Parameter Inference of Time Series by Delay Embeddings and Learning Differentiable Operators

Alex Tong Lin[1], Daniel Eckhardt[2], Robert Martin[3], Stanley Osher[4], and Adrian S. Wong[5]

[1, 4]University of California, Los Angeles
[2, 3, 5]Air Force Research Laboratory, Edwards AFB

Abstract

A common issue in dealing with real-world dynamical systems is identifying system parameters responsible for its behavior. A frequent scenario is that one has time series data, along with corresponding parameter labels, but there exists new time series with unknown parameter labels, which one seeks to identify. We tackle this problem by first delay-embedding the time series into a higher dimension to obtain a proper ordinary differential equation (ODE), and then having a neural network learn to predict future time-steps of the trajectory given the present time-step. We then use the learned neural network to backpropagate prediction errors through the parameter inputs of the neural network in order to obtain a gradient in parameter space. Using this gradient, we can approximately identify parameters of time series. We demonstrate the viability of our approach on the chaotic Lorenz system, as well as real-world data with the Hall-effect Thruster (HET).

1 Introduction

Studying the dynamical system arising from real-world phenomenon is a common approach to understanding said phenomenon. But it is most often the case that we don't have a mathematical description of the dynamical system, and instead we must settle with observed data. Thus, many researchers and practitioners are starting to use machine learning to ascertain properties of these systems, wherein there is ample data. A common approach to understanding dynamical systems is to infer the governing equations, whether represented symbolically or as a neural network. In our approach, we are not necessarily trying to reconstruct these equations, but rather to infer the parameters of the governing equations, even when these equations are unknown.

For example, in tackling the problem of parameter inference from time series data for the Lorenz system (which was developed to be a simplified model of atmospheric convection [15]):

$$\begin{aligned}\frac{dx}{dt} &= \sigma(y-x)\\ \frac{dy}{dt} &= x(\rho-z)-y\\ \frac{dz}{dt} &= xy-\beta z\end{aligned}$$

there are three parameters: σ is the Prandtl number, ρ is the Rayleigh number, and β has no name but is related to the physical proportions of the region under consideration [22]. In our problem

Distribution Statement A: Approved for Public Release; Distribution is Unlimited. PA Clearance AFRL-2022-1216 (Submitted for approval on March 2, 2022)

setup, we only have access to time series data with parameter labels, and our goal is to infer the parameters of new time series.

In this work, we introduce a method that tackles the following scenario: We seek to identify system parameters of unlabeled trajectories from time series data, given that we *only* have trajectory data with labeled system parameters. Our contribution lies in the insight that we can utilize delay-embeddings for time series in order to reconstruct the state-space, which then allows us to use neural networks to learn the velocity operator of the ODE. And because one can backpropagate through the neural network, then we can construct gradients in parameter space in order to perform parameter inference.

2 Related Works

Applying machine learning techniques to dynamical systems have become a recent trend, now that there is an abundance of data. Some researchers have taken to performing system identification by identifying the governing symbolic equations through data [2, 19, 4], while others have taken to representing the governing equations through neural networks [23, 18, 5, 17, 16]. Delay-embeddings in terms of system identification have also been examined in [12, 5]. Although often in system identification, the dynamical parameters are fixed, and thus there is a need for approaches that directly perform parameter inference.

In terms of parameter inference, approaches from an optimal transport view are considered in [24], where they know the form of the equation. In [8], they also take an optimal transport view but infer only one parameter. Parameter inference using the architecture of Echo State Networks has also been explored in [1], where they work on the Lorenz equation.

From the perspective of simulations (and less dynamical systems), using neural networks to infer parameters from a have gone back as far as [9] which was applied to the field of animations and took a control theory view, as well as in [7] which also constructs a gradient in parameter space like us, but is applied to simulations where the full state-space is accessible and without noise.

3 Methods

Here we provide an overview of our method. We first present the method of delay-embeddings for time series in order to reconstruct the state-space of the system. Afterwards, we provide common notation and notions for dynamical systems, such as forward Euler integration. Then we explain the learning phase of the neural network, which seeks to learn the dynamics operator. And finally, we explain the inference stage, where we are able to produce gradients in parameter space in order to infer system parameters.

3.1 Data preprocessing: Delay Embedding

When dealing with deterministic dynamical systems, we need access to the full state space of the system in order to uniquely determine the current state, much less predicting the next one. Failure to fully span the state space makes the system appear non-deterministic, but this is only an artifact of not having a full state. Such issues often arise in experimental time series of nonlinear systems, and are particularly tricky to deal with since the state space and physical model are both unknown. Any measurements of the system is unlikely to be a state variables and, in addition to all this, the number of measured variables is likely less than the dimension of the state space. The introduction of noise in the measurements also adds another layer of complication, which will be discussed toward the end of this subsection.

A clever workaround to all these predicaments is to introduce time delay embedding as a preprocessing step for the data. This embedding technique is possible due to the famous Takens' Embedding Theorem and is widely used in time series analysis, particularly in the presence of chaotic orbits [21]. Takens' Embedding Theorem states that an n-dimensional (possibly fractal)

manifold of a time series can be reconstructed diffeomorphically by creating a time-delay vector of at most $m > 2n$ entries, each with a fixed delay of τ [20]. The number of entries may be less than m and this varies between systems.

$$\mathbf{y}(t) := (y(t), y(t-\tau), y(t-2\tau), \ldots, y(t-(m-1)\tau))$$

The time delay embedding is equivalent to incorporating information of higher order derivatives as surrogate state variables. There are additional considerations, such as having τ large enough for the states $y(t)$ and $y(t-\tau)$. Having τ too small means that the states $y(t)$ and $y(t-\tau)$ are strongly correlated, therefore containing no additional information.

3.1.1 Minimum Embedding

The act of using time delay embedding is akin to unfolding the attractor such that artificially intersecting trajectories are no longer intersecting. Such "artificially intersecting" trajectories are also called "false neighbors". The most prevalent tests involving the determination of minimum embedding dimension will test, in some form or another, the ratio of "neighborliness" observed. The Kennel [14, 13] and Cao [3] algorithms both quantify "neighborliness" by tracking the average distance between a point and its nearest neighbour as the embedding dimension increases. The False First Nearest Neighbor algorithm (Krakovská 2015) tracks "neighborliness" depending on whether nearest neighbors remain nearest neighbors after consecutive embeddings. Both methods are suitable for our application, and because neither of them have a clear advantage, we employ both methods in our determination of minimum embedding.

3.2 The dynamical system

After delay-embedding the time series, we have reconstructed the state space and can now consider the dynamical system. We consider a dynamical system,

$$\frac{d}{dt}\mathbf{x}(t, \boldsymbol{\alpha}) = F(\mathbf{x}(t, \boldsymbol{\alpha}), \boldsymbol{\alpha}), \quad x_0 \in \mathbb{R}^n \tag{1}$$

with $x_0 \in \mathbb{R}^n$, $\boldsymbol{\alpha} \in A \subseteq \mathbb{R}^m$, $\mathbf{x} : [0, T], A \to \mathbb{R}^n$, $F : \mathbb{R}^{n+m} \to \mathbb{R}^n$, and $t \in [0, T]$. For notational simplicity, we write $\mathbf{x}(t) = \mathbf{x}(t, \boldsymbol{\alpha})$, where the dependence on α is implicit. For the system (1), we assume sufficient regularity conditions (e.g. F is uniformly Lipschitz in $\mathbf{x}$ for each $\boldsymbol{\alpha}$), so that existence and uniqueness is guaranteed [6].

Numerically, in order to produce trajectories of the dynamical system one can use various integrators, such as forward Euler,

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + \Delta t\, F(\mathbf{x}(t_k), \boldsymbol{\alpha}) \tag{2}$$

where we divide the time domain into equal step-sizes so that $0 = t_0 < t_1 < \cdots < t_N = T$. There also exist more involved schemes with better convergence properties, e.g. 4th order Runge-Kutta. In our experiments, we stick with the simplicity of forward Euler, which seem sufficient in our cases.

3.3 The Learning Stage: Training to minimize integration error

The Learning Stage in our method requires training a neural network to minimize integration error, i.e. to predict a future time-step of a trajectory, given the current time-step and system parameter. Namely, we place a neural network prior on F in (2) to obtain,

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + \Delta t\, N_\theta(\mathbf{x}(t_k), \boldsymbol{\alpha})$$

For example, the Lorenz system with standard parameters has a fractal box-counting dimension of $n = 2.06$. Takens' theorem states that we need at most $m = 5 > 4.12$ components in the time delay vector to reconstruct the attractor. As it happens, the Lorenz attractor can be reconstructed with just 3 components in the time delay vector.

where we approximate F with N_θ, with θ being the weights of the neural network. Our goal is to minimize the loss function that reduces this forward Euler integration error with respect to the weights θ,

$$\min_\theta \mathbb{E}_{k,\boldsymbol{\alpha}} \left[\frac{1}{2} \left\| \mathbf{x}(t_{k+1}) - (\mathbf{x}(t_k) + \Delta t \, N_\theta(\mathbf{x}(t_k), \boldsymbol{\alpha})) \right\|^2 \right] \tag{3}$$

the expectation being over trajectory points, $\mathbf{x}(t_k)$ and $\mathbf{x}(t_{k+1})$, and system parameters, $\boldsymbol{\alpha}$. More precisely, given that our data consists of trajectories and corresponding system parameters,

$$\left\{ \left(\{\mathbf{x}(t_k, \boldsymbol{\alpha})\}_{k=1}^N, \boldsymbol{\alpha} \right) \right\}_{\boldsymbol{\alpha} \in A}$$

then we want to minimize (3), although experimentally we found it better conditioned to minimize the rearranged loss,

$$\min_\theta \mathbb{E}_{k,\boldsymbol{\alpha}} \left[\frac{1}{2} \left\| \frac{\mathbf{x}(t_{k+1}) - \mathbf{x}(t_k)}{\Delta t} - N_\theta(\mathbf{x}(t_k), \boldsymbol{\alpha}) \right\|^2 \right] \tag{4}$$

After sufficient training of the neural network, N_θ, to minimize the loss (4), we now present the Inference Stage of our method, which will produce a gradient in the system parameter space A.

3.4 The Inference Stage: Producing gradients in parameter space

In the Inference Stage, we now complete our goal in identifying the corresponding system parameters for trajectories in which these parameters are unknown. After training a neural network $N_\theta = N_\theta(\mathbf{x}, \boldsymbol{\alpha})$ as in the Learning Stage, we now have a computationally differentiable operator – namely N_θ – with which we can produce gradients with respect to system parameters. Of course, N_θ is differentiable because it is just the product of matrices and activation functions. Then given a trajectory $\{\mathbf{x}(t_k)\}_{k=1}^N$ whose corresponding system parameters is unknown, this parameter (or an equivalent parameter) must be the argument minimum of the following loss

$$\min_{\boldsymbol{\alpha}} \mathbb{E}_k \left[\frac{1}{2} \left\| \frac{\mathbf{x}(t_{k+1}) - \mathbf{x}(t_k)}{\Delta t} - N_\theta(\mathbf{x}(t_k), \boldsymbol{\alpha}) \right\|^2 \right]$$

which we note is essentially (4), except we are minimizing with respect to $\boldsymbol{\alpha}$ and the expectation is now only in k. Now since N_θ is differentiable, being a neural network, then we have available the following gradient descent update rule:

$$\boldsymbol{\alpha}_{j+1} = \boldsymbol{\alpha}_j - h \, \nabla_{\boldsymbol{\alpha}} \left(\mathbb{E}_k \left[\frac{1}{2} \left\| \frac{\mathbf{x}(t_{k+1}) - \mathbf{x}(t_k)}{\Delta t} - N_\theta(\mathbf{x}(t_k), \boldsymbol{\alpha}) \right\|^2 \right] \right)$$

with h the gradient step-size. In order to compute the gradient with respect to $\boldsymbol{\alpha}$, we perform automatic differentiation all the way through to the input layer of N_θ. We note that experimentally, we actually use gradient descent with momentum, but present here the simpler gradient descent for clearer exposition.

We demonstrate the effectiveness of our approach on the chaotic Lorenz system, as well as real-world data with the Hall-effect Thruster (HET) [11].

4 Experiments

We examine our method on two datasets: the Lorenz system, and the Hall-effect thruster (HET), the latter being real-world data.

In all experiments, we use a feed-forward neural network with an input layer, three hidden layers, and an output layer. The hidden layers each have 2,000 nodes, and the activation functions are rectified linear units (ReLUs). When we perform the inference stage, in all cases we run stochastic gradient descent with momentum for $20,000$ iterations, which seems to work in most cases. Further hyperparameters are detailed in Appendix A.

4.1 Lorenz System

Here we investigate our method on the Lorenz system:

$$\begin{aligned} \frac{dx}{dt} &= \sigma(y - x) \\ \frac{dy}{dt} &= x(\rho - z) - y \\ \frac{dz}{dt} &= xy - \beta z \end{aligned} \tag{5}$$

which is a system of three ordinary differential equations, with three system parameters: σ, ρ, and β. It was first studied by Edward Lorenz as a simplified model of atmospheric convection, and most notably exhibits chaotic solutions for certain parameter values and initial conditions.

Our training dataset consists of trajectories for parameters in the range:

$$9 \leq \sigma \leq 11, \quad 27 \leq \rho \leq 29, \quad 2 \leq \beta \leq 4,$$

with a discrete grid spacing of 0.2 for each parameter, e.g. $(\sigma, \rho, \beta) = (9.2, 28.6, 3.4)$, or $(\sigma, \rho, \beta) = (10.8, 27.0, 3.2)$. For each parameter, we produce a trajectory to time $t = 1,000$, with $100,000$ time-steps (so $\Delta t = 0.01$). The parameter ranges were chosen to contain the canonical example parameters of the chaotic solution: $(\sigma, \rho, \beta) = (10, 28, 8/3)$, which is plotted in Figure 1.

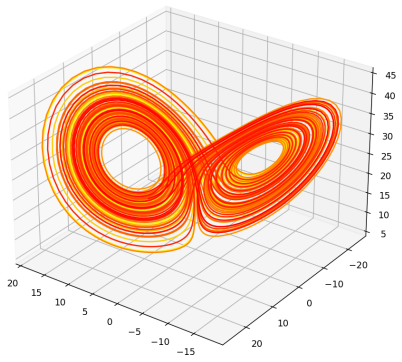

Figure 1 – Canonical example trajectory of the Lorenz system, with $\sigma = 10$, $\beta = 8/3$, and $\rho = 28$.

Our testing dataset consists of 1,000 trajectories for parameters sampled uniformly in the range,

$$9 \leq \sigma \leq 11, \quad 27 \leq \rho \leq 29, \quad 2 \leq \beta \leq 4,$$

Similar to the training set, these trajectories go up to $t = 1,000$ with $100,000$ time-steps. For all trajectories in both training and testing, the initial condition is $\mathbf{x}(0) = (0, 1, 1.05)$.

We also note that within the parameter ranges we have chosen, there is a phase transition where the behaviour moves from chaotic with 2 equilibrium points, to stable with 1 equilibrium point. This demonstrates the ability of our method to deal with such transitions in the data.

We apply our method on two cases: case 1 is utilizing the original, full state-space given in (5), and case 2 is only using the time series given by the x-coordinate in (5), but time-delaying to reconstruct the state-space. Case 2 aligns more with real-world situations, where only time series data is available.

4.1.1 Using the original state space

We examine our method's performance on inferring parameters when using the original state space given in (5). After training on the dataset and evaluating our performance on the test set as

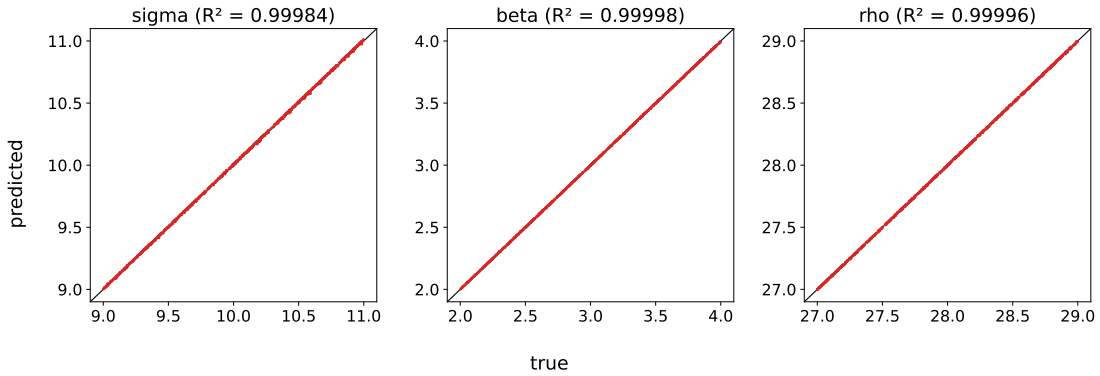

Figure 2 – The inferred parameters are plotted against the true parameters, when the method has access to the original, full state space. The red dots are the model's inferred parameters vs. the true values. For σ we have $R^2 = 0.99984$, for β we have $R^2 = 0.99998$, and for ρ we have $R^2 = 0.99996$. This demonstrates good fit, and that our method can infer the correct parameters from trajectories pretty well.

detailed above in Section 4.1, we compute the R^2 value for σ, β, and ρ in Figure 2, and we plot the model's inferred parameters v.s. the true parameters. We see that for σ we have $R^2 = 0.99984$, for β we have $R^2 = 0.99998$, and for ρ we have $R^2 = 0.99996$. These statistics demonstrate that the model does well in inferring the correct parameters from trajectores.

4.1.2 Using only the time series

Now we examine the performance of our method when we just take the x-coordinate of the Lorenz system in (5), and then delay-embed this to reconstruct the state-space, or more precisely to have a representation of the state-space that is diffeomorphic to the original. In Figure 3, we show an example of a time series and its delay-embedding in 3-dimensions. However, in our experiment, we chose a delay-embedding dimension of 7-dimensions, inline with the guarantees of Takens' theorem.

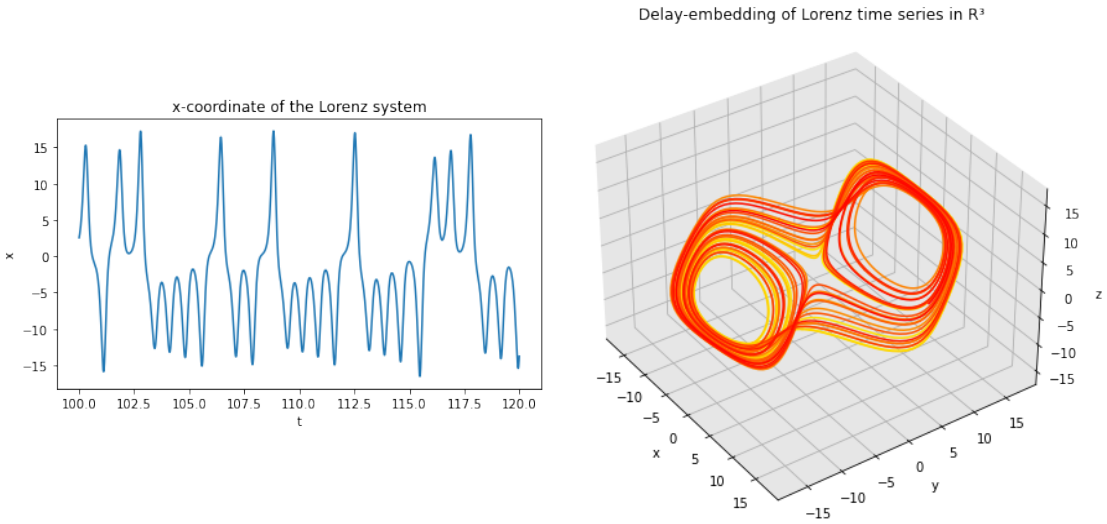

Figure 3 – Left: Time series plot for the trajectory with parameters $(\sigma, \beta, \rho) = (10, 8/3, 28)$. Right: The delay-embedding of the time series on the left into $\mathbb{R}^3$. The time-delay is $t = 0.16 = 16\Delta t$..

In Figure 4, we plot the model's inferred parameters v.s. the true parameters. Here, for σ we

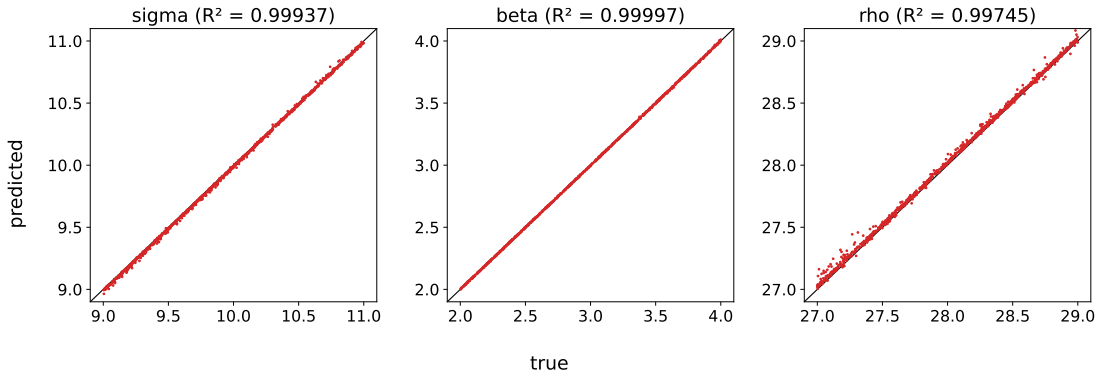

Figure 4 – The inferred parameters are plotted against the true parameters, after delay-embedding the Lorenz time series. The red dots are the model's inferred parameters vs. the true values. For σ we have $R^2 = 0.99937$, for β we have $R^2 = 0.99997$, and for ρ we have $R^2 = 0.99745$. This demonstrates good fit, and that our method can infer the correct parameters from trajectories pretty well.

have $R^2 = 0.99937$, for β we have $R^2 = 0.99997$, and for ρ we have $R^2 = 0.99745$. It seems the ρ parameters does the worse, but overall these statistics demonstrates that our method does pretty well in inferring the true parameters just from *time series* data with parameter labels. As expected though, we perform slightly worse as compared to when we have the original, full state-space. We conjecture this may be that the original system is easier to learn as it is a second-order polynomial of the spatial variables and parameters, whereas the delay-embedding induces a mapping that, while diffeomorphic, may not be as nice as a second-order polynomial. This line of investigation is saved for future work.

4.2 Hall-effect Thruster

In this section, we examine our method on real data: the Hall-effect Thruster (HET), named after its discoverer Edwin Hall, which is a type of ion thruster for spacecraft propulsion, where the propellant is accelerated by an electric field [10].

In our case, the HET data comes as a 2-dimensional list of time series – namely there are 11×31 time series data, each with a million time-steps. This means we have two parameters which we call α and β, with α ranging uniformly from 1 to 3 amperes discretized into 11 values, and β ranging uniformly from 150 to 500 volts, discretized into 31 values. But this depends on the calibration and specifics of the thruster, and to remain calibration agnostic, we normalize the specific values of α and β to:

$$-1 \leq \alpha \leq 1, \quad -3 \leq \beta \leq 3,$$

so this means the 11 values α will take on are $-1, -0.8, -0.6, \ldots, 0.8, 1$, and the 31 values that β will take on are $-3, -2.8, -2.6, \ldots, 2.8, 3$. We also set $\Delta t = 0.001$. In Figure 5 we show an example of a time series, and its delay-embedding in $\mathbb{R}^3$. For our method, we delay-embed the time series in 12-dimensions, with a time-delay of $t = 0.01$. In our case as well, we let the training set and testing set be the same.

Using our method on HET trajectories, which contains real-world noise, we compute the R^2 values for the inferred parameters v.s. the true parameters in Figure 6. We see that for α we have $R^2 = 0.92420$, and for β we have $R^2 = 0.98100$. As expected, we don't do as well compared with Lorenz, but for real, noisy data, our method seems to perform well. Clearly, the R^2 values are heavily being affected by outliers, and when we remove these, of course the R^2 value improves. We leave it to future work to examine how we can prevent outliers.

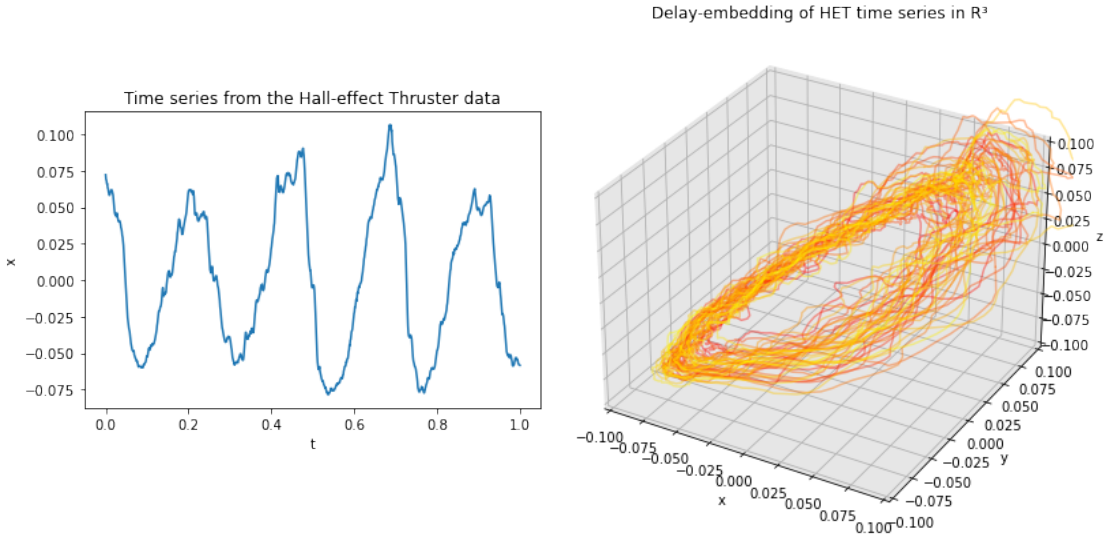

Figure 5 – Left: Time series plot for the HET trajectory with parameters $(\alpha, \beta) = (0, 1)$. Right: The delay-embedding of the time series on the left into $\mathbb{R}^3$. The time-delay is $t = 0.01 = 10\Delta t$.. We note that here we are now dealing with (real) noisy data.

5 Conclusion

In this work, we present a method to infer the dynamical parameters of time series data. We do this by delay-embedding the time series, in order to reconstruct the state space, and then learning the dynamical system operator with a neural network. And because we can compute gradients of neural networks, this allows us to construct a gradient in parameter space, from which we can infer the dynamical parameters. From our experiments on synthetic and real data, we have demonstrated the efficacy of our method, and shown it is a viable approach to inferring parameters of time series.

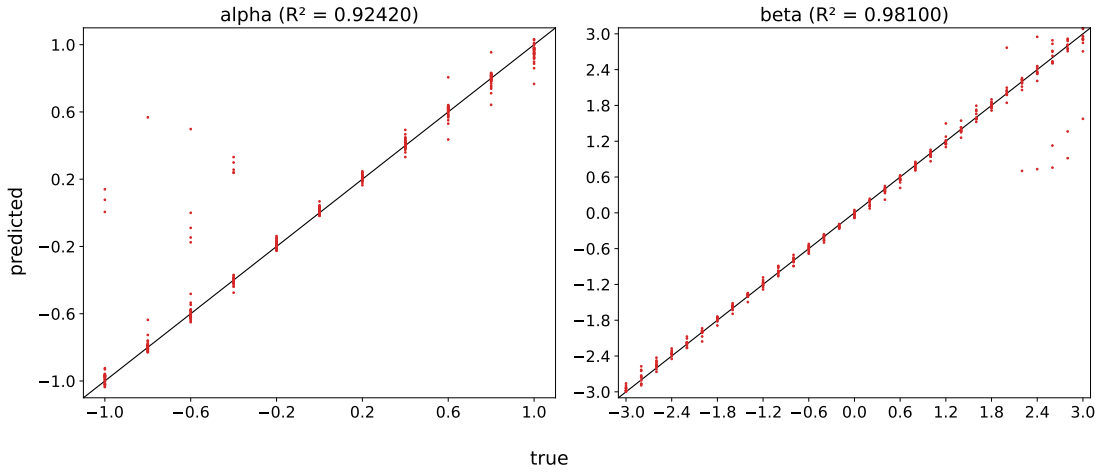

Figure 6 – Inferred parameters are plotted against the true parameters, after delay-embedding the HET time series. The red dots are the model's inferred parameters vs. the true values. For α we have $R^2 = 0.92420$, for β we have $R^2 = 0.98100$. While not as good as the Lorenz data, taking into account this is real, noisy data, our method seems to perform well. We also note there are outliers here, and of course removing them improves the R^2 values.

6 Acknowledgements

Alex Tong Lin and Stanley Osher were supported by Air Force Office of Scientific Research (AFOSR) Multidisciplinary University Research Initiative Grant FA9550-18-1-0502, ONR N00014-18-1-2527, N00014-18-20-1-2093, N00014-20-1-2787, Air Fare AWARD 16-EPA-RQ-09. Dan Eckhardt, Robert Martin, and Adrian Wong were supported by AFOSR LRIR FA9550-20RQCOR098 (PO: Fred Leve).

References

[1] Oreoluwa Alao, Peter Y Lu, and Marin Soljacic. Discovering dynamical parameters by interpreting echo state networks. In *NeurIPS 2021 AI for Science Workshop*, 2021.

[2] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

[3] Liangyue Cao. Practical method for determining the minimum embedding dimension of a scalar time series. *Physica D: Nonlinear Phenomena*, 110(1-2):43–50, 1997.

[4] Kathleen Champion, Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45):22445–22451, 2019.

[5] Kathleen P. Champion, Steven L. Brunton, and J. Nathan Kutz. Discovery of nonlinear multiscale systems: Sampling strategies and embeddings. *SIAM Journal on Applied Dynamical Systems*, 18(1):312–333, 2019.

[6] Earl A Coddington and Norman Levinson. *Theory of ordinary differential equations*. Tata McGraw-Hill Education, 1955.

[7] Filipe de Avila Belbute-Peres, Kevin Smith, Kelsey Allen, Josh Tenenbaum, and J Zico Kolter. End-to-end differentiable physics for learning and control. *Advances in neural information processing systems*, 31, 2018.

[8] C. M. Greve, K. Hara, R. S. Martin, D. Q. Eckhardt, and J. W. Koo. A data-driven approach to model calibration for nonlinear dynamical systems. *Journal of Applied Physics*, 125(24):244901, 2019.

[9] Radek Grzeszczuk, Demetri Terzopoulos, and Geoffrey Hinton. Neuroanimator: Fast neural network emulation and control of physics-based models. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 9–20, 1998.

[10] Richard Hofer and David Jacobson. Development and characterization of high-efficiency, high-specific impulse xenon hall thrusters. *NASA*, 07 2004.

[11] Richard Robert Hofer. *Development and characterization of high-efficiency, high-specific impulse xenon Hall thrusters*. University of Michigan, 2004.

[12] Mason Kamb, Eurika Kaiser, Steven L Brunton, and J Nathan Kutz. Time-delay observables for koopman: Theory and applications. *SIAM Journal on Applied Dynamical Systems*, 19(2):886–917, 2020.

[13] Matthew B Kennel and Henry DI Abarbanel. False neighbors and false strands: A reliable minimum embedding dimension algorithm. *Physical review E*, 66(2):026209, 2002.

[14] Matthew B Kennel, Reggie Brown, and Henry DI Abarbanel. Determining embedding dimension for phase-space reconstruction using a geometrical construction. *Physical review A*, 45(6):3403, 1992.

[15] Edward N. Lorenz. Deterministic nonperiodic flow. *Journal of Atmospheric Sciences*, 20(2):130 – 141, 1963.

[16] Elisa Negrini, Giovanna Citti, and Luca Capogna. A neural network ensemble approach to system identification. *CoRR*, abs/2110.08382, 2021.

[17] Elisa Negrini, Giovanna Citti, and Luca Capogna. System identification through lipschitz regularized deep neural networks. *Journal of Computational Physics*, 444:110549, 2021.

[18] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Multistep neural networks for data-driven discovery of nonlinear dynamical systems. *arXiv preprint arXiv:1801.01236*, 2018.

[19] Samuel Rudy, Alessandro Alla, Steven L. Brunton, and J. Nathan Kutz. Data-driven identification of parametric partial differential equations. *SIAM Journal on Applied Dynamical Systems*, 18(2):643–660, 2019.

[20] Tim Sauer, James A Yorke, and Martin Casdagli. Embedology. *Journal of statistical Physics*, 65(3):579–616, 1991.

[21] Floris Takens. Detecting strange attractors in turbulence. In *Dynamical systems and turbulence, Warwick 1980*, pages 366–381. Springer, 1981.

[22] G. Wallis. Sparrow, c., the lorenz equations: Bifurcations, chaos, and strange attractors. berlin-heidelberg-new york, springer-verlag 1982. xii, 269 s., 91 abb., dm 54,—. us $ 21.60. isbn 3-540-90775-0 (applied mathematical sciences 41). *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, 64(1):71–71, 1984.

[23] Yi-Jen Wang and Chin-Teng Lin. Runge-kutta neural network for identification of dynamical systems in high accuracy. *IEEE Transactions on Neural Networks*, 9(2):294–307, 1998.

[24] Yunan Yang, Levon Nurbekyan, Elisa Negrini, Robert Martin, and Mirjeta Pasha. Optimal transport for parameter identification of chaotic dynamics via invariant measures, 2021.

A Hyperparameters

In all our experiments, we use a feed-forward neural network with an input layer, 3 hidden layers, and an output layer. The 3 hidden layers have 2,000 nodes each. And the activation function is the rectified linear unit (ReLU).

A.1 Lorenz

For the learning phase, the training hyperparameters for our neural network are:

- Learning rate: 10^{-4},
- Batch size: 100
- Error threshold: 0.01.

For the inference stage, we use stochastic gradient descent with momentum, with the following hyperparameters:

- Learning rate: 10^{-3},
- Batch size: 100,
- Momentum: 0.99
- Maximum iterations: $20,000$.

A.2 Hall-effect Thruster

For the learning phase, the training hyperparameters for our neural network are:

- Learning rate: 10^{-5},
- Batch size: 200,
- Max epochs: 2^6.

For the inference stage, we use stochastic gradient descent with momentum, with the following hyperparameters:

- Learning rate: 10^{-1},
- Batch size: 200,
- Momentum: 0.5,
- Maximum iterations; $20,000$.