Learning the dynamics of autonomous nonlinear delay systems

Xunbi A. Ji XUNBIJ@UMICH.EDU
Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI 48109, USA

Gábor Orosz OROSZ@UMICH.EDU
Department of Mechanical Engineering, and Department of Civil Environmental Engineering, University of Michigan, Ann Arbor, MI 48109,USA

Editors: N. Matni, M. Morari, G. J. Pappas

Abstract

In this paper, we focus on learning the time delay and nonlinearity of autonomous dynamical systems using trainable time delay neural networks. We demonstrate that, with delays trained together with weights and biases, the trained neural networks may approximate the right hand side of delay differential equations. It is shown that data collected from the vicinity a stable equilibrium or limit cycle do not contain rich enough dynamics, therefore the trained networks can have very poor generalization. However, including data about the transient behavior can significantly enhance the performance, and similar improvements can be achieved when data collected near a chaotic attractor is utilized. We also evaluate how the learning performance is affected by the selected loss function and measurement noise. Numerical results are presented for learning examples: Mackey-Glass equation and a predator-prey model.

Keywords: trainable time delay neural networks, autonomous nonlinear systems

1. Introduction

Time delay systems widely exist in the physical world and they are usually modelled with delay differential equations which have been be studied extensively using analytical and numerical techniques (Insperger and Stépán, 2011; Fridman, 2014; Breda et al., 2015). Besides using first principles when deriving the equations with explicit time delays, data-driven methods are also a crucial part for obtaining time delay system models. Most of these data-driven methods require some knowledge on the nonlinearity or the time delays of the systems. For instance, the input delay was estimated online together with the parameters of linear system in (Ren et al., 2005). The state delay was identified in (Bayrak and Tatlicioglu, 2013) with knowledge of nonlinearity, while in (Schulze and Unger, 2016) the state delay was estimated from frequency-domain data using a reduced-order model. Multi-model approach with expectation maximization algorithm was applied to identify the time delays from input-output data in discrete time in (Chen et al., 2014). The nonlinearity can be learned with the help of sparse identification of nonlinear dynamics in continuous time when the delay is known; see (Breda et al., 2022).

Meanwhile, neural networks and deep learning algorithms have demonstrated their powerful ability in function approximation, which may greatly benefit the learning of time delay systems from data. On one hand, time delays were introduced in static feed-forward neural networks for speech recognition in (Waibel et al., 1989). This was further extended to adaptive delays which can

be trained together with weights and biases. The corresponding algorithms were typically implemented in discrete time, where the delays were introduced to represent the memory effects with no physical meaning; see (Lin et al., 1992; Yen, 1994; Yazdizadeh and Khorasani, 2002). On the other hand, delays were also incorporated into continuous-time Hopfield networks (Marcus and Westervelt, 1989) for modeling the oscillatory behavior caused by hardware delays. These networks are treated as delay differential equations, and the delays have clear physical meaning. The stability and bifurcations in continuous-time delay networks were studied, e.g., by Campbell et al. (1999) and by Zhang et al. (2014). Also, Ren and Rad (2007) provided an online algorithm for estimating a single input delay in a time delay neural network in continuous time.

Recently, following the successful implementations of the neural ordinary differential equations by Chen et al. (2018), neural differential equations are gaining popularity in the field of deep learning. Neural delay differential equations with a single fixed delay were introduced in (Zhu et al., 2021) for capturing trajectories that intersect in lower dimensional state spaces. These continuous-time neural networks are considered as a type of recurrent neural networks since the network output of the previous time step affects the current network input when performing recursive simulations. These simulations, however, drastically slow down the training, especially when the state contains time delays; see (Zhu et al., 2021; Ji and Orosz, 2022). To accelerate the training process, the direct output of the network, which gives the time derivative of the state, can be used in the loss function as in Ji et al. (2021). Since the neural networks with trainable delays are able to represent the right hand sides of delay differential equations, they are interpretable and can be analyzed using the tools developed for delay differential equations.

In this paper, we study the algorithms orignally proposed in Ji et al. (2021), and apply it to learn the dynamics of nonlinear autonomous delay systems from limited set of trajectories. Different from learning systems with control input, stable autonomous systems converge to steady-state solutions, such us equilibria or limit cycles. We demonstrate that once the data used for training is from the vicinity of a steady state solution, the model trained by such data does not generalize well. We propose to include transients in the data in order to improve the learning performance and demonstrate that similar improvements can be observed when utilizing data from chaotic motion. We investigate the learning performance while considering different loss function and examine the robustness of the algorithms under measurement noise. In Section 2, we briefly introduce the learning algorithm with time delay neural networks and define a group of loss functions based on state derivative. In Section 3, we summarize the existing challenges in learning the autonomous time delay systems. Examples of learning the dynamics of the Mackey-Glass equation and a predator-prey model are presented in Section 4. Conclusions are drawn and future work is laid out in Section 5.

2. Learning algorithm with neural networks

Autonomous time delay systems can be formulated as

$$\dot{x}(t) = \mathcal{G}(x_t), \tag{1}$$

where $x \in \mathbb{R}^n$ and $x_t \in C([-\tau_{\max}, 0], \mathbb{R}^n)$ represents the space of continuous functions. Namely, we define $x_t(\vartheta) := x(t+\vartheta), \vartheta \in [-\tau_{\max}, 0]$ where $\tau_{\max} > 0$ is the maximum delay. The functional $\mathcal{G}$ maps the Banach space $C([-\tau_{\max}, 0], \mathbb{R}^n)$ to $\mathbb{R}^n$. For systems with finite number of delays $0 \leq \tau_1, \tau_2, \ldots, \tau_d \leq \tau_{\max}$, (1) can be written as

$$\dot{x}(t) = \mathbf{g}(\mathbf{x}_t), \tag{2}$$

where

$$\mathbf{x}_t = \begin{bmatrix} x(t-\tau_1) \\ \vdots \\ x(t-\tau_d) \end{bmatrix} \in \mathbb{R}^{nd}, \tag{3}$$

and $\mathbf{g}$ is a nonlinear function such that $\mathbf{g}\colon \mathbb{R}^{nd} \to \mathbb{R}^n$. The particular form of $\mathbf{g}$ and the values of the delays $\tau_1, \tau_2, \ldots, \tau_d$ may not be known and the task is then to obtain the approximate system

$$\hat{\dot{x}}(t) = \mathrm{net}(\hat{\mathbf{x}}_t), \tag{4}$$

via learning from the trajectories $x(t)$. Here net represent the nonlinear map between the input and output of a neural network. This way the neural network represents the right hand side of a delay differential equation. Below we specify how such networks can be constructed.

Trainable time delay neural networks: Consider a time delay neural network described by

$$\begin{aligned} z_1^t &= f(W_1 \mathbf{z}_0^t + b_1), \\ z_l^t &= f_l\big(W_l z_{l-1}^t + b_l\big), \quad l \geq 2, \ldots, L, \end{aligned} \tag{5}$$

where $\mathbf{z}_l^t$ denotes the output of layer l at time t. The input of the network $\mathbf{z}_0^t$ consists of the delayed values of the signal z_0^t collected in the vector

$$\mathbf{z}_0^t = \begin{bmatrix} z_0^{t-\tau_1} \\ \vdots \\ z_0^{t-\tau_d} \end{bmatrix}, \tag{6}$$

with input delays $0 \leq \tau_1, \cdots, \tau_d \leq \tau_{\max}$. That is when we use the time delay neural network to learn the dynamics of (2), we use $\mathbf{z}_0^t = \mathbf{x}_t$; cf. (3) and (6). However, assuming that trajectory data is available at time moments $t = i\Delta t$, $i \in \mathbb{Z}$, and that delays evolve continuously during the learning process, $t - \tau$ may not fall on one of the mesh points. By defining $\alpha \in [0, 1)$ such that $\tau = (j + \alpha)\Delta t$, the delayed state can be expressed using linear interpolation:

$$x(t-\tau) = x(t-(j+\alpha)\Delta t) \approx (1-\alpha)x(t-j\Delta t) + \alpha x(t-(j+1)\Delta t). \tag{7}$$

Then, as first proposed in Ji et al. (2020), the delays can be trained together with the weights and biases via back-propagation. After the network is trained, we can use the delay differential equation (4) for analysis and numerical simulation. Note that to obtain $\hat{x}(t)$ with high accuracy simulations may be performed with much smaller time steps than Δt and with any preferred numerical integration scheme

$$\hat{x}(t) = \mathrm{DDEsolver}(\mathrm{net}, x_0). \tag{8}$$

Design of loss function: Loss functions can be constructed using the distance between the time derivative predicted by the neural network and the time derivative given by the data with the help of vector norms. If $\dot{x}(t)$ is not measured, it can be obtained via numerical differentiation, e.g., Euler's method or central difference method at the mesh points, that is, $\dot{x}(i\Delta t)$ is derived from $x(i\Delta t)$, $x((i+1)\Delta t)$, and $x((i-1)\Delta t)$. At the same time $\hat{\dot{x}}(i\Delta t) = \mathrm{net}(\mathbf{x}_{i\Delta t})$ is a direct prediction given by the neural network. Then one may construct the loss function

$$\mathcal{L} = \frac{1}{N} \sum_{i \in \mathrm{batch}} \big\| \hat{\dot{x}}(i\Delta t) - \dot{x}(i\Delta t) \big\|, \tag{9}$$

where N is the number of output data used for updating within one batch and i marks the location of the data point along the trajectory. The norm $\|\cdot\|$ can be chosen as any distance measure. In this paper, we examine the different distance measures and assign weights for each data point depending on its location along the trajectory. In other words, we consider a weighted loss function

$$\mathcal{L}_p = \frac{1}{nN} \sum_{i \in \text{batch}} \sum_{j=1}^{n} w_i \big(\hat{\dot{x}}_j(i\Delta t) - \dot{x}_j(i\Delta t)\big)^p. \tag{10}$$

In particular, we compare results for the average loss with $w_i = 1$ and the weighted loss with $w_i = \frac{1}{i\Delta t}$ while using $p = 1, 2$.

One can also use the distance between the data $x(t)$ and the neural network simulation and define the loss

$$\mathcal{L} = \frac{1}{H} \sum_{i=1}^{H} \big\|\hat{x}(i\Delta t) - x(i\Delta t)\big\|, \tag{11}$$

where H is the simulation horizon from each given initial history x_0. In this paper, we train the networks using the derivative loss (10), and highlight that small derivative loss (10) does not always lead to a small simulation loss (11). Compared to the work in Ji and Orosz (2022) where the simulation loss is used for training, the usage of only the derivative loss has much less computation and serves as an alternative. We also thoroughly discuss the limitations of this alternative method and show how the data quality (including the richness of the dynamics and measurement noise) affects the learning performance. An example of learning two-state time delay system further illustrates the potential of this algorithm.

Backpropagation with respect to time delays: In the training algorithms, the gradient of the loss function with respect to the parameters are necessary for the iterative update. The gradients of the loss (10) with respect to the weights and biases, i.e., $\frac{\partial \mathcal{L}}{\partial W_l}$ and $\frac{\partial \mathcal{L}}{\partial b_l}$, can be calculated via backpropagation. The gradient of the loss with respect to the time delay at time t is given by

$$\frac{\partial \mathcal{L}}{\partial \tau_k} = \frac{\partial \mathcal{L}}{\partial x(t-\tau_k)} \frac{\partial x(t-\tau_k)}{\partial \tau_k} = \frac{\partial \mathcal{L}}{\partial x(t-\tau_k)} \big(-\dot{x}(t-\tau_k)\big), \tag{12}$$

where $\frac{\partial \mathcal{L}}{\partial x(t-\tau_k)}$ is the gradient with respect to the delayed state and $\dot{x}(t-\tau_k)$ is the time derivative of x at $t-\tau_k$. As $x(t)$ is only available at the discrete time steps $t = i\Delta t$, $i \in \mathbb{Z}$ then $t - \tau$ does not necessarily coincide with a mesh point. Again using $\tau = (j+\alpha)\Delta t$ with $\alpha \in [0, 1)$ we obtain

$$\dot{x}(t-\tau) = \dot{x}(t-(j+\alpha)\Delta t) \approx \frac{x(t-j\Delta t) - x(t-(j+\alpha)\Delta t)}{\alpha \Delta t} \approx \frac{x(t-j\Delta t) - x(t-(j+1)\Delta t)}{\Delta t}, \tag{13}$$

where in the first approximation we used Euler's method while in the second one we utilized the linear interpolation (7). In this work, we use the MATLAB nnet toolbox to calculate the gradients. Once the gradients are obtained, one can choose any gradient-based methods for updating the parameters in the network. In this paper, we use adaptive moment estimation; see (Kingma and Ba, 2015). The learning rates used for the weights and biases and those used for time delay parameter can be different. Here, these are tuned manually while observing the training loss in multiple epochs.

3. Existing challenges in learning

Data quality and system complexity: Learning the autonomous delay systems that converge to a stable equilibria or stable limit cycles may present fundamental challenges due to the lack of dynamics in data. This may be especially difficult when the equilibrium or the limit cycle is strongly attractive, that is, when the leading eigenvalue of the infinitesimal generator has small real part or when the leading eigenvalue of the monodromy operator is small in magnitude; see (Insperger and Stépán, 2011). In such cases, the steady state solution occupies a large portion of the data and we say that the data is of "low quality" (even if it does not contain noise). If the transient part does not have enough portion in the training data, it is often treated as an "outlier" by the training algorithms, therefore the trained network does not generalize well. Incorporating more transient data can be achieved by setting the initial conditions further away from the attractors. We remark that autonomous systems with high behavior complexity typically generate "high quality" data for learning, even without transients. For instance, models learned from chaotic attractors usually generalize well (Lin et al., 1992; Ji and Orosz, 2022; Goldmann et al., 2022).

While higher behavior complexity facilitates better learning, increasing the system complexity typically makes learning more difficult. For example, a scalar system is easier to learn compared to a system with vector valued states as in the latter case the learning often gets stuck in local minima. Increasing the complexity of the networks may help one to avoid such undesired behavior.

Derivative vs. simulation loss: As mentioned in (Ji et al., 2021), there is a fundamental difference between using the derivative loss (9) and the simulation loss (11). Once we train the network using the direct prediction of derivatives, the inputs to the network are taken from the data, which are always accurate, i.e., we use $\mathrm{net}(\mathbf{x}_t)$ instead of $\mathrm{net}(\hat{\mathbf{x}}_t)$ in (4). Contrarily, once simulating the network, the inputs are obtained from previous simulation steps while generating the solution of (4) from the initial history x_0; see (8). Thus, the prediction can deviate from the training data and may enter regions in the state space where the network is untrained. One example of inconsistency between derivative and simulation loss is when the network learns numerical differentiation instead of the right hand side. In this case, the network gives small derivative loss with a small delay but indeed it fails in terms of simulation. Such behavior was also observed in (Goldmann et al., 2022), where a one-step-ahead prediction loss, which is equivalent to the derivative loss, was considered, and the loss function had a local minima at a very small delay value.

Robustness to measurement noise: One may observe from the gradient calculation (12) and the loss function (9) that the algorithms rely heavily on the state derivatives. If the states are measured with added noise, the time derivatives obtained via numerical differentiation and the state estimates obtained via linear interpolation are largely affected by the noise and the time step used. Smaller time step Δt leads to larger noise in the derivative, while larger time step leads error in the differentiation. For example, consider that white noise with zero mean and standard deviation $\sigma_x = 0.02$ is added to x through the measurements, then the time derivatives obtained from Euler's methods with $\Delta t = 0.05$ results in additive noise with zero mean and standard deviation $\sigma_{\dot{x}} = \frac{\sqrt{2}\sigma_x}{\Delta t} = 0.57$ for $\dot{x}$. If the central difference method is used to obtain the derivative, the standard deviation of the noise added to $\dot{x}$ reduces to 0.28. Thus, choosing a robust numerical differentiation method and applying appropriate filtering are critical for calculating the time derivatives accurately, which is an important research topic on its own; see (Fioretti and Jetto, 1989; Van Breugel et al., 2020).

4. Learning Results

Mackey-Glass equation: First, we examine the algorithm to learn the dynamics of the Mackey-Glass equation (Mackey and Glass, 1977). This scalar autonomous time delay system is given as

$$\dot{x}(t) = \frac{\beta x(t-\tau)}{1+(x(t-\tau))^{\delta}} - \gamma x(t), \tag{14}$$

and we use $\beta = 4, \gamma = 2, \delta = 9.65$. For delay $\tau = 0.5$, the system approaches a stable limit cycle (of period $T_p \approx 1.57$), while for $\tau = 1$, the system exhibits chaotic behaviour. For both τ values we use MATLAB dde23 to generate 10 trajectories using constant initial histories $x(t) \equiv c$, $t \in [-\tau_{\max}, 0]$ where $c = 0.5 + i/9, i = 0, \ldots, 9$. To demonstrate the influence of the data quality, we train the neural network with trajectories of the same length while using different sections from the simulations. As shown in Fig. 1, we consider two types of training data sets, one discards the transients and the other includes the initial constant history. The length of all training trajectories is set to be $T_{\rm tr} = 7$ (on the left of the solid line) and the testing part is the following data segment of length $T_{\rm ts} = 3$. Note that, the length of the trajectory does not affect the training performance as long as it is larger than the maximum length of the history $T_{\max}$ and the trajectories cover enough area of the state space. In other words, the dynamics of the trajectories is more important than the length of the trajectories. The sampling time step is $\Delta t = 0.05$. We set $\tau_{\max} = 1$ for the limit cycle data and $\tau_{\max} = 2$ for chaotic data. The data on the left of dashed line is only used as initial history and the output data is obtained from the right of this line.

We use a two-hidden-layer network with 5 neurons with $\tanh$ type of nonlinearity in each layer to learn the Mackey-Glass dynamics. The input vector to the network is constructed using states with one fixed zero delay and one trainable delay. We first conduct massive training with different data sets and loss functions. For each combination of training data set and loss function, we run the training process 100 times from different initial parameters (with maximum iteration 2000 in each epoch, batch size $N = 10$, learning rate $\eta_\theta = 0.1$ for weights and biases and learning rate $\eta_\tau = 0.001$ for the delay). The trainable delay τ is initialized uniformly between $[0, \tau_{\max}]$, biases are initialized as zeros, weights are initialized using the Glorot initialization (Glorot and Bengio, 2010).

The final learned delay corresponding to the minimal training loss is plotted in Fig. 2. The blue histograms represent the probability of the delay converging to a certain value and the orange dots mark the learned delay and the corresponding training loss. Besides the minima around the ground truth delay, we observe some other local minima that the networks may converge to. Observe that including the history and using weighted loss can help to reduce the chance of converging to undesired local minima.

For limit cycle data, the average $\mathcal{L}_2$ loss performs significantly better than the average $\mathcal{L}_1$ loss when the initial history is included; cf. panels (b) and (c). The reason behind this is that the $\mathcal{L}_1$ loss tends to ignore "outliers" in the data, while here the "outliers" are the transients which are the most important parts in terms of learning the dynamics. The network generalizes well outside the training data set only if these transients are taken into account. This also explains why the weighted loss performs better: it assigns more weights to the transient part of the data. We can also see that the chaotic data has better delay learning performance compared to the limit cycle data since the training data covers a larger part of the state space. With the initial history included and using weighted $\mathcal{L}_2$ loss, both training data sets lead to a very high success ratio of learning the correct delay; see panels (d) and (h).

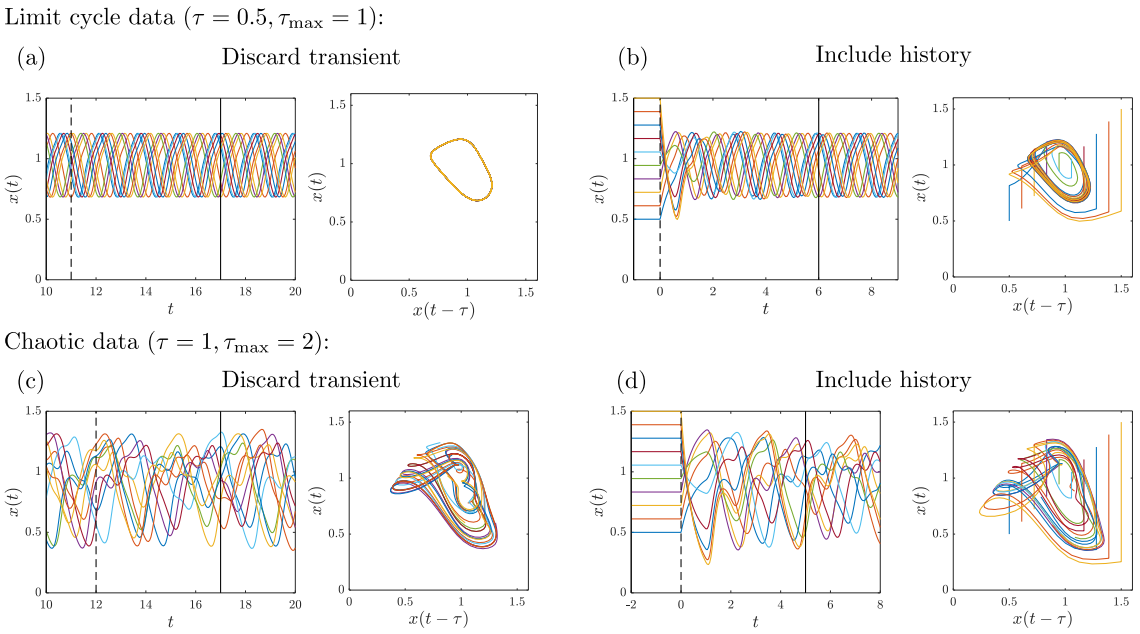

Figure 1: Training and testing data for learning Mackey-Glass equation. Top row: simulation data generated for $\tau = 0.5$; bottom row: simulation data generated for $\tau = 1$. The solid line divides the data into training and testing and the data on the left of the dashed line is given as initial history (whose length corresponds to the maximum allowed delay $\tau_{\max}$). We consider different segments of data for training: (a), (c) are the segments without the initial transients, starting from $t = 10$; (b), (d) are the segments with initial constant history, starting from $t = -\tau_{\max}$.

In addition to checking the learning of the correct delay, we examine the network predictions on training and testing data. The first row of Fig. 3 shows two epochs of training from same initial delay value 0.8. In panel (a) one run (purple) converges to $\tau = 1$ while the other (green) converges to $\tau = 0.45$. Limit cycle data (generated with $\tau = 0.5$) without transients are used for training and the average $\mathcal{L}_1$ loss is chosen. Since the limit cycle data is repeating itself, there is no difference whether we use average vs. weighted loss and whether we use $\mathcal{L}_1$ vs. $\mathcal{L}_2$ loss. Even though the derivatives fit very well on the training set for both runs in panel (b), the simulation of the network with delay $\tau = 1$ (purple) diverges in panel (c). This demonstrates that small derivative loss does not guarantee small simulation loss. During the simulation small derivative errors accumulate and the trajectory may leave the region of training data. In the region where the network was not trained, larger errors of the derivative prediction can result in unstable behavior. In panel (d) the networks are tested while setting the time delay $\tau = 0.5$ for a trajectory starting from a constant initial history $x(t) \equiv 0.5$. The network which learned $\tau = 0.45$ gives good predictions along the limit cycle, but it fails to capture the transients. This shows that the network trained only on data from the vicinity of the attractor generalizes poorly.

For comparison, the result for a network trained on chaotic data (generated with $\tau = 1$) including the initial history and using the weighted $\mathcal{L}_2$ loss is shown in the second row of Fig. 3. Panel (e) demonstrates that the correct delay $\tau = 1$ is learned in this case. This results in good performance

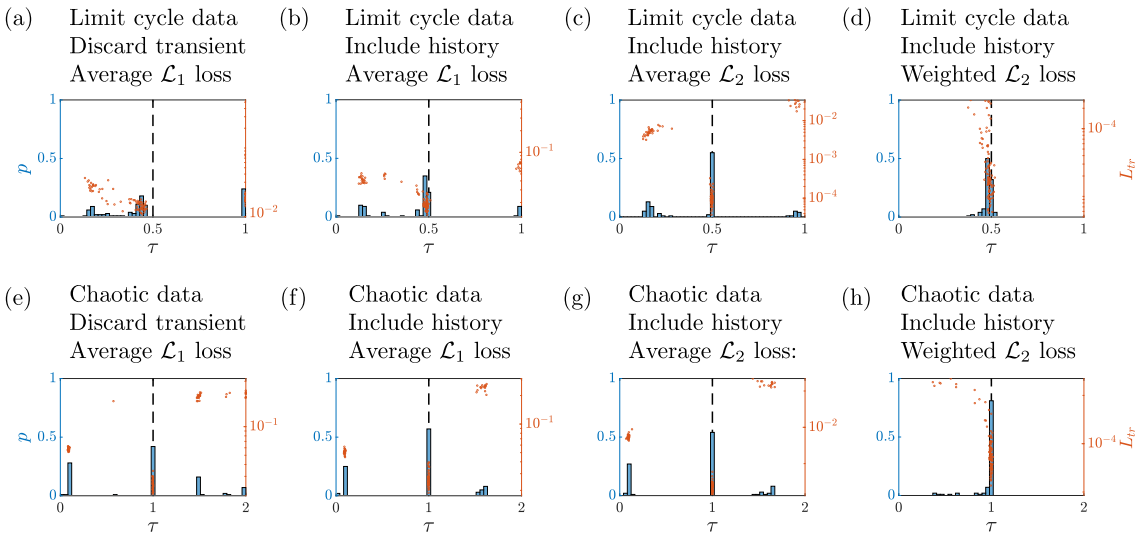

Figure 2: Convergence of 100 training epochs from initial delay values between $[0, \tau_{\max}]$. The left axis shows the (measured) probability of the learned time delay and the right axis shows the corresponding training loss. The dashed line indicates the ground truth time delay.

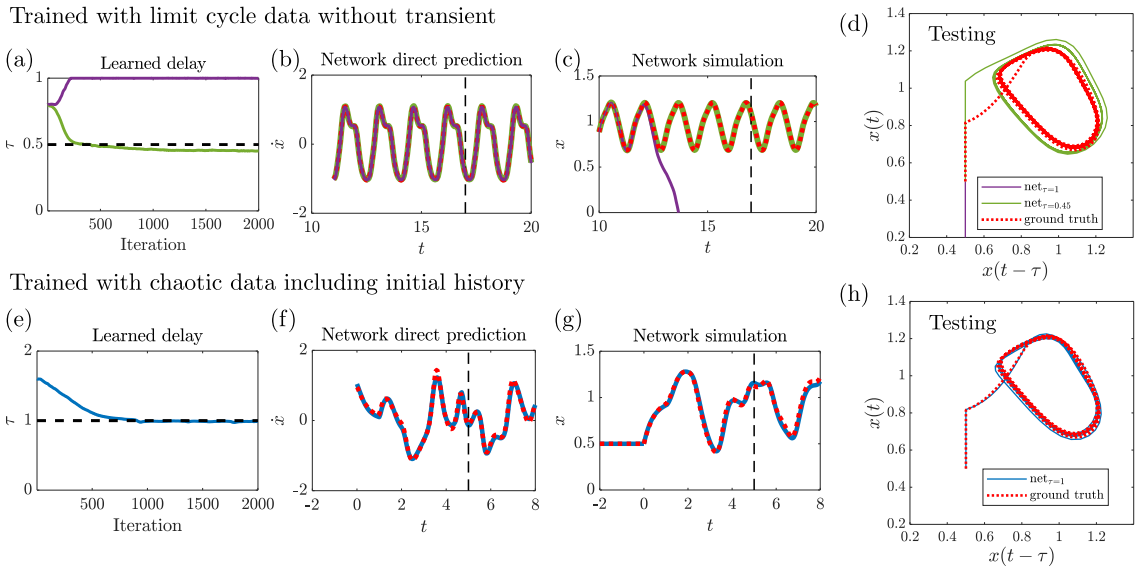

Figure 3: The performance of networks trained with different data. (a)-(d): using limit cycle data and average $\mathcal{L}_1$ loss; (e)-(h): using chaotic data and weighted $\mathcal{L}_2$ loss. (d) and (h): the trained networks are tested using the initial history $x(t) \equiv 0.5$ and given delay $\tau = 0.5$.

both for derivative and simulation as shown in panels (f) and (g). Moreover the network is also able to give good predictions when the delay is set to $\tau = 0.5$. This shows the independence of the learned delay and nonlinearity, and highlights the generalizability of the network when it is trained on high-quality chaotic data.

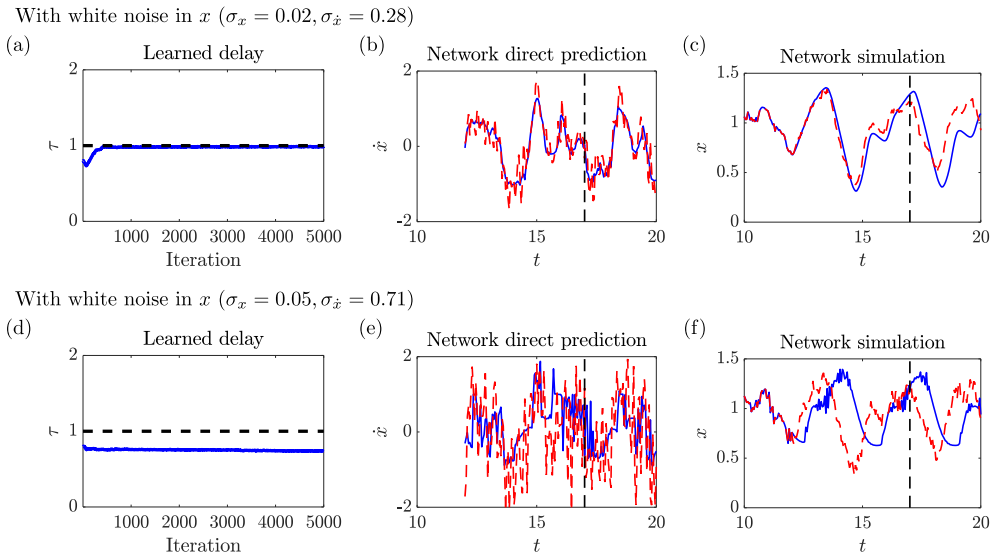

Figure 4: Training performance under two different levels of white noise. Both networks have the same initial delay 0.8 and are trained on chaotic data using weighted $\mathcal{L}_2$ loss.

We also present the learning performance under different levels of measurement noise in Fig. 4. As shown in the top row, the delay is still learned when small noise is added to the data, however, the noise does affect the learning of the nonlinearity as shown by the decreased accuracy of prediction in panels (b) and (c) . The bottom row demonstrates that for stronger noise, the state derivatives fail to guide the learning of the delay. One may try to reduce the noise by pre-processing the data but this is beyond the scope of this paper.

Predator-prey model: Here we learn the dynamics of a two-state autonomous nonlinear delay system, which is used to model the dynamics of a predator-prey dynamics

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} rx_1(t) - sx_1^2(t) - px_1(t)x_2(t) \\ bpx_1(t-\tau)x_2(t-\tau) - dx_2(t) - mx_2^2(t) \end{bmatrix}, \tag{15}$$

where the time delay τ indicates the maturation time of the predators. Simulation trajectories are generated with parameters $r = 1, a = 1, p = 4, b = 1.2, m = 0.1, d = 1, \tau = 2$ and constant history $x(t) \equiv 0.1, 0.2, 0.3, 0.4$. The training data set contains trajectories of length 34 ($t \in [-\tau_{\max}, 30]$) with time step $\Delta t = 0.1$. A two-hidden-layer network with $\tanh$ type of nonlinearity, one fixed zero delay and one trainable delay is used considering $\tau_{\max} = 4$. Training is performed with maximum iteration 5000, batch size $N = 50$, and learning rates $\eta_\theta = 0.01, \eta_\tau = 0.001$.

Figure 5 shows one training result with no noise added to the data while using the weighted $\mathcal{L}_1$ loss. This system is not sensitive to the choice of loss measurement, i.e., both $\mathcal{L}_1$ and $\mathcal{L}_2$ loss would work, since the initial part is not too much of an outlier compared to the rest of the trajectory. Although the performance is similar, $\mathcal{L}_1$ loss does provide faster convergence in this example. The initial delay is set to 3 and it converges to the correct value of $\tau = 2$ during learning. The trained network gives good predictions on the state derivative, however, the simulation of the network on the last trajectory is not so accurate. This again shows inconsistency between direct prediction and the simulation prediction. We remark that part of the last trajectory is very close to the boundary of training data set and the portion of training data at that part of the state space is small.

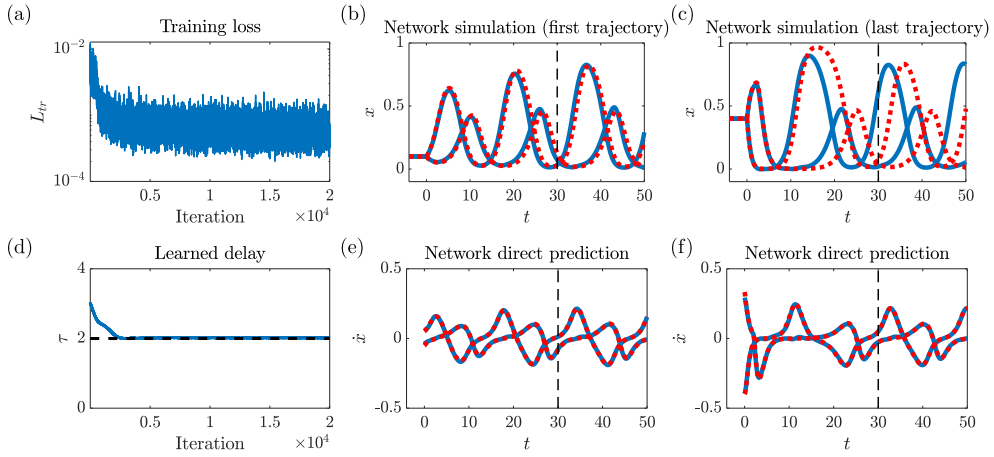

Figure 5: Training performance of the predator-prey model.

To show the robustness of the delay learning algorithm againts noise, we conduct another 100 training runs from different initial weights and biases but same initial delay 3 for different levels of noise added. We define the success ratio of learning the delay as the number of learned delay falling in the interval 2 ± 0.15 divided by the number of conducted runs. We observe from the Fig. 6, the success ratio of learned delay drops sharply as the noise level exceeds some limit.

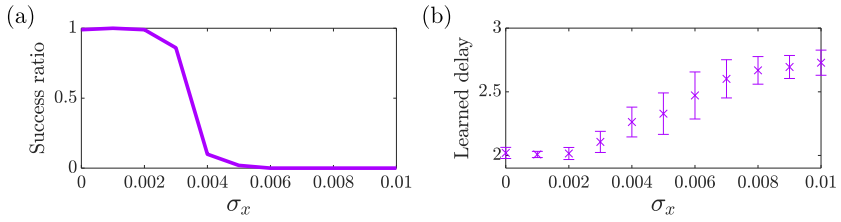

Figure 6: Evaluation of learning the delay from noisy data. (a) success ratio of learning the delay for different levels of noise; (b) distribution of 100 learned delays. All the training runs are initialized with different weights and biases but with the same initial delay 3.

5. Conclusion and future direction

We presented tools and discussed some fundamental challenges for learning autonomous nonlinear delay systems. Without external excitation, it is difficult to obtain data with rich dynamics, therefore the trained network may generalize poorly. Weighing the initial transient data or choosing the mean squared error instead of the absolute error can help to learn the nonlinearity and the time delays. Using direct derivative output in the loss function does not guarantee small error in the network simulation and the learning performance may be sensitive to measurement noise. As a potential solution, one may use the simulation error for validation, or consider the network simulation directly in the training loss function. Introducing external inputs to generate trajectories may also help in learning the dynamics of time delay systems. The state of the delay systems learned in this paper were fully observed, and time delay in the states are constant and discrete. Future studies include partially observed system/input-output data and learning time-varying or state dependent delays.

References

Alper Bayrak and Enver Tatlicioglu. Online time delay identification and control for general classes of nonlinear systems. *Transactions of the Institute of Measurement and Control*, 35(6):808–823, 2013.

Dimitri Breda, Stefano Maset, and Rossana Vermiglio. *Stability of Linear Delay Differential Equations*. Springer, 2015.

Dimitri Breda, Nicola Demo, Alessandro Pecile, and Gianluigi Rozza. Data-driven methods for delay differential equations. Oral Presentation at the 17th IFAC Workshop on Time Delay Systems, 2022.

Sue Ann Campbell, S Ruan, G Wolkowicz, and J Wu. Stability and bifurcation of a simple neural network with multiple time delays. *Fields Institute Communications*, 21(4):65–79, 1999.

Lei Chen, Biao Huang, and Fei Liu. Multi-model approach to nonlinear system identification with unknown time delay. *IFAC Proceedings Volumes*, 47(3):9388–9393, 2014.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *Advances in Neural Information Processing Systems*, 2018.

S Fioretti and L Jetto. Accurate derivative estimation from noisy data: a state-space approach. *International Journal of Systems Science*, 20(1):33–53, 1989.

Emilia Fridman. *Introduction to Time-Delay Systems*. Birkhäuser, 2014.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

Mirko Goldmann, Claudio R. Mirasso, Ingo Fischer, and Miguel C. Soriano. Learn one size to infer all: Exploiting translational symmetries in delay-dynamical and spatiotemporal systems using scalable neural networks. *Physical Review E*, 106:044211, 2022.

Tamás Insperger and Gábor Stépán. *Semi-Discretization for Time-Delay Systems*. Springer, 2011.

Xunbi A. Ji and Gábor Orosz. Learning time delay systems with neural ordinary differential equations. In *Proceedings of the 17th IFAC Workshop on Time Delay Systems*, 2022.

Xunbi A. Ji, Tamás G. Molnár, Sergei S. Avedisov, and Gábor Orosz. Feed-forward neural networks with trainable delay. In *Proceedings of the 2rd Conference on Learning for Dynamics and Control*, volume 120, pages 127–136. PMLR, 2020.

Xunbi A. Ji, Tamás G. Molnár, Sergei S. Avedisov, and Gábor Orosz. Learning the dynamics of time delay systems with trainable delays. In *Proceedings of the 3rd Conference on Learning for Dynamics and Control*, volume 144, pages 930–942. PMLR, 2021.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *Proceedings of the 3rd International Conference on Learning Representations*, 2015.

Daw-Tung Lin, Judith E Dayhoff, and Panos A Ligomenides. Adaptive time-delay neural network for temporal correlation and prediction. In *Intelligent Robots and Computer Vision XI: Biological, Neural Net, and 3D Methods*, volume 1826, pages 170–181. International Society for Optics and Photonics, 1992.

Michael C Mackey and Leon Glass. Oscillation and chaos in physiological control systems. *Science*, 197(4300):287–289, 1977.

CM Marcus and RM Westervelt. Stability of analog neural networks with delay. *Physical Review A*, 39(1):347, 1989.

XM Ren, Ahmad B Rad, PT Chan, and Wai Lun Lo. Online identification of continuous-time systems with unknown time delay. *IEEE Transactions on Automatic Control*, 50(9):1418–1422, 2005.

Xue-Mei Ren and Ahmad B Rad. Identification of nonlinear systems with unknown time delay based on time-delay neural networks. *IEEE Transactions on Neural Networks*, 18(5):1536–1541, 2007.

Philipp Schulze and Benjamin Unger. Data-driven interpolation of dynamical systems with delay. *Systems & Control Letters*, 97:125–131, 2016.

Floris Van Breugel, J. Nathan Kutz, and Bingni W. Brunton. Numerical differentiation of noisy data: A unifying multi-objective optimization framework. *IEEE Access*, 8:196865–196877, 2020.

Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):328–339, 1989.

Alireza Yazdizadeh and Khashayar Khorasani. Adaptive time delay neural network structures for nonlinear system identification. *Neurocomputing*, 47(1-4):207–240, 2002.

Gary G. Yen. Adaptive time-delay neural control in space structural platforms. In *Proceedings of the IEEE International Conference on Neural Networks (ICNN'94)*, volume 4, pages 2622–2627. IEEE, 1994.

Huaguang Zhang, Zhanshan Wang, and Derong Liu. A comprehensive review of stability analysis of continuous-time recurrent neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 25(7):1229–1262, 2014.

Qunxi Zhu, Yao Guo, and Wei Lin. Neural delay differential equations. In *International Conference on Learning Representations*, 2021.