Learning Simple Chaotic Flows

William Huang wh629@nyu.edu
Department of Mathematics
Courant Institute of Mathematical Sciences
New York University
New York, NY 10012, USA

Sachin Satish Bharadwaj ssb638@nyu.edu
Department of Mechanical and Aerospace Engineering
Tandon School of Engineering
New York University
New York, NY 11201, USA

Editor: William Huang and Sachin Satish Bharadwaj

Abstract

The world is filled with processes that are extremely dynamical and chaotic. A pressing problem in science has always been trying to predict chaotic systems. Such predictions will bear tremendous impact on the world and its dynamics. Here, we try to apply a nascent approach to "learning" chaos through neural networks by taking a set of three dimensional autonomous ODEs of chaotic flows to understand the efficacy of the amalgamation of **Chaos and Machine Learning.** We find that given the sophistication of these ODEs and our neural network, we are able to reasonably predict simple chaotic flows.

Keywords: Machine learning, Deep learning, Dynamical systems, Neural Networks

1. Introduction

Dynamical systems are ubiquitous and have the greatest impact on almost everything we observe. Unfortunately, not all dynamical systems are analytically tractable and have closed form solutions. This is due to the intrinsic randomness that arises purely due to the system dynamics. These systems are "ergodic" and "chaotic" and their dynamics are hard to forecast, let alone classify. While these processes are very common, it is almost impossible to predict the state of the system in the next moment given the state of the system at a specific time. The key signatures of chaos in the context of machine learning are the facts that: (a) the system is either quasi periodic or has no periodicity at all, and (b) the systems are deterministic for each trajectory, but the trajectories diverge drastically even with the slightest perturbation of initial conditions. A brief note on Chaos, Lyapunov Exponents and Strange Attractors is given in Appendix C.

Recently in Pathak et al. (2018) and Pathak et al. (2017), state of the art reservoir computing/learning methods have been implemented to understand a model free prediction of a system that evolves in a spatio-temporal chaotic manner. Immediately following this work in Nakai and Saiki (2018), they studied methods to understand the dynamics of variables in

chaotic fluids using similar reservoir computing techniques. On parallel grounds, there have been other works, such as in Rudy et al. (2019) and Cestnik and Abel (2019), where they have tried to understand and predict the trajectories of other chaotic systems using similar deep learning techniques to understand chaotic double pendulums and oscillatory systems. The success of such techniques motivates us to use the framework proposed in Rudy et al. (2019) to learn several simple chaotic systems found in Sprott (1994) to gain insights into the behavior and performance of deep learning in the context of chaos.

2. Method

In this section, we first describe the problem statement and framework for the network and then explain the methodology used in this report.

2.1 Problem Formulation and Model

For each chaotic system, we consider the underlying dynamics of the form,

$$\frac{d}{dt}\mathbf{x}(t) = f(\mathbf{x}(t)),$$

where f is unknown. However, we are provided measurements $\mathbf{Y} = [y_1, \ldots, y_m]$ representing measurements of the true state x_j at time t_j corrupted by some measurement noise ν_j,

$$\mathbf{y}_j = \mathbf{x}_j + \nu_j.$$

The discrete-time map from $\mathbf{x}_j$ to $\mathbf{x}_{j+1}$ is represented by

$$\mathbf{x}_{j+1} = F(\mathbf{x}_j) = \mathbf{x}_j + \int_{t_j}^{t_{j+1}} f(\mathbf{x}(\tau))d\tau,$$

where we use the Runge-Kutta scheme to approximate $F(\mathbf{x}_j)$. Under this framework, we use the neural network to approximate the vector field of the dynamics through $\hat{f}_\theta$ to predict $\hat{\mathbf{x}}_{j+1} = \hat{F}_\theta(\mathbf{x}_j)$. Additional information about the network can be found in Appendix A.

2.2 Experimental Procedure

Our procedure follows that in Rudy et al. (2019), fitting the network to 12 of the simple chaotic systems in Sprott (1994). Namely, we simulate each system starting near one of its critical points for 25,000 time steps (sampling data at every 10th step) and add 5% error to the trajectory. We then fit the neural network to the single noisy trajectory data and use the learned vector field to generate a new trajectory. Using both simulations, we then visually and quantitatively evaluate the accuracy of the model.

Given the known dynamical equation and measurement noise, we are able to quantify two measurements of error: E_f and E_N. These represent the relative squared L^2 error of the learned vector field and the mean L^2 difference in learned measurement noise, respectively.

$$E_f = \frac{\sum_{j=1}^{m} \|f(\mathbf{x}_j) - \hat{f}_\theta(\mathbf{x}_j)\|_2^2}{\sum_{j=1}^{m} \|f(\mathbf{x}_j)\|_2^2} \qquad E_N = \frac{1}{m}\sum_{j=1}^{m} \|\nu_j - \hat{\nu}_j\|_2^2 = \frac{1}{m}\sum_{j=1}^{m} \|\mathbf{x}_j - \hat{\mathbf{x}}_j\|_2^2$$

Finally, to understand how the learned vector field generalizes to different initial conditions, we apply several perturbations to one of the x, y, or z coordinates and compare the trajectories between the known dynamics equation and learned vector field.

3. Experimental Results and Conclusion

Here we use cases A, B, E, and F to illustrate the abilities and the shortcomings of the neural net. Definitions of the system equations can be found in Appendix B.

3.1 Chaotic Trajectories

Inspecting Figure 1 for systems A, B, and E we show the chaotic trajectories of the systems as obtained by (a) RK4 time integration of the system (in blue), (b) trajectories that the neural net has learned (in red), and (c) RK4 and NN trajectories after some initial perturbation. Upon inspection of the first two columns in systems A and B, we see that the learned dynamics perform fairly well. However, system E gives an example of where the neural net fails to capture the dynamics of its training data. In fact, it misses a turning point! We hypothesise that these turning points represent a form of **hidden stratification** in the data. These points occur with such low frequency that the network finds an optimal objective value while failing to capture these turns due to equally weighted errors.

The second observation comes from the third columns of systems A, B, and E, that depict the trajectories of the systems after some initial perturbation. These represent the generalization of the learned vector field to other initial conditions. Quite remarkably, the network learns to trace the trajectory fairly well even for a slightly perturbed chaotic system! We know that chaotic systems behave differently for different initial conditions, yet the network is able to roughly trace the RK4 trajectory of the perturbed state. To improve generalization, it may benefit to train the network on an ensemble of trajectories.

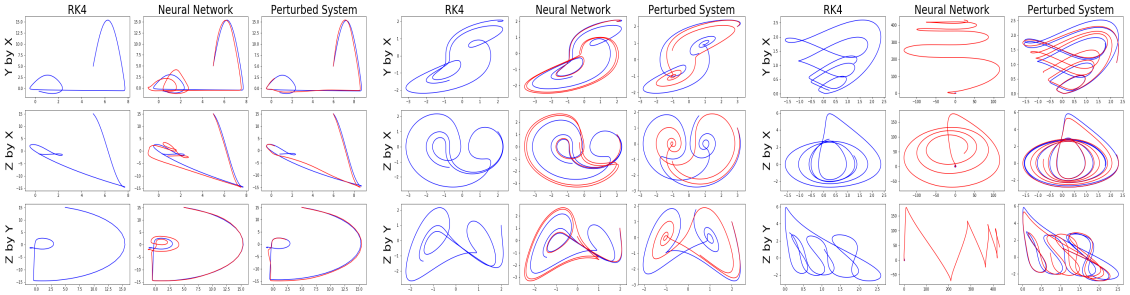


Figure 1: Chaotic trajectories

Let us now consider Figure 2 for system F, where the network "completely" fails to capture the single turning point. This system represents an extreme case of the potential hidden stratification discussed above. From the first set plots, we can see that the system stays at a certain point for most of the trajectory and quickly shoots off and returns. In the second set of plots, we see the network misses this return and continues evolve the system beyond this turning point.

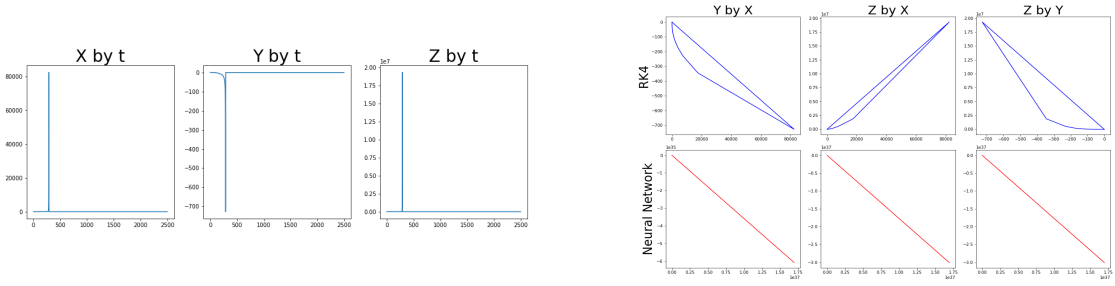
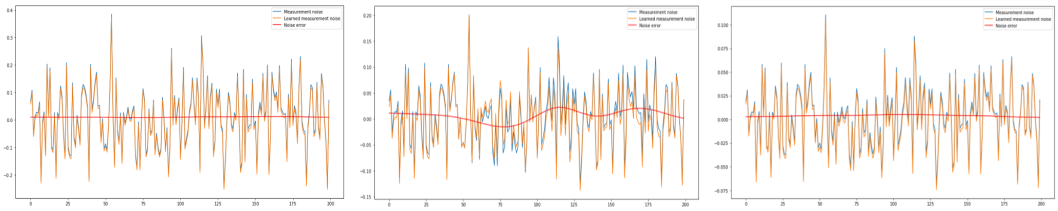


Figure 2: Trajectory of system F

3.2 Measurement Errors

In the below table we summarize the model error measurements for each of these systems. Additional measurements can be found in Appendix B.

Case	Reported Lyapunov	E_f	E_N
A	0.014	0.000728	0.002358
B	0.210	0.000146	0.000835
E	0.078	0.000188	0.000303
F	0.117	1.115e+11	1.005732



A B E

Figure 3: Time Evolution of Measurement Noise Error

From the table we can see that systems A, B, and E have comparably good vector field and measurement noise errors. However, when evaluating the plots along with these measurements we conclude that additional evaluation metrics may be required. While systems B and E have similar metrics as A, the plots above show worse performance in either the networks generalization (in system B) or its performance on the trained data set (in system E). Note that E_N represents an average error of the measurement noise, and Figure 3 above show the measurement noise errors over time.

Appendix A: Description of Neural Network

The discrete-time mapping allows the model to focus on approximating the dynamics f by fitting the neural network

$$\hat{f}_\theta(\mathbf{x}) = \left(\prod_{i=1}^{l} C_{g_i}\right)(\mathbf{x}) \text{where } g_i(\mathbf{x}) = \sigma_i(\mathbf{W}_i\mathbf{x} + \mathbf{c}_i),$$

where C_g is the composition operator with g and $\sigma(\mathbf{x})$ is the smooth activation function,

$$\sigma(\mathbf{x}) = \begin{cases} e^{\mathbf{x}} - 1 & \text{for } \mathbf{x} \leq 0 \\ \mathbf{x} & \text{for } \mathbf{x} > 0 \end{cases}.$$

The model then uses this approximation to approximate $\hat{\mathbf{x}}_{j+1} = \hat{F}_\theta(\mathbf{x}_j)$.

To derive the objective function, the model assumes distributions for the timestepper error, measurement error, and neural network parameters as $\xi_j = \mathbf{x}_{j+1} - \hat{F}_\theta(\mathbf{x_j}) \sim \mathcal{N}(0, \sigma_\xi^2\mathbf{I})$, $\nu_j = \mathbf{y}_j - \mathbf{x_j} \sim \mathcal{N}(0, \sigma_\nu^2\mathbf{I})$, and $\mathbf{W}_{i,jk} \sim \mathcal{N}(0, \sigma_w^2)$, respectively, giving,

$$\mathcal{L}(\theta, \mathbf{N}) = \sum_{j=1}^{m-1} \|\hat{F}_\theta(\mathbf{y}_j - \hat{\nu}_j) + \hat{\nu}_{j+1} - \mathbf{y}_{j+1}\|_2^2 + \gamma\|\hat{\mathbf{N}}\|_F^2 + \beta\sum_{i=1}^{l}\|\mathbf{W}_i\|_F^2,$$

where $\gamma = \sigma_\xi^2\sigma_\nu^{-2}$ and $\beta = \sigma_\xi^2\sigma_w^{-2}$. This objective function can be viewed as the traditional maximum-likelihood estimator of the prediction error $(\mathbf{y}_{j+1} - \hat{\mathbf{y}}_{j+1})$ followed by regularization terms for the measurement noise and model parameter, which allows the network to simultaneously learn the dynamics and measurement noise.

Appendix B: Cases from Sprott (1994) with Experimental Errors

Case	Equations	Reported Lyapunov	E_f	E_N
A	$\dot{x} = y$ $\dot{y} = -x + yz$ $\dot{z} = 1 - y^2$	0.014	0.000728	0.002358
B	$\dot{x} = yz$ $\dot{y} = x - y$ $\dot{z} = 1 - xy$	0.210	0.000146	0.000835
C	$\dot{x} = yz$ $\dot{y} = x - y$ $\dot{z} = 1 - x^2$	0.163	0.000303	0.000644
D	$\dot{x} = -y$ $\dot{y} = x + z$ $\dot{z} = xz + 3y^2$	0.103	1.825e+12	0.997514
E	$\dot{x} = yz$ $\dot{y} = x^2 - y$ $\dot{z} = 1 - 4x$	0.078	0.000188	0.000303
F	$\dot{x} = y + z$ $\dot{y} = -x + 0.5y$ $\dot{z} = x^2 - z$	0.117	1.115e+11	1.005732
G	$\dot{x} = 0.4x + z$ $\dot{y} = xz - y$ $\dot{z} = -x + y$	0.034	2.575e+3	0.000129
H	$\dot{x} = -y + z^2$ $\dot{y} = x + 0.5y$ $\dot{z} = x - z$	0.012	4.362e+13	0.990734
I	$\dot{x} = -0.2y$ $\dot{y} = x + z$ $\dot{z} = x + y^2 - z$	0.012	4.362e+13	0.990734
J	$\dot{x} = 2z$ $\dot{y} = -2y + z$ $\dot{z} = -x + y + y^2$	0.076	2.636e+35	1.172828
N	$\dot{x} = -2y$ $\dot{y} = x + z^2$ $\dot{z} = 1 + y - 2z$	0.076	0.003779	0.000797
R	$\dot{x} = 0.9 - y$ $\dot{y} = 0.4 + z$ $\dot{z} = xy - z$	0.062	0.000849	0.000598

Appendix C: Chaos, Lyapunov Exponents and Strange Attractors

Chaos can range from a trajectories of people in a market to turbulent flow of clouds in the atmosphere. Though one might see some order, this perceived order is localised to specific scales. Even in the absence of a universally accepted definition of chaos, we quote Strogatz (2018), who gives a rather intuitive one : "An **aperiodic long-term** behavior in a **deterministic** system that exhibits **high sensitivity to initial conditions.**". This sensitivity to initial condition is the hallmark of chaos. Thus, this high dependence on initial condition makes chaos very unpredictable. This inability to predict motivates us to try and test different methods of machine learning to learn chaos. This is in fact our nascent and a humble effort to see how far one can go with even the simplest of methods to understand the intrinsic chaos in nature.

Interestingly, even with high sensitivity to initial conditions, there still exists a set of initial conditions where each point in that set plumes out trajectories that end up in the same set after a long time. These "strange" sets are called **"Strange" Attractors**. So, if a system enters into such a domain of "attraction", then the trajectory shall remain in that attraction basin. The extent up to which two slightly differing initial points diverge with time is quantified by what is called the **Lyapunov Exponent**

The Lyapunov exponent is calculated from the following:

$$|\delta \mathbf{R}(t)| = e^{\lambda t} |\delta \mathbf{R}(0)|$$

Here λ is the Lyapunov Exponent. This quantifies how trajectories diverge over time, starting from small a initial separation. Hence the exponent is given by:

$$\lambda = \lim_{t \to \infty} \lim_{|\delta \mathbf{R}(0)| \to 0} \frac{1}{t} \ln \frac{|\delta \mathbf{R}(t)|}{|\delta \mathbf{R}(0)|}$$

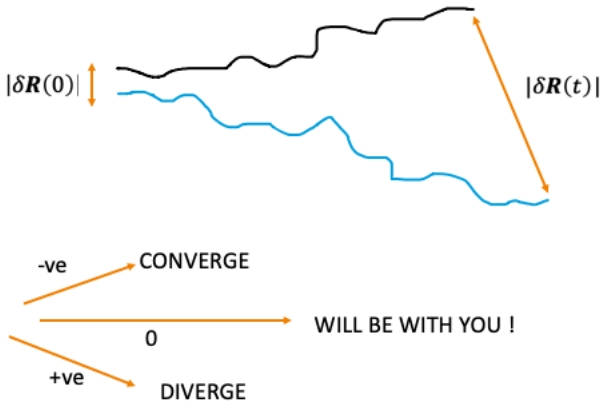


Figure 4: Lyapunov Exponent

In this study we also tried to compute an approximate Lyapunov exponent and they seem to show that the exponents converge to a constant over time, as expected. We show in the following figure, the evolution of the Lyapunov exponents over time, for different initial perturbations. Their convergence indicate that the system is able to capture the "trend" of the dynamics fairly well, for at least a few selection of systems.

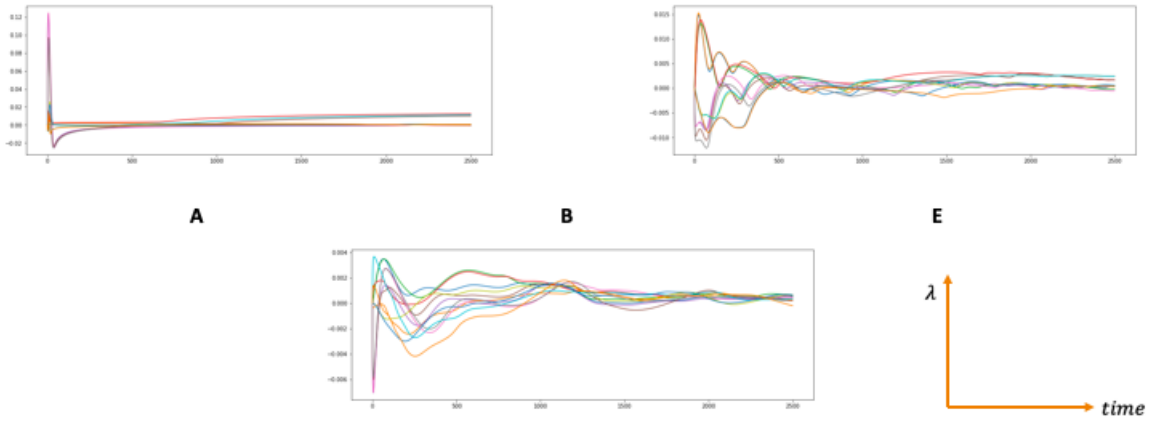


Figure 5: Time Evolution of Lyapunov Exponent

On the other hand, what are **Strange Attractors**? Very briefly, an attractor is a closed set A which is :

1. An **invariant set** : Trajectories starting from this set, end up in the same set.
2. **Attracts** an open set of initial set of conditions : Meaning that, there exists a small region around A, such that, trajectories starting off of this region will get attracted into A.
3. A **minimal set** : There is no proper subset of A that satisfies the above two conditions.
4. They are highly initial condition dependent and form **fractal sets**, hence termed "Strange".
5. They thus concentrate the dynamics of the system around them if the trajectory every enters a domain of the attraction.

References

Rok Cestnik and Markus Abel. Inferring the dynamics of oscillatory systems using recurrent neural networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(6):063128, 2019.

Kengo Nakai and Yoshitaka Saiki. Machine-learning inference of fluid variables from data using reservoir computing. *Physical Review E*, 98(2):023111, 2018.

Jaideep Pathak, Zhixin Lu, Brian R Hunt, Michelle Girvan, and Edward Ott. Using machine learning to replicate chaotic attractors and calculate lyapunov exponents from data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(12):121102, 2017.

Jaideep Pathak, Brian Hunt, Michelle Girvan, Zhixin Lu, and Edward Ott. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Physical review letters*, 120(2):024102, 2018.

Samuel H Rudy, J Nathan Kutz, and Steven L Brunton. Deep learning of dynamics and signal-noise decomposition with time-stepping constraints. *Journal of Computational Physics*, 396:483–506, 2019.

J Clint Sprott. Some simple chaotic flows. *Physical review E*, 50(2):R647, 1994.

Steven H Strogatz. *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. CRC Press, 2018.