Network Architecture

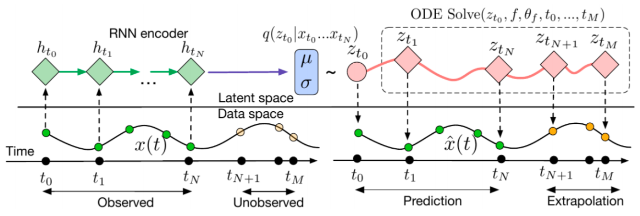

Figure 2. Architecture of the Neural ODE model

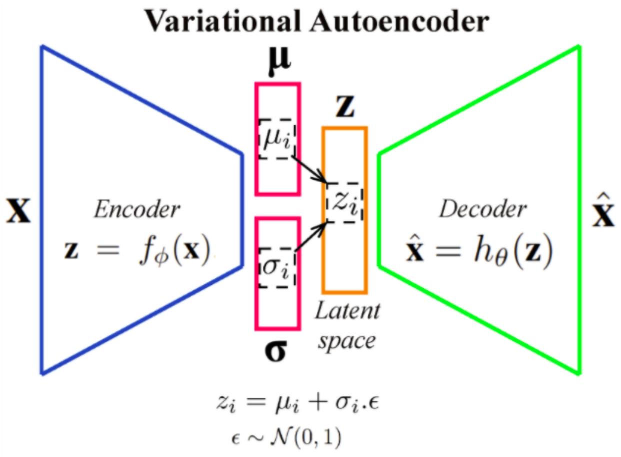

Figure 3. Architecture of a VAE

Fig. 1 illustrates the general structure of the adopted architecture. It follows the principle of Variational Auto Encoders (VAE) depicted in Fig. 2. The trajectory is learned in a latent space (**Z**) connected to the observation space (**X**) through an encoder and a decoder. The encoder is a Recurent Neural Network (RNN) and the decoder a simple feed forward network.

The model takes as input a sample of N consecutive data points sampled from the Rössler attractor $(w_{t_0}, ..., w_{t_N})$. This trajectory is encoded by the RNN encoder into a probabilistic latent state $q(z_{t_0}|w_{t_0}, ..., w_{t_N})$. A state z_{t_0} is then sampled from the distribution $q \sim \mathcal{N}(\mu, \sigma)$ using the reparametrization trick. This z_{t_0} serves as the initial state from which the trajectory must be reconstructed.

In the latent space, the ODE solver (Ordinary Differential Equation) computes the latent trajectory $(z_{t_1}, z_{t_2}, ..., z_{t_N})$: it takes as input the initial state z_{t_0}, the system function g ($\frac{dz}{dt} = g(z_t)$) and the instants of reconstruction $(t_0, t_1..., t_N)$, and returns a (good) approximation of $z_{t_i} = z_{t_0} + \int_{t_0}^{t_i} g(z_t)dt = z_{t_0} + \int_{t_0}^{t_i} \frac{dz}{dt}dt$ for ever t_i. The latent trajectory is the decoded to an observable trajectory $(\hat{w}_{t_0}, \hat{w}_{t_1}, ..., \hat{w}_{t_N})$.
To sum up, our objective is to model the data, that is we want to find the distribution of the data $p(W)$. To do that, we train an encoder $q(z|W)$ that project our data W into latent variable space Z, and a decoder module composed of an ODENet and a FFNet that generates data given a latent variable: $p(W|z)$.

2.2. Size of the latent space

For RAM problematic I use 4000 observations Delage and al. state that when time series are not infinitely long (¿32,000 values) or noisy-free, the embedding dimension, d_E, and the time delay, $T = N \times delta\_t$,are correlated. They propose to determine the time window length $t_w = (d_E - 1)T$, that is the time spanned by the embedding vector. In our case,

2.3. Objective function - ELBO

The reasoning behind the ELBO objective function (ELBO=Evidence Lower BOund) is clearly explained by [2] and gives the following expression:

$$ELBO(q) = \log p(w) - KL(q(z)||p(z|w))$$

where KL is the Kullback-Leibler divergence and w is the observed state (x, y, z). KL measures the closeness between the two distribution q and $p(.|w)$, it is always positive and is null when the two distribution are equal. The objective is to minimize KL divergence, which is equivalent to maximizing the ELBO (minimizing -ELBO).

2.4. Code

The model is implemented in python using pytorch. The ODENet is from the *torchdiffeq* library[1]. The VAE model itself is an adaptation of the latent_ode.py from *torchdiffeq/examples*.

3. Results

3.1. Parameters of the model

The model is trained from on the 10,000 first points of the Rössler attractor (from the initial state). 1000 trajectories of 500 data points each are sampled. These trajectories are fed into the network 1000 times.

The model is trained using Adam optimizer with a learning rate of 0.01.

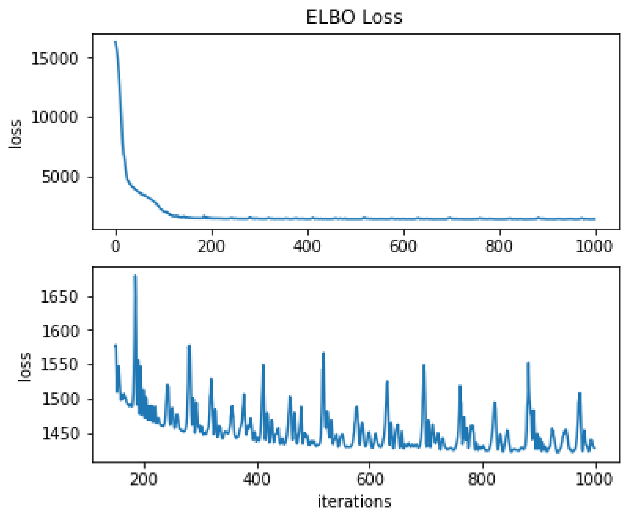

Figure 4. ELBO Loss during training

[1]https://github.com/rtqichen/torchdiffeq

3.2. Evaluation of the dynamical system learned

3.2.1 Probability density function

The density of probability learned can be visually evaluated by sampling the latent space and passing it through the decoder. The distribution learned is shown in the figure bellow. The global visual reconstruction is acceptable: the Rössler attractor is identifiable. However, the trajectory diverges rapidly from the true trajectory as shown in figure 6.

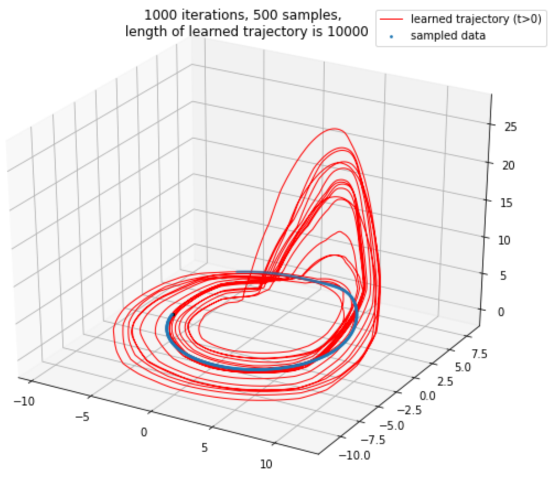

Figure 5. Visual output of the model. In red: prediction of 10000 points of the Rössler attractor after 1000 epochs. In blue: the sample data from which is extrapolated the red line

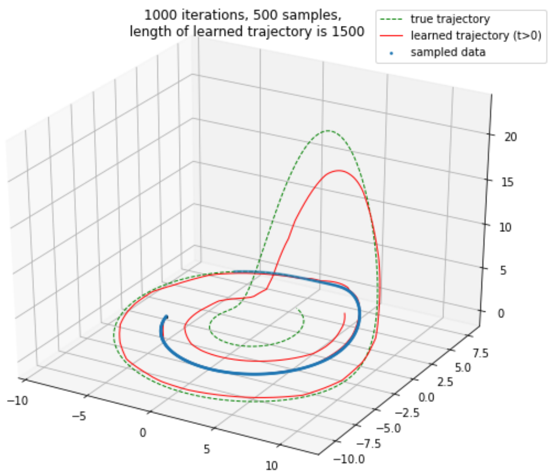

Figure 6. 1500 first points of the imitation (red) and the real Rössler attractor (green). In blue: the sample data from which is extrapolated the red line

3.2.2 Time correlation

The correlations between the real signal and the predicted trajectory (10,000 points) is presented in the following table. The y-axis correlation is only of 0.2, which is low.

Axis	correlation coefficient
x	0.221
y	0.205
z	0.089

3.2.3 Spectral domain

The dynamic model can also be evaluated in the frequency domain. The power density of the model and the real attractor are shown in the figure bellow (Fig. 7). The periodicity along the x- and y-axis has been partially learned: the power density of the imitation is a smooth/spread version of that of the real attractor.

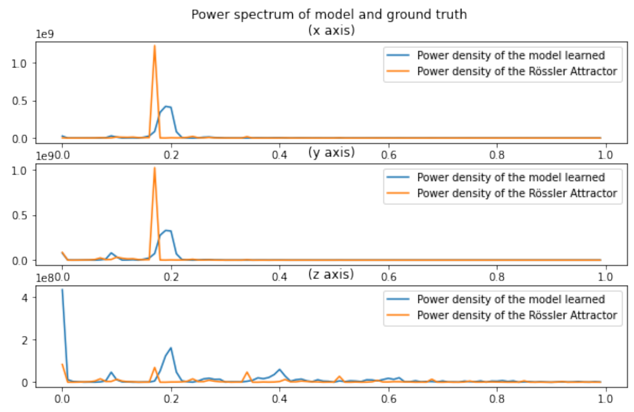

Figure 7. Power density along each axis of the ground truth (orange) and the imitation (blue) (trajectories of 10,000 points)

4. Conclusion

The imitation of a dynamic system can be approached in continuous-time, using a VAE associated with a ODE solver. The function of the attractor g is learned by the decoder, which then allows to generate the trajectory. The visual reconstruction shows similarity with the real attractor but at a closer look it is clear that the generated trajectory diverges quickly from the real one. The evaluation of the model shows that the pseudo-periods of the attractor have been identified but only partially learned as the time-correlation corroborates.

References

[1] Delage, O. and Bourdier, A. (2017) Selection of Optimal Embedding Parameters Applied to

Short and Noisy Time Series from Rössler System. Journal of Modern Physics, 8, 1607-1632. https://doi.org/10.4236/jmp.2017.89096

[2]David M. Blei et al. (2018). Variational Inference: A Review for Statisticians.