Exploiting the Intrinsic Dynamics of a Driven Pendulum for Machine Learning

Rishi Vora
MS21113

IISER Mohali

2024-11-16

Reservoir computing (RC) is an umbrella term for a number of different machine learning techniques that use dynamical systems[1], also referred to as "reservoir". Usually, the system's rich transient (i.e. short term) dynamics are used to perform temporal (time-dependent) computations. However, it can also be applied to spatial (time-independent) tasks.

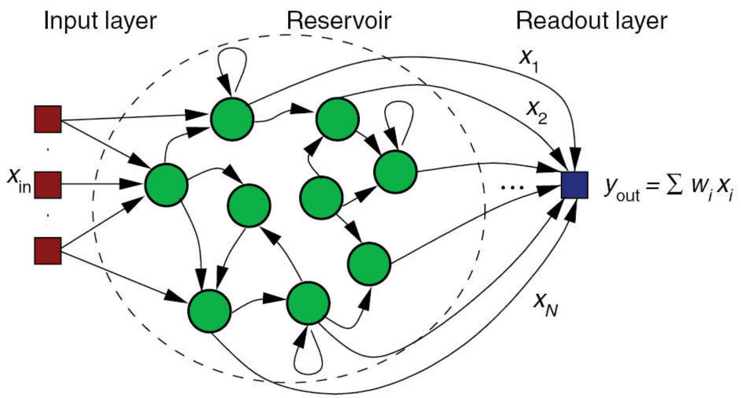

Figure 1: Schematic of a typical RC [1]

[1] A *dynamical system* is a rule for time evolution on a state space.

1.1. How can we use it?

Particularly, we look at the driven pendulum as a dynamical system. Moreover, we will see how it:

- performs in temporal tasks
- performs in spatial tasks
- performs under noise

2.1. The system

Consider a pendulum with length l, the bob of mass m, periodically driven by a force of amplitude F, in an environment with damping coefficient b.

The equation of motion is given by:

$$\frac{\mathrm{d}^2 x}{\mathrm{d}t^2} = -\frac{g}{l}\sin(x) - k\frac{\mathrm{d}x}{\mathrm{d}t} + f\,\mathrm{sgn}(\sin(\omega t))$$

where $f = \frac{F}{m}, k = \frac{b}{m}$.

2.1. The system

Here are some numerical simulations by varying the parameters,

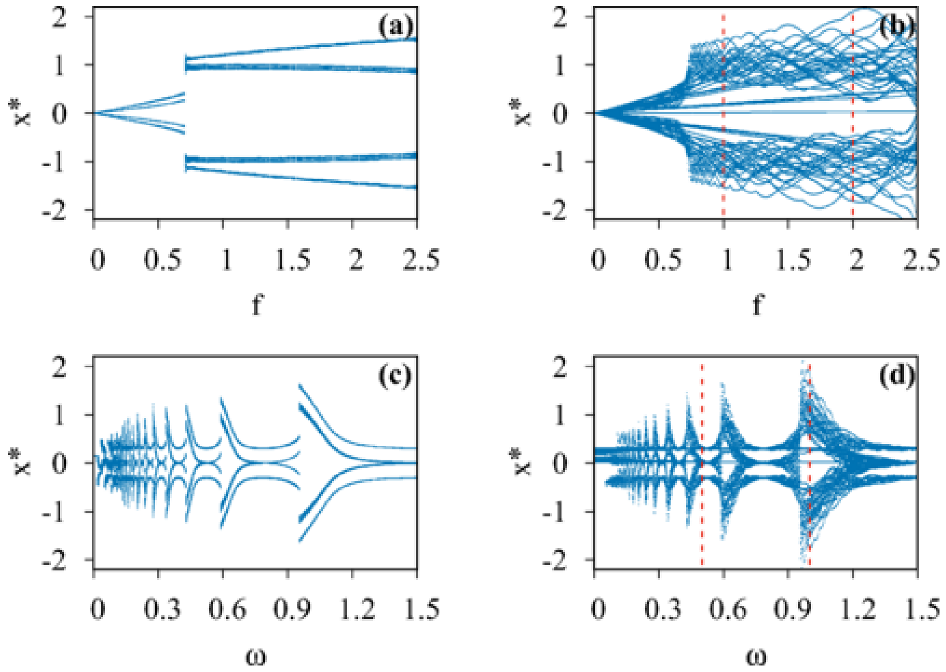

Figure 2: Bifurcation diagram of the driven pendulum [2]. (a,b) have fixed $\omega = 1.0$ while (c,d) have fixed $f = 1.5$. (a,c) show the asymptotic behaviour of the system while (b,d) show the transient dynamics.

2.1. The system

Here are some numerical simulations by varying the parameters,

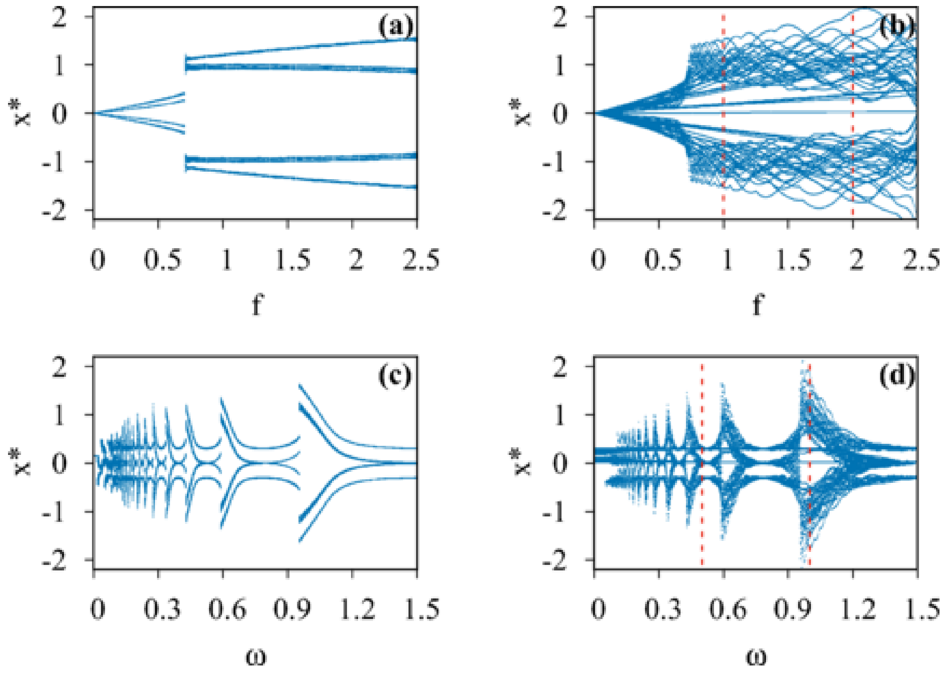

Figure 2: Bifurcation diagram of the driven pendulum [2]. (a,b) have fixed $\omega = 1.0$ while (c,d) have fixed $f = 1.5$. (a,c) show the asymptotic behaviour of the system while (b,d) show the transient dynamics.

The transient dynamics look rich! We can use this to encode our data.

3.1. Choice of encoding

Encoding options we have:

3.1. Choice of encoding

Encoding options we have:

1. Encode with initial condition

3.1. Choice of encoding

Encoding options we have:

1. Encode with initial condition
    - Different trajectories can evolve similarly (not a one-to-one mapping)
    - Leads to input information loss, hence inefficient

3.1. Choice of encoding

Encoding options we have:

1. Encode with initial condition
    - Different trajectories can evolve similarly (not a one-to-one mapping)
    - Leads to input information loss, hence inefficient
2. Encode with system parameters

3.1. Choice of encoding

Encoding options we have:

1. Encode with initial condition
   - Different trajectories can evolve similarly (not a one-to-one mapping)
   - Leads to input information loss, hence inefficient
2. Encode with system parameters
   - Use amplitude (f) and frequency (ω)
   - Each point in the $f - \omega$ parameter space corresponds to a unique trajectory, hence more efficient

3.1. Choice of encoding

Encoding options we have:

1. Encode with initial condition
   - Different trajectories can evolve similarly (not a one-to-one mapping)
   - Leads to input information loss, hence inefficient
2. Encode with system parameters
   - Use amplitude (f) and frequency (ω)
   - Each point in the $f - \omega$ parameter space corresponds to a unique trajectory, hence more efficient

So clearly, we shall work with the parameters.

3.1. Choice of encoding

We will be working with 1D data, so varying either parameter is fine. So lets try both ways!

3.1. Choice of encoding

We will be working with 1D data, so varying either parameter is fine. So lets try both ways!

3.1.1. Amplitude encoding

Encode input data using forcing amplitude f (with ω fixed).

- Linearly map the input data to the chosen range of $f \in [f_{\text{min}}, f_{\text{max}}]$.

3.1. Choice of encoding

We will be working with 1D data, so varying either parameter is fine. So lets try both ways!

3.1.1. Amplitude encoding

Encode input data using forcing amplitude f (with ω fixed).

- Linearly map the input data to the chosen range of $f \in [f_{\text{min}}, f_{\text{max}}]$.

3.1.2. Frequency encoding

Encode input data using forcing frequency ω (with f fixed).

- Linearly map the input data to the chosen range of $\omega \in [\omega_{\text{min}}, \omega_{\text{max}}]$.

3.2. Regression

3.2. Regression

Let $u(t) = [u(t_1); u(t_2); ...; u(t_L)]$[1] denote the input signal and $v(t) = [v(t_1); v(t_2); ...; v(t_L)]$ denote the corresponding output signal.

[1] all arrays with semicolon $(;)$ as separator represent column vectors

3.2. Regression

Let $u(t) = [u(t_1); u(t_2); ...; u(t_L)]$[1] denote the input signal and $v(t) = [v(t_1); v(t_2); ...; v(t_L)]$ denote the corresponding output signal.

1. Perform encoding
   - Amplitude encoding: $u(t_i) \xrightarrow{[f_{\min}, f_{\max}]} (f_i, \omega)$

   So $u(t) \to [(f_1 \omega_1), (f_2, \omega_2), ..., (f_i, \omega_i)]$ where all $\omega_i = \omega$

   - Frequency encoding: $u(t_i) \xrightarrow{[\omega_{\min}, \omega_{\max}]} (f, \omega_i)$

   So $u(t) \to [(f_1 \omega_1), (f_2, \omega_2), ..., (f_i, \omega_i)]$ where all $f_i = f$

[1] all arrays with semicolon $(;)$ as separator represent column vectors

3.2. Regression

Let $u(t) = [u(t_1); u(t_2); \ldots; u(t_L)]$[1] denote the input signal and $v(t) = [v(t_1); v(t_2); \ldots; v(t_L)]$ denote the corresponding output signal.

1. Perform encoding
   - Amplitude encoding: $u(t_i) \xrightarrow{[f_{\min}, f_{\max}]} (f_i, \omega)$

   So $u(t) \to [(f_1 \omega_1), (f_2, \omega_2), \ldots, (f_i, \omega_i)]$ where all $\omega_i = \omega$

   - Frequency encoding: $u(t_i) \xrightarrow{[\omega_{\min}, \omega_{\max}]} (f, \omega_i)$

   So $u(t) \to [(f_1 \omega_1), (f_2, \omega_2), \ldots, (f_i, \omega_i)]$ where all $f_i = f$

2. For $t = t_i$, run the reservoir with $(x, \dot{x}) = (0, 0)$ and (f_i, ω_i) corresponding to $u(t_i)$

[1] all arrays with semicolon $(;)$ as separator represent column vectors

3.2. Regression

3. Record reservoir state at sampling rate of $\kappa\Omega$ $(\kappa \in \mathbb{Z})$ for N cycles
    - For amplitude encoding: $\Omega = \omega$ (frequency of driving force)
    - For frequency encoding: $\Omega = \omega_0$ (natural frequency of the oscillator)

Store it as $S_i = [x(0); x(\tau); \ldots; x(\kappa N\tau)]$

3.2. Regression

3. Record reservoir state at sampling rate of $\kappa\Omega$ ($\kappa \in \mathbb{Z}$) for N cycles
   - For amplitude encoding: $\Omega = \omega$ (frequency of driving force)
   - For frequency encoding: $\Omega = \omega_0$ (natural frequency of the oscillator)

   Store it as $S_i = [x(0); x(\tau); ...; x(\kappa N \tau)]$
4. Repeat (2) and (3) for all $u(t_i)$ where $i = 1, 2, 3, ..., L$

3.2. Regression

3. Record reservoir state at sampling rate of $\kappa\Omega$ ($\kappa \in \mathbb{Z}$) for N cycles
   - For amplitude encoding: $\Omega = \omega$ (frequency of driving force)
   - For frequency encoding: $\Omega = \omega_0$ (natural frequency of the oscillator)

   Store it as $S_i = [x(0); x(\tau); ...; x(\kappa N\tau)]$
4. Repeat (2) and (3) for all $u(t_i)$ where $i = 1, 2, 3, ..., L$
5. The reservoir state vector corresponding to $u(t_i)$
   - for nontemporal tasks will be $X_i \equiv S_i$
   - for temporal tasks will be $X_i \equiv [w_0 S_{i-m}, w_1, S_{i-m-1}, ..., w_m S_i]$ where w_j are linearly decreasing weights.
     m is the finite memory which allows us to achieve *fading memory* effect to process temporal data. So, the first m number of input points are used only to generate the dynamics.

3.2. Regression

6. All X_i are stacked to form state vector matrix $\mathfrak{R} \equiv [X_1, X_2, ..., X_L]$. The output signal $v(t)$ is used for regression to connection matrix W.

$$v = W\mathfrak{R}$$

For training, we calculate W using the known $v(t)$ as

$$W = v\mathfrak{R}^{-1}$$

4.1. Description

- Nontemporal:
  Approximate the following polynomial

$$f(x) = (x-3)(x-2)(x-1)x(x+1)(x+2)(x+3)$$

  for $x \in [-3, 3]$

- Temporal:
  Infer missing variable ($y(t)$ or $z(t)$) from the given state variable ($x(t)$) for a chaotic Lorenz system given by

$$\dot{x} = 10(y-x)$$
$$\dot{y} = x(28-z) - y$$
$$\dot{z} = xy - \frac{8}{3}z$$

4.2. Results

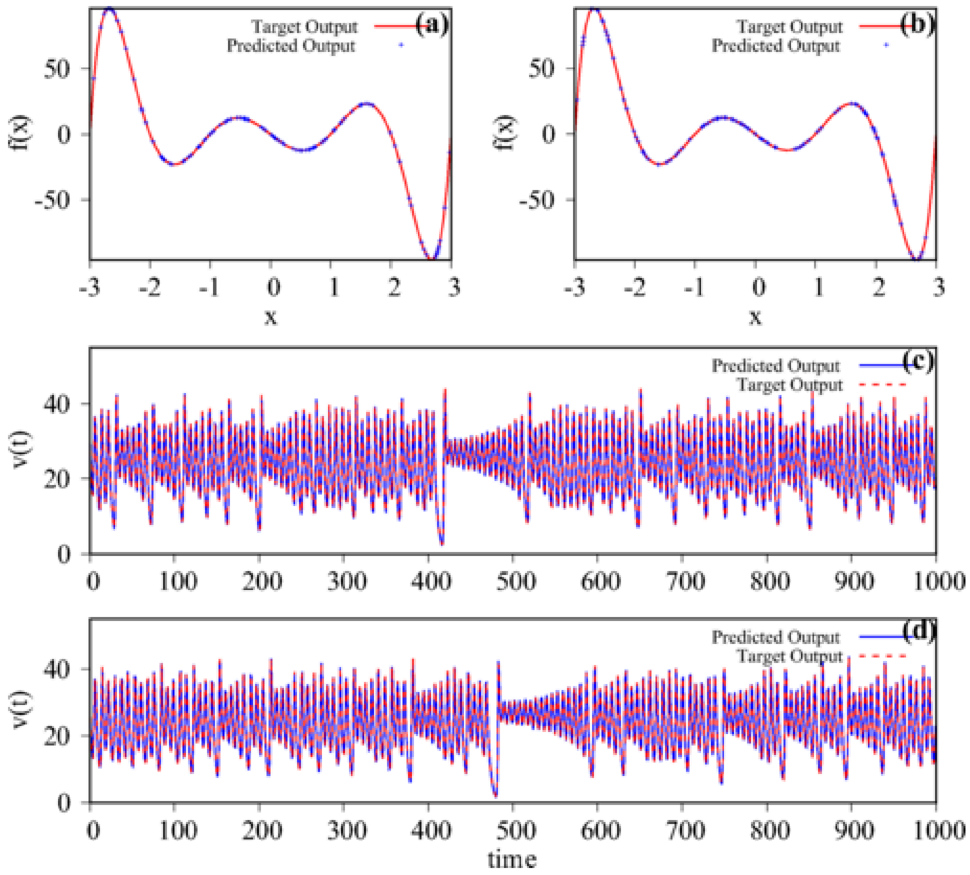

Figure 3: The comparison of predicted output with the target for (a, b) task 1 and (c, d) task 2. (a) and (c) are the results obtained with the amplitude encoding scheme, and (b) and (d) are those obtained with frequency encoding. [2]

4.2. Results

Input encoding	Task 1	Task 1 (with noise)	Task 2	Task 2 (with noise)
f	10^{-10}	10^{-2}	10^{-5}	10^{-5}
ω	10^{-8}	10^{-3}	10^{-5}	10^{-5}

Table 1: Comparison of performance, as quantified by the RMSE [2]. For Task 1, the reservoir was trained with 500 data points; for Task 2, the reservoir was trained with temporal data of length 5000. The results for Task 2 are for $x(t) \to z(t)$ prediction. For testing performance in the presence of noise, each state variable was perturbed with a random noise, uniformly distributed in the range $[-0.01 : 0.01]$.

- A simple driven pendulum holds great potential to be used as a reservoir for machine learning tasks!
- Showed construction of such a reservoir
- Excellence in both temporal and nontemporal tasks
- It works great in noisy conditions, too!

- A simple driven pendulum holds great potential to be used as a reservoir for machine learning tasks!
- Showed construction of such a reservoir
- Excellence in both temporal and nontemporal tasks
- It works great in noisy conditions, too!

Thank you!

Bibliography

[1] G. Van der Sande, D. Brunner, and M. C. Soriano, "Advances in photonic reservoir computing," *Nanophotonics*, vol. 6, pp. 561–576, 2017, doi: 10.1515/nanoph-2016-0132.

[2] S. Mandal, S. Sinha, and M. D. Shrimali, "Machine-learning potential of a single pendulum," *Physical review. E*, vol. 105, 2022, doi: 10.1103/physreve.105.054203.

Questions?