Learning Dynamical Systems Using Dynamical Systems

The Reservoir Computing Approach

Doctoral Dissertation submitted to the
Faculty of Informatics of the Università della Svizzera Italiana
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

presented by
Pietro Verzelli

under the supervision of
Cesare Alippi and Lorenzo Livi

January 2022

Dissertation Committee

Andrea Emilio Rizzoli	Universitá della Svizzera italiana, Switzerland
Rolf Krause	Universitá della Svizzera italiana, Switzerland
Juan-Pablo Ortega	Nanyang Technological University, Singapore
Peter Ashwin	University of Exeter, UK
Alessio Micheli	Universitá di Pisa, Italy

Dissertation accepted on 10 January 2022

Research Advisor

Cesare Alippi

Co-Advisor

Lorenzo Livi

PhD Program Director

Walter Binder

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Pietro Verzelli
Lugano, 10 January 2022

To life and my love for it.

Intelligence does not aspire to
emancipate itself, but to submit.
Truth is the shine of necessity.

Nicolás Gómez Dávila

Abstract

Dynamical systems have been used to describe a vast range of phenomena, including physical sciences, biology, neurosciences, and economics just to name a few. The development of a mathematical theory for dynamical systems allowed researchers to create precise models of many phenomena, predicting their behaviors with great accuracy. For many challenges of dynamical systems, highly accurate models are notably hard to produce due to the enormous number of variables involved and the complexity of their interactions. Yet, in recent years the availability of large datasets has driven researchers to approach these complex systems with machine learning techniques. These techniques are valuable in settings where no model can be formulated explicitly, but not rarely the working principles of these models are obscure and their optimization is driven by heuristics. In this context, this work aims at advancing the field by "opening the black-box" of data-driven models developed for dynamical systems. We focus on Recurrent Neural Networks (RNNs), one of the most promising and yet less understood approaches. In particular, we concentrate on a specific neural architecture that goes under the name of Reservoir Computing (RC). We address three problems: (1) how the *learning* procedure of these models can be understood and improved, (2) how these systems encode a *representation* of the inputs they receive, and (3) how the *dynamics* of these systems affect their performance. We make use of various tools taken from the theory of dynamical systems to explain how we can better understand the working principles of RC in dynamical systems, aiming at developing new guiding principles to improve their design.

Acknowledgements

First of all, I would really like to thank Professor Cesare Alippi and Professor Lorenzo Livi for their careful guidance. They guided me throughout this quest with gentle yet firm suggestions. This achievement is mainly due to them. I would also like to thank Professor Peter Tiňo for his precious advice during my time spent in Birmingham. I also thank Professor Filippo Maria Bianchi for his help.

I thank all my family members for the support that they gave me during the course of my studies and for all the advice and suggestions gently offered. They were never refused.

I am really grateful to my colleagues and friends since they made my time in the lab a joyful one. I thank Daniele Z. for the calculations we did together, Daniele G. for his boldness, Alberto for his friendship, Andrea for his nice temper, Ivan for the gym sessions, Matteo for the cigarettes. I'd like also to thank Alessandro, Stefano, Slobodan, and Kartik, as it was always a pleasure to talk to them. A special thank goes to Lorenzo for the many adventures we had and to Elena for her incredible kindness.

Among my friends in Como, it is foremost to thank Marta whose company was always appreciated, way more than I was able to express. I'd also like to thank Stefania and Elena for their niceness towards me in these years. I thank Alessandro for the Ikea trip and all that followed and I thank Letizia for her suggestions regarding my love struggles. I thank Alessia for the deep thoughts we shared on many occasions and Claudio for his attitude towards life. I also thank Gaia for her spirit and Clara for her advice on how to use Instagram.

A special thanks to my friends of the Comitato Ventotene: together we fought many battles. In particular, I'd like to thank Jolyon, Luca M., Luca B., Dario, Claudia, Jovana, Nadia, and Beatrice. I thank Alexandra for our long friendship and Michalina for the long philosophical discussion we had and the short time we spent together, I also thank the many researchers I've met in these four years, Ayn Rand, Lucio Battisti, Nietzsche, Battiato, Federico Fiumani, D. Hofstadter, Gödel, Escher, Bach, Asimov, Von Neumann, Mozart, N.N. Taleb, Šklovskij, Barthes, Poincaré, Buzzati, Morselli, Dalla, Phil Lynott, Conan Doyle, Dick, Tarkovsky, David Lynch, Joni Mitchell, Buckingham and Nicks, Dario Bressanini, Bill Withers, John Fogerty, Bernhard, Cioran, and all the persons I have met because of the phase flow.

And finally, I would like to thank Chiara, simply for everything.

Contents

Chapter 1

Introduction

Predicting the future has always been a crucial goal for humans. In a certain sense, it can be said that it is the fundamental task of science: while it is basically always possible to explain past observations, well-grounded scientific theories must be able to predict future outcomes of experiments. Otherwise, they are rejected (or, at least, questioned).

The study of phenomena that change in time goes under the name of Dynamical Systems theory. Even if the origin of this field may be dated back to Isaac Newton and his study of mechanics, the foundation of the modern theory of dynamical systems is usually attributed to Henri Poincaré. He laid down the basis of the local and global analysis of nonlinear differential equations, introducing concepts like stable and unstable manifolds, and using a geometric approach to tackle these problems. The idea underlying this approach is that reality obeys some unchanging laws that can be discovered and described using the language of mathematics. Hence, with a perfect knowledge of the equations governing a natural system, its future behaviour can be perfectly forecast. This faith in the predicting role of mathematical modelization was evocatively exposed in 1814 by Pierre-Simon Laplace in his book *Essai philosophique sur les probabilités* (Philosophical Essay on Probability):

> We must consider the present state of Universe as the effect of its past state and the cause of its future state. An intelligence that would know all forces of nature and the respective situation of all its elements, if furthermore it was large enough to be able to analyze all these data, would embrace in the same expression the motions of the largest bodies of Universe as well as those of the slightest atom: nothing would be uncertain for this intelligence, all future and all past would be as known as present.

Difficulties related to this view were already evident to Poincaré, who realized that some problems (like the famous *three-body problem*) were *impossible* to solve, in the sense of obtaining precise formulas for the motion of the objects under study. But an even more radical form of uncertainty was discovered. Leaving aside the stochastic nature of quantum phenomena, also fully-deterministic systems may display an inherent unpredictability, which goes under the name of *deterministic chaos*. The findings of Edward Lorenz are usually considered the beginning of the study of chaotic phenomena. He talked about his discovery in an interview:

> I wanted to examine some of the solutions in more detail. I had a small computer in my office then so I typed in some of the intermediate conditions which the computer had printed out as new initial conditions to start another computation and then went out for awhile. When I came back I found that the solution was not the same as the one I had before. The computer was behaving differently. I suspected computer trouble at first. But I soon found that the reason was that the numbers I had typed in were not the same as the original ones. These were rounded off numbers. And the small difference between something retained to six decimal places and rounded off to three had amplified in the course of two months of simulated weather until the difference was as big as the signal itself. And to me this implied that if the real atmosphere behaved in this method then we simply couldn't make forecasts two months ahead. The small errors in observation would amplify until they became large.

The discovery that a 3-dimensional dynamical system displays *sensitive dependence to initial conditions* undermined the belief about the predictability of systems, even when they are modeled to be deterministic and low-dimensional. Chaos theory has greatly developed since then due to theoretical studies and the numerical simulations made possible by the improvement of computational power.[1] But yet, the unpredictability is still at the core of these systems. This means that even when having a perfect model of a phenomenon, an accurate prediction of its future state might be impossible.

Yet, the key-role of models in today science is far from being obsolete. In fact, the discovery that even simple models can display complex behaviors led

[1]Lorenz's electronic computer, the Royal McBee, was able to perform only sixty (60!) multiplications a second.

researchers to provide simple explanations to complicated phenomena. Scientists in the past century have been able to describe behaviors which appeared heterogeneous and intricate simply by starting from few assumptions. Particle physics, neuroscience, fluidodynamics and, more recently, complex networks are just some notable examples of this trend. Basically any scientific field involves the creation of a (mathematical) model encoding the fundamental feature of a natural (or artificial) phenomenon.

But for many of the most challenging applications in science exact models are notably hard to produce due to the enormous number of variables involved and the complexity of their interactions. Yet, in recent years the incredible amount of available data has driven researchers to study these complex systems by using machine learning techniques. These techniques are valuable in settings where no model can be formulated explicitly. This approach led to incredible achievements in fields like image classification, text and speech processing, and autonomous driving, just to name a few. Machine learning uses the concept of *model* in a novel sense: instead of formulating a mathematical model that aims at encoding the fundamental feature of the object under study, the models used in machine learning are constructed to learn a specific behavior, which would mimic the one of the system of interest. This might seem to be something alien to the scientific method: instead of understanding the rules governing reality, one simply tries to imitate its behavior. This issue generates some vibrant discussions among experts, not only from an epistemological point of view, but also regarding the usage of machine learning in real word tasks, where some guarantees might be needed.

In this work, we focus on a particular class of machine learning models, namely Reservoir Computing (RC), which is particularly suited for the study of dynamical systems (and time-related tasks in general), aiming at advancing our understanding of this class models in order to better exploit their potential. This is a particularly intriguing quest as these models have a two-fold relationship with dynamical systems: on one hand, they are used to approximate and predict dynamical systems; therefore, they can be applied in their study. On the other hand, they *are* dynamical systems, meaning that the theory of dynamical systems can be used to study them. We explore this intriguing connections in order to advance the knowledge in the field, explaining some salient behavior of RC systems. Our research is aimed at *opening the black box* of such systems, with the belief that a better understanding of their behaviours would lead to an improvement in their performance too.

1.1 Structure of the Thesis

This work is organized as follows. Chapter 2 introduces the problem setting and presents the state of the art. The RC framework is discussed in Chapter 3, along with the most popular implementation called Reservoir Computing Network (RCN). The thesis is then divided into three parts:

- Part I focuses on the *Learning* mechanism of RC systems. Chapter 4 explains how these systems are trained and which hyper-parameters affect their performance and need to be optimized. Chapter 5 introduces a model which provably alleviates the hyper-parameters optimization.
- In Part II, we study how the input *Representation* is encoded into the reservoir states. Chapter 6 discusses the role played by memory, while Chapter 7 examines the *Echo State Property*, a fundamental property of RC systems. Chapter 8 presents a theoretical analysis of input-to-state representation.
- Part III discusses the *Dynamics* of RC systems. In Chapter 9 the relationship between dynamical and computational properties of systems is shortly presented. In Chapter 10 a minimal model displaying the main computational features of RCN is studied. Chapter 11 introduces the topic of synchronization of dynamical systems and presents its implications in the context of RC training and performance.

Finally, in Chapter 12, conclusions are drawn and possible future directions are sketched. Various appendices, concerning insights or technical details, conclude this work.

1.2 Notation

We use $\mathbb{R}$ to denote the set of *real numbers*, $\mathbb{Z}$ for the *integers* and $\mathbb{N}$ for the *naturals*. For vectors, we use bold lower case letters, e.g., $\boldsymbol{v}$, while for matrices we use bold upper case letters, e.g., $\boldsymbol{M}$. Functions are denoted with bold font when their image is a vector-space, whereas we use regular font when they are applied element-wise. For instance, $\boldsymbol{f}(\boldsymbol{x})$ is a generic function of $\boldsymbol{x} = [x_1, x_2, \dots, x_N]$, while $f(\boldsymbol{x})$ is a short-hand notation for $[f(x_1), f(x_2), \dots, f(x_N)]$. Time derivatives use the dot-notation, $\dot{\boldsymbol{x}} = \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}$. We reserve the letter $\boldsymbol{s}$ for the *states* of the *source system* generate by the map $\boldsymbol{g}$. We use $\boldsymbol{u}$ to denote the *observables* of such system. The *reservoir states* are denoted by $\boldsymbol{r}$ and the reservoir update function by $\boldsymbol{f}$. We refer to continuous-time systems by using the function notation $\boldsymbol{x}(t)$ (as they are, in fact, functions of time), while we use the subscript for discrete-time one, $\boldsymbol{x}_t$. Concerning the reservoir architecture, we will denote by $\boldsymbol{W}$ the *reservoir connection matrix* and by ρ its *spectral radius*. The matrix (or vector) mapping the input into the reservoir states is denoted by $\boldsymbol{w}$. We use the symbol σ for the activation function.

1.3 Papers Published from this Thesis

Part of this thesis presents the results of published research. Here we briefly describe these publications.

- In [Verzelli et al., 2018] we studied which characteristics of the networks are needed to display the main dynamical features of Recurrent Neural Networks (RNNs). This was done by proposing a binary RCN (bRCN), namely a RCNs whose weights and states are binary. Due to its simplicity, an exact theoretical analysis of the feature of such a model was possible, shedding the light on the working principles of RNNs. This work in presented in Chapter 10.
- In [Verzelli et al., 2019], we studied the properties of RCNs equipped with an activation function that projects the network state on a hyper-sphere at each time-step. First, we prove that the proposed model is a universal approximator just like regular RCNs and we give sufficient conditions to support this claim. Our theoretical analysis shows that the behavior of the resulting network is never chaotic. This leads to networks that are globally stable for any hyper-parameter configuration. The proposed activation function allows the model to display a memory capacity comparable with the one of linear Echo State Networks (ESNs), but its intrinsic nonlinearity makes it capable to deal with tasks for which rich dynamics are required. The neural network introduced in this paper is presented and discussed in Chapter 5 while Chapter 6 contains an experimental analysis of its memory properties.
- In [Verzelli, Alippi, Livi and Tiňo, 2021] we proposed a methodology for explaining how linear reservoirs encode inputs in their internal states. Our theoretical findings allow to express the system state in terms of the *controllability matrix* $\mathscr{C}$ and the *network encoded input* $\boldsymbol{v}$. Our results show that reservoirs with a cyclic topology give the richest possible encoding

of input signals, yet they also offer one the most parsimonious reservoir parametrization. Some experimental facts that motivate this work are introduced in Chapter 6, while the content of this paper is discussed in Chapter 8.

- In [Verzelli, Alippi and Livi, 2021], we laid down the groundwork for establishing and analyzing the working principles of RC within the theoretical framework of synchronization between dynamical systems. We showed that the presence of a synchronization function allows to formally consider the reservoir states as an unsupervised, high-dimensional representation of an unknown source system that generates the observed data. We showed that the realizability of learning, defined as the possibility of perfectly solving the task, crucially depends on the existence of a function connecting the reservoir states with the source system states: the presence of Generalized Synchronization (GS) implies the existence of a synchronization function playing an analogous role, which is found in an unsupervised way in RC. Moreover, the presence of such a synchronization function allows one to make use of the ergodicity of the source system to grant results on the generalization error for a given task. This work in presented in chapter 11.

Chapter 2

Problem Formulation

In this preliminary chapter, we introduce the reader to the main topics that we are going to address. Section 2.1 presents the setting, describing dynamical systems. Section 2.2 describes the tasks that one faces when dealing with temporal data, namely *prediction*, *generation* and *classification*. In Sec. 2.3 the main data-driven approaches to the problem are presented.

2.1 Dynamical Systems

In this section we briefly introduce the topic of dynamical systems. A complete and more formal discussion is contained in Appendix A.

2.1.1 Discrete-Time Dynamical Systems

Let us consider a discrete-time autonomous *source* system, described by:

$$\boldsymbol{s}_{t+1} = \boldsymbol{g}(\boldsymbol{s}_t) \tag{2.1}$$

where $\boldsymbol{s}_t \in \mathbb{R}^{d_s}$ denotes the system *state* at time t. Assume the update function $\boldsymbol{g}$ to be differentiable and invertible, and that $\boldsymbol{s}_t$ asymptotically approaches and stays in a bounded attractor, $\mathscr{A}_s$. We will be interested in the situation where $\boldsymbol{g}$ is unknown and we do not have direct access to the source system states.

2.1.2 Continuous-Time Dynamical Systems

Continuous-time dynamical systems are usually described in term of differential equations:

$$\dot{\boldsymbol{s}}(t) = \boldsymbol{q}(\boldsymbol{s}(t)) \tag{2.2}$$

where we use the dot-notation for derivatives $\dot{\boldsymbol{s}} := ds/dt$. Note that here we describe the system in terms of its variation in time (the derivative reflects this fact): it tells us how the system is changing, as opposed to (2.1) which directly describes where the system will be at time t.

To recover this concept, we need the notion of *flow* $\boldsymbol{Q}_t$:

$$\boldsymbol{s}(t) = \boldsymbol{Q}_t(\boldsymbol{s}_0) \tag{2.3}$$

i.e., the flow evolves the initial point $\boldsymbol{s}_0$ in time.

The flow is expressed in terms of the integral of (2.2)

$$\boldsymbol{Q}_t(\boldsymbol{s}) = \boldsymbol{s} + \int_0^t \mathrm{d}t' \boldsymbol{q}(\boldsymbol{s}(t')) \tag{2.4}$$

2.1.3 Discretization

Discrete-time systems can be thought of as a discretization of continuous ones. In particular imagine, as common in practical cases, that our system is varying continuously in time, but we are only able to sample it every Δt seconds. We then have a series of measurements of the system $x(0), x(\Delta t), x(2\Delta t), x(3\Delta t), \dots$ and so on. The relation which relates each observation can be given in terms of flow (2.3):

$$\boldsymbol{s}(k\Delta t + \Delta t) = \boldsymbol{Q}_{\Delta t}\boldsymbol{s}(k\Delta t) \tag{2.5}$$

where k is the index of our sample. If we write $\boldsymbol{s}_k := \boldsymbol{s}(k\Delta t)$ the equation above becomes:

$$\boldsymbol{s}_{k+1} = \boldsymbol{Q}_{\Delta t}(\boldsymbol{s}_k) \tag{2.6}$$

which is just (2.1) where $\boldsymbol{Q}_{\Delta t}$ plays the role of the map and k is the time index.

2.1.4 Observables

In many contexts, one does not have access to the full state of the system, $\boldsymbol{s}$ but only to a limited set of *observables* (sometimes called *measurements)* $\boldsymbol{u} \in \mathbb{R}^{d_u}$

$$\boldsymbol{u}_t = \boldsymbol{h}(\boldsymbol{s}_t) \tag{2.7}$$

When $\boldsymbol{h}$ is the identity, one has complete knowledge of the system state.

We consider the case where one is given a set of observations of the system $\{\boldsymbol{u}_t\}$ $t = 0, 1, \ldots, T-1$ but no information is provided about its generating equation (2.1).

2.1.5 Noise

In the modelling of dynamical systems, one faces two kind of noise:

Dynamical Noise If the system dynamics is affected by a noise term $\boldsymbol{\xi}_t$ we use the term *dynamical noise*. When the noise is additive, this means that equation (2.1) reads:

$$\boldsymbol{s}_t = \boldsymbol{g}(\boldsymbol{s}_{t-1}) + \boldsymbol{\xi}_{t-1} \tag{2.8}$$

Measurement Noise If the noise affects the observables instead, the term *measurement noise* is used and we denote it with the symbol $\boldsymbol{\epsilon}_t$. If the noise is additive, (2.7) reads:

$$\boldsymbol{u}_t = \boldsymbol{h}(\boldsymbol{s}_t) + \boldsymbol{\epsilon}_t \tag{2.9}$$

This two kind of noise are not mutually exclusive and, in real applications, one should generally account for both. In this work, which mainly deals we methodological issues, we will mainly consider noiseless systems.

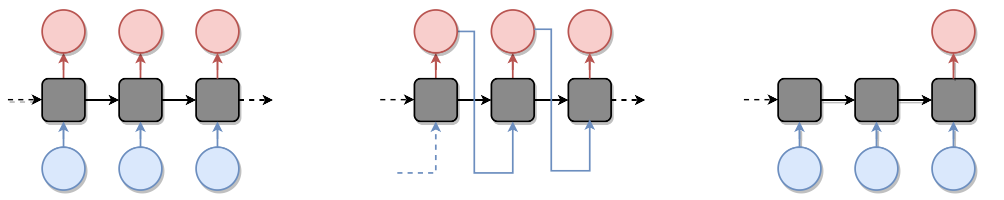
2.1.6 Modelling

In this setting, we have a series of observation $\{\boldsymbol{u}_t\}$ given by (2.7) that are produced by a dynamical system like (2.1). The modelization of the system generating can be divided into different settings, according to the amount of information one has [Bezruchko and Smirnov, 2010].

White Box This is the case in which the structure of the system model is perfectly known, but $\boldsymbol{g} = \boldsymbol{g}_{\boldsymbol{\theta}}$ depends on some parameters $\boldsymbol{\theta}$ which need to be estimated.

Grey Box In the grey box setting, one has only partial knowledge of the system model (2.1). In particular, only some components of the function $\boldsymbol{g}$ are unknown.

Black Box When no information about the source system (2.1) is available, one is said to be in the *Black Box* setting. Here, a model which is able to estimate salient features of the dynamical system must be constructed relying solely on the set of observations.



(a) Prediction *(b)* Generation *(c)* Classification

Figure 2.1. The possible tasks one deals with in dynamical systems. The blue circles represent the inputs while the red dots are the outputs generated by the model (the black box).

In this work, we focus on *Black box* modelling only. In particular we are interested in estimating some salient features of the dynamical systems relying only on a series of observations (2.7), without any further information about it.

2.2 Tasks

Given this setting, there are three different tasks that one may be interested in studying, namely: *prediction*, *generation*, or *classification*.

2.2.1 Prediction

In this case, along with the set of observables $\{\boldsymbol{u}_t\}$ we are also given a set of *targets* $\{\boldsymbol{y}_t\}$. We assume that the target, just like the observable, is a function of the (unknown) system-state only:

$$\boldsymbol{y}_t = \boldsymbol{k}(\boldsymbol{s}_t). \tag{2.10}$$

It makes sense to consider the observables and the targets as pairs $\{(\boldsymbol{u}_t, \boldsymbol{y}_t)\}$ in which $\boldsymbol{u}_t$ is the input and $\boldsymbol{y}_t \in \mathbb{R}^{d_y}$ is the desired output. The task here is to learn how to produce an approximated target $\hat{\boldsymbol{y}}_t$ for $t \geq T$, while receiving a new $\boldsymbol{u}_t$. In some contexts, this framework goes under the name of *input-output systems identification*, as one needs to model an input-output relation.

This scenario assumes that for $t = 0, 1, \ldots, T-1$ we can access both the values of the observables $\boldsymbol{u}_t$ and the corresponding targets $\boldsymbol{y}_t$. This is usually called *training set*. Then, for $t \leq T$ (*testing set*), we still have access to $\boldsymbol{u}_t$ but not to $\boldsymbol{y}_t$. The approach consists in using the data of the training phase to learn how to generate an estimated target $\hat{\boldsymbol{y}}_t$, which can be used in the testing (or operational) phase. An example is depicted in Fig. 2.2.

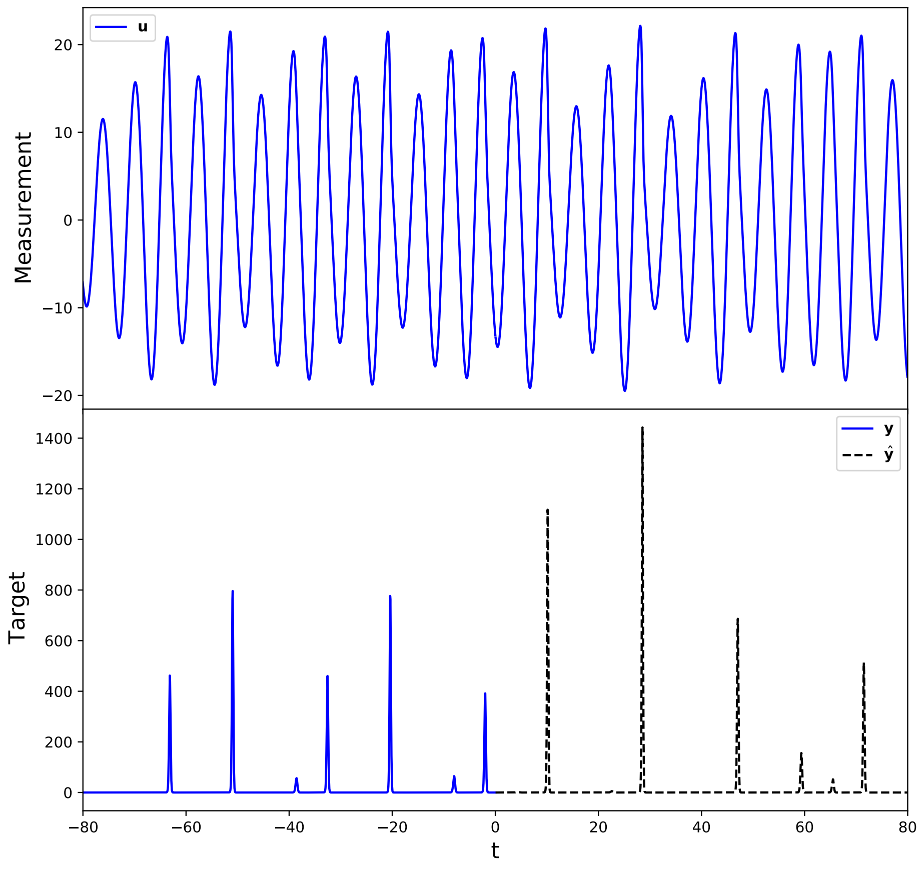


Figure 2.2. An example of a prediction problem, where both $\boldsymbol{u}$ and $\boldsymbol{y}$ are mono-dimensional. The input value $\boldsymbol{u}$ is always provided (top figure), while the target $\boldsymbol{y}$ is only accessible at training time, i.e., for $t < 0$ (bottom figure, blue solid line). The goal is to generate a prediction $\hat{\boldsymbol{y}}$ for $t > 0$ by using the input only. Here, the source system is the Rössler system (see Appendix C.2), the input is $u(t) = x(t)$ while the target is $y(t) = z^2(t)$, where $x(t)$ and $z(t)$ are, respectively, the first and the third coordinate of the Rössler system.

Practical applications include:

Forecasting In the forecasting scenario [Bianchi et al., 2017; Berry et al., 2015], one aims at predicting the future value of the input at given *forecasting horizon* d, i.e., one has $\boldsymbol{y}_t = \boldsymbol{u}_{t+d}$.

State Reconstruction In the reconstruction scenario [Brunton et al., 2017; Lu et al., 2017; Zimmermann and Parlitz, 2018] the task is to reconstruct the full state of the system by observing a limited part of it, i.e., $\boldsymbol{y}_t = \boldsymbol{s}_t$, the dimension of the input $\boldsymbol{u}$ being smaller than the dimension of the state $\boldsymbol{s}$.

Denoising (or Filtering) In the denoising [Krishnagopal et al., 2020; Antonik et al., 2018] scenario one assumes that the input can be decomposed as $\boldsymbol{u}_t = \boldsymbol{x}_t + \boldsymbol{n}_t$, where $\boldsymbol{x}_t$ is the signal of interest and $\boldsymbol{n}_t$ is an *additive noise* term which we wish to filter out. When, as discussed above, we assume that our input comes from a deterministic dynamical system, this situation implies that the observation function (2.7) is noisy, i.e., we have *observational noise*. The target for this task is given by the de-noised signal $\boldsymbol{y}_t = \boldsymbol{x}_t$.

From a theoretical point of view, the above tasks can be formulated using the theory of *filters* (see for example Grigoryeva and Ortega [2018] and references therein) between the input $\boldsymbol{u}_t$ and the target $\boldsymbol{y}_t$. Yet, they are usually considered distinct problems because, in practice, they present different difficulties. So, they benefit from dedicated solutions.

2.2.2 Generation

In this scenario, the goal is to generate an approximation of the squence $\hat{\boldsymbol{u}}_t$ for $t \geq T$. This requires our system to be able to run *autonomously* in the generative phase: the system goal is to learn how to generate $\boldsymbol{u}_{t+1}$ when receiving $\boldsymbol{u}_t$ as input. This is done using the training data. Then, in the generative phase, the system runs autonomously by using the predicted output as input, implementing a *feedback* mechanism. This task is particularly difficult (and interesting) when the system to be learned is *chaotic*. Chaoticity implies a sensitive dependence to the initial conditions, which - roughly speaking - means that trajectories starting from two different points will eventually diverge, even when they are extremely close to each other. This makes it inherently impossible to generate accurate predictions in the long term because even an exact knowledge of the system would still lead to a divergence due to measurement or approximation errors, or noise. So, when predicting chaotic systems, it is challenging even to define what a good prediction is. However, one would generally like to have two properties:

- A **Short-Term Accuracy** in which the predicted trajectory remains close to the true one. This can be evaluated in several ways: the most common one being the *forecast horizon*. It is defined as the time between the start of a prediction and the point where it deviates from the test data more than a fixed threshold.

- A **Long-Term Climate Replication**, which consists in correctly reproducing some important features of the system attractor, instead of focusing on the point-wise similarity. This can be achieved by studying global properties of the attractor of the reproduced system, like the *Lyapunov exponents* or the *Correlation Dimension*.

Notable instance of this task can be found in Pathak, Hunt, Girvan, Lu and Ott [2018]; Vlachas et al. [2018]; Haluszczynski and Räth [2019]; Liu et al. [2020]; Fan et al. [2020]; Pathak, Wikner, Fussell, Chandra, Hunt, Girvan and Ott [2018]; Qi and Majda [2020] to name a few.

2.2.3 Classification

In this scenario, the data consist of pairs $\{\boldsymbol{U}_i, \boldsymbol{l}_i\}$, where $\boldsymbol{U} = \{\boldsymbol{u}_t^{(i)}\}$ is a series of T_i observations and $\boldsymbol{l}_i$ is a (possibly multidimensional) *label* associated to it. The index i spans from 1 to N. One needs to learn a rule to produce the correct label when a series of observations is presented so that it can generate an approximated label associated with previously unseen time series. Instances of this scenario can be found in [Fawaz et al., 2019], [Baydogan and Runger, 2016] or [Bianchi et al., 2020].

2.3 Data-Driven Modelling of Dynamical Systems

Various techniques have been developed to approximate dynamical systems from data observations. A popular approach is based on the approximation of some salient features of the system by making use of Reproducing Kernel Hilbert Space (RKHS) [Bouvrie and Hamzi, 2017]. Koopman analysis [Koopman, 1931]) was recently exploited in this direction [Brunton et al., 2021], starting from the fundamental ideas introduced by Schmid [2010] when developing the Dynamic Mode Decomposition (DMD). These approaches can also be used in more sophisticated ways to study fundamental characteristics of the system of interest [Brunton et al., 2017]. A popular algorithm that is based on looking for sparse solutions in the model space is known as Sparse Identification of Nonlinear Dynamics (SINDy) [Brunton et al., 2016]. A comparison of various methods for prediction was recently tested on a large collection of dynamical systems Gilpin [2021].

Another way of facing this problem is by using neural networks to approximate the system. This can not be done without a careful design of the neural architecture, i.e., exploiting a proper *inductive bias* Karniadakis et al. [2021]. The first attempt of using neural networks to approximate dynamical systems was conducted by González-García et al. [1998]. More advanced techniques have been developed lately, based on exploiting various properties of the underlying dynamical systems [Regazzoni et al., 2019; Gilpin, 2020; Liu and Theodorou, 2019; Qi and Majda, 2020]. When present, it is also possible to exploit the Hamiltonian [Greydanus et al., 2019] or Lagrangian [Cranmer et al., 2020] structure of the system to properly tailor a neural network to perform predictions.

Recently, a review on how data-driven methods can prove useful in dynamical systems was conducted by Berry et al. [2020]. In Karniadakis et al. [2021] physics application are presented, while Brunton et al. [2020] discuss the role played by machine learning in fluid dynamics.

Notably, methods based on dynamical systems also have proved useful to study neural networks and to understand their working principles [Liu and Theodorou, 2019; Weinan, 2017]. Moreover, this relationship can be also exploited to design novel training strategies [Gaedke-Merzhäuser et al., 2020; Chaudhari et al., 2019]: e.g., the popular momentum-based methods can also be seen as instances of this trend [Kingma and Ba, 2014].

Chapter 3

Reservoir Computing

In this Chapter we present the main object of study of this work, the RC framework. In Section 3.1 we introduce some basic terminology and briefly describe how the machine learning approach for *temporal tasks* differs from the *non-temporal* one. In Section 3.2 we provide a compact introduction to Recurrent Neural Network (RNN), discussing the problems that their training mechanism raise. We also mention some relevant applications. The Reservoir Computing (RC) framework is presented in Section 3.3, in which we expose its features. We also mention the most common implementation, the Reservoir Computing Network (RCN) and show that the architecture is a *universal function approximator*. Finally in 3.4 we review recent achievements in RC and mention some relevant applications. We also list some popular variations of the basic RCN.

3.1 Temporal Tasks

In the context of machine learning, a *problem* or a *task* is defined as the issue of estimating a functional relationship between an input $\{\boldsymbol{u}_t\}$ and a desired output $\{\boldsymbol{y}_t\}$, where $t = 0, 1, \ldots, T$ and T is the number of data. In *non-temporal* tasks, data-points are assumed to be independent and the goal is to learn a function $\boldsymbol{\psi}(\boldsymbol{u}_t) =: \hat{\boldsymbol{y}}_t$ which produces correct approximations of the output. This is usually done by defining an *Error* or *Loss* function $E(\hat{\boldsymbol{y}}_t, \boldsymbol{y}_t)$ and then finding a $\boldsymbol{\psi}$ such that it is minimized. In a *temporal* task, the input and the output are signals in the (discrete) time t. Then, the function to learn is not limited to a point-to-point relation, but has general form $\boldsymbol{\psi}(\boldsymbol{u}_t, \boldsymbol{u}_{t-1}, \boldsymbol{u}_{t-2}, \ldots)$. We can say that the function $\boldsymbol{\psi}$ in a temporal task has *memory* and the output at time t depends also on the inputs at preceding times, contrary to the previous case, which in fact does not have a time structure.

Managing memory is a difficult problem as, in general, the input history might be infinite. One needs to create a representation of the signal, i.e., define a function $\boldsymbol{f}$ which creates and encoding of the signal, $\boldsymbol{x}_t = f(\boldsymbol{u}_t, \boldsymbol{u}_{t-1}, \boldsymbol{u}_{t-2}, \dots)$, when processed by $\boldsymbol{\psi} = \boldsymbol{\psi}(\boldsymbol{x}_t) = \hat{\boldsymbol{y}}_t$ Since the number of arguments of a function cannot be unbounded, one has to fix a past-horizon D and use a regressive model which only considers the last D inputs, $\boldsymbol{x}_t = \boldsymbol{f}(\boldsymbol{u}_t, \boldsymbol{u}_{t-1}, \dots, \boldsymbol{u}_{t-(D-1)})$. Choosing $\boldsymbol{f}$ in this way results in model with a fixed memory, which might exceed or fall short the memory required for the task. Selecting the optimal D might be hard and requires some cross-validation.
A way to overcome this problem is to define a state-space representation of the input in a recursive way, as:

$$\boldsymbol{x}_t = \boldsymbol{f}(\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1}). \tag{3.1}$$

Which may be interpreted an input-driven dynamical system.[1] The estimated output is then constructed as $\hat{\boldsymbol{y}}_t = \boldsymbol{\psi}(\boldsymbol{x}_t)$. The underlying assumption is that $\boldsymbol{x}_t$ will encode all the relevant information about the history of the input, so that the temporal problem can be translated into a non-temporal one, i.e., finding the mapping $\boldsymbol{x}_t \to \hat{\boldsymbol{y}}_t$ using the time-independent mapping $\boldsymbol{\psi}$.

The datasets are usually divided into a *Training Set* in which the model is trained and a *Test Set* (unseen during the training) in which it is tested.[2] As a convention, we will denote with negative t the training-set and with positive t the test set, such that the testing phase starts a $t = 0$.

3.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a family of neural networks for processing sequential data. They are modeled as an input-driven discrete-time dynamical system. The most simple RNN was proposed by Elman [1990] and its model reads:

$$\boldsymbol{x}_t = \boldsymbol{f}(\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1}) := \sigma_h(\boldsymbol{W}_h \boldsymbol{h}_{t-1} + \boldsymbol{V}_h \boldsymbol{u}_{t-1} + \boldsymbol{b}_h) \tag{3.2a}$$
$$\hat{\boldsymbol{y}}_t = \boldsymbol{\psi}(\boldsymbol{x}_t) = \sigma_y(\boldsymbol{W}_y \boldsymbol{y}_t + \boldsymbol{b}_y) \tag{3.2b}$$

[1]Some authors use a different index convention for the time indexes of the arguments of f, namely they write: $\boldsymbol{x}_t = f(\boldsymbol{x}_{t-1}, \boldsymbol{u}_t)$ instead of using $\boldsymbol{u}_{t-1}$, to highlight the fact that the information about the $(t+1)$-th input is contained into the $(t+1)$-th state. We prefer to use the alternative notation as it better translates into the dynamical systems framework. They are, however, perfectly analogous, being just a matter of labeling.

[2]We omit, for simplicity, the distinction between the test and the validation set.

where $\boldsymbol{u}_t$ is the input vector at time t, $\boldsymbol{x}_t$ is the hidden layer vector, $\boldsymbol{y}_t$ is the output, $\boldsymbol{V}_h$,$\boldsymbol{W}_h$,$\boldsymbol{W}_y$, $\boldsymbol{b}_h$ and $\boldsymbol{b}_y$ are the network's parameters, and σ_h and σ_y are the activation functions (applied element-wise). RNNs are generally trained by Back-Propagation Through Time (BPTT), which consists of "unfolding" the computational graph and then propagating the gradient through it.

By (3.2), we can see that RNNs involve the composition of the same function multiple times. This results in the gradient being propagated through multiple applications of the same function, which may lead to exploding or (most frequently) vanishing phenomena [Bengio et al., 1994; Pascanu, Mikolov and Bengio, 2013]. This problem is particularly evident when one is trying to model long-term dependencies, as the long-term interactions (involving the multiplication of many Jacobians) are given exponentially smaller weights than short-term ones.

For this reason, Hochreiter and Schmidhuber [1997] introduced the Long Short-Term Memory (LSTM), an architecture explicitly designed to overcome this problem. In there, the idea of using self-loops to produce paths where the gradient can easily flow was developed. This is achieved through the use of *gates*, which explicitly control the computational flow. The current version of the LSTM was proposed in Gers et al. [2000], where an additional gate - called *forget gate* - is introduced to make the weight on this self-loop conditioned on the context, rather than fixed. Another popular gated architecture is called Gated Recurrent Unit (GRU), proposed by Cho et al. [2014], and it is based on a similar idea. We mention that Neural Ordinary Differential Equations (NODEs) [Chen et al., 2018] can be also described as a RNN with continuous time. This fact was recently exploited to design a system with varying time constants [Hasani et al., 2021].

3.3 Reservoir Computing

RC is a computational paradigm developed independently by Jaeger [Jaeger, 2001; Jaeger and Haas, 2004] (ESNs), Maas [Maass et al., 2002] (Liquid State Machines (LSMs)), and Tiňo [Tiňo and Dorffner, 2001] (Fractal Predicting Machines (FPMs)).[3] The basic idea is to create a representation of the input signal using an untrained RNN, called *the reservoir*, and then use a trainable *readout* layer to generate the network output.

[3]Actually, the idea of replacing optimization with randomization in RNNs had already been explored by Schmidhuber in the ′90s [Schmidhuber and Hochreiter, 1996].

The working principle of RC relies upon creating a representation of the input sequence by feeding it to an untrained dynamical system, *the reservoir*, which aims at encoding all relevant dynamics associated with the input. A generic equation for a reservoir would read:

$$\boldsymbol{r}_t = \boldsymbol{f}(\boldsymbol{r}_{t-1}, \boldsymbol{u}_{t-1}) \tag{3.3}$$

Learning focuses solely on the *readout* function, which is trained to generate the desired output, given the encoded dynamics and the task at hand.

$$\hat{\boldsymbol{y}}_t = \boldsymbol{\psi}_{\boldsymbol{\theta}}(\boldsymbol{r}_t) \tag{3.4}$$

Here $\boldsymbol{\theta}$ stands for the set of tunable parameters which characterize the particular form of the readout function. The parameters are selected by a fitting procedure yielding parameter configuration $\hat{\boldsymbol{\theta}}$, which is optimal according to some criteria. In this work, when it does not lead to ambiguity, we will drop the $\hat{\boldsymbol{\theta}}$-notation and simply write $\boldsymbol{\psi}$ for $\boldsymbol{\psi}_{\hat{\boldsymbol{\theta}}}$.

3.3.1 Reservoir Computing Network

Practically, (3.3) is usually implemented in the form of a neural network. In this case, we refer to it as Reservoir Computing Network (RCN). [4] Its equation reads:

$$\boldsymbol{r}_{t-1} = \sigma(\boldsymbol{W}\boldsymbol{r}_{t-1} + \boldsymbol{w}\boldsymbol{u}_{t-1} + \boldsymbol{b}) \tag{3.5}$$

where the reservoir *size* is N, $\boldsymbol{r}_t \in \mathbb{R}^N$ is the reservoir *state* at *time* t, $\boldsymbol{W} \in \mathbb{R}^{N\times N}$ is the reservoir *connection matrix*, $\boldsymbol{b} \in \mathbb{R}^N$ is the *bias*; $\boldsymbol{u}_t \in \mathbb{R}^d_{\text{in}}$ is the *input* at time t, $\boldsymbol{w} \in \mathbb{R}^{N\times d_{\text{in}}}$ is the *input matrix* and σ is the *activation function* (usually tanh), applied element-wise.

When the readout is *linear*, we will use $\boldsymbol{W}_{\text{out}} \in \mathbb{R}^{d_{\text{out}}\times N}$ to denote its weights, where d_{out} is the dimension of $\boldsymbol{y}$. Then, (3.4) reads:

$$\hat{\boldsymbol{y}}_t = \boldsymbol{W}_{\text{out}} \cdot \boldsymbol{r}_t \tag{3.6}$$

A diagram representing the RCN architecture is depicted in Fig. 3.1.

[4] We prefer this name over ESN, which is more popular in the engineering community, as the reference to the *echo state* might be misleading, as we will see in Chapter 7.

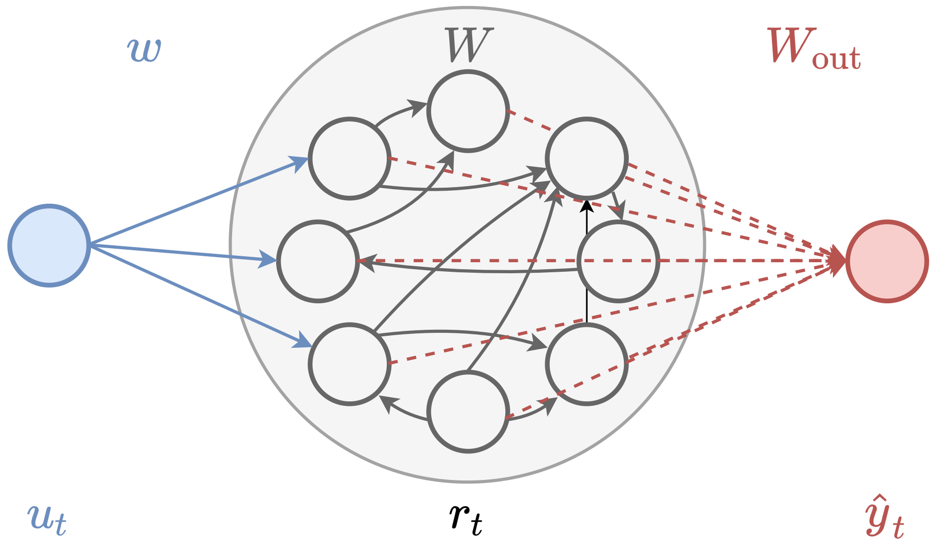


Figure 3.1. A schematic representation of a RCN. The dashed lines are the only tunable weights, which are learned using the training data.

3.3.2 Universal Function Approximation Property

The fact that recurrent neural networks are universal function approximators has been proven in previous works [Siegelmann and Sontag, 1995; Hammer, 2000] and some results on the universality of reservoir-based computation are given in[Maass et al., 2002; Hammer and Tiňo, 2003]. Recently, Grigoryeva and Ortega [2018] proved the universality of RCN: in there, they showed that, given a task, there exist a RCN which is able to solve it with any given accuracy.

More in detail, let us define a *squashing function*:

Definition 1. A *squashing function* is a map $f : \mathbb{R} \to [-1, 1]$ that is *non decreasing* and *saturating*, i.e., $\lim_{x\to\pm\infty} f(x) = \pm 1$.

Then, RCN in the form (11.35),(11.36) for which σ is a squashing function is a universal function approximator provided it satisfies the *contractivity condition*, i.e., that for each $\boldsymbol{x}$ and $\boldsymbol{y}$ in the domain of the activation function σ, it holds:

$$\|\sigma(\boldsymbol{x}) - \sigma(\boldsymbol{y})\| \leq \|\boldsymbol{x} - \boldsymbol{y}\| \tag{3.7}$$

We highlight that this work discusses the *expressivity* of the RCN model class, i.e., the existence of such a RCN is granted but this does not mean that any random initialization of the reservoir weight will work. The question about the possibility of solving any task using a RCN in which only the readout is trained (i.e., the *learnability* of the correct RCN) is addressed in [Gonon et al., 2020a].

The universal approximation property, as exhaustively discussed in [Grigoryeva and Ortega, 2018], can be proved for an RCN of the form (11.35), (11.36) provided that it has the Echo State Property (ESP) (see Chapter (7)).

3.4 Research trends in RC

RC obtained state-of-the-art results in solving various tasks [Lu et al., 2018; Chattopadhyay et al., 2020; Vlachas et al., 2020; Bompas et al., 2020]. It also displayed outstanding results when combined with model-based approaches [Pathak, Wikner, Fussell, Chandra, Hunt, Girvan and Ott, 2018; Doan et al., 2020]. Deep approaches to the RCN have also been used [Gallicchio et al., 2017, 2020], along with more sophisticated training strategies [Løkse et al., 2017; Herteux and Räth, 2020]. The simplicity of the approach makes RC also amenable for theoretical investigations [Gonon et al., 2020c; Grigoryeva and Ortega, 2018; Rodan and Tino, 2010; Tiňo, 2020; Goudarzi et al., 2016; Ganguli et al., 2008].

The research interest in RC in the contests of dynamical systems was recently renewed due to the outstandings results obtained in [Pathak, Hunt, Girvan, Lu and Ott, 2018]. Successive studies were devoted to better understand these findings [Lu et al., 2017, 2018; Lu and Bassett, 2020]. Also the state-reconstruction capability of RC was exploited in a similar fashion [Lu et al., 2017; Weng et al., 2019]. Moreover, results from dynamical systems theory have applied to the study of RC, concerning the dimensions of the learned attractor Carroll [2020b, 2021a], the stability of the reservoir system Gallicchio et al. [2021]; Engelken et al. [2020]; Vogt et al. [2020] and other relevant features Carroll [2020c,a]. These findings have been used to design better suited RC systems Carroll [2021b]; Haluszczynski and Räth [2019]; Herteux and Räth [2020].

RC is also particularly appealing for neuromorphic computing [Neftci et al., 2017; Neftci, 2018] and other hardware implementations [Appeltant et al., 2011; Larger et al., 2012], but also a bucket of water [Fernando and Sojakka, 2003] or road traffic [Ando and Chang, 2021] have been used as reservoirs. See Tanaka et al. [2019] for a recent review.

3.4.1 Popular Variations

Linear RCN We call a RCN linear when its activation function, i.e., σ in (3.5) is the identity. Its equation reads:

$$\boldsymbol{r}_{t+1} = \boldsymbol{W}\boldsymbol{r}_t + \boldsymbol{w}\boldsymbol{u}_t + \boldsymbol{b} \tag{3.8}$$

When using a linear readout like (3.6), linear RCNs are as expressive as an autoregressive models [Verzelli, Alippi, Livi and Tiňo, 2021; Bollt, 2021]. Therefore, they are interesting mainly for theoretical reasons, as their simple form renders the mathematics amenable and many analytical results regarding their properties are already known. The tractability comes from the fact that (3.8) can be expanded recursively:

$$\begin{aligned}
\boldsymbol{r}_{t+1} &= \boldsymbol{W}\boldsymbol{r}_t + \boldsymbol{w}\boldsymbol{u}_t + \boldsymbol{b} && (3.9)\\
&= \boldsymbol{W}\underbrace{(\boldsymbol{W}\boldsymbol{r}_{t-1} + \boldsymbol{w}\boldsymbol{u}_{t-1} + \boldsymbol{b})}_{r_t} + \boldsymbol{w}\boldsymbol{u}_t + \boldsymbol{b} && (3.10)\\
&= \boldsymbol{W}^2\underbrace{(\boldsymbol{W}\boldsymbol{r}_{t-2} + \boldsymbol{w}\boldsymbol{u}_{t-2} + \boldsymbol{b})}_{r_{t-1}} + \boldsymbol{W}(\boldsymbol{w}\boldsymbol{u}_{t-1} + \boldsymbol{b}) + \boldsymbol{w}\boldsymbol{u}_t + \boldsymbol{b} && (3.11)\\
&= \boldsymbol{W}^3\underbrace{(\boldsymbol{W}\boldsymbol{r}_{t-3} + \boldsymbol{w}\boldsymbol{u}_{t-3} + \boldsymbol{b})}_{r_{t-2}} + \boldsymbol{W}^2(\boldsymbol{w}\boldsymbol{u}_{t-2} + \boldsymbol{b}) + \boldsymbol{W}(\boldsymbol{w}\boldsymbol{u}_{t-1} + \boldsymbol{b}) + \boldsymbol{w}\boldsymbol{u}_t + \boldsymbol{b} && (3.12)
\end{aligned}$$

and so on, leading to:

$$\boldsymbol{r}_{t+1} = \sum_k \boldsymbol{W}^k(\boldsymbol{w}\boldsymbol{u}_{t-k} + \boldsymbol{b}) = \sum_k \boldsymbol{W}^k\boldsymbol{w}\boldsymbol{u}_{t-k} + \sum_k \boldsymbol{W}^k\boldsymbol{b} \tag{3.13}$$

in which $\boldsymbol{W}^0$ is the identity.

Continuous-Time RCN It is possible to define RCN in continuous time. This is typically done by defining the network in terms of an Ordinary Differential Equation (ODE) of the form:

$$\dot{\boldsymbol{r}}(t) = -\frac{1}{\tau}\boldsymbol{r} + \boldsymbol{f}(\boldsymbol{r}(t), \boldsymbol{u}(t)) \tag{3.14}$$

which is then numerically integrated. Here, τ is an hyperparameter controlling the leakage of the network. Notable instances can be found in Lu et al. [2018]; Carroll [2020b].

Leaky RCN The time-derivative which appears on (3.14) can be approximated using the Euler discretizaion with time-step δ as:

$$\dot{\boldsymbol{r}}(t) \approx \frac{\boldsymbol{r}_{t+1} - \boldsymbol{r}_t}{\delta} \tag{3.15}$$

which, rearranging the terms, leads to the *leaky RCN* Jaeger [2001]; Jaeger et al. [2007]:

$$\boldsymbol{r}_t = (1 - \frac{\delta}{\tau})\boldsymbol{r}_{t-1} + \delta \boldsymbol{f}(\boldsymbol{r}_{t-1}, \boldsymbol{u}_{t-1}) \tag{3.16}$$

in this model, δ is treated as a hyperparameter to be optimized. Introducing the leakage allows the system to perform what is called a *time-warp* Jaeger et al. [2007]; Tallec and Ollivier [2018], i.e., being invariant to the change of the input time-scale. The idea of considering the discrete-time RCN as a discretization of a continuous-time model can also be exploited in more sophisticate ways, like [Gallicchio, 2021] which aims at improving the reservoir stability.

Deep RCN Just like Feed-forward networks, RNNs can be constructed using multiple (recurrent) layers, leading to deep architectures Pascanu, Gulcehre, Cho and Bengio [2013]. Thus, *deep RCNs* Gallicchio et al. [2017] can be constructed by stacking different (untrained) layers of the form:

$$\boldsymbol{r}_t^{(l+1)} = \boldsymbol{f}^{(l)}(\boldsymbol{r}_{t-1}^{(l)}, \boldsymbol{u}_{t-1}^{(l)}) \tag{3.17}$$

where l is the layer index. The input signal is then fed to the first layer, $\boldsymbol{u}_t^{(0)} = \boldsymbol{u}_t$, while for $l \geq 1$ the input is given by an untrained mask of the preceding layer:

$$\boldsymbol{u}_t^{(l+1)} = \boldsymbol{w}^{(l+1)}\boldsymbol{r}_t^{(l)}. \tag{3.18}$$

This approach showed encouraging results, in particular for the managing of multiple time-scales Gallicchio and Micheli [2011]; Gallicchio et al. [2017, 2020, 2018].

Part I

Learning

Chapter 4

Training

In this chapter we describe the training procedure for RC systems. In Section 4.1, we present the Reservoir Learning Algorithm, which is the most popular procedure for training RCNs, which we name Reservoir Learning Algorithm (RLA), highlighting some of its features. We also mention some other alternative techniques that appeared in the literature. The main hyperparameters that one needs to optimize are discussed in Section 4.2, along with their properties. We focus on describing different topologies for the reservoir connection matrix $\boldsymbol{W}$, which will play a major role in our analysis of memory We also discuss different possible readout functions that one may use instead of the simple linear one, and the spectral radius ρ, as it is the most studied hyperparameter.

4.1 Training Strategies

RC is characterized by an untrained dynamical *reservoir* and by a *readout* which learns to solve the task, but different training procedures can be explored. These procedures are generally less computationally expensive than BPTT, but their performance tends to be sensibly affected by the hyper-parameters choice. So, a hyper-parameters exploration needs to be carried to obtain good results.

4.1.1 Reservoir Learning Algorithm

The most common way for training RC-systems is the one developed by Jaeger [2001] in his seminal work. We refer to it as the RLA. It is based on three phases, whose current naming was given in [Lu et al., 2018]:

Listening In the *listening* phase, the training measurements $\{\boldsymbol{u}_t\}$ are used as input to the reservoir (3.3). This creates a series of states $\{\boldsymbol{r}_t\}$ that will be later used for fitting.

Fitting *Fitting* consists in determining the *readout function*, $\boldsymbol{\psi}_{\boldsymbol{\theta}}$, which reads the reservoir state $\boldsymbol{r}_t$ and provides an estimate for the output $\boldsymbol{y}_t$. Parameters $\boldsymbol{\theta}$ are selected by a fitting procedure yielding a parameter configuration $\hat{\boldsymbol{\theta}}$.

Predicting When fitting is completed, the system can be used to predict new target values (*predicting phase*) for previously unseen data.

A schematic representation of the RLA is depicted in Figure 4.1.

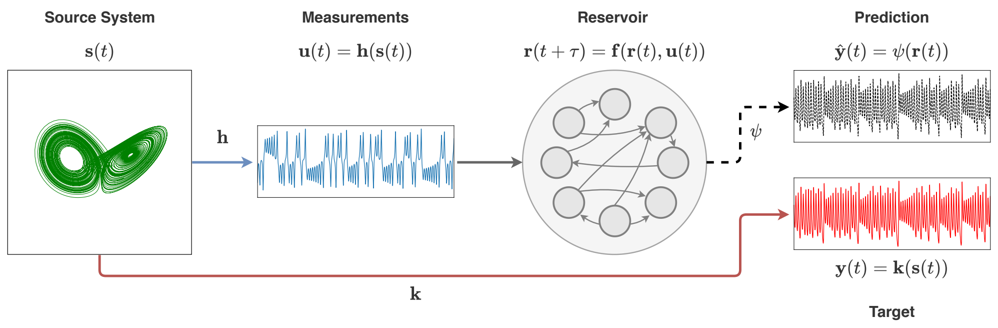

Source System
$\mathbf{s}(t)$
Measurements
$\mathbf{u}(t) = \mathbf{h}(\mathbf{s}(t))$
Reservoir
$\mathbf{r}(t+\tau) = \mathbf{f}(\mathbf{r}(t), \mathbf{u}(t))$
Prediction
$\hat{\mathbf{y}}(t) = \psi(\mathbf{r}(t))$
$\mathbf{h}$
ψ
$\mathbf{k}$
$\mathbf{y}(t) = \mathbf{k}(\mathbf{s}(t))$
Target

Figure 4.1. Diagram representing the RC framework trained with the RLA for a predicting scenario The source system $\boldsymbol{s}_t$ evolves autonomously and generates the targets $\boldsymbol{y}(t)$ and the input measurements $\boldsymbol{u}_t$. The latter is coupled to the reservoir $\boldsymbol{r}_t$ so that its dynamics are dependent on (i.e., driven by) $\boldsymbol{u}_t$. The readout $\boldsymbol{\psi}$ is then trained to generate the prediction $\hat{\boldsymbol{y}}_t$, which is an approximation of $\boldsymbol{y}_t$.

It is important to point out that this is an *offline* learning procedure: the enriched input representation is first created through the reservoir states and only then the readout function is computed in one shot.

4.1.2 Fitting using Least-Squares

The most popular way of training the readout in the fitting phase is based on the Least-Squares. Let us assume that our training data have times $t \in [t_0, 0]$. First we discard the initial states, as they may include some transient effects, so that we only use states starting from $t_s > t_0$. Our goals is then to find the $\boldsymbol{W}_{\text{out}}$ that minimizes the Mean Square Error (MSE):

$$\mathrm{MSE}(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) = \frac{1}{T_{\mathrm{train}}} \sum_{t=t_s}^{t_f} \|\boldsymbol{y}_t - \hat{\boldsymbol{y}}_t\|_2^2 \tag{4.1}$$

where $\boldsymbol{Y} = [\boldsymbol{y}_{t_s}, \ldots, \boldsymbol{y}_0]$ and $\hat{\boldsymbol{Y}} = [\hat{\boldsymbol{y}}_{t_s}, \ldots, \hat{\boldsymbol{y}}_0]$. Each estimation $\hat{\boldsymbol{y}}_t$ is given by (3.6).

To solve this problem, we define the state matrix as:

$$\boldsymbol{R} = \begin{bmatrix} | & | & | & | \\ \boldsymbol{r}_{t_s} & \boldsymbol{r}_{t_s+1} & \cdots & \boldsymbol{r}_0 \\ | & | & | & | \end{bmatrix} \tag{4.2}$$

The minimum of (4.1) can be found in closed form by means of the pseudo-inverse of this matrix:

$$\boldsymbol{W}_{\mathrm{out}} = \boldsymbol{Y}\boldsymbol{R}^{\dagger} \tag{4.3}$$

where

$$\boldsymbol{R}^{\dagger} := \boldsymbol{R}^{\top}(\boldsymbol{R}\boldsymbol{R}^{\top})^{-1}. \tag{4.4}$$

Usually, the problem requires some form of regularization and slightly more complex algorithms are used, like the *Ridge Regression*, which uses a regularized pseudo inverse which reads:

$$\boldsymbol{R}^{\dagger}_{\lambda} := \boldsymbol{R}^{\top}(\boldsymbol{R}\boldsymbol{R}^{\top} - \lambda\mathbb{I})^{-1}, \tag{4.5}$$

in which $\mathbb{I}$ is the *identity matrix* and λ is a *regularization parameter*.

4.1.3 Generating scenario

In the *generating scenario*, one uses the predicted output $\hat{\boldsymbol{y}}_t$ as an input, i.e., one sets $\boldsymbol{u}_t = \hat{\boldsymbol{y}}_t$. This leads to two very different situations: in the listening phase the reservoir behaves like an externally-driven dynamical system (this is named *open loop* phase). This procedure is usually named *teacher forcing* [Williams and Zipser, 1989]. Instead, in the predicting phase the system output behaves as feedback (*closed loop* phase), leading to an autonomous dynamical system of the form:

$$\boldsymbol{r}_{t+1} = \boldsymbol{f}(\boldsymbol{r}_t, \psi(\boldsymbol{r}_t)) \tag{4.6}$$

Online learning procedures may be exploited as well. For instance, in [Lu and Bassett, 2020] the (linear) readout is trained using a simple delta-rule of the form

$$\boldsymbol{W}_{\text{out}}^{(t+1)} = \boldsymbol{W}_{\text{out}}^{(t)} + \alpha(\boldsymbol{\Delta}_t \cdot \boldsymbol{r}_t^\top) \tag{4.7}$$

which aims at gradually minimizing the *discrepancy* $\boldsymbol{\Delta}_t := \boldsymbol{y}_t - \hat{\boldsymbol{y}}_t$ between the target and the output. A more sophisticated approach named *FORCE* learning [Sussillo and Abbott, 2009] was explicitely designed for the generating case. The authors make use of the Recursive Least-Squares (RLS) [Plackett, 1950] algorithm to directly adapt the network prediction $\hat{\boldsymbol{y}}$ to the desired output $\boldsymbol{y}$, avoiding the teacher forcing.

4.2 Hyperpameters

The simplicity of the RC training procedure comes at the cost of a complex hyper-parameters optimization, which is required due to the high variability of performance. RCNs are known to be sensitive to the setting of hyper-parameters like the Spectral Radius (SR), the input scaling and the sparseness degree [Jaeger and Haas, 2004], which critically affect their behavior and, hence, the performance on the task at hand. Various strategies can be applied to conduct this hyper-parameters optimization [Doan et al., 2021; Ferreira et al., 2013; Thiede and Parlitz, 2019] but it is hard to assess the performance of such procedures [Lukoševičius and Uselis, 2019; Racca and Magri, 2021; Haluszczynski and Räth, 2019]. Moreover, some efforts have been devoted to eliminating the randomness in the reservoir initialization to reduce the stochasticity of the optimization procedure [Rodan and Tino, 2010; Gallicchio et al., 2020]. In this section, we list the parameters which are more relevant for the RCN and states how some properties of the reservoir depends on them. A detailed guide with suggestion on how to tune them can be found in [Lukoševičius, 2012].

4.2.1 Reservoir Topology

The matrix connection $\boldsymbol{W}$ is usually generated at random, by sampling its element from a given distribution. Yet, other choices have been explored in the literature.

Delay Line In a *delay line* each neuron is connected to the subsequent one to form a chain-like architecture, so that the reservoir connection matrix $\boldsymbol{W}_{\text{d}}$

reads:

$$\boldsymbol{W}_{\mathrm{d},ij} = \delta_{i,j-1} \tag{4.8}$$

where δ is the *Kronecker delta*. Note that the last neuron in the chain is not connected to the first one. Moreover, the input weights vector is $\boldsymbol{w}_{\mathrm{d}} = (1, 0, 0, \ldots, 0)$, meaning that the input enters the network only through the first neuron of the chain. Mathematically, such a model setting corresponds to the n-th order AR model, which is a really popular and studied tool in time series analysis and system identification Bittanti [2019].

Cyclic Reservoir A reservoir is said to be cyclic when each neuron is connected to another one, in a way that they form a ring. The reservoir matrix of a cyclic reservoir has the form:

$$W_{\mathrm{c},ij} = \delta_{i,j-1} \tag{4.9}$$

where, with an abuse of notation, $\delta_{0,-1} := \delta_{0,n-1}$. From the product of Kroenecker deltas, it follows that:

$$W^2_{\mathrm{c},ij} = \sum_k W_{\mathrm{c},ik} W_{\mathrm{c},kj} = \sum_k \delta_{i,k-1}\delta_{k,j-1} = \delta_{i,j-2} \tag{4.10}$$

and the same holds for higher powers, so that $W^p_{\mathrm{c},ij} = \delta_{i,j-p}$.

Random Reservoir In a random reservoir the entries of the reservoir matrix $\boldsymbol{W}_{\mathrm{p}}$ are Independent and Identically Distributed (i.i.d.) random variables of null mean and variance equal to ρ/N.

$$W_{\mathrm{r},ij} \sim \mathscr{P}\left(0, \frac{\rho^2}{N}\right) \tag{4.11}$$

In the sequel, we consider the generic ij component of the matrix to be drawn from a Gaussian distribution, i.e., $\mathscr{P}\left(0, \frac{\rho^2}{N}\right) = \mathscr{N}\left(0, \frac{\rho^2}{N}\right)$ as the choice of the particular distribution does not effect the behavior [Zhang et al., 2011].

Wigner Reservoir The diagonal elements are distributed as in (4.11), i.e., $W_{\mathrm{w},ii} \sim \mathscr{N}(0, \rho_1^2/n)$, while off-diagonal elements follow:

$$W_{\mathrm{w},ij} = W_{\mathrm{w},ji} \sim \mathscr{N}\left(0, \frac{\rho_2^2}{N}\right), \quad i \neq j \tag{4.12}$$

Wigner matrices are symmetric. In this work, we will always set $2\rho_1 = \rho_2 = \rho$. Notably, this leads to $\langle\rho(\boldsymbol{W}_{\mathrm{p}})\rangle = \langle\sigma_{\max}(\boldsymbol{W}_{\mathrm{p}})\rangle = \rho$.

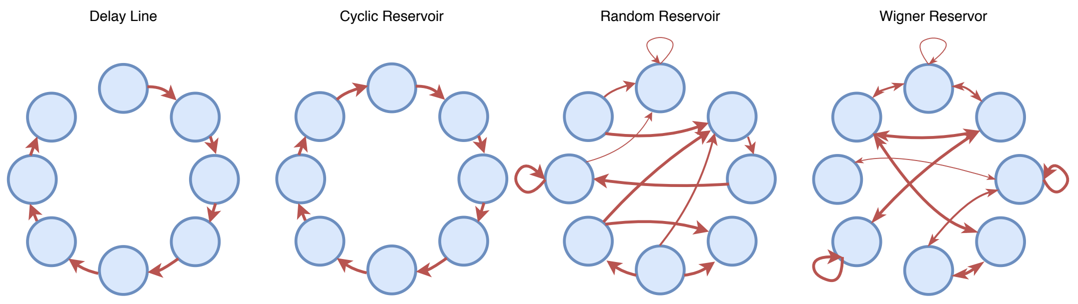


Figure 4.2. The different architectures discussed in this work. From left to right: delay line, cyclic, random and Wigner topology. The thickness of the arrow accounts for the strength of the connection. Notice that it has the same value for all the connections in both the delay line and the cyclic reservoir, while it varies for the other two. Note the presence of self loops in random and Wigner architectures. Also consider that the Wigner reservoir has only symmetric connections (double-headed arrows).

4.2.2 Spectral Radius

Given a square matrix $\boldsymbol{W}$, its Spectral Radius (SR) $\rho = \rho(\boldsymbol{W})$ is defined as the largest absolute value of its eigenvalues. Let $\lambda_1, \dots, \lambda_n$ be the eigenvalues of the square matrix $\boldsymbol{W} \in \mathbb{R}^{n \times n}$. The SR ρ is then defined as:

$$\rho := \max_{i=1,2,\dots,N} \{|\lambda_i|\} \tag{4.13}$$

The name Spectral Radius is due to the fact that all the (possibly complex) eigenvalues of $\boldsymbol{W}$ lie in a circle of radius ρ, see Figure 4.3 In particular when the entries of the matrix W are i.i.d. like in (4.11) the so-called *Circular Laws* applies:

Theorem 1 (Circular Law [Tao et al., 2010]). *Let the Empirical Spectral Distribution (ESD) of a square matrix $\boldsymbol{W} \in \mathbb{R}^{N \times N}$ be defined as:*

$$\mu_W := \frac{1}{N} |\{i : 1 \leq u \leq N, \Re(\lambda_i) \leq s, \Im(\lambda_i) \leq t\}| \tag{4.14}$$

where $|\cdot|$ denotes the cardinality *of a set and $\Re$ and $\Im$ respectively denote the Real and Imaginary part of λ_i, the eigenvalues of $\boldsymbol{W}$. Moreover, let $\boldsymbol{W}_N := \frac{\boldsymbol{W}}{\sqrt{N}}$*

Then, if the entries of of $\boldsymbol{W}$ are i.i.d. complex random variables with mean 0 and variance 1, the ESD of $\boldsymbol{W}_N$ converges to the uniform distribution on the unit disk as $N \to \infty$, both in the strong and weak senses.

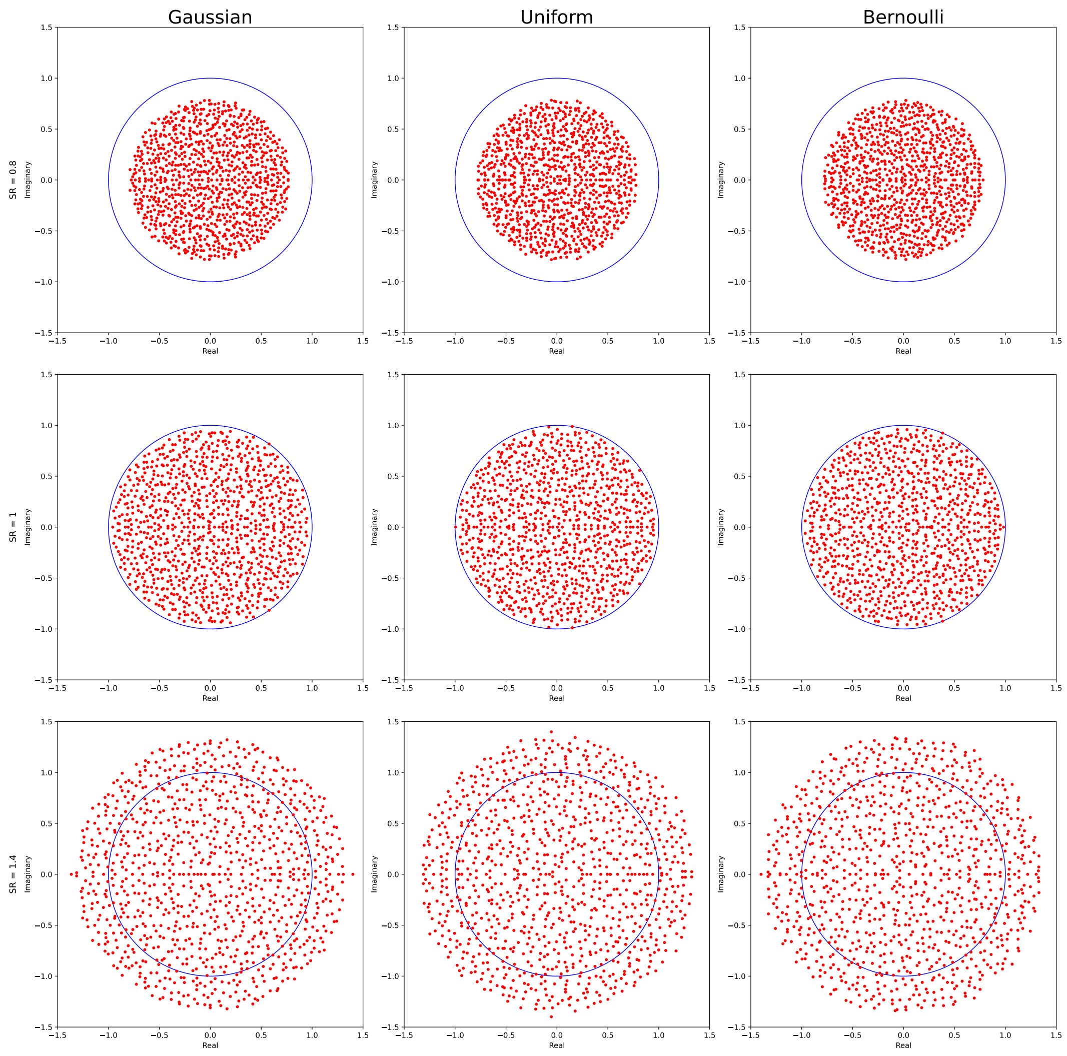


Figure 4.3. Eigenvalues for the of the connectivity matrix for different SR and different distributions for the entries of the matrix. In the "Gaussian" case the entries are drafted from the normal distribution with null mean and unit variance. In the "Uniform" case we used a uniform distribution in $[-1,1]$. "Bernoulli" accounts for entries that are equal to 1 or -1 with equal probability. Matrices are then rescaled to match the reported SR values. The blue line accounts for the unit circle.

In the context of RC and in particular of RCNs, the SR of the connectivity matrix $\boldsymbol{W}$ plays a crucial role as it characterize the dynamics of the reservoir state. In particular it affects the *nonlinearity* of the reservoir and its *memory*, i.e., the ability of encoding past inputs in its state.

The effect of the SR on the nonlinearity can be easility understood when considering the shape of the reservoir of the RCN (11.35) in which the activation function σ is chosen to be tanh [Bollt, 2021]. For small SR, each neuron operates near the origin where the tanh is approximately linear. Increasing the value of the SR drives the network further from 0, thus moving the working point towards the nonlinear regions of tanh.

SR plays a crucial role in the satisfaction of the so-called ESP, as discussed in Section 7.2. The effect on memory of the SR is discussed throughout Chapter 6. In the linear case, i.e., when $\sigma(\boldsymbol{r}) = \boldsymbol{r}$, a the role played by SR can be studied analytically, see Chapter 8 for a detailed analysis.

4.2.3 Readout

In the spirit of RC a complete representation of the driving dynamics system should be present in the reservoir state $\boldsymbol{r}_t$. For this reason, a linear readout like (3.6) is usually the most common choice as it is both easy to train and it avoids overfitting due to its simplicity.

Recently [Lu et al., 2017] a variation of the linear readout accounting for a square term was proposed. Its general form can be written as:

$$\boldsymbol{\psi}(\boldsymbol{r}_t) = \boldsymbol{W}_{\text{out}}^{(1)}\boldsymbol{r}_t + \boldsymbol{W}_{\text{out}}^{(2)}\boldsymbol{r}_t^2 \tag{4.15}$$

in which we introduced the *square state* $\boldsymbol{r}_t^2 := [r_{0,t}^2, r_{1,t}^2, \dots, r_{N,t}^2]$ and $\boldsymbol{W}_{\text{out}}^{(1)}$ and $\boldsymbol{W}_{\text{out}}^{(2)}$ are the readout of the linear and square part respectively. Of course, (4.15) can be also understood as a linear readout which operates on the extended state $\boldsymbol{r}_t^{\text{ext}} := [\boldsymbol{r}_t, \boldsymbol{r}_t^2]$:

$$\boldsymbol{\psi}(\boldsymbol{r}_t) = \boldsymbol{W}_{\text{out}}\boldsymbol{r}_t^{\text{ext}} \tag{4.16}$$

where $\boldsymbol{W}_{\text{out}}$ is simply the concatenation of $\boldsymbol{W}_{\text{out}}^{(1)}$ and $\boldsymbol{W}_{\text{out}}^{(2)}$. This form emphasizes the equivalence with the linear readout (3.6). Note that this procedure is different from a quadratic fitting, in which the mixed components (of the form $r_{i,t}r_{j,t}$) are present.

The readout with quadratic term was first introduced in [Lu et al., 2017] to overcome the impossibility of recovering the full state of the Lorenz system (Appendix C.1) by reading only one of its component, due to the symmetry of its dynamics. This issue was later extensively studied by Herteux and Räth [2020] in which the authors compares various ways to deal with this symmetry. The relationship between the quadratic term of the readout and the symmetry in the source system is explored in Flynn et al. [2021]. Notice that it is also possible to exploit these symmetries through a tailored design of the RCN Barbosa et al. [2021].

4.2.4 Other Hyperparameters

Number of Units The number of units in the reservoir N controls the size of the reservoir. In general, larger reservoirs lead to better results: this is due to the fact that they generally lead to more diverse representation of the input signal into the states and, so, the readout receives a more varied input that is easier to fit. Yet, for this reason, using large reservoirs might lead to overfitting, and some regularization measures must be employed Lukoševičius [2012]. As a lower bound for N, one should consider the minimal number of past-inputs that the network has to remember in order to produce the correct output. See Section 6.1 for a detailed analysis.

Input Scaling The input scaling parameters controls the importance that the input has on the network dynamics. Usually, it is tuned by controlling the norm of the *input matrix* $\boldsymbol{w}$. One first creates a matrix $\tilde{\boldsymbol{w}}$ by sampling its entries at random (usually from a Normal distribution $\mathcal{N}(0,1)$ and the setting $\boldsymbol{w} := \omega\tilde{\boldsymbol{w}}$, where ω is a scalar named *input scaling* factor. Choosing small value of ω means that the input will have a minor influence on the state dynamics, down to the limit case $\omega = 0$ for which the input is completely irrelevant. Instead, setting ω to be large leads to a stronger dependence of the state from the input driving the system. In general, it hard to asses what the optimal value of ω should be for a given task, and some hyper-parameter tuning must be carried by means of cross-validation.

Sparsity In the seminal paper by Jaeger [2001], the *reservoir connection matrix* $\boldsymbol{W}$ was chosen to be sparse: 95% of its connections were set to 0. This was done following the intuition that decoupling the state variables would lead to a richer representation of the input signal they encode. Later studies have empirically discredited this idea [Schrauwen et al., 2007; Xue et al., 2007; Gallicchio and Micheli, 2011; Carroll and Pecora, 2019; Gallicchio, 2020], but sparse designs are still used both as a way to speedup computations and to reduce the number of connections required in hardware implementations of RC. Sparsification of $\boldsymbol{W}$ is usually controlled by a parameter α which controls the probability of the matrix entries W_{ij} to be non-zero, meaning that the matrix is fully connected for $\alpha = 0$. Another possibility is to control the mean degree $\langle k \rangle$ of each neuron. The two approaches are related by the formula $\langle k \rangle = \alpha N$, which is simply the expected value of Bernoulli process. Notably, in a recent work, the sparsification of the *input matrix* $\boldsymbol{w}$ lead to a performance improvement [Gallicchio, 2020].

Bias The role of the bias term ($\boldsymbol{b}$ in Eq. 3.5) has recently gained attention in the community. The bias act as a (constant) shift in the input of each neuron and controls whether the unit will operate near its linear region or near the saturating one, as discussed above for the input signal – which can be viewed as a time varying bias. In Herteux and Räth [2020] the role of bias as a *symmetry breaking* tool was highlighted: there, it is shown that the symmetries in the source systems give raise to the prediction of *mirror attractors* which, in turn, may affect the performance in the task at hand. The bias term breaks this symmetry, thus destroying the mirror attractors.

Chapter 5

Self-Normalizing Activation Function

In this chapter we present a novel activation function for RCNs, namely the *self-normalizing activation function*, based on projecting the reservoir state to an hyper-sphere. In section 5.1 the model is described and a proof that it is a universal function approximator is given. Section 5.2 describes its autonomous dynamics, showing that it cannot display a chaotic behavior. Finally, in Section 5.3 an analysis of the dynamics of networks states when driven by an external input is presented.

5.1 Self-Normalizing Activation Function

Here, we propose a new model for RCN characterized by the use of a particular self-normalizing activation function that provides important features to the resulting network. Notably, the proposed activation function allows the network to exhibit nonlinear behaviors and, at the same time, provides memory properties similar to those observed for linear networks. This superior memory capacity is linked to the fact that the network never displays a chaotic behavior: we will show that the maximum Lyapunov exponent is always zero, implying that the network operates on the Edge of Criticality (EoC). The proposed activation function guarantees that the SR of the reservoir matrix (whose value is used as a control parameter) can vary in a wide range without affecting the network stability.

The proposed self-normalizing activation function is

$$\boldsymbol{a}_t = \boldsymbol{W}\boldsymbol{r}_{t-1} + \boldsymbol{w}\boldsymbol{u}_{t-1} \tag{5.1a}$$
$$\boldsymbol{r}_t = p\frac{\boldsymbol{a}_t}{\|\boldsymbol{a}_t\|} \tag{5.1b}$$

The normalization in Eq. 5.1b projects the network pre-activation a_t onto an $(N-1)$-dimensional hyper-sphere $\mathbb{S}_p^{N-1} := \{\boldsymbol{r} \in \mathbb{R}^N, \|\boldsymbol{r}\| = p\}$ of radius p. Fig. 5.1 illustrates the normalization operator applied to the state. Note that the operation (5.1b) is *not* element-wise like most of activation functions as its effect is global, meaning that a neuron's activation value depends on all other values.

5.1.1 Universal Function Approximation Property

Here, we show that the universal function approximation property also holds for the proposed RCNs model (5.1). To this end, we note that $\sigma_i(\boldsymbol{x}) := x_i/\|\boldsymbol{x}\|$ can be intended as a squashing function (see Def. 1) for each i-th component. As discussed in 3.3.2, a RCN is a universal function approximator provided it satisfies the *contractivity condition*. In order to prove it for our system, let us introduce the the notation $\hat{\boldsymbol{x}} := \boldsymbol{x}/\|\boldsymbol{x}\|$ and $\hat{\boldsymbol{y}} := \boldsymbol{y}/\|\boldsymbol{y}\|$, valid when both norms are non-null. Taking the square of the norm, one gets:

$$\|\sigma(\boldsymbol{x}) - \sigma(\boldsymbol{y})\|^2 = p^2(2 - 2(\hat{\boldsymbol{x}}\cdot\hat{\boldsymbol{y}})) \leq p^2\left(\left(\frac{\|\boldsymbol{x}\|}{\|\boldsymbol{y}\|} + \frac{\|\boldsymbol{y}\|}{\|\boldsymbol{x}\|}\right) - 2\hat{\boldsymbol{x}}\cdot\hat{\boldsymbol{y}}\right) \tag{5.2}$$

The inequality $\frac{a}{b} + \frac{b}{a} \geq 2$ follows from the fact that $(a-b)^2 = a^2 + b^2 - 2ab > 0$ for all $a, b > 0$. Now, we assume $\|\boldsymbol{x}\|, \|\boldsymbol{y}\| > p$ and show that:

$$\begin{aligned}
\|\sigma(\boldsymbol{x}) - \sigma(\boldsymbol{y})\| &\leq p^2 \cdot \left(\frac{\|\boldsymbol{x}\|}{\|\boldsymbol{y}\|} + \frac{\|\boldsymbol{y}\|}{\|\boldsymbol{x}\|} - 2\hat{\boldsymbol{x}}\cdot\hat{\boldsymbol{y}}\right) && (5.3)\\
&\leq \|\boldsymbol{x}\|\|\boldsymbol{y}\|\left(\frac{\|\boldsymbol{x}\|}{\|\boldsymbol{y}\|} + \frac{\|\boldsymbol{y}\|}{\|\boldsymbol{x}\|} - 2\hat{\boldsymbol{x}}\cdot\hat{\boldsymbol{y}}\right) && (5.4)\\
&= \|\boldsymbol{x}\|^2 + \|\boldsymbol{y}\|^2 - \underbrace{2(\|\boldsymbol{x}\|)(\|\boldsymbol{y}\|)\hat{\boldsymbol{x}}\cdot\hat{\boldsymbol{y}}}_{2x\cdot y} && (5.5)\\
&= \|\boldsymbol{x} - \boldsymbol{y}\|_2^2 && (5.6)
\end{aligned}$$

proving the contractivity condition (3.7).

We see that the only condition needed is $||\boldsymbol{x}||, ||\boldsymbol{y}|| > p$, which means that the linear part of the update (5.1a) must map states outside the hyper-sphere of radius p. Finally, by applying properties of norms, we observe that:

$$||\boldsymbol{W} \cdot \boldsymbol{x}_t + \boldsymbol{w} \cdot \boldsymbol{u}_t|| \geq ||\boldsymbol{W} \cdot \boldsymbol{x}_t|| - ||\boldsymbol{w} \cdot \boldsymbol{u}_t|| \geq \sigma_{min}(\boldsymbol{W})p - ||\boldsymbol{w}||||\boldsymbol{u}_{max}|| \tag{5.7}$$

and asking this to be larger than p results in the condition:

$$\sigma_{\min}(\boldsymbol{W}) \geq 1 + \frac{||\boldsymbol{w}||||\boldsymbol{u}_{\max}||}{p} \tag{5.8}$$

where $\sigma_{\min}(\boldsymbol{W})$ is the smallest singular value of matrix $\boldsymbol{W}$ and $||\boldsymbol{u}_{\max}||$ denotes the largest norm associated to an input.

We emphasize that (5.8) results in a sufficient yet not necessary condition that may be understood as requiring that the input will never be strong enough to contrast the expansive dynamics of the system, leading the network state inside the hyper-sphere of radius p. In fact, unless the signal is explicitly designed for violating such a condition, it will very likely not bring the system inside the hyper-sphere as long as the norm of $\boldsymbol{W}$ is large enough compared to the signal variance.

5.2 Network State Dynamics: the Autonomous Case

We now discuss the network state dynamics in the autonomous case, i.e., in the absence of input. This allows us to prove why the network cannot be chaotic.

From now on, we assume $p = 1$ as this does not affect the dynamics, provided that condition (5.8) is satisfied. From (5.1), the system state dynamics in the autonomous case reads:

$$\sigma(\boldsymbol{r}) := \frac{\boldsymbol{W}\boldsymbol{r}}{||\boldsymbol{W}\boldsymbol{r}||} \tag{5.9}$$

At time-step t the system state is given by

$$\boldsymbol{r}_t = \sigma(\boldsymbol{r}_{t-1}) = \frac{\boldsymbol{W}\boldsymbol{r}_{t-1}}{||\boldsymbol{W}\boldsymbol{r}_{t-1}||} = \frac{\boldsymbol{W}}{||\boldsymbol{W}\boldsymbol{W}\frac{\boldsymbol{r}_{t-2}}{\cancel{||\boldsymbol{W}\boldsymbol{r}_{t-2}||}}||}\frac{\boldsymbol{W}\boldsymbol{r}_{t-2}}{\cancel{||\boldsymbol{W}\boldsymbol{r}_{t-2}||}} = \frac{\boldsymbol{W}^2\boldsymbol{r}_{t-2}}{||\boldsymbol{W}^2\boldsymbol{r}_{t-2}||} \tag{5.10}$$

By iterating this procedure, one obtains:

$$\boldsymbol{r}_t := \sigma^t(\boldsymbol{r}_0) = \frac{\boldsymbol{W}^t\boldsymbol{r}_0}{||\boldsymbol{W}^t\boldsymbol{r}_0||} \tag{5.11}$$

where $\boldsymbol{r}_0$ is the initial state. This implies that, for the autonomous case, a system evolving for t time-steps coincides with updating the state by performing t matrix multiplications and projecting the outcome only at the end.

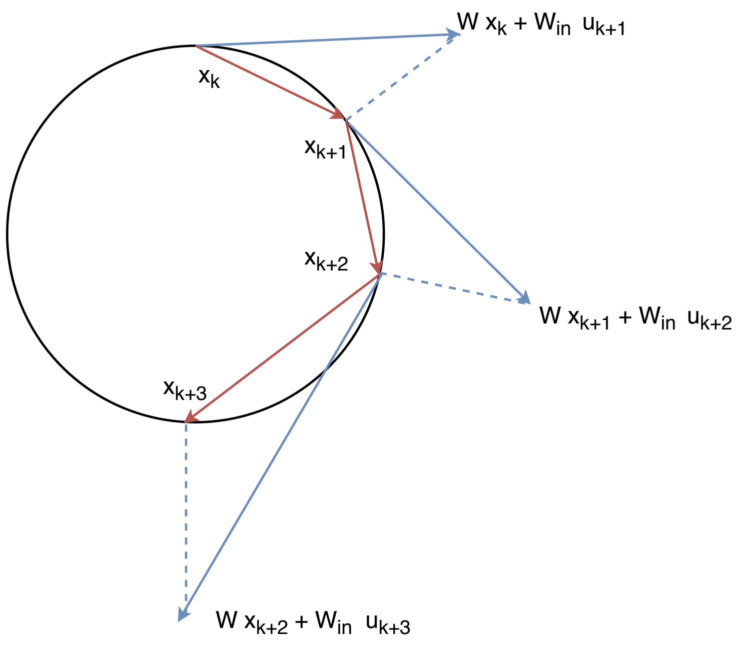


Figure 5.1. Example of the behavior of the proposed model in a 2-dimensional scenario. The blue lines represent the linear update step of Eq. (5.1a), while the dashed lines denote the projection of Eq. (5.1b). The red lines represent the actual steps performed by the system. Note that condition (5.8) accounts for the fact that the linear step must never bring the system state inside the hyper-sphere.

It is worth to comment that this holds also if matrix $\boldsymbol{W}$ changes over time. In fact, let $W_t := \boldsymbol{W}(t)$ be $\boldsymbol{W}$ at time time t. Then, the evolution of the dynamical system reads:

$$\boldsymbol{r}_t := \sigma^t(\boldsymbol{r}_0) = \frac{W_t W_{t-1} \dots W_2 W_1 \boldsymbol{r}_0}{||W_t W_{t-1} \dots W_2 W_1 \boldsymbol{r}_0||} \tag{5.12}$$

Furthermore, note that a system described by matrix $\boldsymbol{W}$ and a system characterized by $\boldsymbol{W}' = a\boldsymbol{W}$ coincide. In turn, this implies that the *SR of the matrix does not alter the dynamics in the autonomous case.*

5.2.1 Edge of Chaos

When tuning the hyper-parameters of RCNs, one usually tries to bring the system close to the EoC, since it is in that region that their performance appears to be optimal [Livi et al., 2017]. This can be explained by the fact that, when operating in that regime, the system introduces rich dynamics without denoting chaotic behavior. See Section 10.2.1

Here, we show that the proposed recurrent model (5.1) cannot enter a chaotic regime. Notably, we prove that, when the number of neurons in the network is large, the maximum (local) Lyapunov exponent cannot be positive, hence neglecting the possibility to introduce a chaotic behavior. To this end, we determine the Jacobian matrix of (5.1b) and then show that, since its spectral radius tends to 1, the maximum Local Lyapunov Exponent (LLE) must be null. The Jacobian matrix of (5.1b) reads:

$$J_{ij} = \frac{\partial}{\partial x_i}\sigma_j(\boldsymbol{r}) = \sum_l^t \frac{\partial \sigma_j}{\partial a_l}\frac{\partial a_l}{\partial x_i} = \frac{W_{ij}}{\|\boldsymbol{a}\|}\left(1 - \frac{a_i a_j}{\|\boldsymbol{a}\|^2}\right) \tag{5.13}$$

where the time index k is omitted to ease the notation. We observe that, asymptotically for large networks ($N \to \infty$), we have that $a_i/\|\boldsymbol{a}\| \to 0$ for each i, meaning that the Jacobian matrix reduces to $J(\boldsymbol{r}) = \boldsymbol{W}/\|\boldsymbol{W}\boldsymbol{r}\|$. As we are considering the case with $p = 1$, we know that $\|\boldsymbol{r}\| = 1$.

This allows us to approximate the norm of $\boldsymbol{W}$ with its SR $\rho = \rho(\boldsymbol{W})$,

$$J \approx \frac{\boldsymbol{W}}{\|\boldsymbol{W}\boldsymbol{r}\|} \approx \frac{\boldsymbol{W}}{\rho}. \tag{5.14}$$

Under this approximation (5.14), the largest eigenvalue of J must be 1 as the SR ρ is the largest absolute value among the eigenvalues of $\boldsymbol{W}$. We thus characterize the global behavior of (5.1b) by considering the maximum Local Lyapunov Exponent (LLE) [Livi et al., 2017], which is defined as:

$$\Lambda := \lim_{t\to\infty} \frac{1}{t} \log\left(\prod_t^k \rho_k\right) \tag{5.15}$$

where ρ_k is the spectral radius of the Jacobian at time-step k. Eq. 5.15 implies that $\Lambda = 0$ as $t \to \infty$, hence proving our claim.

In order to demonstrate that $\Lambda = 0$ holds also for networks with a finite number N of neurons in the recurrent layer, we numerically compute the maximum LLE by considering the Jacobian in (5.13). The results are displayed in Fig. 5.2. Fig. 5.2 panel (a) shows the average value of the maximum LLE with the related standard deviation obtained for different values of SR. Results show that the LLE is not significantly different from zero. In Fig. 5.2 panel (b), instead, we show the Lyapunov spectrum of a network with $N = 100$ neurons in the recurrent layer, obtained for different SR values. Again, our results show that the maximum LLE of (5.1b) is zero for finite-size networks, regardless of the values chosen for the SR.

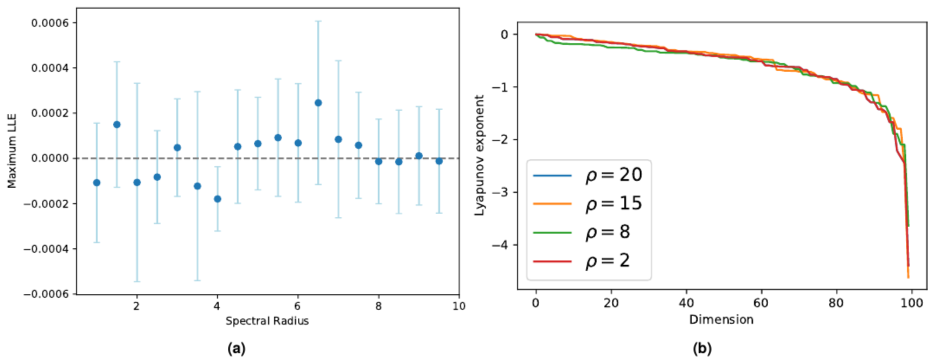


Figure 5.2. Panel (a) shows LLEs for different value of the SR. Each point in the plot represents the mean of 10 different realizations, using a network with $N = 500$ neurons. In panel (b) examples of the Lyapunov spectrum for different 100-dimensional networks are plotted. The Lyapunov exponents are ordered by decreasing magnitude. Note that the largest exponent is always zero.

5.3 Network State Dynamics: the Input-Driven Case

We now study the dynamics of (5.1b) when the system is driven by an input. To simplify the notation, let us define the *effective input* contributing to the neuron activation as

$$\boldsymbol{x}_t := \boldsymbol{w}\boldsymbol{u}_t. \tag{5.16}$$

Accordingly, Eq. 5.1a takes on the following form:

$$\boldsymbol{a}_t = \boldsymbol{W}\boldsymbol{r}_{t-1} + \boldsymbol{x}_t, \tag{5.17}$$

where $\boldsymbol{x}_k$ operates as a time-dependent bias vector. Let us define the normalization factor as:

$$N_t = \|\boldsymbol{W}\boldsymbol{r}_{t-1} + \boldsymbol{x}_t\|, \tag{5.18}$$

Consider (5.1): if we explicitly expand the first steps from the initial state $\boldsymbol{r}_0$ we obtain:

$$r_1 = \frac{\boldsymbol{W}\boldsymbol{r}_0 + \boldsymbol{x}_1}{\|\boldsymbol{W}\boldsymbol{r}_0 + \boldsymbol{x}_1\|} = \frac{\boldsymbol{W}\boldsymbol{r}_0 + \boldsymbol{r}_1}{N_1} = \frac{\boldsymbol{W}}{N_1}\boldsymbol{r}_0 + \frac{1}{N_1}\boldsymbol{x}_1 \tag{5.19}$$

$$r_2 = \frac{\boldsymbol{W}\boldsymbol{r}_1 + \boldsymbol{x}_1}{N_2} = \frac{\boldsymbol{W}^2}{N_2 N_1}\boldsymbol{r}_0 + \frac{\boldsymbol{W}}{N_2 N_1}\boldsymbol{x}_1 + \frac{1}{N_2}\boldsymbol{x}_2 \tag{5.20}$$

$$r_3 = \frac{\boldsymbol{W}\boldsymbol{r}_2 + \boldsymbol{x}_3}{N_3} = \frac{\boldsymbol{W}^3}{N_3 N_2 N_1}\boldsymbol{r}_0 + \frac{\boldsymbol{W}^2}{N_3 N_2 N_1}\boldsymbol{x}_1 + \frac{\boldsymbol{W}}{N_3 N_2}\boldsymbol{x}_2 + \frac{1}{N_3}\boldsymbol{x}_3 \tag{5.21}$$

So that the general case can be written as:

$$r_t = \frac{\boldsymbol{W}\boldsymbol{r}_{t-1} + \boldsymbol{x}_t}{N_t} = \frac{\boldsymbol{W}^t}{N_t N_{t-1}\dots N_1}\boldsymbol{r}_0 + \frac{\boldsymbol{W}^{t-1}}{N_t N_{t-1}\dots N_1}\boldsymbol{x}_1 + \frac{\boldsymbol{W}^{t-2}}{N_t N_{t-1}\dots N_2}\boldsymbol{x}_2 + \dots + \frac{1}{N_t}\boldsymbol{x}_t \tag{5.22}$$

Then, the state at time-step t can be written as:

$$\boldsymbol{r}_t = M^{(t,0)}\boldsymbol{r}_0 + \sum_{k=1}^{t} M^{(t,k)}\boldsymbol{x}_k \tag{5.23}$$

where

$$M^{(t,k)} := \frac{\boldsymbol{W}^{t-k}}{\prod_{l=k}^{t} N_l}. \tag{5.24}$$

By looking at (5.23), it is possible to note that each $\boldsymbol{x}_t$ is multiplied by a time-dependent matrix, i.e., the network's final state $\boldsymbol{r}_t$ is obtained as a linear combination of all previous inputs. Eq. 5.24 allows one to study the memory properties of these networks from a theoretical perspective, see Appendix B for details. An experimetal analysis is presented in Chapter 6, Sections 6.3 and 6.4.

Part II

Representation

Chapter 6

Memory

In this Chapter we analyze the *memory* of RC systems, as it is fundamental a feature that systems dealing with temporal data needs to display. In Section 6.1 we introduce the *Memory Capacity*, which is the most popular measure of memory in the field. The rest of the Chapter is dedicated to experiments concerning the memory. In particular, Section 6.2 compares the memory of different architectures, showing that the circle-topology achieves leading results. Section 6.3 compares different activation functions, including the self-normalizing activation described in Chapter 5. Finally, 6.4 explores the *Memory-Nonlinearity Trade-off* typical of dynamical systems, in which the self-normalizing activation show better results when both memory and nonlinearity are required.

6.1 Memory Capacity

To quantify the memory of a RC system, one can measure its ability to represent past events in its state. The Memory Capacity (MC) [Jaeger, 2002] was defined with this intent, as a measure correlating the past inputs with the network outputs. More in detail, consider a system like (3.3) driven by a 1-dimensional signal $\{u_t\}$. We then train the network to reproduce the input with a given delay τ, i.e., we set the target $y_t = u_{t-\tau}$ and we obtain our approximation $\hat{y}_t$ through the readout (3.4). Then we define:

$$\mathrm{MC}_\tau := \frac{\mathrm{Cov}^2(u_{t-\tau}, \hat{y}_t)}{\mathrm{Var}(s_t)\mathrm{Var}(y_t)}, \tag{6.1}$$

where Cov and Var are the *Covariance* and the *Variance*, respectively. Eq.6.1 can be understood as the amount of information about the value of the input signal that can be retrieved from the state after τ time-steps have elapsed. Usually, one chooses u_t to be an i.i.d. signal, so that the knowledge of different – more recent – time-steps cannot be exploited by the network. Plotting the values of (6.1) as a function of τ for a given system results in the *Memory Curves* [Jaeger, 2002] (see Section 6.2).

The total short-term *Memory Capacity* is then obtained by:

$$\mathrm{MC} = \sum_{\tau=0}^{\infty} \mathrm{MC}_d \tag{6.2}$$

Note that in [Jaeger, 2002] it is proven that, under the assumption of i.i.d. input signal, for *any* RNN with N recurrent units, the maximum MC achievable is N. This result was later extended in [Gonon et al., 2020b], where a bound for nonlinear networks is provided and conditions for the MC to equal N are given.

The analysis of the memory capacity (as measured by the ability of the network to reconstruct or remember past inputs) of input-driven systems plays a fundamental role in the study of RCNs [Tiňo and Rodan, 2013; Goudarzi et al., 2016; Jaeger, 2002; Ganguli et al., 2008].

Moreover, it has been recently shown that optimizing memory capacity does not necessarily lead to networks with higher prediction performance [Marzen, 2017].

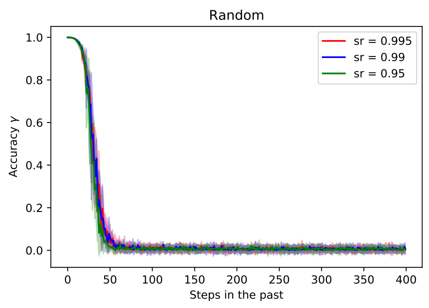
6.2 Memory Curves

In order to study the memory properties of linear RCNs, we design networks trained to remember a random i.i.d. input. We generate inputs of length T and split them in a *training set* ranging in $(-L_{\text{train}}, 0)$ and a *test set* ranging in $(0, L_{\text{test}})$. We set these values so that the training set contains $L_{\text{train}} = 1000$ samples and the test set $L_{\text{test}} = 500$ samples. In the task under consideration, the network is trained to reproduce past input (a white noise signal) at a given past-horizon τ, so that $y_t = u_{t-\tau}$. The input signal $\{u_t\}$ is chosen to be Gaussian i.i.d. white noise, $u_t \sim \mathcal{N}(0, 1)$.

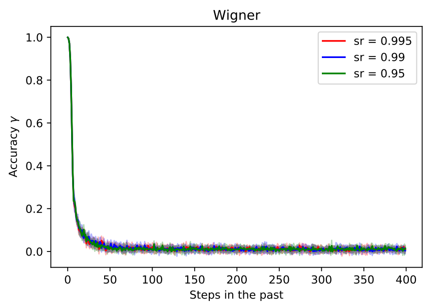
The readout is finally configured through a least-squares procedure and its *accuracy* is then evaluated on the test set as $\gamma = \max\{1 - \mathrm{NRMSE}, 0\}$, where the Normalized Root Mean Squared Error (NRMSE):

$$\mathrm{NRMSE} := \sqrt{\frac{\sum_{t=t_0}^{T}(y_t - \hat{y}_t)^2}{\sum_{t=t_0}^{T}(y_t - \overline{y})^2}} \tag{6.3}$$

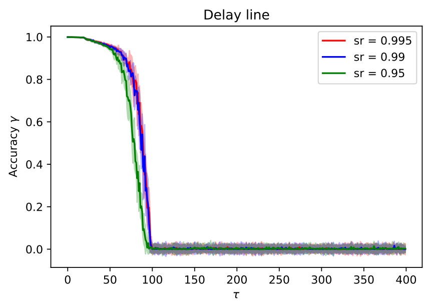
y_t denotes the system output at time t, $\overline{y} := \frac{1}{T}\sum_{t=t_0}^{T} y_t$ is its empirical average and $\hat{y}_t$ stands for the predicted output. Note that (6.3) is an estimation of the memory capacity (6.1).



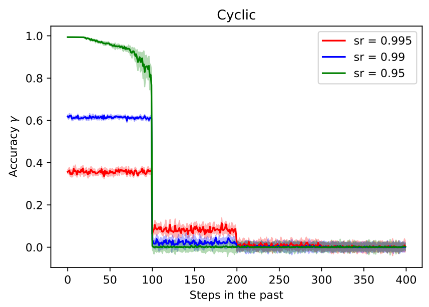
(a)



(b)



(c)

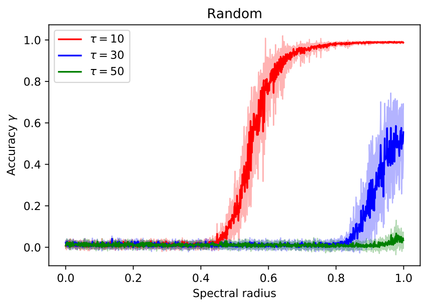


(d)

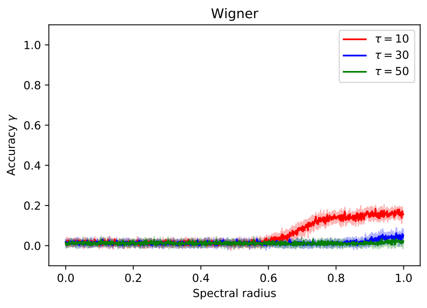
Figure 6.1. Memory curves for the Random (a), Wigner (b), Delay Line (c) and Cyclic (d) reservoirs with different values of ρ, for a reservoir of $N = 100$ neurons. Plotted values are averages over 10 different realizations, with the shaded area accounting for a standard deviation: a side effect of plotting data in this way is that the value may be negative even if the accuracy is defined as a positive quantity. Note that having a high ability to reconstruct recent inputs ($\tau < N$) compromises the capacity to remember the more distant ones.

In Figure 6.1 the *Memory Curves*, are plotted for four different architectures. The Random and the Wigner architectures appear to have a short memory. Their accuracy dramatically decreases as τ grows. The behavior of the cyclic reservoir appears to be radically different. As described in Rodan and Tino [2010], the performance does not decrease gradually, but remains almost constant for some time and then abruptly decreases. The drop in performance occurs when $\tau = N$, N being the number of neurons in the reservoir. Note that in [Jaeger, 2002, Section 3] a similar shape for the memory curve is obtained by using an “almost unitary” reservoir matrix, where all singular values equal a constant $C < 1$. We note that also the cyclic reservoir $\boldsymbol{W}_c$ shares this feature, explaining why the results are similar.

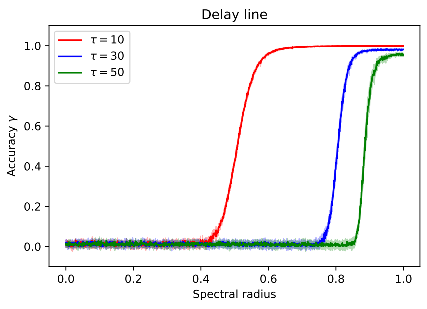
We comment on how the SR affects the performance: when the SR is close to one, the accuracy in the reconstruction of $\boldsymbol{u}_{t-\tau}$ is lower for recent inputs samples (i.e., smaller τ) but higher for distant in time ones. In other words, choosing a large ρ allows the network to better remember the distant past, at the price of compromising its ability to remember the recent one.



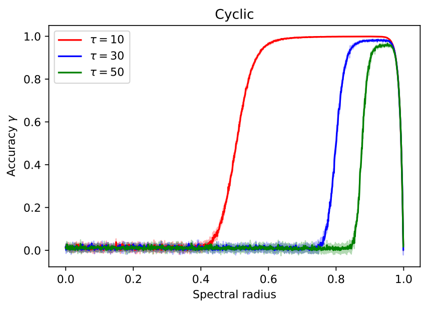
(a)



(b)



(c)



(d)

Figure 6.2. Accuracy in remembering an i.i.d. past-input as function of the spectral radius. All the networks have $N = 100$ neurons. Plotted values are averages over 10 different realizations, with the shaded area accounting for a standard deviation: a side effect of plotting data in this way is that the value may be negative even if the accuracy is defined as a positive quantity.

We investigate the impact of the spectral radius on memory capacity in Figure 6.2, where the accuracy γ of the four architectures in recalling a past input (at various τ) is plotted as function of the SR. We see that, a larger SR is required to correctly recall inputs that are further in past (but for which $\tau < N$), since the SR controls the magnitude of the $\phi_j^{(k)}$, i.e., the permanence of u_k on the state. We notice that the Random and the Wigner architectures show a similar behavior, with the former displaying a superior performance than the latter. Instead, the Cyclic network has the same behavior as the SR increases, but displays an abrupt fall as it approaches 1.

6.3 Different Activation Functions

We now study how the activation function σ affects the memory of RCNs. We compare models with three different σ: a tanh activation, a linear one (3.8) and the self-normalizing activation discussed in Chapter 5 (5.1). We use fixed, but reasonable hyper-parameters for all networks, since in this case we are only interested in analyzing the network behavior on different tasks. In particular, we selected hyper-parameters that worked well in all cases taken into account; in preliminary experiments, we noted that different values did not result in substantial (qualitative) changes of the results. The number of neuron N is fixed to 1000 for all models. For linear and nonlinear networks, the input scaling (a constant scaling factor of the input signal) is fixed to 1 and the SR equals $\rho = 0.95$. For the proposed model (5.1), the input scaling is chosen to be 0.01, while the SR is $\rho = 15$. For the sake of simplicity, in what follows we refer to RCNs resulting from the use of (5.1) as "spherical reservoir".

To evaluate the performance of the network, we use the accuracy metric defined as $\gamma = \max\{1 - \text{NRMSE}, 0\}$, where the NRMSE is defined in (6.3).

In the following, we report the accuracy γ while varying τ in the past for various benchmark tasks. Each result in the plot is computed using the average over 20 runs with different network initializations. The networks are trained on a data set of length $L_{\text{train}} = 5000$ and the associated performance is evaluated using a test set of length $L_{\text{test}} = 2000$. The shaded area represents the standard deviation. All the considered signals were normalized to have unit variance.

White noise In this task, the network is fed with white noise uniformly distributed in the $[-1, 1]$ interval. Results are shown in Fig.6.3, panel (a). We note that networks using the spherical reservoir have a performance comparable with linear networks, while tanh networks do not correctly reconstruct the signal when τ exceeds 20. See Appendix B for a theoretical analysis of these results.

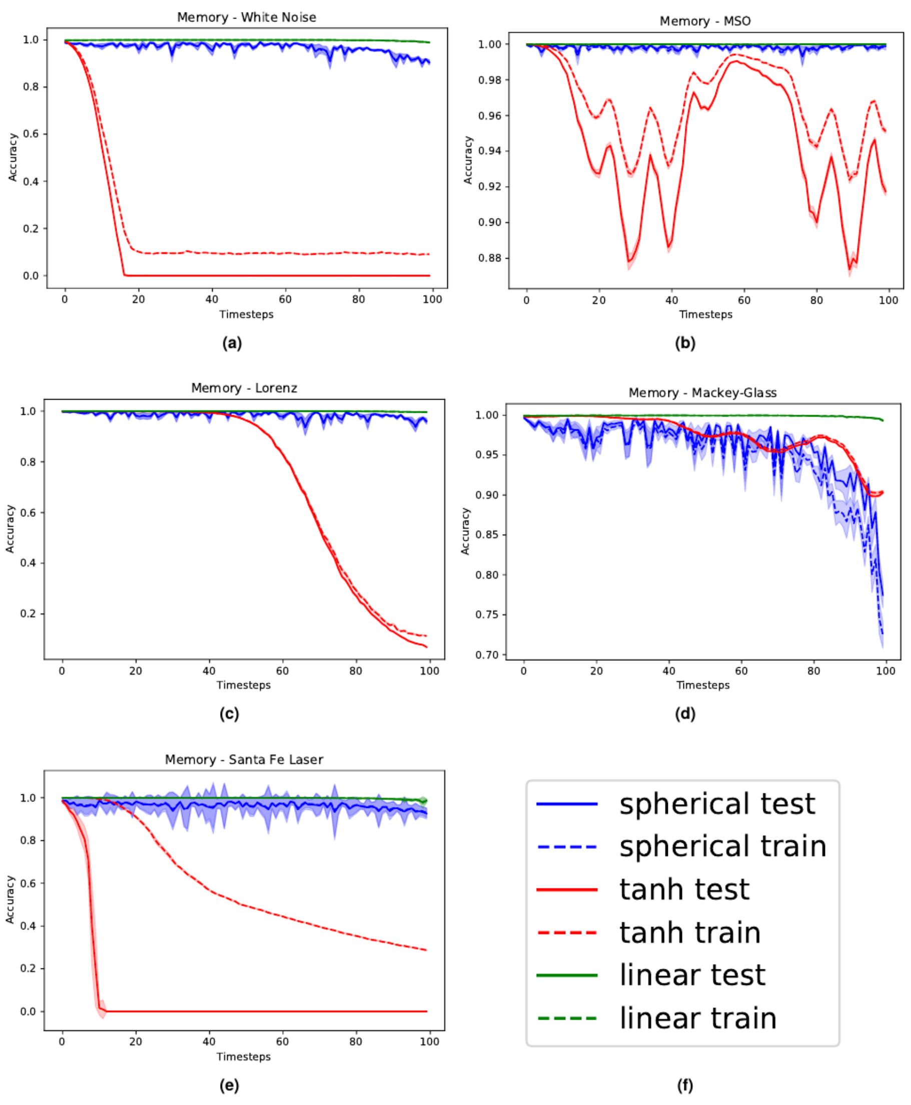


Figure 6.3. Results of the experiments on memory for different benchmarks. Panel (a) displays the white noise memorization task, (b) the Multiple Superimposed Oscillator (MSO), (c) the x-coordinate of the Lorenz system, (d) the Mackey-Glass series and (e) the Santa Fe laser dataset. As described in the legend (f), different line types account for results obtained on training and test data. The shaded areas represent the standard deviations, computed using 20 different realization for each point.

Multiple superimposed oscillators The network is fed with Multiple Superimposed Oscillator (MSO) with 3 incommensurable frequencies. See Appendix C.5 for details. Results are shown in Fig.6.3, panel (b). We note the performance of the linear and the spherical reservoirs are again similar and both outperform the network using the hyperbolic tangent. The accuracy peak when $\tau \approx 60$ is due to the fact that the autocorrelation of the signal reaches its maximum value at that time-step.

Lorenz series We simulated Lorenz system (See Appendix C.1) in a chaotic configuration then fed the network with the x-coordinate only. Results are shown in Fig.6.3, panel (c). Also in this case, while the accuracy for spherical and linear networks does not seem to be affected by τ, the performance of networks using the tanh activation dramatically decreases when τ is large. This stresses the fact that non-linear networks are significantly penalized when they are requested to memorize past inputs.

Mackey-Glass system We simulated the Mackey-Glass system (Appendix C.3) and use it an input signal. Results are shown in Fig.6.3, panel (d). Note that in this case the performance of the network with spherical reservoir is comparable with the one obtained using the hyperbolic tangent and both of them are outperformed by the linear networks.

Santa Fe laser dataset The Santa Fe laser dataset (Appendix C.4) is a chaotic time series obtained from laser experiments. Results are shown in Fig.6.3, panel (e). Also in this case the hyperbolic tangent networks do not manage to remember the signal, while the other systems show the usual behavior.

6.4 Memory-Nonlinearity Trade-off

It is known that RCNs are characterized by a memory–nonlinearity trade-off Dambre et al. [2012]; Verstraeten et al. [2010]; Inubushi and Yoshimura [2017], in the sense that introducing nonlinear dynamics in the network degrades memory capacity.

Here, we evaluate the capability of RCNs to deal with tasks characterized by tunable memory and non-linearity features [Inubushi and Yoshimura, 2017]. The network is fed with a signal $\{u_t\}$ constituted of univariate random variables drawn from a uniform distribution in $[-1, 1]$. The network is then trained to produce the following target output:

$$y_t = \sin(\nu \cdot u_{t-\tau}). \tag{6.4}$$

We see that τ accounts for the memory required to correctly solve the task, while ν controls the amount non-linearity involved in the computation. For each configuration of τ and ν chosen in suitable ranges, we run a grid search on the range of admissible values of SR and input scaling. Notably, we considered 20 equally-spaced values of the SR and for the input scaling. Again, we compare models with three different activation σ: a tanh activation, a linear one (3.8) and the self-normalizing activation discussed in Chapter 5 (5.1). For networks using hyperbolic tangent, the SR varies in $[0.2, 3]$; $[0.2, 1.5]$ for linear networks, and $[0.2, 10]$ for networks with spherical reservoir. The input scaling always ranges in $[0.01, 2]$.[1] We then choose the hyper-parameters configuration that minimizes (6.3) on a training set of length $L_{\text{train}} = 500$ and then assess the error on a test set with $L_{\text{test}} = 200$.

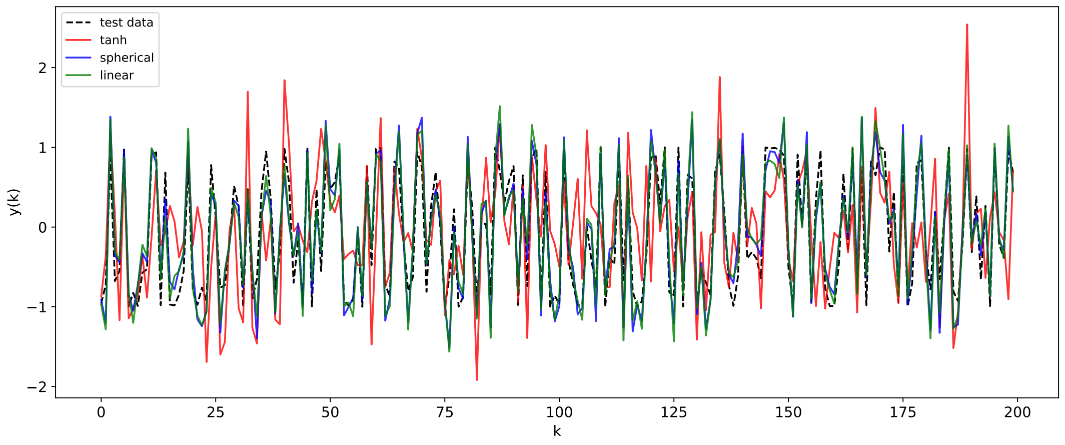


Figure 6.4. Comparison of the network prediction the memory–nonlinearity task for $\nu = 2.5$ and $\tau = 10$. The hyper-parameters of the networks are the same used to generate Fig. 6.3. Here the accuracy values are $\gamma_{\text{tanh}} = 0.12$, $\gamma_{\text{spherical}} = 0.63$ and $\gamma_{\text{linear}} = 0.61$.

In Figures 6.5, 6.6, and 6.7, we show the NRMSE for the task described above for different values of ν and τ, and the ranges of the input scaling factor and the spectral radius which performed best for the hyperbolic tangent, linear and spherical activation function, respectively. The results of our simulations agree

[1]This choice of exploring large values of SR for the hyperspherical case is motivated by the fact that condition (5.8) must be satisfied in order for the network to work properly: choosing a large SR will also lead to larger $\sigma_{\min}(W)$, as discussed in subsection 3.3.2.

with those reported in [Inubushi and Yoshimura, 2017] and, most importantly, confirm our theoretical prediction: the proposed model possess memory of past inputs that is comparable with the one of linear networks but, at the same time, it is also able to perform nonlinear computations. This explains why the proposed model denotes the best performance when the task requires both memory and nonlinearity, i.e., when both τ and ν are large. Predictions obtained for a specific instance of this task requiring both features are given in Fig 6.4, showing how the proposed model outperforms the competitors.

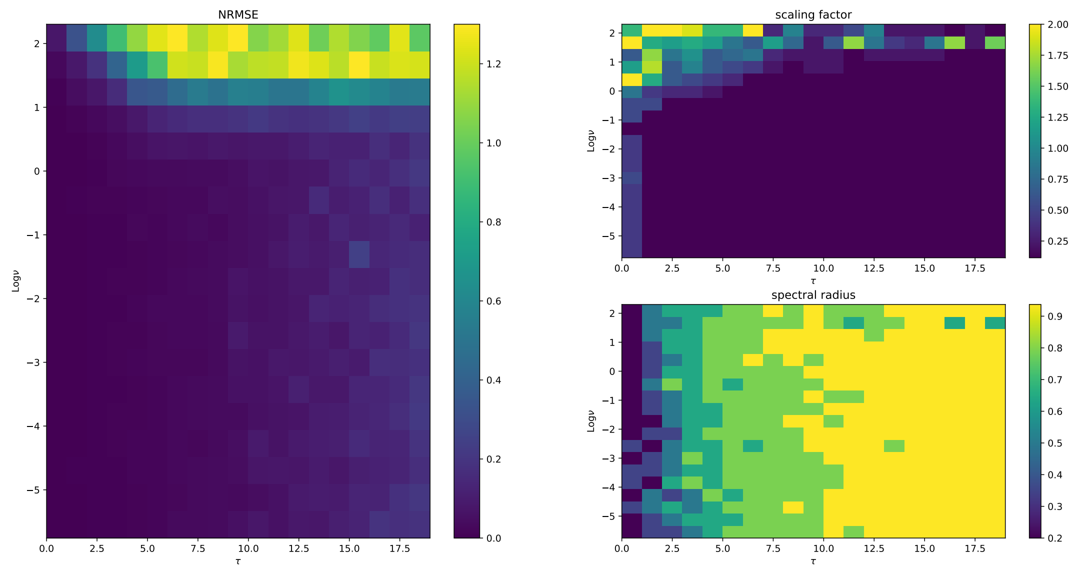


Figure 6.5. Results for the hyperbolic tangent activation function. The network performs as expected: the error grows with the memory required to solve the task. The choice of the spectral radius displays a pattern, where larger SRs are preferred when more memory is required. The scaling factor tends to be small for almost every configuration.

We emphasize that in order to explore the memory-nonlinearity relationship we followed the experimental design proposed in [Inubushi and Yoshimura, 2017]: our goal is to study the memory property of the proposed model and not to develop a model specifically designed to maximize the compromise between memory and nonlinearity. The reader interested in models that aim at explicitly tackling the memory-nonlinearity problem may refer to recently-developed hybrid systems [Inubushi and Yoshimura, 2017; Di Gregorio et al., 2018] which try to deal with the memory–nonlinearity trade–off by combining, in different ways, linear and non-linear units so that the network can exploit a combination of

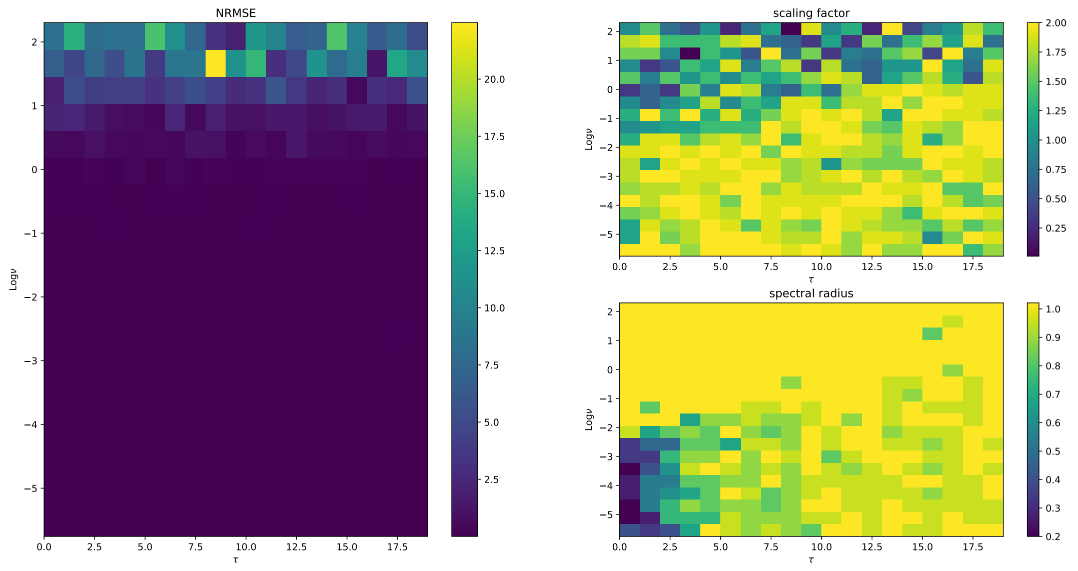


Figure 6.6. Performance of linear networks. We note that τ seems to have no significant effect on the performance. In fact, we note very large errors when the nonlinearity of the task is high. The choice of the scaling factor and of the spectral radius shows a really weak tendency to certain values, indicating that the performance is only weakly influenced by the hyper-parameters.

them according to the problem at hand. These approaches introduce new hyper-parameters which, basically, allow to control the memory–nonlinearity trade–off.

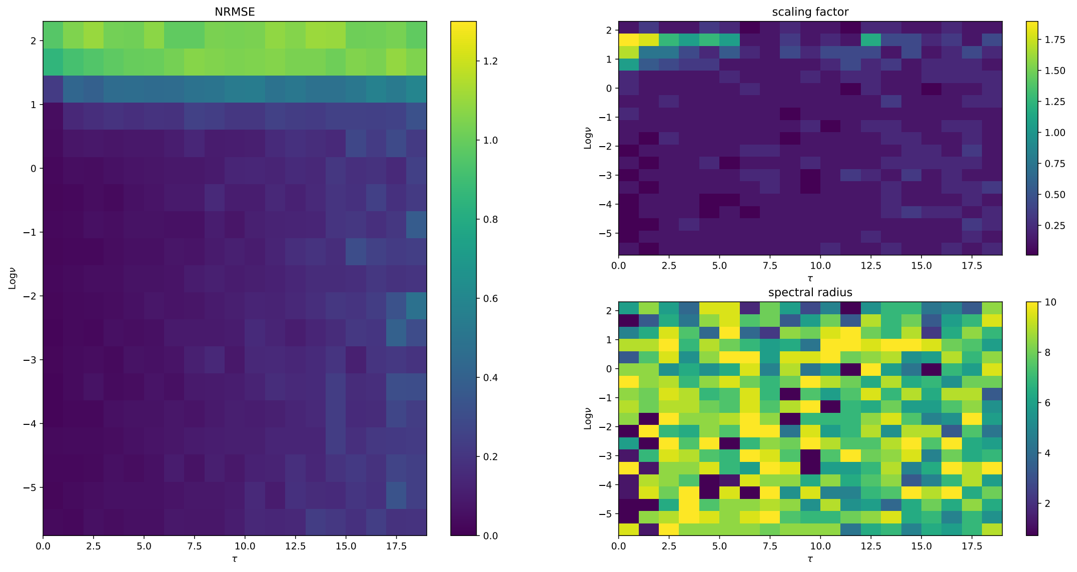


Figure 6.7. Performance of the proposed model. We see that the network performs reasonably well for all the tasks displaying only a weak dependency on the memory. Moreover the spectral radius does not to play any role in the network performance. The choice of the scaling factor denotes similar patterns with the hyperbolic tangent case.

Chapter 7

Echo State Property

This chapters describes and comment the *Echo State Property (ESP)*, a property at the base of the working principle of RCN, regarding the possibility of encoding the signal history into the network state. After an introduction and various analogous definitions (Section 7.1, we discuss the impact that the Spectral Radius play in the property (Section 7.2). In Section 7.3, we contextualize the ESP in the framework of dynamical systems by means of the Takens' Theorem.

7.1 The Echo-State Property

A crucial challenge in machine learning is managing to have a representation of the data which allows solving the problem at hand i.e., to have a "good" (in some sense) *feature map*. When using Reproducing Kernel Hilbert Space (RKHS) the feature map is implicitly provided by the chosen kernel, which must be designed carefully. Neural networks instead learn their own feature map. When learning dynamical systems, the data are time-varying signals, which one must be able to correctly represent in a state. In the context of RC, the possibility of having a correct representation of the driving signal was given the name of Echo State Property (ESP).

The ESP was introduced in the seminal work by Jaeger [2001] as a necessary property for an effective and reliable computation. Basically, the ESP consists in requiring that the reservoir state asymptotically (w.r.t. time) depends only on the received input (i.e., the reservoir state *echoes* the input) and does not depend on the initial conditions of the reservoir. Such a definition takes into account a specific input sequence, with values within a compact set $\mathscr{U}$; in practical applications, the input will always be bounded. Also, the compactness of the reservoir state-space is required, but it is automatically guaranteed if one considers bounded nonlinear activation functions (like tanh).

Definition 2 (Original Echo State Property [Jaeger, 2001])**.** The system has the Echo State Property (ESP) if for every input sequence $\{\boldsymbol{u}_t\}$ there exists an *input echo function*

$$\boldsymbol{E} = (e^1, \ldots, e^N) \tag{7.1}$$

where $e_i : \mathcal{U}^{-\mathbb{N}} \rightarrow \mathbb{R}$. such that for all left-infinite input histories $\ldots, \boldsymbol{u}_{t-1}, \boldsymbol{u}_t$ the system state at time t is given by:

$$\boldsymbol{r}_t = \boldsymbol{E}(\ldots, \boldsymbol{u}_{t-1}, \boldsymbol{u}_t) \tag{7.2}$$

The motivation behind the original ESP formulation is the following: for learning to be effective, the current network state $\boldsymbol{r}_t$ must be uniquely determined by the input sequence $\{\boldsymbol{u}_t\}$.

Definition 3 (Compatibility)**.** We say that a state sequence $\{\boldsymbol{r}_t\}$ is *compatible* with a bounded input sequence $\{\boldsymbol{u}_t\}$ when, for all t:

$$\boldsymbol{r}_{t+1} = \boldsymbol{f}(\boldsymbol{r}_t, \boldsymbol{u}_t)$$

Definition 4 (Echo State Property [Jaeger, 2001])**.** The system has the Echo State Property (ESP) if for every input sequence $\{\boldsymbol{u}_t\}$, for any state sequences $\{\boldsymbol{r}_t\}$ and $\{\boldsymbol{l}_t\}$ compatible with $\{\boldsymbol{u}_t\}$ it holds that $\boldsymbol{r}_t = \boldsymbol{l}_t$ for each t.

A different definition might be given, going forward in time (as opposed to Def. 4 that uses a backward approach).

Definition 5 (Forward Specification of ESP [Yildiz et al., 2012])**.** The system has the Echo State Property (ESP) if and only if it has the *uniform state contraction* property, i.e., there exist a null sequence $\{\boldsymbol{\delta}_t\}$ such that for every input sequence $\{\boldsymbol{u}_t\}$ and for all $\{\boldsymbol{r}_t^1\}$ and $\{\boldsymbol{r}_t^2\}$ compatible with $\{\boldsymbol{u}_t\}$ it holds that for all t $\|\boldsymbol{r}_t^1 - \boldsymbol{r}_t^2\| \leq \delta_t$. It can be equivalently stated as:

$$\lim_{t \to \infty} \sup_{u \in \mathcal{U}} \|\boldsymbol{r}_t^1 - \boldsymbol{r}_t^2\| = 0$$

We highlight that, in order to be equivalent to Def. 4, a stricter condition on the convergence must be adopted. This is discussed in [Jaeger, 2001].

7.2 ESP and Spectral Radius

Notably, even though most theoretical results assume the ESP to hold [Grigoryeva and Ortega, 2018; Hart et al., 2020], existing sufficient conditions are too restrictive [Yildiz et al., 2012] to be used in most practical applications and necessary ones seem to suffice in most cases [Zhang et al., 2011; Basterrech, 2017]. In practice, some less restrictive criteria to verify satisfaction of the ESP have been proposed over time [Yildiz et al., 2012; Manjunath and Jaeger, 2013; Caluwaerts et al., 2013; Verstraeten et al., 2007] as well as a general formulation for the ESP accounting for multiple, stable responses to a driving input sequence [Ceni et al., 2020].

Theorem 2 (ESP – Sufficient Condition [Jaeger, 2001]). *Assume a RCN like* (11.35). *Let its* activation function σ *be a* squashing function *and let its* reservoir connection matrix $\boldsymbol{W}$ *have a Maximum Singular Value (MSV)* $\sigma_{max}(\boldsymbol{W}) < 1$. *Then the RCN has the ESP for all inputs* $\{\boldsymbol{u}_t\}$.

For a Random Reservoir (4.11), the expected value for the SR is $\langle \rho(\boldsymbol{W}_\mathrm{p}) \rangle = \rho$ [Rivkind and Barak, 2017], while the MSV $\langle \sigma_\mathrm{max}(\boldsymbol{W}_\mathrm{p}) \rangle = 2\rho$ [Rudelson and Vershynin, 2010]. So, the condition on the MSV of Theorem 2 result in SR $\rho(\boldsymbol{W}) < \frac{1}{2}$, which is smaller than the values used in most implementations.

Theorem 3 (ESP – Necessary Condition [Jaeger, 2001]). *Assume a RCN like* (11.35). *Let its* activation function σ *be a* squashing function *and let its* reservoir connection matrix $\boldsymbol{W}$ *have a SR* $\rho(\boldsymbol{W}) > 1$. *Then, the RCN* does not *have the ESP when driven with a* null input sequence.

Theorem 3 is often misinterpreted as stating that a RCN cannot possibly have the ESP if its spectral radius exceeds unity. This is not the case, as the theorem refers only to RCNs which are fed with a null signal (which is the same as having no input, i.e., running autonomously). See [Yildiz et al., 2012] for a detailed discussion about this misinterpretation.

7.3 Dynamical System Representation of the Reservoir

7.3.1 Takens's Theorem

Let us consider a dynamical system like (2.1) and a 1-dimensional measurement of the form (2.7):

$$\boldsymbol{s}_t = \boldsymbol{g}(\boldsymbol{s}_{t-1}) \tag{7.3a}$$

$$u_t = h(\boldsymbol{s}_t) \tag{7.3b}$$

Here, each orbit of (7.3a) $\{\boldsymbol{s}_t\}$ corresponds to a time realization of (7.3b) $\{\boldsymbol{u}_t\}$. Since (7.3a) is an autonomous system, the initial condition $\boldsymbol{s}_{t_0}$ determines the entire evolution of the system and, consequently, the entire realization of the observable. This means that the entire sequence $\{\boldsymbol{u}_t\}$ depends on the initial condition, in the sense that different initial conditions will lead to different state sequences. We now wonder whether the opposite is true or not, i.e., if one can reconstruct the state of the system by observing a series of one-dimensional measures like (7.3b). The Takens's theorem deals with this question. In order to state the theorem, we now introduce a *delay vector* $\boldsymbol{x}_t$ constructed from the observation of our dynamical system as:

$$\boldsymbol{x}_t = \begin{bmatrix} x_t^1 \\ x_t^2 \\ \cdots \\ x_t^D \end{bmatrix} = \begin{bmatrix} u_t \\ u_{t-1} \\ \cdots \\ u_{t-(D-1)} \end{bmatrix} = \begin{bmatrix} h(\boldsymbol{s}_t) \\ h(\boldsymbol{s}_{t-1}) \\ \cdots \\ h(\boldsymbol{s}_{t-(D-1)}) \end{bmatrix} =: \boldsymbol{F}_D(\boldsymbol{s}_t) \tag{7.4}$$

where $\boldsymbol{F}_D : \mathbb{R}^d \rightarrow \mathbb{R}^D$ is the *D-delay map*. Its smoothness depends on the smoothness of $\boldsymbol{g}$ and h.

If we assume that the motion of (7.3a) occurs in manifold $\mathscr{M}$ of dimension d, we call $\mathscr{H}_D \subset \mathbb{R}^D$ its image under the effect of $\boldsymbol{F}_D$. So, we can now present a different formulation of the question above: is $\boldsymbol{F}_D$ a diffeormorphism? The answer to this question in given by the Takens's Embedding Theorem, which we now state.

Theorem 4 (Takens's Embedding Takens [1981]). *Let $\mathscr{M}$ be a compact d-dimensional C^2 manifold. For almost any pair of function $\boldsymbol{g}$ and h, which are continuously differentiable on $\mathscr{M}$, the mapping*

$$\boldsymbol{F}_D : \mathscr{M} \rightarrow \mathbb{R}^D$$

given by (7.4) *is a diffeomorphism for almost any $D > 2d$.*

This implies that $\mathscr{H}_D$ is an embedding of $\mathscr{M}$ which, in turn, means that each vector $\boldsymbol{x}$ on $\mathscr{H}_D$ corresponds to a unique vector $\boldsymbol{s}$ on $\mathscr{M}$. So $\boldsymbol{x}_t$ can be used as a state vector to describe the dynamics of the original system (7.3). More specifically, an operator $\boldsymbol{\Gamma}$ might be introduced, such that:

$$\boldsymbol{x}_{t+1} = \boldsymbol{\Gamma}(\boldsymbol{x}_t) \tag{7.5}$$

where

$$\boldsymbol{\Gamma} := \boldsymbol{F}_D \circ \boldsymbol{g} \circ \boldsymbol{F}_D^{-1}. \tag{7.6}$$

It is clear from (7.4) that $x^i_{t+1} = \Gamma^i(\boldsymbol{x}) = x^{i-1} = u_{t-i}$ for $i \leq 2$. The only component which actually needs to be predicted is the first one, i.e.,

$$u_{t+1} = x^1_{t+1} = \Gamma^1(\boldsymbol{x}_t) := \Psi(u_t, u_{t-1}, \dots, u_{t-(D-1)}). \tag{7.7}$$

Where Ψ is a function that is able to produce the next element of our time series (u_{t+1}) by reading the past D elements – i.e., the vector $\boldsymbol{x}_t$. The function Ψ is guaranteed to exist by Theorem 4, in particular by Eq. 7.6.

In real-world applications, it is impossible to check whether the conditions of Theorem 4 are fulfilled and the dimension d is not known, so that it may appear that it has few practical implications. Yet, the theoretical value of this result is high. In fact, theorem 4 provides a formal justification for the autoregressive models (introduced in [Yule, 1927]), which aim at predicting the future of time series from past observations.

7.3.2 Takens in Reservoir Computing

The ESP was not introduced specifically for the dynamical system framework. By (7.2) one has that the reservoir state r_t must be an echo of input time series $\{u_t\}$. In our framework, we know that our input time-series is given by measurement from a dynamical system (Eq. (7.3)). So, our reservoir plays the role of $\boldsymbol{x}_t$ in (7.4), and we can consider the input echo function (7.1) a delay map in which $D = \infty$.

$$\boldsymbol{r}_t = \boldsymbol{E}(\dots, u_{t-1}, u_t) = \boldsymbol{F}_\infty(\boldsymbol{s}_t). \tag{7.8}$$

But Theorem 4 guarantees that a finite amount of delays D will suffice, in fact providing a solid ground for the use of RC (where D is at most equal to the reservoir size N) for dynamical system. So, assuming that our reservoir can be written as

$$\boldsymbol{r}_t = \boldsymbol{F}_D(\boldsymbol{s}_t), \tag{7.9}$$

any generic target in the form $\boldsymbol{y}_t = \boldsymbol{k}(\boldsymbol{s}_t)$ (Eq.2.10) can in principle estimate through RC, as:

$$\boldsymbol{y}_t = \boldsymbol{k}(\boldsymbol{s}_t) = \boldsymbol{k}(\boldsymbol{F}_D^{-1}(\boldsymbol{r}_t)) \tag{7.10}$$

so that one simply needs to set

$$\psi \equiv k \circ \boldsymbol{F}_D^{-1} \tag{7.11}$$

as readout. In fact it was recently shown that RC can be interpreted as generalization of Taken's Theorem [Grigoryeva et al., 2020]. This is particularly evident when considering linear reservoirs, as in (7.4) one can recognize the delay line (4.8). In fact, in [Grigoryeva et al., 2021] the authors prove that it is possible to embed an attractor using a linear reservoir, generalizing Theorem 4 to a class of functions larger than the delay functions like (7.4).

Chapter 8

Input-to-State Representation

In this Chapter, a novel method to study linear RCN is presented. It is based on the Cayley-Hamilton (CH) theorem, which states that any real square matrix satisfies its own characteristic equation. In Section 8.1 we show that CH theorem allows one to rewrite the state equation of a linear network (Eq. 3.13) as a finite sum of signals. We call this signals *network encoded inputs*. The states of the network can then be constructed by multiplying this signals by the *controllability matrix*. In Section 8.2 we study the network encoded inputs by examining their relation with the signal driving the network. This analysis discloses some properties of the role played by the SR in the encoding of the signal. In Section 8.3, we dive further into this analysis by studying the role that the different topologies introduced in Section 4.2.1 have on the input representation. In particular we make prediction about the memory capacity of various models by studying the controllability matrix associated with each topology. Finally in Section 8.4 we study the nullspace of these controllability matrices as a prominent factor for the network MC. Our results explains the experimental findings in Section 6.2.

8.1 Controllability Matrix and Network Encoded Input

Here we develop a representation for a linear RCN (see 3.4.1) in which we study the network state making use of the CH theorem (see Appendix D.1 for details).

The Cayley-Hamilton (CH) theorem states that every real square matrix satisfies its own characteristic equation, implying that

$$\boldsymbol{W}^N = \varphi_{N-1}\boldsymbol{W}^{N-1} + \varphi_{N-2}\boldsymbol{W}^{N-2} + \cdots + \varphi_1\boldsymbol{W} + \varphi_0\boldsymbol{I} \qquad (8.1)$$

where the φ_i are the negated coefficients of the characteristic polynomial . Accordingly, any power of matrix $\boldsymbol{W}$ can be written as a linear combination of the first $N-1$ powers, where N is the matrix order (and also the size of the reservoir):

$$\boldsymbol{W}^k = \sum_{j=0}^{N-1} \phi_j^{(k)} \boldsymbol{W}^j \tag{8.2}$$

where the apex k denotes the fact that the N coefficients are expansion coefficients of the k-th power of $\boldsymbol{W}$. In Appendix D.2 we also show how the coefficients $\phi_j^{(k)}$ can be written in terms of φ_j in (8.1).

The update of a linear RCN is given by (3.8), which we repeat here for readability:

$$\boldsymbol{r}_{t+1} = \boldsymbol{W}\boldsymbol{r}_t + \boldsymbol{w}u_t. \tag{8.3}$$

Note that we set, for simplicity, the bias $\boldsymbol{b} = \boldsymbol{0}$. Here we consider a network driven by a *one-dimensional left-infinite signal* $\{u_t\}_{t=-\infty}^{0}$.

We are interested in describing the current state $\boldsymbol{r}_0$ in terms on this input. By inserting (8.2) in (3.13), we obtain:

$$\boldsymbol{r}_0 = \sum_{k=0}^{\infty}\sum_{j=0}^{N-1} \phi_j^{(k)} \boldsymbol{W}^j \boldsymbol{w} u_{-k} \tag{8.4}$$

$$= \sum_{j=0}^{N-1} \boldsymbol{W}^j \boldsymbol{w} \sum_{k=0}^{\infty} \phi_j^{(k)} u_{-k} = \sum_{j=0}^{N-1} \boldsymbol{W}^j \boldsymbol{w} v_j \tag{8.5}$$

where

$$v_j := \sum_{k=0}^{\infty} \phi_j^{(k)} u_{-k} \tag{8.6}$$

is what we call the *network encoded input*. It is useful to interpret $\boldsymbol{v} = (v_0, v_1, \ldots, v_{N-1})$ as a vector with N components, which "encodes" the left-infinite input signal $\boldsymbol{u}$ in the spatial representation provided by the network. In order for the v_j term to exist, the sum in (8.6) must converge; we will discuss this issue in the next section. We emphasize the fact that the sum over j (the dimensionality of our system) is a *finite* sum with N terms, as opposed to the infinite sum over k (the time index).

Inspired by well-known tools from control theory (see e.g., [Sontag, 2013]), we define the *controllability matrix* of the reservoir as

$$\mathscr{C} = [\boldsymbol{w} \quad \boldsymbol{W}\boldsymbol{w} \quad \boldsymbol{W}^2\boldsymbol{w} \quad \dots \quad \boldsymbol{W}^{N-1}\boldsymbol{w}] \tag{8.7}$$

Then, the state-update equation (3.8) becomes

$$\boldsymbol{r}_0 = \mathscr{C}\boldsymbol{v} \tag{8.8}$$

and the output (3.4) can then be expressed as:

$$y_0 = \boldsymbol{r}\mathscr{C}\boldsymbol{v}, \tag{8.9}$$

i.e., the readout filters the input according to the controllability matrix.

8.2 The Encoded Input Signal

From (8.9), we see that the possibility for the readout to produce the correct output (i.e., the output that solves the task at hand) depends on two distinct elements: the controllability matrix $\mathscr{C}$ (function of $\boldsymbol{W}$ and $\boldsymbol{w}$) and the network encoded input $\boldsymbol{v}$ (which depends on $\boldsymbol{W}$ and $\boldsymbol{u}$).

Under the assumption of bounded inputs $u_{-k} \in [-U, U], \forall k$, we see that

$$|v_j| = \left| \sum_{k=0}^{\infty} \phi_j^{(k)} u_{-k} \right| \leq U \sum_{k=0}^{\infty} \left| \phi_j^{(k)} \right|$$

allowing us to focus on the properties of the $\phi_j^{(k)}$.

These terms are the element of $\boldsymbol{v}$, which we rewrite as:

$$\begin{bmatrix} v_0 \\ v_1 \\ \dots \\ v_{N-2} \\ v_{N-1} \end{bmatrix} = \begin{bmatrix} \sum_{k=0}^{\infty} \phi_0^{(k)} u_{-k} \\ \sum_{k=0}^{\infty} \phi_1^{(k)} u_{-k} \\ \dots \\ \sum_{k=0}^{\infty} \phi_{N-2}^{(k)} u_{-k} \\ \sum_{k=0}^{\infty} \phi_{N-1}^{(k)} u_{-k} \end{bmatrix} \tag{8.10}$$

In Appendix D we show that for $k < N$, $\phi_j^{(k)} = \delta_{kj}$ (Eq. D.6). This implies that the first $N-1$ time steps are simply the inputs:

$$\begin{aligned}\begin{bmatrix} v_0 \\ v_1 \\ \dots \\ v_{N-2} \\ v_{N-1} \end{bmatrix} &= \begin{bmatrix} u_0 + \sum_{k=N}^{\infty} \phi_0^{(k)} u_{-k} \\ u_{-1} + \sum_{k=N}^{\infty} \phi_1^{(k)} u_{-k} \\ \dots \\ u_{-(N-2)} + \sum_{k=N}^{\infty} \phi_{N-2}^{(k)} u_{-k} \\ u_{-(N-1)} + \sum_{k=N}^{\infty} \phi_{N-1}^{(k)} u_{-k} \end{bmatrix} \\ &= \begin{bmatrix} u_0 \\ u_{-1} \\ \dots \\ u_{-(N-2)} \\ u_{-(N-1)} \end{bmatrix} + \begin{bmatrix} \sum_{k=N}^{\infty} \phi_0^{(k)} u_{-k} \\ \sum_{k=N}^{\infty} \phi_1^{(k)} u_{-k} \\ \dots \\ \sum_{k=N}^{\infty} \phi_{N-2}^{(k)} u_{-k} \\ \sum_{k=N}^{\infty} \phi_{N-1}^{(k)} u_{-k} \end{bmatrix}\end{aligned} \tag{8.11}$$

Then, we observe that the terms corresponding to time-step $k = N$ follow from Eq. D.4:

$$\begin{aligned}\begin{bmatrix} v_0 \\ v_1 \\ \dots \\ v_{N-2} \\ v_{N-1} \end{bmatrix} = \begin{bmatrix} u_0 \\ u_{-1} \\ \dots \\ u_{-(N-2)} \\ u_{-(N-1)} \end{bmatrix} + \begin{bmatrix} u_{-N}\varphi_0 \\ u_{-N}\varphi_1 \\ \dots \\ u_{-N}\varphi_{N-2} \\ u_{-N}\varphi_{N-1} \end{bmatrix} + \\ + \begin{bmatrix} \sum_{k=N+1}^{\infty} \phi_0^{(k)} u_{-k} \\ \sum_{k=N+1}^{\infty} \phi_1^{(k)} u_{-k} \\ \dots \\ \sum_{k=N+1}^{\infty} \phi_{N-2}^{(k)} u_{-k} \\ \sum_{k=N+1}^{\infty} \phi_{N-1}^{(k)} u_{-k} \end{bmatrix}\end{aligned} \tag{8.12}$$

successive terms corresponding to time steps $k > N$ can be computed by using (D.12). This procedure shows that, in general, the inputs from 0 to $N-1$ time steps in the past will *always* appear in their original form, and the "mixing" will begin starting from the N-th time step in the past.

In Appendix D we also show that the coefficients $\phi_i^{(k+1)}$ of (8.2) can be recursively expressed in terms of $\phi_i^{(k)}$ as:

$$\begin{bmatrix} \phi_0^{(k+1)} \\ \phi_1^{(k+1)} \\ \vdots \\ \phi_{N-2}^{(k+1)} \\ \phi_{N-1}^{(k+1)} \end{bmatrix} = \boldsymbol{M} \begin{bmatrix} \phi_0^{(k)} \\ \phi_1^{(k)} \\ \vdots \\ \phi_{N-2}^{(k)} \\ \phi_{N-1}^{(k)} \end{bmatrix} \tag{8.13}$$

where $\boldsymbol{M}$ is the *Frobenius companion matrix* of $\boldsymbol{W}$ (see section D.3 for details). Note that the characteristic polynomial of $\boldsymbol{M}$ is that of $\boldsymbol{W}$; as such, the two matrices share the same eigenvalues. Thus, the series (8.6) converges, for bounded inputs, when $\boldsymbol{W}$ has a spectral radius smaller than 1. Note that most theoretical results (see e.g., Jaeger [2001]; Grigoryeva and Ortega [2018]; Tiňo [2020]) require the MSV to be smaller than one, which is a stricter condition compared to ours, as SR≤MSV; it follows that our analysis can be applied to a larger class of reservoirs.

The $\boldsymbol{v}$ vector can be written as:

$$\begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_{N-2} \\ v_{N-1} \end{bmatrix} = \begin{bmatrix} \sum_{k=0}^{\infty} \phi_0^{(k)} u_{-k} \\ \sum_{k=0}^{\infty} \phi_1^{(k)} u_{-k} \\ \vdots \\ \sum_{k=0}^{\infty} \phi_{N-2}^{(k)} u_{-k} \\ \sum_{k=0}^{\infty} \phi_{N-1}^{(k)} u_{-k} \end{bmatrix} \tag{8.14}$$

In Appendix D we show that for $k < N$, $\phi_j^{(k)} = \delta_{kj}$ holds. We also note that terms corresponding to time-step $k = N$ follow from (8.1). This means that (8.14) can be written as:

$$\begin{bmatrix} v_0 \\ v_1 \\ \cdots \\ v_{N-2} \\ v_{N-1} \end{bmatrix} = \begin{bmatrix} u_0 \\ u_{-1} \\ \cdots \\ u_{-(N-2)} \\ u_{-(N-1)} \end{bmatrix} + \begin{bmatrix} u_{-N}\varphi_0 \\ u_{-N}\varphi_1 \\ \cdots \\ u_{-N}\varphi_{N-2} \\ u_{-N}\varphi_{N-1} \end{bmatrix} + \begin{bmatrix} \sum_{k=N+1}^{\infty} \phi_0^{(k)} u_{-k} \\ \sum_{k=N+1}^{\infty} \phi_1^{(k)} u_{-k} \\ \cdots \\ \sum_{k=N+1}^{\infty} \phi_{N-2}^{(k)} u_{-k} \\ \sum_{k=N+1}^{\infty} \phi_{N-1}^{(k)} u_{-k} \end{bmatrix} \tag{8.15}$$

All other terms in (8.15) corresponding to time steps $k > N$ can be computed according to (8.2). Note that (8.15) implies that a larger SR amplifies the contribution of past inputs over the more recent ones, since the input reproducibility property is controlled by ρ^{j+pn}. We see that, a larger SR is required to correctly recall inputs that are further in past (but for which $\tau < N$), since the SR controls the magnitude of the $\phi_j^{(k)}$, i.e., the permanence of u_k on the state. This explains the results obtained in Section 6.2, as we will now discuss.

8.3 Topologies

In general, the inputs from 0 to $N-1$ steps back in time will *always* appear in their original form, and the cross-contribution starts only from u_{-N} backwards in time.

We will make use of this result to analytically examine the properties of the different networks topologies introduced in subsection 4.2.1. Moreover, by deriving the expression for the $\phi_i^{(k)}$ we can study how the network is able to recall its past inputs. In general, if the $\phi_i^{(k)}$v are large then the network will not be able to recall the inputs, since the input u_{-j} can only be read through $v_j = u_{-j} + \sum_{k=N}^{\infty} \phi_j^{(k)} u_{-k}$. It follows that having large expansion coefficients $\phi_j^{(k)}$ prevents the network from being able to recall its past inputs. We will show in the next section that, when we can derive an analytical expression for the $\phi_i^{(k)}$, it is possible to anticipate how the network recalls its past inputs. Note that, as implied by (8.9), for a linear network this is deeply related to its expressive power, since the network output is basically a linear combination of past inputs. The inputs accessibility to the readout is also due to $\mathscr{C}$, which is a property of the network only, as it does not depend on any input signal. A detailed discussion about the relation between the network properties and the rank of the controllability matrix $\mathscr{C}$ was recently presented in Gonon et al. [2020b]. There, the authors prove that the memory capacity for linear reservoirs equals the rank of $\mathscr{C}$. In the following, we analyze the different architectures described in Section 4.2.1 discussing the properties of their controllability matrices.

Random Reservoir In the random reservoir case the property of $\mathscr{C}$ can be studied by considering the expected values of the norm of its columns,which describes how the system accesses past inputs. Let us consider the N-by-N matrix $\boldsymbol{W}_r$ in (9) and a vector with N components $\boldsymbol{w} = \{w_j\} \sim \mathscr{N}(0, \frac{1}{N})$. Since w_j are generated independently, the expected value of the squared norm of the random vector $\boldsymbol{w}$ is $l(\boldsymbol{w}) = N\langle w_i^2 \rangle$. We drop the r in $\boldsymbol{W}_r$ to simplify the notation. We now study $\boldsymbol{z} := \boldsymbol{W}\boldsymbol{w}$ and obtain:

$$\langle z_i^2 \rangle = \langle (\boldsymbol{W}\boldsymbol{w})_i^2 \rangle = \langle (\sum_j W_{ij} w_j)^2 \rangle = N \langle W_{ij}^2 \rangle \langle w_i^2 \rangle \tag{8.16}$$

where the last equality follows from the independence of the zero-mean entries of $\boldsymbol{W}$ and $\boldsymbol{w}$. Now, by the way we constructed $\boldsymbol{W}$ and $\boldsymbol{w}$, we see that $\langle W_{ij}^2 \rangle = \rho^2/N$ and that $\langle w_i^2 \rangle = 1/N$. This results in:

$$\langle z_i^2 \rangle = N \langle W_{ij}^2 \rangle \langle w^2 \rangle = N \frac{\rho^2}{N} \frac{1}{N} = \frac{\rho^2}{N} \tag{8.17}$$

This means that $l(\boldsymbol{z}) = N\langle z_i^2 \rangle = \rho^2$ and that the standard deviation is $\sqrt{\langle z_i^2 \rangle} = \frac{\rho}{\sqrt{N}}$.
From the above the first column of $\mathscr{C}$ has a euclidean norm $\|\boldsymbol{w}\| = 1$, the second

ρ , the third ρ^2; the last one $\rho^{(N-1)}$. Since ρ must be smaller than 1, the components of the last columns of $\mathscr{C}$ shrink quickly. This fact explains the shading observed in the column of $\mathscr{C}$ for the random case of Fig. 8.1 and Fig. 8.2.

Wigner Reservoir For the Wigner case the effect is emphasized by the correlations introduced by the symmetry of $\boldsymbol{W}_{\mathrm{w}}$.

Delay Line The controllability matrix $\mathscr{C}$ for the delay line and the cyclic reservoir can instead be described in exact terms. A sample of each case in provided in Fig. 8.1 and Fig. 8.2 for $N = 100$ and $N = 1000$, respectively. For the delay line a complete analysis of the network output can be carried.

It is easy to see that, applying $\boldsymbol{W}_{\mathrm{d}}$ to a vector $\boldsymbol{v} = (v_1, v_2, \ldots, v_n)$ results in a vector

$$\boldsymbol{v}' := \boldsymbol{W}_{\mathrm{d}ij}\boldsymbol{v} = (0, v_1, \ldots, v_{N-1})$$

and because of the associativity of the matrix product, we see that applying $\boldsymbol{W}_{\mathrm{d}ij}$ to a vector k times results in permuting the vector k times and the substituting the first k elements with the same number of 0v. So, the controllability matrix for the delay line is:

$$\mathscr{C}_{\mathrm{d}} = [\boldsymbol{w}_{\mathrm{d}} \quad \boldsymbol{W}_{\mathrm{d}}\boldsymbol{w}_{\mathrm{d}} \quad \ldots \quad \boldsymbol{W}_{\mathrm{d}}^{N-1}\boldsymbol{w}_{\mathrm{d}}] \tag{8.18}$$

which would be a lower diagonal matrix for a generic $\boldsymbol{v}$ but for $\boldsymbol{w}_{\mathrm{d}} = (1, 0, \ldots, 0)$ is just the identity.

Now, consider the fact that

$$\boldsymbol{W}_{\mathrm{d}}^{N} = \boldsymbol{0} \tag{8.19}$$

The CH theorem implies that any higher power will be null as well. So we simply have:

$$v_0 = u_0$$

$$v_1 = u_1$$

and so on, because all the $\phi_j^{(m)}$ for $m > N$ are null. If we define $\boldsymbol{v}_{\mathrm{d}} := (u_0, u_{-1}, u_{-2}, \ldots, u_{-(N-1)})$:

$$y_0 = \boldsymbol{r} \cdot \mathscr{C}_{\mathrm{d}} \cdot \boldsymbol{v}_{\mathrm{d}} = \boldsymbol{r} \cdot \boldsymbol{I} \cdot \boldsymbol{v}_{\mathrm{d}} = \sum_{i=0}^{N-1} r_i u_{-i} \tag{8.20}$$

which, as expected, is simply a regressive model of order N.

Cyclic Reservoir The characteristic polynomial of $\boldsymbol{W}_\mathrm{c}$ is $\lambda^N = 1$ so that the CH Theorem implies:

$$\boldsymbol{W}_\mathrm{c}^n = \boldsymbol{I} \tag{8.21}$$

Meaning that, for all $m > N$,

$$\boldsymbol{W}_\mathrm{c}^m = \sum_{j=0}^{N-1} \phi_j^{(m)} \boldsymbol{W}_\mathrm{c}^j = \boldsymbol{W}_\mathrm{c}^\mu \tag{8.22}$$

where $\mu := m \mod N$. Note that, in general:

$$(a\boldsymbol{W}_\mathrm{c})^m = a^m \boldsymbol{W}_\mathrm{c}^\mu \tag{8.23}$$

So, if in our reservoir we fix $\boldsymbol{W} = \rho \boldsymbol{W}_\mathrm{c}$ (where ρ is the parameter controlling the spectral radius) we obtain a number of simplifications. First of all, the elements of $\boldsymbol{v}$ assume a regular form. For example:

$$\begin{aligned} v_0 &= u_0 + \rho^N u_{-N} + \rho^{2n} u_{-2n} + \dots \\ v_1 &= u_{-1} + \rho^N u_{-(N+1)} + \rho^{2n} u_{-(2n+1)} + \dots \end{aligned}$$

so that their general form is

$$v_j = \sum_{k=0}^{\infty} \phi_j^{(k)} u_{-k} = \sum_{p=0}^{\infty} \rho^{pn} u_{-j+pn} \tag{8.24}$$

Moreover, the controllability matrix $\mathscr{C}_\mathrm{c}$ assumes a simple form. If we define the i-time permuted input weight vector as:

$$\boldsymbol{w}^{(i)} := \boldsymbol{W}_\mathrm{c}^i w \tag{8.25}$$

we obtain:

$$\mathscr{C}_\mathrm{c} = [\boldsymbol{w} \quad \rho \boldsymbol{w}^{(1)} \quad \rho^2 \boldsymbol{w}^{(2)} \quad \dots \quad \rho^{N-1} \boldsymbol{w}^{(N-1)}] \tag{8.26}$$

so that:

$$y_0 = (r_0, r_1, \cdots, r_{N-1}) \mathscr{C}_\mathrm{c} \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ \vdots \\ v_{N-1} \end{pmatrix} \tag{8.27}$$

The output can be written in compact form by defining:

$$\tilde{v}_j = \sum_{p=0}^{\infty} \rho^{j+pn} u_{-j+pn} \tag{8.28}$$

$$\tilde{\mathscr{C}}_{\mathrm{c}} = [\boldsymbol{w} \quad \boldsymbol{w}^{(1)} \quad \boldsymbol{w}^{(2)} \quad \dots \boldsymbol{w}^{(N-1)}] \tag{8.29}$$

so that, finally:

$$y_0 = \boldsymbol{r} \tilde{\mathscr{C}}_{\mathrm{c}} \tilde{\boldsymbol{v}} \tag{8.30}$$

The fact that, as suggested in [Rodan and Tino, 2010], $\boldsymbol{w}$ should be non-periodic for the network to work at its best, is now evident. In fact, if $\boldsymbol{w}$ is periodic, it means that some columns of $\hat{\mathscr{C}}_{\mathrm{c}}$ are linearly related and, therefore, the rank degenerates, as supported by theoretical arguments in [Tiňo, 2020].

Note that u_{-j} is only accessible through the term $v_j = u_{-j} + \rho^N u_{-(j+N)} + \dots$ and, in order to do that, it must hold that $u_{-j} \gg \rho^N u_{-(j+N)}$. This may suggest to choose small spectral radii, but the smaller the spectral radius, the faster the decay of the memory, since $\hat{v}_j = \rho^j v_j$. This confirms previous intuitions Rodan and Tino [2010], stating that by choosing a small spectral radius, the network preserves an accurate representation of recent inputs, at the expense of losing the ability to recall remote ones. Conversely, if one sets a large spectral radius (i.e., close to 1) the network will be able to (partially) recall inputs from the past, but its memory of more recent inputs will decrease.

8.4 The Nullspace of C and the Network Memory

By (8.9) one can understand how the rank of the controllability matrix $\mathscr{C}$ is associated with the degrees of freedom (the effective number of parameters used by the model to solve the task at hand) that can be exploited by the readout (i.e., the "complexity" of the model).

Note from Figures 8.1 and 8.2 that the cyclic reservoir always has the highest rank of $\mathscr{C}$, while the Wigner the lowest. The difference increases with the number of neurons. [1]

The fact that $\mathscr{C}$ is not full-rank is linked to the presence of the *nullspace*.[2] This means that there are some network encoded inputs $\boldsymbol{v}$ which are mapped to $\mathbf{0}$ by

[1]Note that this is coherent with the findings in [Tiňo, 2020], since the $\boldsymbol{Q}$ defined in that work is simply $Q = \mathscr{C}^\top \mathscr{C}$ and the number of motifs is related to the rank of $\boldsymbol{Q}$ (and so, of $\mathscr{C}$).

[2]What we call the nullspace is practically the *effective* nullspace detected up to the numerical precision, computed using the Numpy dedicated function [Harris et al., 2020].

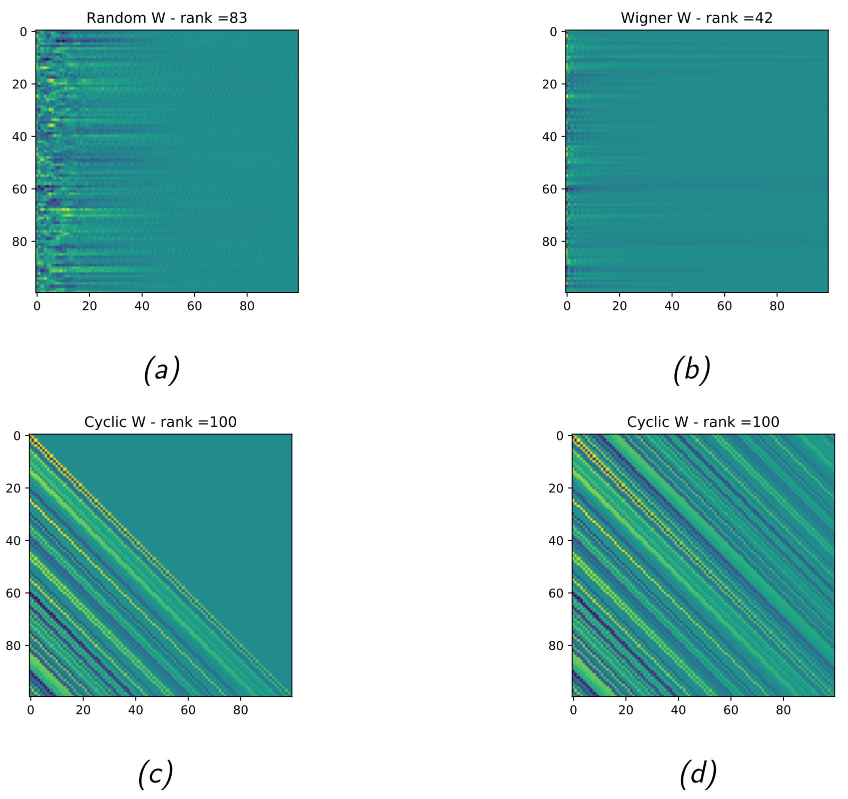


Figure 8.1. The controllability matrix and its rank for different architectures. The Spectral Radius is $\rho = 0.99$ and the reservoirs has $N = 100$ neurons. The four architectures share the same randomly-generated $\boldsymbol{w}$.

$\mathscr{C}$ (8.8) and hence are indistinguishable by the readout perspective. In Fig. 8.3, we plot the rank of $\mathscr{C}$ as a function of the reservoir dimension N. In the experiments using Wigner and cyclic reservoirs, the spectral radius ρ and the maximum singular value σ_{max} coincide and their values are set to 0.995 (Fig. 8.3a) and 0.9 (Fig. 8.3b). For the random reservoir, ρ and σ_{max} are distinct, so we design an experiment where the spectral radius is fixed and another one where the maximum singular value is set (we remind the reader that $\langle \rho \rangle = \frac{1}{2}\langle \sigma_{\text{max}} \rangle$). But what is the shape of the basis of this nullspace? We show its basis in two cases (see Fig. 8.4). The controllability matrix obtained with a cyclic reservoir does not have a nullspace for such a value of ρ, since $\mathscr{C}$ is full-rank. Note that, in order to interpret each vector in Fig. 8.4 as a time series, one must consider the last inputs seen as the ones closer to the origin. Given this interpretation, we clearly see how the memory is linked to the rank of $\mathscr{C}$: the reservoirs' ability to recall past inputs depend on the rank of $\mathscr{C}$ as inputs which only differ in the far-away

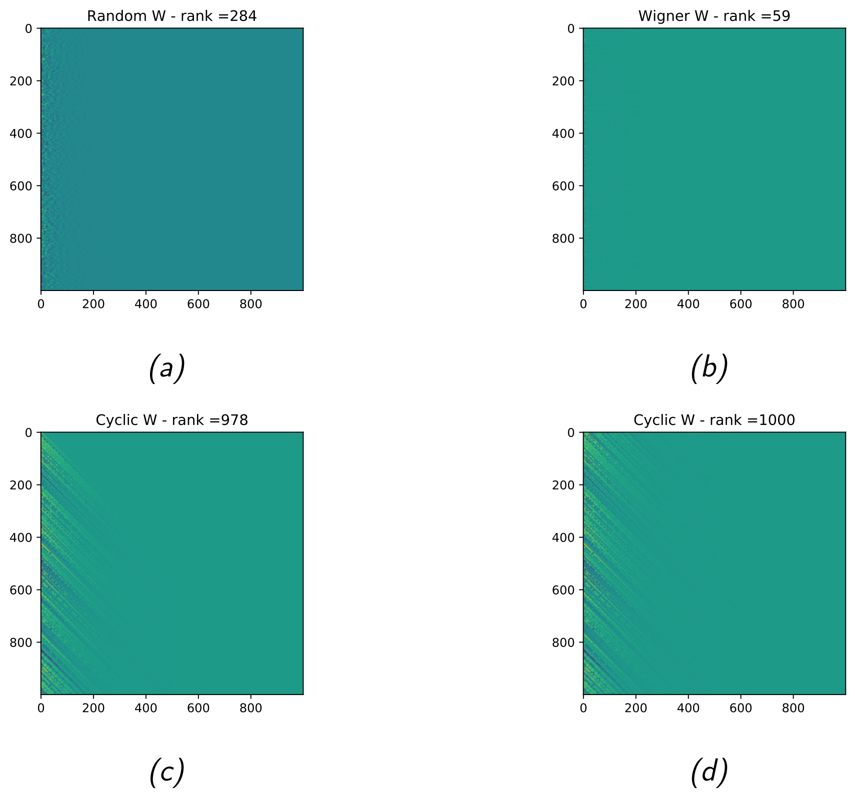


Figure 8.2. The controllability matrix and its rank for different architectures. The Spectral Radius is $\rho = 0.99$ and the reservoirs has $N = 1000$ neurons. The four architectures share the same randomly-generated $\boldsymbol{w}$.

past are mapped to the same final state.

Let us consider an example. Let $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ be two network encoded inputs, which differ only in the last $N-m$ elements. Denote by $\boldsymbol{x}_0^i$ the final state of the system after being fed by the signal $\boldsymbol{v}_i$. We can then write $\boldsymbol{v}_2 = \boldsymbol{v}_1 + \boldsymbol{d}$, where $\boldsymbol{d}$ encodes the difference between the two representations. We see that the first m elements of $\boldsymbol{d}$ are null. So, we can write:

$$\boldsymbol{r}_0^2 = \mathscr{C}\boldsymbol{v}_2 = \mathscr{C}(\boldsymbol{v}_1 + \boldsymbol{d}) = \mathscr{C}\boldsymbol{v}_1 + \mathscr{C}\boldsymbol{d} = \mathscr{C}\boldsymbol{v}_1 + \boldsymbol{0} = \boldsymbol{r}_0^1 \tag{8.31}$$

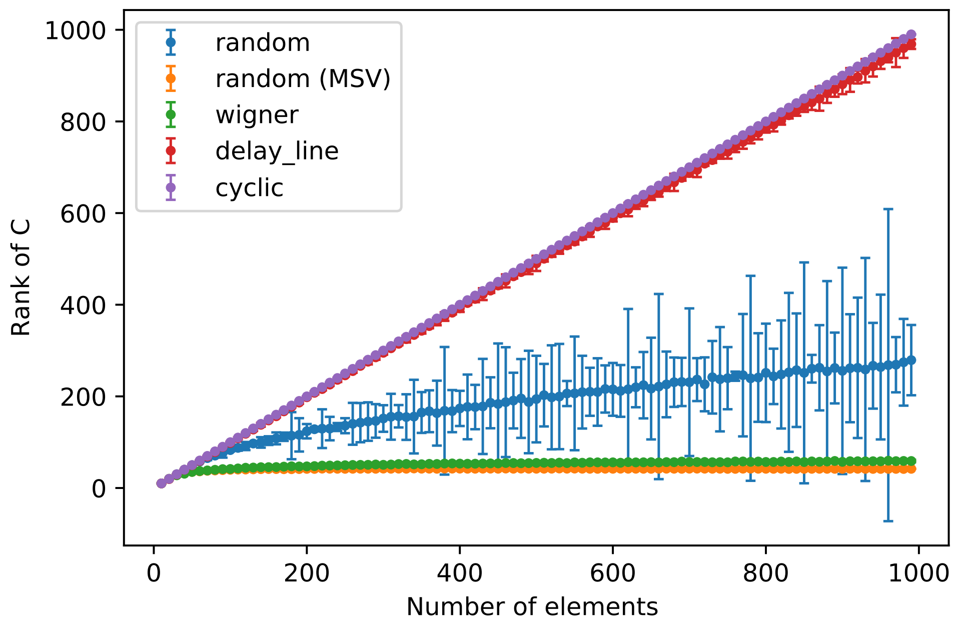
since $\boldsymbol{d}$ lives in the nullspace of $\mathscr{C}$. This results in the network not being able to distinguish between the two signals.

8.4.1 A Note on Numerical Issues

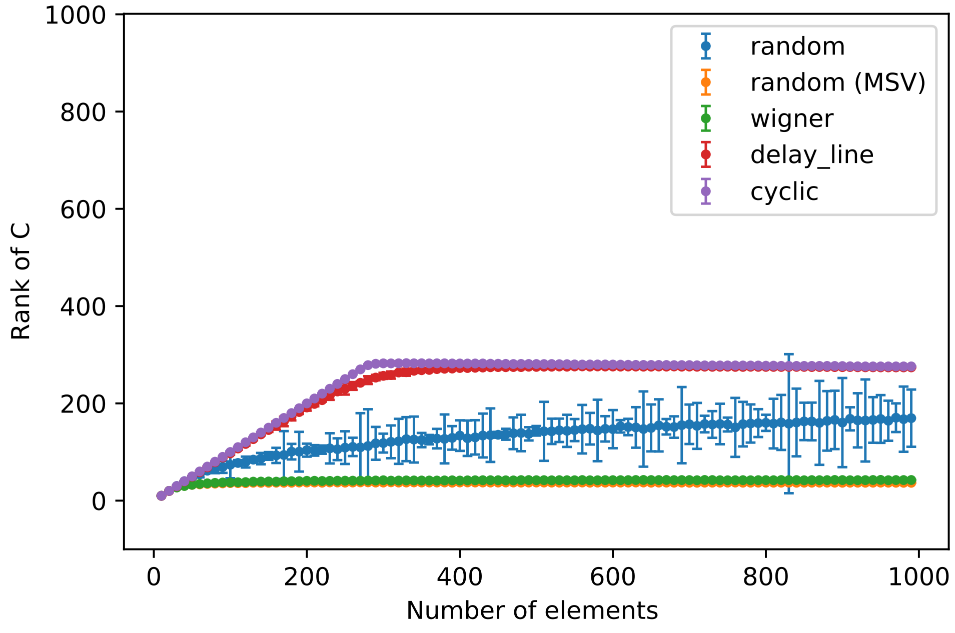
It may be argued that all the above results are due to numerical issues. Theoretical results claim that a system like 8.3 constructed using a random reservoir have a full-rank controllability matrix with probability one [Luh and O'Rourke, 2021]. Moreover, also Wigner reservoirs are controllable with high probability [O'Rourke and Touri, 2015]. The fact that the properties of power iterations render the columns of $\mathscr{C}$ almost linearly dependent is a known issue is methods relying on Krylov subspaces, which can be solved adopting a proper normalization scheme. See, for instance, [Liesen and Strakos, 2013]. The topic in also briefly discussed, in the context of RC, in [Gonon et al., 2020b].

Yet, computing the rank of $\mathscr{C}$ in this way lead to the prediction that the MC of linear RCNs is the same for all values of SR smaller than 1. This is clearly in contrast with the experimental findings in Chapter 6, which shows a strong dependency on ρ. The same holds for the different topologies: the difference in performance should not be observed according to theoretical arguments about the rank of $\mathscr{C}$, as all topologies lead to a full rank matrix. We argue that different topologies and values for the SR lead to different spectra for the eigenvalues of $\mathscr{C}$ which, in turns, correspond to different values for the matrix rank when computed numerically. We tried to convey this concept using the term *effective rank* in the above. In fact, various ways have been proposed to deal with this fact, which is known since the introduction of RCN Jaeger [2002].

On a side note, we mention that the equivalence between RNN and Turing Machines is exactly due to the finite precision of neural networks Siegelmann and Sontag [1995]. The field of *Super-Turing Computation* studies the properties of systems with infinite precision [Siegelmann, 2003].

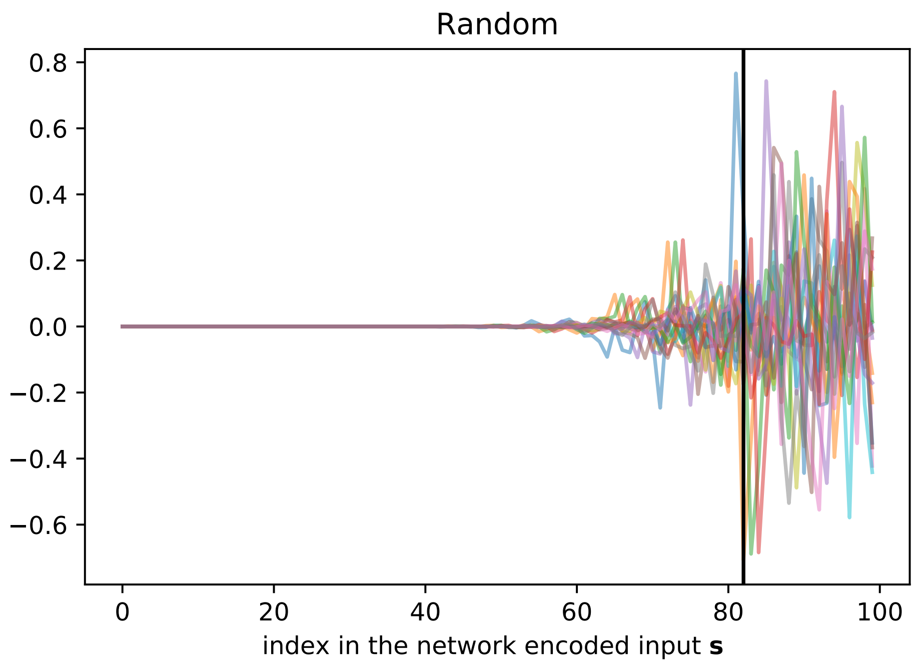


(a)

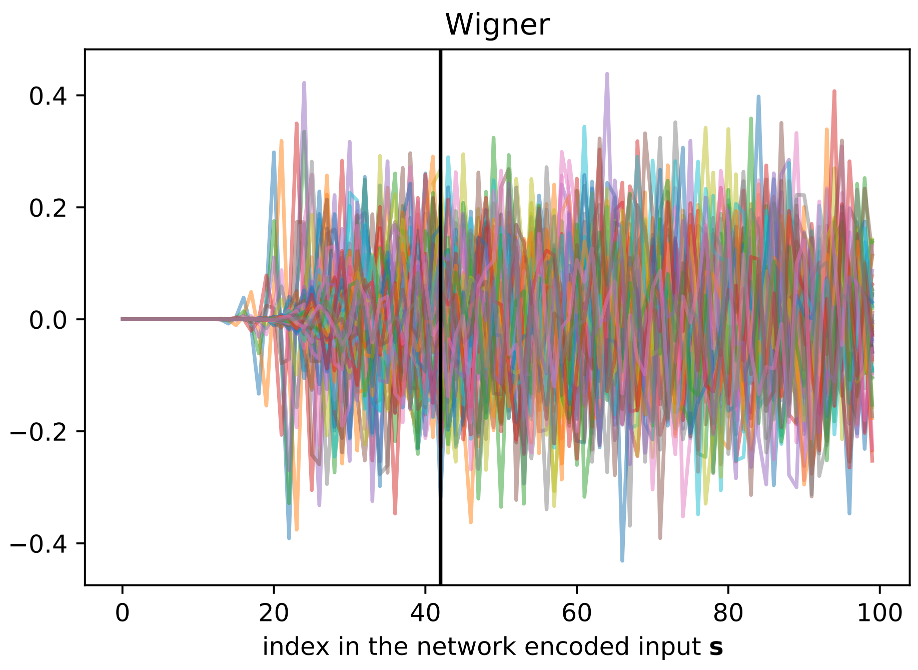


(b)

Figure 8.3. Ranks of the controllability matrix $\mathscr{C}$ as a function of the reservoir dimension N, for $\rho = 0.995$ (a) and $\rho = 0.9$ (b). Note the saturation of the delay line and the cyclic reservoir, which happens, because ρ is not close enough to 1 for its powers to be numerically distinguishable from zero.



(a)



(b)

Figure 8.4. Nullspace basis for a Random (a) and a Wigner (b) reservoir matrices. Each curve represents a basis vector of the nullspace of $\mathscr{C}$, where the x-axis accounts for the vector components. In both cases, we set the spectral radius to $\rho = 0.99$, while the reservoirs size is $N = 100$. The vertical black lines represent the ranks of $\mathscr{C}$, i.e., the dimensions of the image spaces.

Part III

Dynamics

Chapter 9

Computing with dynamical systems

When using a dynamical system, like an RNN, to perform computation, the results will be determined by its dynamics. The final behavior will depend on the input driving the system, on the selected hyper-parameters, and the training strategy adopted, rendering it a very complex problem. Since in RC the dynamical part is not trained, it is easier to address this issue and many results have been derived. In this chapter, a summary of the most important results concerning the dynamics of computational systems (with a focus on RNN) is reported in Sec. 9.1. Sec. 9.2 discuss the concept of Edge of Criticality (EoC) which raised great interest in the community.

9.1 Computing with dynamical systems

The relationship between dynamics and computation has been explored in the scientific literature (e.g., see [Prokopenko et al., 2019] and reference therein). An interesting problem is to determine the computational capacities of neural networks. It is known that RNNs are Turing-complete [Siegelmann and Sontag, 1995], but the question of how to properly train them still remains open. For example, in [Casey, 1996] they study second-order networks, or in [Ashwin and Postlethwaite, 2018] where noisy networks are exploited. Neural Turing Machines [Graves et al., 2014] explicitly address the matter. In fact, dynamical systems seem to have some inherent computational properties [Dambre et al., 2012], for example, they display a trade-off between the nonlinearity of their dynamics and their memory Verstraeten et al. [2010]. This fact is of particular interest in RNNs, as it has been known for a long time that they can display chaotic behavior Sompolinsky et al. [1988], which would render the computation unfeasible. This led the researchers to search for the hyper-parameters configuration and network architecture that would avoid the erratic behavior associated with chaos. For example in Chang et al. [2019]; Kerg et al. [2019] various properties of the connectivity matrices were used to stabilize the dynamics, while in Laurent and von Brecht [2016] a specific gating architecture was designed to avoid chaos. In general, the gating mechanism can be studied using tools from dynamical systems [Jordan et al., 2021]. More specifically, it was recently shown that it enables the *time warping* (i.e., the possibility of managing different timescales) in RNN Tallec and Ollivier [2018]. This fact had also been used in the context of RC [Jaeger et al., 2007].

9.2 Edge of Criticality

Many dynamical systems display a change in their qualitative behavior when their parameters vary. This naturally leads to a classification of their behavior. In particular, many systems display an *ordered* regime, in which the dynamics are regular, and a *disordered* one, where they are irregular. The latter is sometimes referred to as the *chaotic* phase, even if it does not always match the definition of chaoticity that is used in the dynamical system literature.

RCNs are known to be sensitive to the setting of hyper-parameters like the SR, the input scaling and the sparseness degree [Jaeger and Haas, 2004], which critically affect their behavior and, hence, the performance on the task at hand. Fine-tuning of hyper-parameters requires cross-validation or ad-hoc criteria for selecting the best-performing configuration. Experimental evidence [Legenstein and Maass, 2007; Gallicchio, 2018; Legenstein and Maass, 2007; Livi et al., 2017] and some results from the theory [Bertschinger and Natschläger, 2004; Rajan et al., 2010; Rivkind and Barak, 2017] show that RCNs performance is usually maximized in correspondence of a very narrow region in hyper-parameter space called Edge of Chaos or Edge of Criticality (EoC) [Langton, 1990]. However, we comment that beyond such a region RCs display disordered dynamics, resulting in useless and unreliable computations. At the same time, it is everything but trivial configuring the hyper-parameters to lie on the EoC still granting a non-chaotic behavior.

Chapter 10

Binary Reservoirs

RCNs display a wide range of dynamics, which depends on the values of various hyperparameters. In this chapter, we present a minimal model of RCN, having binary states and weights. Its simplicity permits a theoretical analysis of some important aspects of the transition to chaos. We study how the dynamics of these networks are affected by their hyperparameters. In particular, we derive a closed-form expression for the EoC in the autonomous case (Section 10.1), which is in perfect agrement with the empirical data. We also perform simulations in order to assess their behavior in the case of noisy neurons and in the presence of a signal (10.1). We propose a theoretical explanation for the fact that the variance of the input plays a major role in characterizing the EoC.

10.1 Binary RCN

10.1.1 System Description

In this section, we introduce binary RCNs (bRCNs) and study the dynamics of a RCN constituted of binary neurons $r_i \in \{-1, +1\}$ for $i = 1, 2, ..., N$ and binary weights $W_{ij} \in \{-1, 0, +1\}$ for $i, j = 1, 2, ..., N$ (the zero value accounts for the fact that two neurons may not be linked). The bRCNs system model simplifies as:

$$\boldsymbol{S}_t := \boldsymbol{W}\boldsymbol{r}_{t-1} + \boldsymbol{u}_{t-1} \tag{10.1}$$

$$\boldsymbol{r}_t = \operatorname{sgn}(\boldsymbol{S}_t) \quad , \tag{10.2}$$

$\boldsymbol{u}_t$ is the input signal, which we consider to be *unfiltered* ($\boldsymbol{v} = \boldsymbol{1}$, i.e., the all-ones vector), for simplicity. The sgn function is applied element-wise, being an activation function. When $\boldsymbol{u}_t = 0$ for every t (i.e., there is no input), we say that the system is *autonomous*. The study of the autonomous system plays an important role, since it allows us to investigate analytically the network dynamics and its properties. Reservoir connections $\boldsymbol{W} = (W_{ij})$ are instantiated according to the Erdős–Rényi model where each link W_{ij} is created with probability α. If the link is generated, the weight value is set to 1 with probability p or -1 with probability $1-p$.

The proposed bRCN model is controlled by three hyperparameters:

1. the *number of neurons* in the network N;
2. the *mean degree* of the network $\langle k \rangle := \alpha N$;
3. the *asymmetry* in the weights values $d := p - \frac{1}{2}$.

These hyperparameters are related to α and p, although they are easier to understand: in fact, $\langle k \rangle$ has a natural interpretation in terms of mean neuron degree that does not depend on the network size N. The choice of using d is due to the symmetry of the model around the zero value and to the fact that, with this choice, a positive (negative) value of the hyperparameter accounts for majority of positive (negative) weights. Note that $\langle k \rangle$ can vary *continuously* from 0 to N and $d \in (-\frac{1}{2}, \frac{1}{2})$. A similar model was proposed in Rohlf and Bornholdt [2002], but in their work the weights assume a positive or negative value with equal probability, i.e., their model corresponds to ours in the $p = 0$ case.

10.1.2 Edge of criticality in binary RCNs

Here, we study two networks with the same weight matrix $\boldsymbol{W}$, that are in the states $\boldsymbol{r}_t$ and $\boldsymbol{r}'_t$; the latter refers to the perturbed network and differs only in one neuron whose state is flipped. The goal is here to understand under which conditions the time evolution of the perturbed network differs from the original one, i.e., whether the perturbation will spread and significantly impact the network behavior or not.

For $N \to \infty$, the fraction of positive-valued neurons is equal to the probability for a neuron of being positive, namely $P_+ = p$, while the fraction of negative neurons is $P_- = q = 1 - p$. By comparing the original network with the perturbed one, the probability that the flipped neuron will have an influence on a neuron connected to it will be given by two terms: the probability that neuron state is positive P_+ multiplied by the probability of switching from positive to negative π_{+-}, plus an analogous terms accounting for the negative part (P_- and π_{-+} respectively. Assuming that $\pi_{+-} = P_- = q$ (i.e., that the probability of turning negative from positive is equal to the probability of being negative) and, analogously, that $\pi_{-+} = P_+ = p$ (that can be seen as a formulation of the *annealed approximation* introduced in [Derrida and Pomeau, 1986]), one obtains:

$$P_+ \cdot \pi_{+-} + P_- \cdot \pi_{-+} = pq + qp = 2p(1-p) \tag{10.3}$$

We now define k_O and k_I as the mean *out-degree* and *in-degree* of a neuron, respectively. Since a single neuron has influence on k_O neurons, the expected number of changes is given by $2p(1-p)k_O$, to which one has to add the fact that at least one neuron has changed due to the flip. Therefore, if this number is bigger than half of the mean incoming links of a neuron, i.e., $k_I/2$, then the perturbation will dominate the network dynamics and will propagate. Since in an Erdös–Rényi graph $k_I = k_O = \langle k \rangle$, we obtain the following condition for the onset of chaos:

$$1 + 2p(1-p)\langle k \rangle > \frac{\langle k \rangle}{2} \tag{10.4}$$

which using $d := p - 1/2$ can be rewritten as:

$$\langle k \rangle < k_c := \frac{1}{2d^2} \tag{10.5}$$

Note that the mean degree $\langle k \rangle$ in (10.5) plays a “stabilizing” role (i.e., the higher the degree, the larger the magnitude of d required for chaos), as opposed to Random Boolean Networks (RBNs), where increasing the mean degree leads towards a chaotic region [Kauffman, 1969].

10.2 Experiments

In Sec. 10.2.1, we assess the agreement of our theoretical expression for the EoC with simulations. Initially, we consider an autonomous bRCN. In Sec. 10.2.2 we analyze the effects of perturbations on network trajectories. In Sec. 10.2.3 we take into account the effect of white Gaussian noise on the theoretical predictions of the EoC; in Sec. 10.2.4 we consider periodic inputs.

10.2.1 Edge of Criticality

In order to assess the agreement between the prediction given by Eq. 10.5 and experimental results, we conducted an exploration of the parameter space. Here, we exploit the fact that our neurons assume binary states only and consider their

Shannon entropy H as an indicator for the transition to chaos. The entropy H was computed considering a time average $\overline{H}$,

$$\overline{H} := \frac{1}{T - t_0} \sum_{n=t_0}^{T} H(\boldsymbol{r}_t) \tag{10.6}$$

where the entropy of a configuration $\boldsymbol{r}$ is estimated as $H(\boldsymbol{r}) := -\rho(\boldsymbol{r})\log(\rho(\boldsymbol{r})) - (1-\rho(\boldsymbol{r}))\log(1-\rho(\boldsymbol{r}))$, in which $\rho(\boldsymbol{r})$ is the number of neurons whose state is $+1$ and $1-\rho(\boldsymbol{r})$ the number of neurons with state -1. We expect Eq.(10.6) to be be almost zero in the frozen regime and almost one in the chaotic one, with a sharp region of intermediate values that we consider to be the edge of chaos.

In order to explore the parameter space, we run a series of simulations using a network of $N = 1000$ neurons with different random initial conditions and connection matrices $\boldsymbol{W}$, generated using specific values of $\langle k \rangle$ and d. In Eq. 10.6, we used $T = 300$ time-steps and $t_0 = 100$ accounting for an initial transient from the initial state to a stationary condition. Results are showed in Fig. 10.1 and demonstrate almost perfect agreement with the theoretical result (10.5).

10.2.2 Effects of Perturbations on State Evolution

In order to assess the effect of chaos on the network behavior, we compare the evolution of a bRCN instantiated with weight matrix $\boldsymbol{W}$ but different initial conditions. Starting from a random initial condition, we generated 50 additional initial conditions by flipping the state of a single neuron (as in Sec. 10.1.2). Here, an initial condition $\boldsymbol{r}_0$ is randomly generated with a biased probability $c = 0.6$ for a neuron to assume a positive value. This is necessary as the network in the frozen phase could reach two different stable states – one with almost all positive neurons and another one with almost all negative neurons. Finally, we note that the value of c has no impact on the EoC. We let the original and perturbed networks evolve, and take into account the (normalized) Hamming distance, D_H, between trajectories.

Results are summarized in Fig. 10.2. We observe that, in the ordered phase, perturbations on the initial state have no effect on the network evolution and the Hamming distance of the perturbed trajectory from the original one is zero. As d decreases (i.e., the networks approach the chaotic regime), we observe how the Hamming distance significantly increases, leading to chaos. Note that the maximum value achievable by the (normalized) Hamming distance is 0.5, corresponding to the distance of two random binary vectors (a larger distance would imply a negative correlation). In the same set of figures, we show three additional indicators, called *Energy*, *Activity*, and *Entropy*. The mean *Energy*, defined

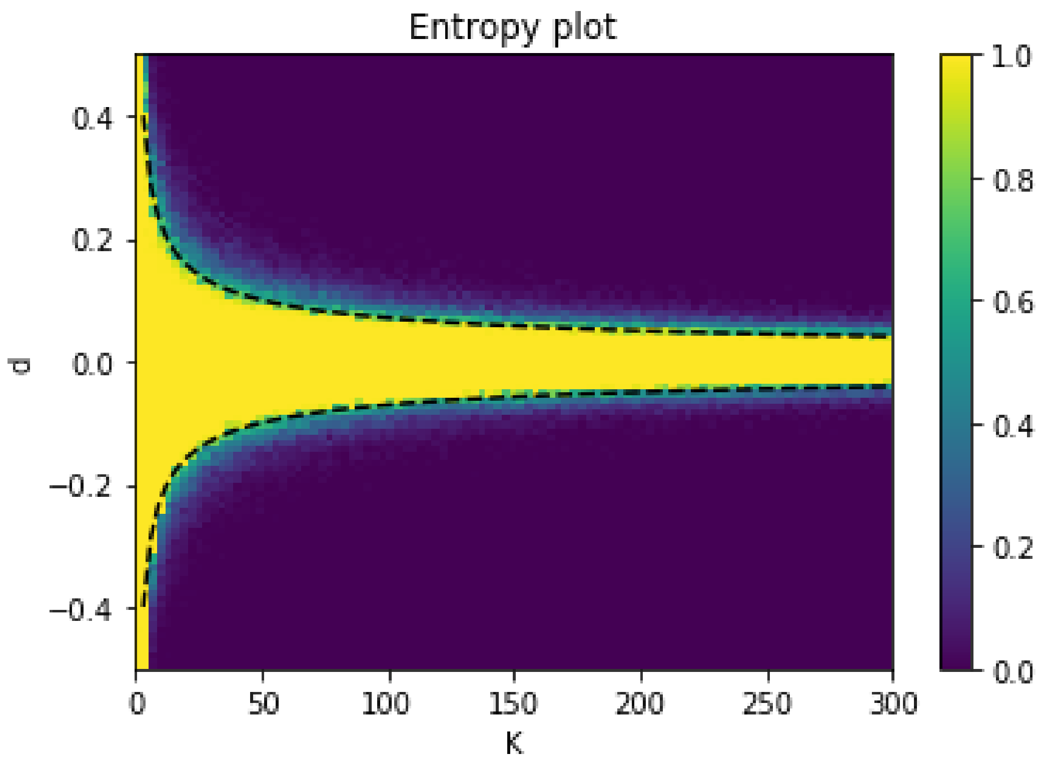


Figure 10.1. Values of $\overline{H}$ for different configurations of the asymmetry and the mean degree. The experiment shows a good agreement with the predicted EoC region (dashed line), where we observe an abrupt change of the entropy from 0 to 1.

as $E(\boldsymbol{r}_t) := \frac{1}{N}\sum_{i=1}^{N} r_t^i$, quantifies the average number of positive and negative neurons. In the frozen phase, the network almost instantly evolves towards values close to 1 (cfr. the role of c, discussed above), and then rapidly decreases to 0, which is the expected value in the chaotic phase. The mean *Activity* of network at time-step n is defined as the (normalized) Hamming distance of the current state w.r.t the previous one, $A(\boldsymbol{r}_t) = D_H(\boldsymbol{r}_t, \boldsymbol{r}_{t-1})$, i.e., the number of neurons that changed their states in one step. As expected, networks operating in chaotic regimes are characterized by an elevated activity. Lastly, we plot the evolution of the *Entropy* (10.6) over time. As expected from the theory, transitioning to a chaotic regime is signaled by a sharp increase of entropy.

10.2.3 Impact of Noise in the EoC

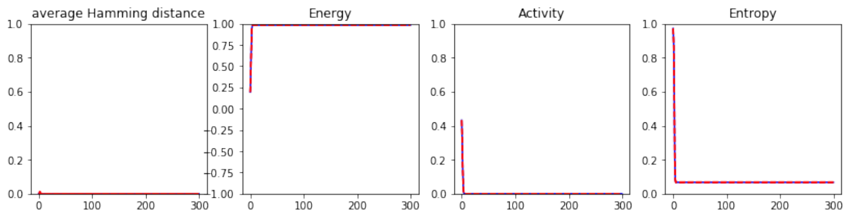
Here, we study how the EoC is influenced when considering an independent noise term for each neuron, $r_{t+1}^i = \text{sgn}\big(S_t^i + \nu \cdot \langle k \rangle \cdot \xi_t^i\big)$, where ν is the *noise gain*, $\xi^i \sim \mathcal{N}(0,1)$, and $\boldsymbol{S}$ is the same as in Eq.(10.1). The choice of scaling the noise with $\langle k \rangle$ was made to account for the fact that the network stability increases with it, as we discuss below.

To explore the dependency from ν, we ran an experiment where we fixed $\langle k \rangle$ and plotted ν versus d. Results are shown in Fig. 10.3. We can recognize three regimes: (1) for low noise values, the chaotic region remains almost constant; (2) for intermediate values, the chaotic region linearly expands with the noise intensity up until (3) there is only chaos. We repeated the experiments with different values of $\langle k \rangle$ (not shown) and they all confirm the same linear expansion of the chaotic region (in units of $\langle k \rangle$). To verify this fact for a wider range of $\langle k \rangle$, we repeated the experiment in Fig. 10.1 with noise intensity $\nu = 0.1$. It is possible to observe in Fig. 10.4 how the EoC maintains its shape for lower values of $\langle k \rangle$, while for higher average degrees it deviates from the theoretical prediction and the chaotic region depends on d only. We explain this fact as follows. Neurons can only assume 1 or −1 values. The probability of a neuron having j positive inputs is then $P_k(j) = \binom{k}{j} p^j q^{k-j}$ where k is its in-degree. If we consider that $j = \frac{k+s}{2}$, where s is the value of the *sum of the positive and negative inputs* (whose sign determines the value of the neuron), then we obtain $s = 2j - k$. The expectation of j is $\langle j \rangle = pk$, so that the expectation of s and its variance are:

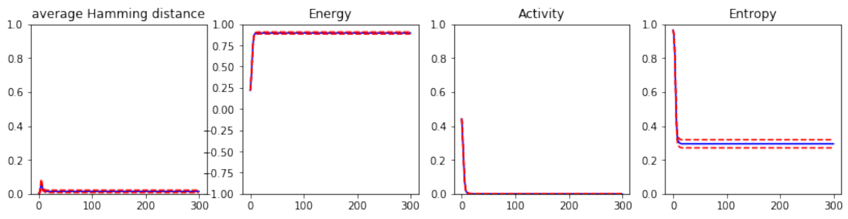
$$\langle s \rangle = k(p-q) = 2kd \tag{10.7}$$

$$\langle (s - \langle s \rangle)^2 \rangle = 4kpq = k(1-4d^2) \tag{10.8}$$

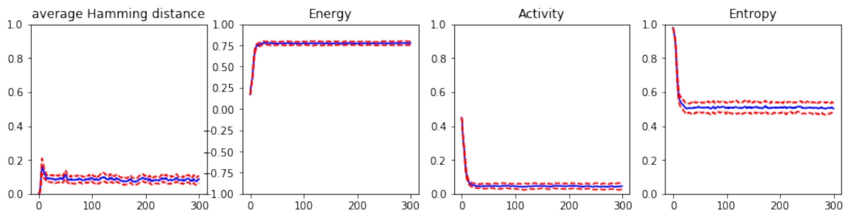
Note that these values are related to a single neuron. For a general understanding of the network behavior, one simply uses $\langle k \rangle$ instead of k in the expressions



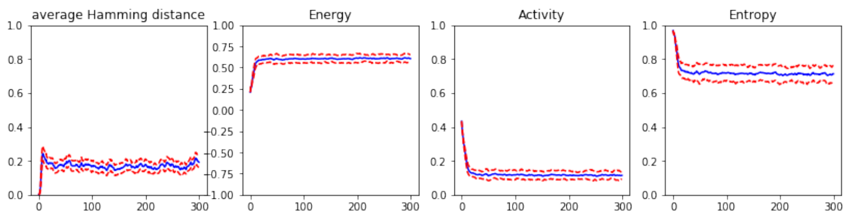
(a) $d = 0.25$



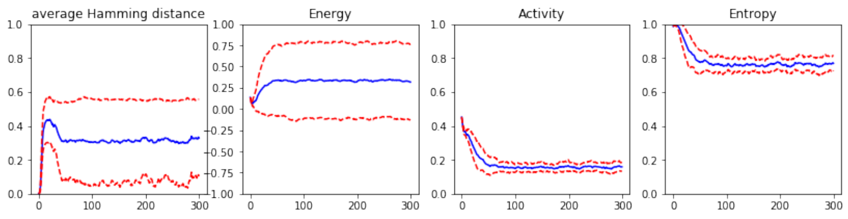
(b) $d = 0.184$



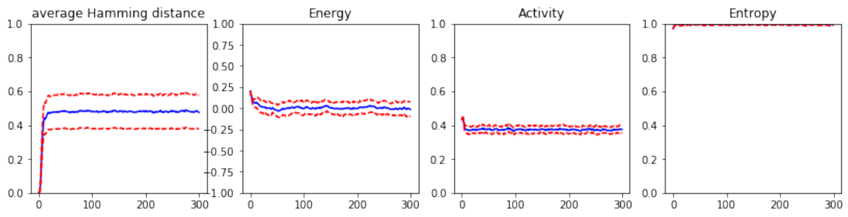
(c) $d = 0.157$



(d) $d = 0.144$



(e) $d = 0.131$



(f) $d = 0.105$

Figure 10.2. Mean values of the Hamming distance, Energy, Activity and Entropy of the 50 perturbed networks, with $N = 1000$ and $K = 22$ for selected values of d (see 10.2.2). The x-axis represents time. The values of the quantities are plotted in blue, while the dashed red lines show the variance. The predicted system should turn chaotic for $d < 1/\sqrt{2\langle k \rangle} \approx 0.15$, according to the theoretical formula.

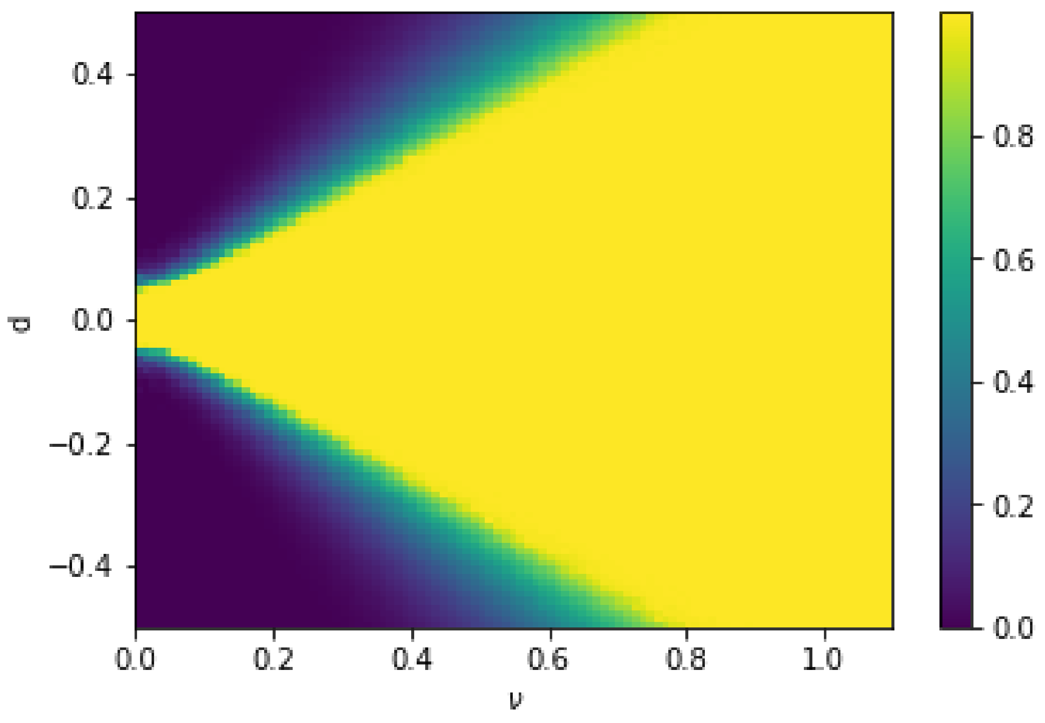


Figure 10.3. Values of $\overline{H}$ for different configurations of d and the ν. Note that ν is multiplied by the mean degree, which is here fixed to $\langle k \rangle = 200$.

above, so that it is possible to consider $\sigma^2 := \langle k \rangle (1{-}4d^2)$ as a *mean-field variance* of the total inputs to neurons. The impact of the noise on the network can then be studied considering the ratio between ν and σ^2. As previously discussed, the noise expands the chaos region linearly with its ν.

The noise we are considering has a standard deviation $\theta_k = \nu \cdot \langle k \rangle$. This leads us to a formula for the chaotic region which, for $\langle k \rangle \gg 1$, is $|d| < a \cdot \nu + b$. This relation, as shown in Fig. 10.4, does not depend on $\langle k \rangle$. As such, having Gaussian noise with standard deviation θ, the formula is $|d| < a \cdot \frac{\theta}{\langle k \rangle}$, or in terms of $\langle k \rangle$, we have $\langle k \rangle < k_c^{\text{noise}} := \frac{a\theta}{|d|}$. In our experiments, constant a was determined as $a \approx 0.65$.

10.2.4 Impact of a Signal

As for the noise, the magnitude of the signal should have a major role in the EoC, but this time the chaotic region should reduce instead of expanding, since the signal is known to suppress chaos in certain conditions [Rajan et al., 2010].

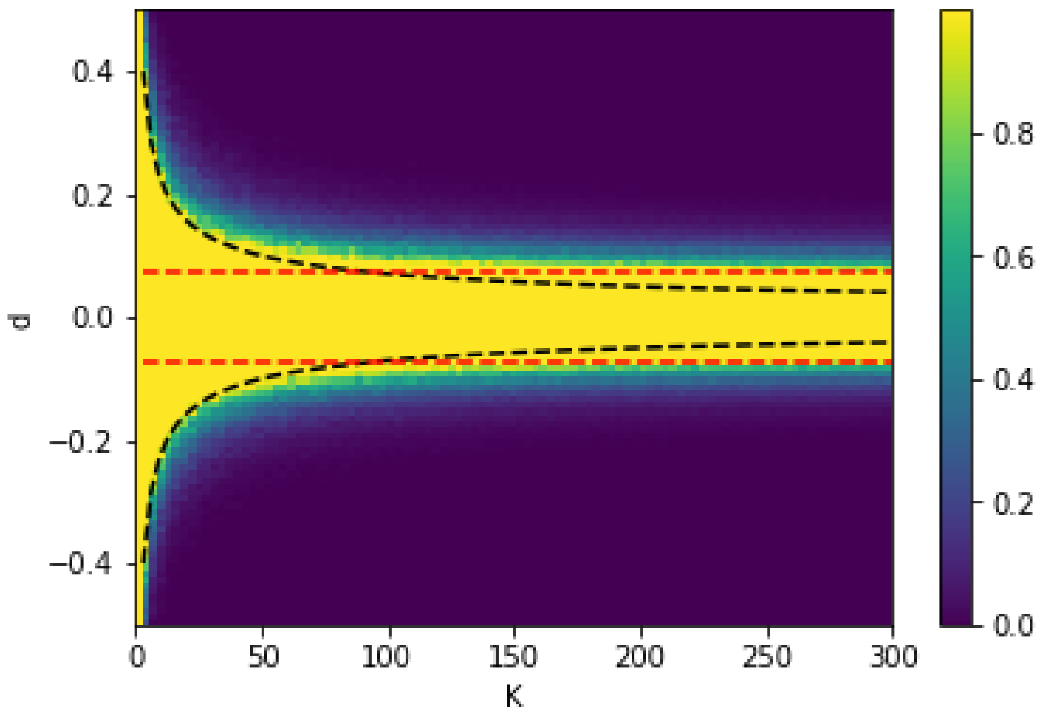


Figure 10.4. The same experiment of Fig.10.1, but with the presence of a noise term with $\nu = 0.1$ (multiplied by the mean degree, so that it increases along the x-axis). Note how for higher degree the chaos region is constant (the predicted value is the red dashed line), deviating from the autonomous-case prediction (red dashed line).

The signal introduces a correlation among neurons, which makes the annealed approximation ineffective. We drive the network with the signal as in (10.1), but we scale $\boldsymbol{u}_t$ with a *gain* factor A, since we are interested in its usage as an hyper-parameter and not in relation with $\langle k \rangle$. We initially feed the network with white noise (note that this is different from what we did in 10.2.3, since in this case the noise is the same for each neuron): from Fig. 10.5 one can observe how the chaotic region rapidly shrinks as A increases, but a region with an intermediate value of entropy expands (linearly). This is due to the fact the signal prevents the system from collapsing in a stable state, keeping the entropy above zero.

In Fig. 10.6 we show the results obtained for the MSO (see Appendix C.5): again we note how the chaotic region shrinks as A increases, with the appearance of the region characterized by intermediate entropy values which, instead, expands.

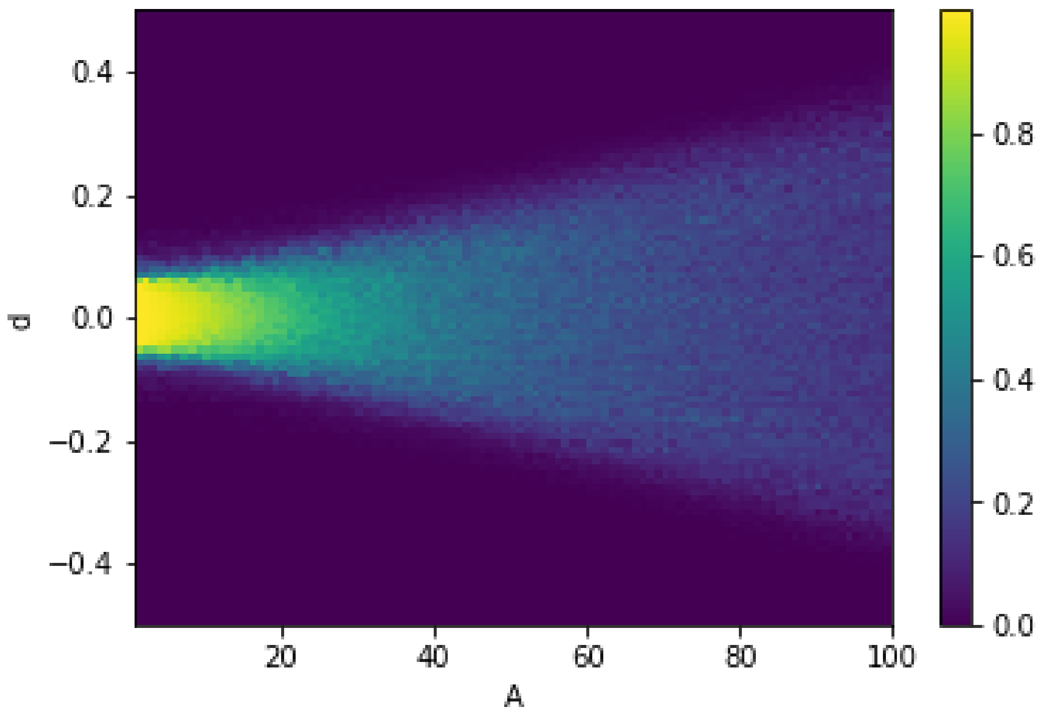


Figure 10.5. Network driven by white noise. Values of $\overline{H}$ for different configurations of A and d. The mean degree was fixed to $\langle k \rangle = 150$.

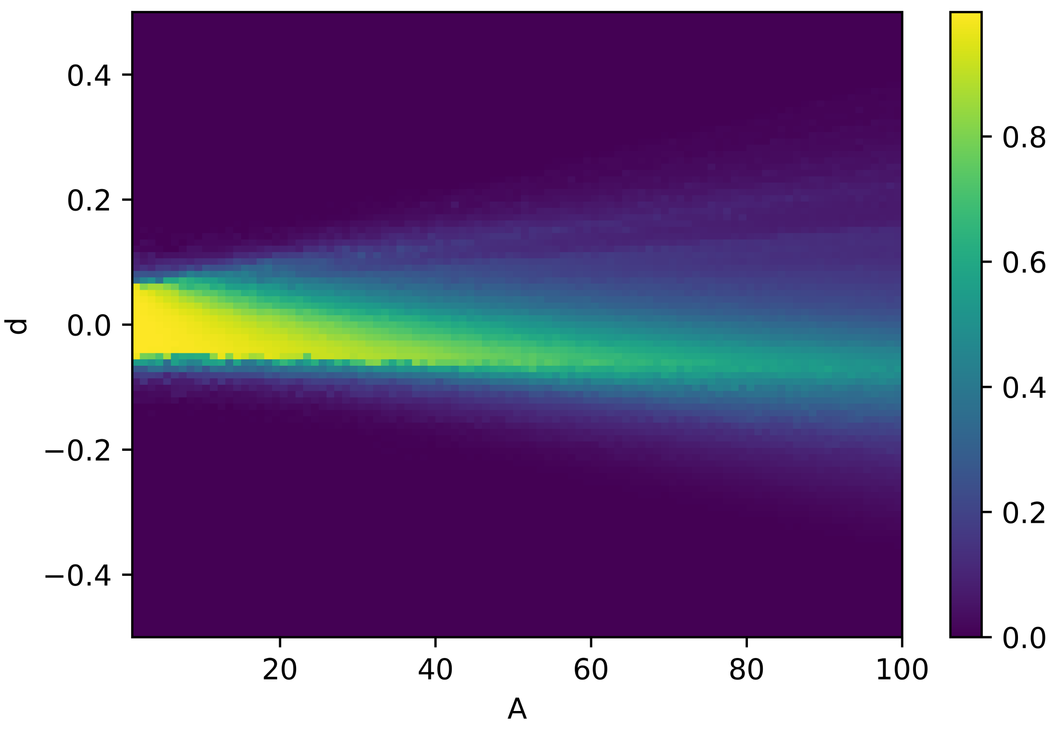


Figure 10.6. Network driven with the sum of three sinusoids. Values of $\overline{H}$ for different configurations of A and d. The mean degree was fixed to $\langle k \rangle = 150$.

Chapter 11

Synchronization

In this chapter, we propose a general framework to study RC systems, which is based on *Synchronization* of dynamical systems. Synchronization is a well established topic with a solid theoretical backbone. We discuss how results coming form this theory can be exploited when dealing with temporal tasks in machine learning and show that they are particularly suited to RC. Section 11.1 introduces the topic of synchronization for dynamical system, building up the theoretical background needed in the rest of the chapter. In section 11.1, we introduce synchronization between identical systems, synchronization in drive-response systems and finally *GS*, which is the main tool we will use to study RC systems. In Section 11.2, we analyze the role played by GS when training a RC system to solve a generic task. In particular, we show how RC allows the reservoir to correctly encode the system generating the input signal into its dynamics. We also discuss necessary and sufficient conditions for the learning to be feasible in this approach. Moreover, we explore the role that ergodicity plays in this process, showing how its presence allows the learning outcome to apply to multiple input trajectories. Finally, in Section 11.3 we show that satisfaction of the GS can be measured by means of the Mutual False Nearest Neighbors index, which makes effective to practitioners theoretical derivations.

11.1 Synchronization

11.1.1 Synchronization of Identical Systems

Following Ott [2002], we start by recalling the concept of *sensitive dependence on initial conditions*. Consider two identical d-dimensional chaotic systems, say

a and b, described by:

$$\boldsymbol{x}_t^a = \boldsymbol{F}(\boldsymbol{x}_{t-1}^a) \tag{11.1a}$$
$$\boldsymbol{x}_t^b = \boldsymbol{F}(\boldsymbol{x}_{t-1}^b) \tag{11.1b}$$

where the function $\boldsymbol{F}$ is the same for both systems. If the initial conditions differ even slightly, then the chaotic nature of the system will lead to exponential divergence: the two systems posses the same attractor but their motion will be uncorrelated over time.

In this context, an instance of chaos synchronization consists of designing a coupling between the two systems such that the two trajectories, $\boldsymbol{x}_t^a$ and $\boldsymbol{x}_t^b$, become identical asymptotically with time. That is, if $\boldsymbol{x}_t^a \approx \boldsymbol{x}_t^b$ then $\|\boldsymbol{x}_t^a - \boldsymbol{x}_t^b\| \to 0$ as $t \to \infty$. A possible coupling for (11.1) might be:

$$\boldsymbol{x}_t^a = \boldsymbol{F}(\boldsymbol{x}_{t-1}^a) + \boldsymbol{c}^a \left(\boldsymbol{x}_{t-1}^a - \boldsymbol{x}_{t-1}^b\right) \tag{11.2a}$$
$$\boldsymbol{x}_t^b = \boldsymbol{F}(\boldsymbol{x}_{t-1}^b) + \boldsymbol{c}^b \left(\boldsymbol{x}_{t-1}^b - \boldsymbol{x}_{t-1}^a\right) \tag{11.2b}$$

The $\boldsymbol{c}^a = [c_{a,1}, c_{a,2}, \dots c_{a,d}]$ and $\boldsymbol{c}^b = [c_{b,1}, c_{b,2}, \dots c_{b,d}]$ are the *coupling constants*. If all the c^a's are null, we say that there is one-way coupling from a to b, since the state of a influences b but b does no influence a. If $c_{a,i} \neq 0$ and $c_{b,i} \neq 0$ for at least one i, we say that there is a two-way coupling. See Fig. 11.1 for an example.

System in (11.2) is, as a whole, a $2d$-dimensional dynamical system resulting from the coupling of the two original systems. Note that if synchronization is achieved, $\boldsymbol{x}_t^a = \boldsymbol{x}_t^b$: this means that the coupling terms are null.

In the $2d$-dimensional state-space of system (11.2), the synchronized state $\boldsymbol{x}^a = \boldsymbol{x}^b$ represents an d-dimensional invariant manifold. On this manifold, (11.2) reduces to (11.1).

11.1.2 Drive-Response Systems

In this section we introduce the concept of GS and relate it to the concept of ESP. To do so, we start by considering the source system (2.1) together with the reservoir (3.3) in a *drive-response system*:

$$\boldsymbol{s}_t = \boldsymbol{g}(\boldsymbol{s}_{t-1}) \qquad \text{(Drive)} \tag{11.3a}$$
$$\boldsymbol{r}_t = \boldsymbol{f}(\boldsymbol{r}_{t-1}, \boldsymbol{h}(\boldsymbol{s}_{t-1})) \qquad \text{(Response)} \tag{11.3b}$$

Where (11.3a) is the *drive* and (11.3b) the *response*. Together, they form an autonomous $(d_s + d_r)$-dimensional dynamical system which can be written as:

$$\boldsymbol{x}_t = \boldsymbol{G}(\boldsymbol{x}_{t-1}) \tag{11.4}$$

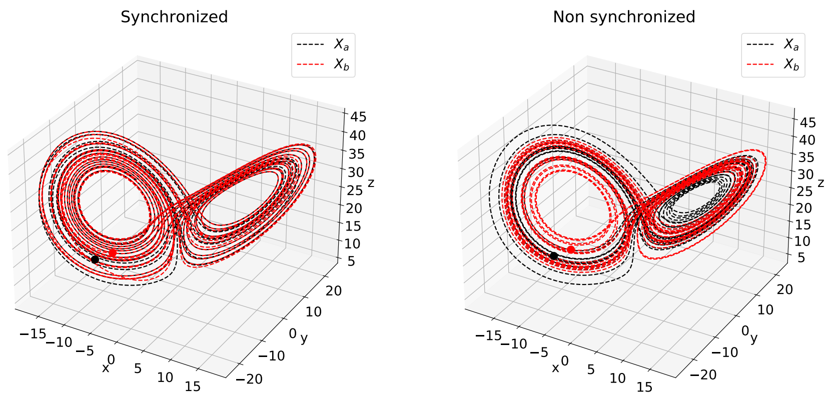


Figure 11.1. Two examples of (11.2) in which $\boldsymbol{F}$ is the Lorenz system (Appendix C.1). In both panels, $\boldsymbol{x}_0^a$ and $\boldsymbol{x}_0^b$ are the same but on the left one the coupling strength is enough to lead to synchronization. In the right panel the coupling is too weak for the synchronization to occur.

where $\boldsymbol{x}$ is simply the concatenation of $\boldsymbol{s}$ and $\boldsymbol{r}$ and, accordingly, $\boldsymbol{G}$ represents the concatenation of the action of $\boldsymbol{g}$ and $\boldsymbol{f}$.

Let us now assume that (11.4) has an attractor $\mathscr{A}$ with a basin of attraction $\mathscr{B}$.[1] This attractor can be expressed as:

$$\mathscr{A} = \mathscr{A}_s \times \mathscr{A}_r$$

where $\mathscr{A}_s$ (respectively, $\mathscr{A}_r$) is the projection of $\mathscr{A}$ onto the d_s(respectively, d_r) coordinates of system (11.3a) (respectively, (11.3b)). The same holds for the basin of attraction $\mathscr{B}$ of $\mathscr{A}$, which can be expressed in an analogous way as

$$\mathscr{B} = \mathscr{B}_s \times \mathscr{B}_r$$

It is important to note that, because (11.3a) is an *autonomous dynamical system*, $\mathscr{A}_s$ is its attractor and $\mathscr{B}_s$ its basin of attraction. The nature of $\mathscr{A}_r$ and $\mathscr{B}_r$ is more complex, as (11.3b) is a non-autonomous dynamical system, for which the definition of attractor is more complicated (and non-uniquely defined, see Appendix A.2 for a brief discussion): for our purpose, $\mathscr{A}_r$ and $\mathscr{B}_r$ can be simply thought of as sets obtained by a projection of the whole system considering only the variables related to the response system.

[1]This assumption is required just to simplify the exposition. For the case where the system has multiple attractors see, for instance, Lu and Bassett [2020].

11.1.3 Complete Synchronization and Asymptotic Stability

In the framework introduced for system 11.3, we now introduce a *driven replica subsystem*:

$$\tilde{\boldsymbol{r}}_t = \boldsymbol{f}(\tilde{\boldsymbol{r}}_{t-1}, \boldsymbol{h}(\boldsymbol{s}_{t-1})) \tag{11.5}$$

Note that $\boldsymbol{f}$ is the same as in (11.3b). We then take the sequence of states $\boldsymbol{s}_t$ from (11.3a) and use $\boldsymbol{h}(\boldsymbol{s})$ to feed the replica subsystem (11.5). The complete synchronization [Pecora and Carroll, 1990] between the response (11.3b) and its replica (11.5) is defined as the identity of the trajectories of $\boldsymbol{r}$ and $\tilde{\boldsymbol{r}}$. In more formal terms, we are requiring the *asymptotic stability* of the response with respect to the replica subsystem [Boccaletti et al., 2002, Sec. 3.6].

Definition 6 (Asymptotic stability)**.** A dynamical system is said to be *asymptotically stable* if, for any two copies $\boldsymbol{r}$ and $\tilde{\boldsymbol{r}}$ of the system driven by the same input $\boldsymbol{u}_t$ and starting from different initial conditions in B_r, it holds that

$$\lim_{t\to\infty} \|\boldsymbol{r}(t, \boldsymbol{u}_{t+1}) - \tilde{\boldsymbol{r}}(t, \boldsymbol{u}_{t+1})\| = 0 \tag{11.6}$$

In our case, $\boldsymbol{u}_t = \boldsymbol{h}(\boldsymbol{s}_t)$. The state of the full dynamical system is now constituted by (11.3) and (11.5), and thus it is $d_s + 2d_r$ dimensional. The synchronized state $\hat{\boldsymbol{r}} = \boldsymbol{r}$ represents an $(d_s + d_r)$-dimensional manifold embedded in the state-space of the full system.

11.1.4 Generalized Synchronization

As stated in the introduction, the concept of synchronization has been raising interest in the RC community. Here we introduce the GS, which is a generalization of the concept of synchronization for non-identical systems. A short introduction of the simpler case in which the synchronization occurs between identical system can be found in Appendix 11.1.1). We introduce GS following the definition used in Parlitz [2012]:

Definition 7 (Generalized Synchronization)**.** A system like (11.3) possesses the property of Generalized Synchronization (GS) when there exists a transformation

$$\boldsymbol{\phi} : \quad \mathbb{R}^{d_s} \to \mathbb{R}^{d_r} \tag{11.7}$$

$$\boldsymbol{s} \mapsto \boldsymbol{\phi}(\boldsymbol{s}) \tag{11.8}$$

mapping the states of the drive (11.3a) into the states of the response (11.3b) for which:

$$\lim_{t\to\infty} \|\boldsymbol{r}_t - \boldsymbol{\phi}(\boldsymbol{s}_t)\| = 0 \tag{11.9}$$

This means that the response state $\boldsymbol{r}$ is asymptotically given by the state of the driving system $\boldsymbol{s}$ and there exists a invariant set $\mathscr{M}$ in the full state-space of the system defined by the equation:

$$\boldsymbol{r} = \boldsymbol{\phi}(\boldsymbol{s}). \tag{11.10}$$

i.e., $\mathscr{M} := \{(\boldsymbol{s},\boldsymbol{r}) \in \mathscr{A} : \boldsymbol{r} = \boldsymbol{\phi}(\boldsymbol{s})\}$. If we assume that $\boldsymbol{\phi}$ is smooth, $\mathscr{M}$ is indeed a manifold and we refer to it as the *synchronization manifold*. Moreover we can define $\mathscr{B}_{\mathscr{M}}$ as the set of initial conditions for which (11.9) holds. Then $\mathscr{B}_{\mathscr{M}} \subseteq \mathscr{B}$. As noted in Rulkov et al. [1995], if a synchronizing relationship of the form (11.10) occurs, it means that the motion of the system in the full space has collapsed onto a subspace which is the manifold of the synchronized motion $\mathscr{M}$. This manifold is invariant, in the sense that $\boldsymbol{r}_t = \boldsymbol{\phi}(\boldsymbol{s}_t)$ implies $\boldsymbol{r}_{t+1} = \boldsymbol{\phi}(\boldsymbol{s}_{t+1})$. Moreover, (11.9) implies that such a manifold must be attracting Parlitz [2012].

Since the relationship defined in (11.9) should hold on the attractor $\mathscr{A}_s$, which the drive system approaches asymptotically, it makes sense to write the attractor of the response system as $\mathscr{A}_r = \boldsymbol{\phi}(\mathscr{A}_s)$.

We assume $\boldsymbol{\phi}$ to be smooth (which can be theoretically granted for a large class of systems Grigoryeva et al. [2020]). This is necessary as simple existence $\mathscr{M}$ is not enough to guarantee any regularity, even when it is attracting [Stark, 1999]. In general, distinguishing the case in which $\boldsymbol{\phi}$ does not exist from when it is very very irregular is impossible, at least practically Parlitz [2012], so the smoothness of $\boldsymbol{\phi}$ is a fundamental assumption. The case in which the synchronization function exists but is complicated or even fractal is called *Weak Synchronization* Pyragas [1996]; this case is not taken into account in our paper.

If $\boldsymbol{\phi}$ equals the identity transformation, this general definition of synchronization coincides with the definition of identical synchronization (see Sec. 11.1.1).

11.2 Generalized Synchronization and Learning

11.2.1 ESP and GS

The assumption that the input $\boldsymbol{u}_t$ is given by (2.7) (i.e., it is a function of the state of an autonomous dynamical system) makes it possible to explore the matter in more depth. We start by noticing that, in this framework, a sequence $\{u_t\}$ is uniquely defined by an initial condition of (2.1) as:

$$\boldsymbol{u}_t = \boldsymbol{h}(\boldsymbol{s}_t) = \boldsymbol{h}(\boldsymbol{g}^t(\boldsymbol{s}_0))$$

which holds for any t as $\boldsymbol{g}$ is assumed to be invertible. This means that each left-infinite sequence of measurements can be uniquely associated to a state of the source system (2.1) so that:

$$E(\ldots,\boldsymbol{u}_{t-1},\boldsymbol{u}_t) = E(\ldots,\boldsymbol{h}\circ\boldsymbol{g}^{-1}\circ\boldsymbol{s}_t,\boldsymbol{h}\circ\boldsymbol{s}_t)$$

which is clearly a function of $\boldsymbol{s}_t$ only and is, in fact, (11.10). Within our framework, the existence of an input echo function is equivalent to the existence of a synchronization function, i.e.:

$$E(\ldots,\boldsymbol{u}_{t-1},\boldsymbol{u}_t) = \boldsymbol{\phi}(\boldsymbol{s}_t)$$

Yet, the analogy is not perfect as the ESP requires $\boldsymbol{\phi}$ to be *unique*. Non-uniqueness means that there exists $p > 1$ different synchronization manifolds, each one given by a different synchronization function $\mathscr{M}^i := \{(\boldsymbol{s},\boldsymbol{r}) : \boldsymbol{r} = \boldsymbol{\phi}^i(\boldsymbol{s})\}, i = 1,\ldots,p$. In [Grigoryeva et al., 2020] the authors show that this phenomenon can be avoided by ensuring local contractivity of $\boldsymbol{f}$, i.e. $\boldsymbol{f}$ should operate as a contraction on each separate manifold $\mathscr{M}^i$.

When evaluating the reservoir system performance (which is needed in order to perform hyper-parameters tuning), one usually compares single realizations of the reservoir and of the input signal, i.e. a specific instance of the reservoir with its initial condition is trained on an input signal. This means that the uniqueness is not practically exploited in most practical context, and sometimes it might even be detrimental (see the concept of “echo index” introduced in [Ceni et al., 2020]): for this reason, in this paper we simply explore the GS disregarding its uniqueness.

11.2.2 Unsupervised System Reconstruction During the Listening Phase

In the scenario depicted above, one uses the reservoir $\boldsymbol{r}$ to create a representation of the input, which is finally processed by the readout $\boldsymbol{\psi}$. The goal is to generate a mapping from $\boldsymbol{s}$ to $\boldsymbol{y}$ and then to use such readout for generating $\hat{\boldsymbol{y}}_t$ for values of $\boldsymbol{u}_t$ which are not in the training set (i.e., for $t \geq 0$). Since $\boldsymbol{s}$ is unknown, what one really assumes is that it is possible to predict $\boldsymbol{y}$ from the knowledge of the whole history of $\boldsymbol{u}$. This is, in fact, an implication of *Takens embedding theorem* Takens [1981] and the feasibility of such a procedure was recently proved in the context of RC by Hart et al. [2020][2].

[2]Note that what they call *echo state map* (see Theorem 2.2.2 in Hart et al. [2020]) corresponds to the synchronization function in (11.10)

It is really important to emphasize the fact that we only consider the case where the fitting of the readout *does not* affect the reservoir dynamics in any way. The representation of the attractor of $\boldsymbol{s}$ into the reservoir states $\boldsymbol{r}$ by the use of the input sequence $\boldsymbol{u}$ is done in the listening phase, which is (in machine learning parlance) *unsupervised*. The fitting consists of trying to estimate the *static* function $\boldsymbol{k}$ mapping the state $\boldsymbol{s}$ to the desired output $\boldsymbol{y}$, i.e,

$$\hat{\boldsymbol{y}}_t := \boldsymbol{\psi}(\boldsymbol{r}_t) \approx \boldsymbol{k}(\boldsymbol{s}_t) = \boldsymbol{y}_t \tag{11.11}$$

for each t. We now discuss the role that the listening phase has on the learning process.

Let us consider the time interval $(t_s, 0)$, in which we assume that the GS has occurred; remember that we assume negative times for the training phase, so $t_s < 0$. We consider the reservoir states generated in this interval,

$$\boldsymbol{R}_{(t_s,0)} = \begin{bmatrix} | & | & | & | \\ \boldsymbol{r}_{t_s} & \boldsymbol{r}_{t_s+1} & \dots & \boldsymbol{r}_0 \\ | & | & | & | \end{bmatrix} =$$
$$= \begin{bmatrix} | & | & | & | \\ \boldsymbol{r}_{t_s} & \boldsymbol{f}(\boldsymbol{r}_{t_s}, \boldsymbol{u}_{t_s}) & \dots & \boldsymbol{f}(\boldsymbol{r}_{-1}, \boldsymbol{u}_{-1}) \\ | & | & | & | \end{bmatrix} \tag{11.12}$$

GS guarantees that there exists a function mapping the source system states to the reservoir states and also its invariance. This means that

$$\boldsymbol{r}_t = \boldsymbol{\phi}(\boldsymbol{s}_t) \Rightarrow \boldsymbol{r}_{t+1} = \boldsymbol{\phi}(\boldsymbol{s}_{t+1}) = \boldsymbol{\phi}(\boldsymbol{g}(\boldsymbol{s}_t)) \tag{11.13}$$

so that (11.12) can be written as follows:

$$\boldsymbol{R}_{(t_s,0)} = \begin{bmatrix} | & | & | & | \\ \boldsymbol{\phi}(\boldsymbol{s}(t_s)) & \boldsymbol{\phi}(\boldsymbol{s}_{t_s+1}) & \dots & \boldsymbol{\phi}(\boldsymbol{s}_0) \\ | & | & | & | \end{bmatrix} \tag{11.14}$$

Note that $\boldsymbol{\phi}$ is a time-independent function that is the same for all $\boldsymbol{s}$. Since by assumption $d_r > d_s$, we can think of $\boldsymbol{\phi}$ as an attempt to expand the source system state-space (which is unknown) into a higher-dimensional space, in the same fashion as the well-known reproducing kernel Hilbert space mechanism behind kernel methods [Shawe-Taylor and Cristianini, 2004]: the reservoir dynamics performs a sort of nonlinear basis expansion of the (unknown) attractor of $\boldsymbol{s}$. The use of the synchronization function $\boldsymbol{\phi}$ provides a sound theoretical framework to the fitting process, and the relation (11.10) can be seen a sound formulation of the “reservoir trick”; see [Shi and Han, 2007]. Moreover note that such an expansion $\boldsymbol{\phi}$ was not computed or estimated from data, but was obtained as a result of driving the reservoir with the input sequence under consideration: this means that the mapping is “informed” of the dynamics. Accordingly, we can interpret (11.11) as follows:

$$\hat{\boldsymbol{y}}_t = \boldsymbol{\psi}(\boldsymbol{r}_t) = \boldsymbol{\psi}(\boldsymbol{\phi}(\boldsymbol{s}_t)) \approx \boldsymbol{k}(\boldsymbol{s}_t) = \boldsymbol{y}_t \tag{11.15}$$

11.2.3 Learning Realizability

We define the concept of “Realizable Learning” [Shalev-Shwartz and Ben-David, 2014] as the situation where the readout is perfectly able to reconstruct the targets by using the reservoir states. More formally,

Definition 8 (Learning Realizability)**.** We say that the learning is *realizable* if there exists a readout $\boldsymbol{\psi}$ such that,

$$\boldsymbol{y}_t = \boldsymbol{\psi}(\boldsymbol{r}_t),\ \forall t \tag{11.16}$$

The following theorem proves that for the learning to be realizable for a orbit of the source system, there must be a function mapping that orbit into the orbit of the reservoir. First we introduce some notation. Let us denote with $\mathscr{S} \subset \mathscr{A}_s$ the set containing all $\boldsymbol{s}_t$, for all t (this is the *orbit* of a system). Analogously, we define $\mathscr{R} \subset \mathscr{A}_r$ as the set of all $\boldsymbol{r}_t$, for all t. We define $\mathscr{Y}$ as the result of applying $\boldsymbol{k}$ to each point in $\mathscr{S}$, in short $\mathscr{Y} := \boldsymbol{k}(\mathscr{S})$.

Theorem 5. *A necessary condition for learning to be realizable is that for each $\boldsymbol{r} \in \mathscr{R}$ such that $\boldsymbol{\psi}(\boldsymbol{r}) = \boldsymbol{y}$, there exists a function $\mathscr{F} : \mathscr{S} \to \mathscr{R}$ such that $\boldsymbol{r} = \mathscr{F}(\boldsymbol{s})$, where $\boldsymbol{s}$ is such that by $\boldsymbol{k}(\boldsymbol{s}) = \boldsymbol{y}$.*

Proof. Realizability of learning implies that $\boldsymbol{\psi}$ is surjective when mapping $\mathscr{R}$ into $\mathscr{Y}$. The surjectivity of $\boldsymbol{k}$ is guaranteed by the way we constructed $\mathscr{Y}$. But because different source system states could result in the same target, $\boldsymbol{k}$ may not be an injective function. The same holds for $\boldsymbol{\psi}$. We define $\boldsymbol{\psi}^\dagger(\boldsymbol{y})$ as a function mapping each $\boldsymbol{y}$ onto an $\boldsymbol{r}$: if $\boldsymbol{\psi}$ is also injective, then $\boldsymbol{\psi}^\dagger$ corresponds to the inverse of $\boldsymbol{\psi}$, but in general it is not. These functions are called *right-inverse* since $\boldsymbol{\psi} \circ \boldsymbol{\psi}^\dagger$ is the identity but $\boldsymbol{\psi}^\dagger \circ \boldsymbol{\psi}$ is not. Since by definition $\boldsymbol{y}_t = \boldsymbol{k}(\boldsymbol{s}_t)$, it will then be possible to construct the function $\mathscr{F}$ as follows:

$$\mathscr{F} = \boldsymbol{\psi}^\dagger \circ \boldsymbol{k} \tag{11.17}$$

Such a function maps all $\boldsymbol{s}$ into the corresponding targets $\boldsymbol{y}$ and inverts the readout function $\boldsymbol{\psi}$ so that it maps each target to a corresponding reservoir state $\boldsymbol{r}$.

So far, we have shown that (11.17) maps each $\boldsymbol{s}$ to an $\boldsymbol{r}$. We also need to make sure that each $\boldsymbol{r}$ can be written as $\mathscr{F}(\boldsymbol{s})$. This is granted by our assumption (surjectivity of $\boldsymbol{\psi}$) which tells us that each $\boldsymbol{y}$ can be written as $\boldsymbol{\psi}(\boldsymbol{r})$. Then, using an argument analogous to the one above, we can associate each $\boldsymbol{y} \in \mathscr{Y}$ to an $\boldsymbol{s} \in \mathscr{S}$ by defining $\boldsymbol{k}^{\dagger}$. Again, this corresponds to the inverse of $\boldsymbol{k}$ only if $\boldsymbol{k}$ is also injective. Note that, in general, distinct values of $\boldsymbol{r}$ might be associated to the same $\boldsymbol{s}$ (and *viceversa*). This shows that if learning is realizable, then $\mathscr{F}$ must exist. □

The theorem also implies that if $\mathscr{F}$ does not exist, then learning is not realizable. So, any successful training procedure must (i) develop an (implicit) mapping from $\mathscr{S}$ to $\mathscr{R}$ and (ii) find a suitable readout. Yet, the existence of $\mathscr{F}$ *does not necessarily imply* the realizability of learning: we have no guarantees that, in the presence of such a mapping, a readout solving the problem can be found. Moreover, the fact that learning is not realizable *does not necessarily imply* that $\mathscr{F}$ does not exist: the problem might simply be that we are not able to conceive the right readout.

We now prove that, by requiring $\mathscr{F}$ to be injective, we can always construct a readout which correctly solves the problem.

Theorem 6. *A sufficient condition for the learning to be realizable is that there exists a function $\mathscr{F}$ such that for all $\boldsymbol{r} \in \mathscr{R}, \boldsymbol{r} = \mathscr{F}(\boldsymbol{s})$ and $\mathscr{F}$ is injective.*

Before proving the theorem, we make a remark:

Remark 1. In Theorem 6, the condition that $\boldsymbol{r} = \mathscr{F}(\boldsymbol{s})$ must hold for all $\boldsymbol{r} \in \mathscr{R}$ means that $\mathscr{F} : \mathscr{S} \to \mathscr{R}$ must be surjective. This means that when $\mathscr{F}$ is injective, it is in fact bijective and so, invertible.

The proof of the theorem in now trivial:

Proof. As discussed in Remark 1, the injectivity of $\mathscr{F}$ grants the existence of its inverse $\mathscr{F}^{-1}$. The readout function solving (11.16) then exists and it is given by:

$$\boldsymbol{\psi} = \boldsymbol{k} \circ \mathscr{F}^{-1} \tag{11.18}$$

□

The fact that $\mathscr{F}$ is injective means that it always maps distinct $\boldsymbol{s}$ into distinct $\boldsymbol{r}$. Without it, $\mathscr{F}$ may map two distinct $\boldsymbol{s}_1, \boldsymbol{s}_2$ into the same $\boldsymbol{r} = \mathscr{F}(\boldsymbol{s}_1) = \mathscr{F}(\boldsymbol{s}_2)$:

the realizability of learning then depends on whether $\boldsymbol{k}(\boldsymbol{s}_1) = \boldsymbol{k}(\boldsymbol{s}_2) = \boldsymbol{y}$ or not. This is why the existence of $\mathscr{F}$ is not sufficient by itself. An analogous concept was introduced by Maass et al. [2002], where the authors use the term *separation property*. We emphasize the fact that the separation property requires the system to differentiate between distinct input signals ($\{\boldsymbol{u}_t\}$ in our notation), while here the discrimination must be performed on the source system states $\boldsymbol{s}_t$

It is important to note that both $\boldsymbol{k}$ and $\mathscr{F}$ are unknown in our problem setting, so that the theorem only guarantees the possibility of finding the right $\boldsymbol{\psi}$ but does not provide a constructive way of finding it. Therefore, when learning is not realizable it is generally impossible to understand whether the problem is related to $\mathscr{F}$, to $\boldsymbol{\psi}$, or even to the both of them.

Notably, as we will discuss later, this problem can be bypassed by considering the synchronization function $\boldsymbol{\phi}$ as a surrogate for $\mathscr{F}$. As $\boldsymbol{\phi}$ is only related to the dynamical evolution of the reservoir (listening phase), we can discuss its existence and properties disregarding the readout.

This shows the importance of $\boldsymbol{\phi}$ in the context of RC: it can be used to asses the quality of the representation of the unknown source system that the reservoir has encoded in its state. This allows one to disentangle the problem of embedding the input (which is done in an unsupervised way during the listeining phase) from the the problem of finding the best readout to predict the target (which is a supervised problem, faced in the fitting phase). This fact is of particular interest as most of the hyperparameters that are usually optimized (e.g., the spectral radius of the connectivity matrix, its sparsity, the input scaling, the activation function) affect the listening phase only and, therefore, the synchronization. Hence, their analysis and optimization can be performed disregarding the fitting procedure.

Finally, we point out that Theorem 6 formally proves that – as suggested in other works Lu et al. [2018]; Lu and Bassett [2020] – the existence of an invertible synchronization function is sufficient for the RC paradigm to work (provided that the readout is able to correctly approximate the target). We proved that this condition applies not only in the generative frameworks (i.e., when $\boldsymbol{y}_t = \boldsymbol{s}_{t+1}$) which is the one studied in [Lu et al., 2018; Lu and Bassett, 2020], but to any generic target $\boldsymbol{y}_t = \boldsymbol{k}(\boldsymbol{s}_t)$.

11.2.4 Error on the Whole Attractor

Since the readout $\boldsymbol{\psi}$ is generated after the listening phase, we have no guarantees that, in general, it will continue to correctly reproduce the target also in the predicting phase. More in detail, after observing a series of measurements $\boldsymbol{u}_t$

and targets $\boldsymbol{y}_t$ coming from an *unknown* trajectory of the source system $\boldsymbol{s}_t$, we want to learn a readout $\boldsymbol{\psi}$ which is able to predict the targets even for future times.

Since we have assumed that the source system (2.1) has a unique attractor $\mathscr{A}$, this goal can be achieved by learning a readout valid for all the $\boldsymbol{y} = \boldsymbol{k}(\boldsymbol{s})$, for $\boldsymbol{s} \in \mathscr{A}$. In the machine learning parlance, this can be described as follows: a single trajectory plays the role of a sample, while the attractor plays the role of the data-generating process. This becomes possible by assuming the attractor $\mathscr{A}_s$ to be ergodic [Birkhoff, 1931]. In fact, the existence of an ergodic attractor guarantees that a sufficiently long trajectory will be a "good sampling" of the whole attractor (see [Hart et al., 2021] for a discussion about the generative framework). Moreover, as all trajectories starting from the basin of attraction $\mathscr{B}_s$ will approach $\mathscr{A}_s$, this procedure allows to learn a prediction model suitable for a full set of trajectories by observing only one.

To do so, let us define the loss function:

$$\mathscr{L}(\boldsymbol{y}_t, \hat{\boldsymbol{y}}_t) = \|\boldsymbol{y}_t - \hat{\boldsymbol{y}}_t\|_2 \tag{11.19}$$

where $\|\cdot\|_2$ is the L_2-norm. We refer to (11.19) as Root Mean Square Error (RMSE).[3] The learning realizability trivially implies that there exists a readout for which:

$$\frac{1}{T}\sum_{t=t_s}^{T} \mathscr{L}(\boldsymbol{y}_t, \hat{\boldsymbol{y}}_t) = 0 \tag{11.20}$$

since $\mathscr{L}(\boldsymbol{y}_t, \hat{\boldsymbol{y}}_t) = 0$, for each t. Note that time starts at $t = t_s$ because we want to remove transient effects (as our discussion is valid on the attractor only).

By expanding $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$, we get:

$$\frac{1}{T}\sum_{t=t_s}^{T} \mathscr{L}(\boldsymbol{k}(\boldsymbol{s}_t), \boldsymbol{\psi}(\boldsymbol{r}_t)) = 0 \tag{11.21}$$

The existence of a function $\mathscr{F}$ (Thm. 6) allows us to write $\boldsymbol{r}_t = \mathscr{F}(\boldsymbol{s}_t)$, so that our loss becomes

$$\mathscr{L}(\boldsymbol{k}(\boldsymbol{s}_t), \boldsymbol{\psi}(\boldsymbol{r}_t)) = \mathscr{L}(\boldsymbol{k}(\boldsymbol{s}_t), \boldsymbol{\psi}(\mathscr{F}(\boldsymbol{s}_t))) = \mathscr{L}(\boldsymbol{s}_t) \tag{11.22}$$

where the last equality stresses the fact that $\mathscr{L}$ is a function of $\boldsymbol{s}_t$ only (with an abuse of notation on the function $\mathscr{L}$). Taking the limit for $T \to \infty$, we can now

[3]Note that different choices can be made for $\mathscr{L}$ and the results do not depend on its particular form. We use the RMSE because it is the one we use in the esperimental section.

exploit the ergodicity[4] of $\mathscr{A}_s$ and obtain:

$$0 = \lim_{T\to\infty} \frac{1}{T}\sum_{t=t_s}^{T} \mathscr{L}(y_t, \hat{y}_t) = \underbrace{\lim_{T\to\infty} \frac{1}{T}\sum_{t=t_s}^{T} \mathscr{L}(s_t) = \langle \mathscr{L}(s)\rangle_{\mathscr{A}_s}}_{\text{Ergodicity}} \tag{11.23}$$

In (11.23), $\langle \mathscr{L}(s)\rangle_{\mathscr{A}_s}$ denotes the average loss by sampling trajectories over the whole attractor $\mathscr{A}_s$. This means that, if learning is realizable for a single trajectory, then it will be realizable on the whole attractor of the source system.

Note that the crucial part of this approach is the dependence on s only, because only the source system attractor $\mathscr{A}_s$ is assumed to be ergodic.

11.2.5 Synchronization Function

One would like to relax the definition of realizable learning (see Def. 8): in fact, in realistic situations the error is not exactly zero. This is formalized by assuming that $\mathscr{L}(y_t, \hat{y}_t) = \epsilon_t \geq 0, \forall t$, so that:

$$\mathscr{E}_T = \frac{1}{T}\sum_{t=t_s}^{T} \mathscr{L}(y_t, \hat{y}_t) \tag{11.24}$$

As proved in the previous section, if a mapping $\mathscr{F}$ does not exist, then learning cannot be realizable according to Def. 8. But assuming GS to hold, we can make use of the synchronization function ϕ and write:

$$\begin{aligned} \frac{1}{T}\sum_{t=t_s}^{T} \mathscr{L}(y_t, \hat{y}_t) &= \frac{1}{T}\sum_{t=t_s}^{T} \mathscr{L}(k(s_t), \psi(\phi(s_t))) \\ &= \frac{1}{T}\sum_{t=t_s}^{T} \mathscr{L}(s_t) \end{aligned} \tag{11.25}$$

where, again, $\mathscr{L}$ depends only on s_t. In order to make use of the ergodicity of the source system, we need to be sure that the above limit exists. An easy way for guaranteeing this consists of requiring the error to be bounded, i.e., to have $\mathscr{L}(y_t, \hat{y}_t) = \epsilon_t < C$, where $C \geq 0$ is a constant. So, when $\lim_{T\to\infty} E_T$ exists and is finite, one can write:

$$\mathscr{E} = \lim_{T\to\infty} \mathscr{E}_T = \lim_{T\to\infty} \frac{1}{T}\sum_{t=t_s}^{T} \mathscr{L}(s_t) = \langle \mathscr{L}(s_t)\rangle_{\mathscr{A}_s} \tag{11.26}$$

[4] This is indeed the celebrated *Birkhoff's Ergodic Theorem* Birkhoff [1931], which states that for any ergodic system the time average from almost any initial condition equals the space average. For a modern discussion about ergodicity in the context of RC see Hart et al. [2021].

The existence of the synchronization function guarantees that the error for a single trajectory is the same as the error in the whole attractor of the source system. This means that, by assuming GS, we can have some guarantees on the performance of our model even when learning is not realizable, and this is due to the fact that $\mathscr{L}$ depends only on the source system states $\boldsymbol{s}$ when assuming GS. We emphasize that this argument applies not only to future time of the the same trajectory, but also to any trajectory of the source system (2.1) which starts from $\mathscr{B}_s$. Practically speaking, this means that if we have a trajectory of the source system starting at $\boldsymbol{s}'(0) \in \mathscr{B}_s$, the readout that was previously trained will still work, because this new trajectory will still approach the same attractor $\mathscr{A}_s$ and the reservoir will approach the synchronization manifold. Note that if the GS is granted but it is not unique (i.e, we do not have the ESP), the reservoir needs to be initialized with the same initial condition, otherwise it may converge to a different synchronization manifold, which would require a different readout; which always exists as long as the new synchronization function is invertible. The ESP ensures that the readout will be the same, as the synchronization function will be unique. In fact, the definition of the synchronization function in Def. 7 has other implications for the training mechanisms.

Making use of its smoothness along with the attractivity of the synchronization manifold, one can account not only for the error in the approximation, but also for the observational noise of the source system.

In many real situations the input is corrupted by some noise, so that instead of reading just $\boldsymbol{u}_t$ one actually reads $\boldsymbol{u}_t + \boldsymbol{\epsilon}_t$, $\boldsymbol{\epsilon}_t$ being i.i.d. noise. This lead to the following state-update for the reservoir:

$$\begin{aligned} \boldsymbol{r}_{t+1} &= \boldsymbol{f}(\boldsymbol{r}_t, \boldsymbol{u}_t + \boldsymbol{\epsilon}_t) \\ &\approx \boldsymbol{f}(\boldsymbol{r}_t, \boldsymbol{u}_t) + \boldsymbol{f}'(\boldsymbol{r}_t, \boldsymbol{u}_t)\boldsymbol{\epsilon}_t \end{aligned} \tag{11.27}$$

This will affect the reservoir dynamics in general, but when the noise is small we can still hope that the trajectory will not be too far from the one generated without noise. That is, we assume it is possible to write each point as $\boldsymbol{r}_t + \boldsymbol{\eta}_t$. This will be in fact guaranteed by the GS, which requires the synchronization manifold not only to exist, but also to be attractive Kocarev and Parlitz [1996]. Note that $\boldsymbol{\eta}_t$ is not i.i.d. anymore.

The synchronization problem (w.r.t. to the *true* system state) becomes:

$$\boldsymbol{s} = \boldsymbol{\phi}(\boldsymbol{r} + \boldsymbol{\eta}) \tag{11.28}$$

We can make use of the smoothness of $\boldsymbol{\phi}$ to write a first-order approximation of the source system state as follows:

$$\boldsymbol{s} \approx \boldsymbol{\phi}(\boldsymbol{r}) + \boldsymbol{\phi}'(\boldsymbol{r})\boldsymbol{\eta} \tag{11.29}$$

Such an approximation allows us to introduce a measure of *synchronization error* due to noise, which reads:

$$E_n := \|\boldsymbol{s} - \boldsymbol{\phi}(\boldsymbol{r})\| \approx \|\boldsymbol{\phi}'\boldsymbol{\eta}\| \leq \|\boldsymbol{\phi}'\|\|\boldsymbol{\eta}\| \tag{11.30}$$

For the observer task (see 11.3.3), in the common case of a linear readout, $\boldsymbol{\phi}'$ is simply the pseudo-inverse of the readout matrix $\boldsymbol{W}^*_{\text{out}}$, whose singular values are the reciprocal of the singular values of $\boldsymbol{W}_{\text{out}}$. This implies the following bound on the synchronization error due to noise,

$$E_n \leq \|\boldsymbol{W}^*_{\text{out}}\|\|\boldsymbol{\eta}\| = \frac{\|\boldsymbol{\eta}\|}{\min_i \sigma_i(\boldsymbol{W}_{\text{out}})} \tag{11.31}$$

where $\sigma_i(\boldsymbol{W}_{\text{out}})$ denote the non-null singular values of $\boldsymbol{W}_{\text{out}}$.

Finally, we stress that the existence of GS is a property which only involves the source system (2.1) and its coupling with the reservoir (3.3) by means of the measurements (2.7), disregarding the particular task at hand. In fact, in the discussion above, we showed how using the synchronization function $\boldsymbol{\phi}$ one can, in some sense, decouple the learning task and separate the problem of finding a suitable readout from the problem of granting the existence of a mapping from the source system states, $\boldsymbol{s}$, to the reservoir states, $\boldsymbol{r}$.

11.3 Experimental Results

11.3.1 The Mutual False Nearest Neighbors

Identifying GS is hard due to the fact that the synchronization function (11.10) is unknown and may take any form. For this reason, in Rulkov et al. [1995] a method to empirically assess the occurrence of GS from data was proposed under the name of Mutual False Nearest Neighbors (MFNN). It is based on the fact that, under reasonable smoothness conditions for $\boldsymbol{\phi}$, (11.10) implies that two states that are close in state-space of the response system correspond to two close states in the state-space of the driving system. So, we are looking for a geometric connection between the two systems which preserves the neighbor-structure in state space.

Let us assume that we sample trajectories from a dynamical system at a fixed sampling rate, resulting in a series of discrete times $\{t_n\}$. The resulting measurements for the two systems will be $\{\boldsymbol{x}_n\}$ and $\{\boldsymbol{r}_n\}$, for the drive and the response respectively, where we used the notation $\boldsymbol{x}_n := \boldsymbol{x}(t_n)$ and $\boldsymbol{r}_n := \boldsymbol{r}(t_n)$. For each point $\boldsymbol{x}_n$ of the driving system, we seek the closest point from its neighbors, which we will call time index n_{NND}. Then, due to (11.10), the point $\boldsymbol{r}_n = \boldsymbol{\phi}(\boldsymbol{s}_n)$ will be close to $\boldsymbol{r}_{n_{\mathrm{NND}}}$. If the distances between these pairs of points in state-space of both the drive and response systems are small, one can write:

$$\boldsymbol{r}_n - \boldsymbol{r}_{n_{\mathrm{NND}}} = \boldsymbol{\phi}(\boldsymbol{s}_n) - \boldsymbol{\phi}(\boldsymbol{s}_{n_{\mathrm{NND}}}) \approx \boldsymbol{D\phi}(\boldsymbol{s}_n)(\boldsymbol{s}_n - \boldsymbol{s}_{n_{\mathrm{NND}}}) \tag{11.32}$$

where $\boldsymbol{D\phi}(\boldsymbol{s}_n)$ is the Jacobian of $\boldsymbol{\phi}$ evaluated at $\boldsymbol{s}_n$.

Now, we do a similar operation but in the response state space. We look for the closer point to $\boldsymbol{r}_n$ and we index it with n_{NNR}. Again, due to (11.10), it holds:

$$\boldsymbol{r}_n - \boldsymbol{r}_{n_{\mathrm{NNR}}} = \boldsymbol{\phi}(\boldsymbol{s}_n) - \boldsymbol{\phi}(\boldsymbol{s}_{n_{\mathrm{NNR}}}) \approx \boldsymbol{D\phi}(\boldsymbol{s}_n)(\boldsymbol{s}_n - \boldsymbol{s}_{n_{\mathrm{NNR}}}) \tag{11.33}$$

We highlight the fact that (11.33) requires the continuity of $\boldsymbol{\phi}^{-1}$ since points that are close in the response system's domain are also close in the drive system's domain.

So, due to (11.32) and (11.33) we have two different ways of evaluating $\boldsymbol{D\phi}(\boldsymbol{s}_n)$. This leads us to the definition of the MFNN as the following ratio:

$$\mathrm{MFNN}(n) := \frac{\|\boldsymbol{r}_n - \boldsymbol{r}_{n_{\mathrm{NND}}}\|}{\|\boldsymbol{s}_n - \boldsymbol{s}_{n_{\mathrm{NND}}}\|} \frac{\|\boldsymbol{s}_n - \boldsymbol{s}_{n_{\mathrm{NNR}}}\|}{\|\boldsymbol{r}_n - \boldsymbol{r}_{n_{\mathrm{NNR}}}\|} \tag{11.34}$$

If the two systems are synchronized in a general sense, then $\mathrm{MFNN}(n) \approx 1$. If the synchronization relation does not hold, then (11.34) should instead be of the order of (size of the attractor squared)/(distance between nearest neighbors squared) which is, in general, a large number.

Note that in this work we use the full knowledge of the source system to measure the GS by means of MFNN. Generally, one would not have such a knowledge: anyway the MFNN can be used also in this case, as showed in the paper where it was proposed Rulkov et al. [1995], making use of the embedding theorem. For simplicity, we do not deal with this more complex case, since it would not be relevant for the discussion. See Boccaletti et al. [2002, Chapter 7.1] for a review about the empirical detection of synchronization.

Another possible way of assessing GS is to verify the complete synchronization (see Appendix 11.1.3) between multiple copies of the reservoir. While such an approach might be hard to follow when considering physical systems, it poses no problem and can be straightforwardly implemented in the context we are interested in (See Platt et al. [2021]). Note that this method may misidentify some form of synchronization. See Parlitz [2012] for a detailed discussion on the different methods used to empirically detect synchronization.

11.3.2 Reservoir Computing Networks

For simplicity, but without loss of generality, we will only deal with one-dimensional inputs $u_t \in \mathbb{R}$. We use an RCN where the explicit form of the reservoir equation (3.3) reads:

$$\boldsymbol{r}_t = \tanh(\boldsymbol{W}\boldsymbol{r}_{t-1} + \boldsymbol{w}u_{t-1} + \boldsymbol{b}) \tag{11.35}$$

$\boldsymbol{W} \in \mathbb{R}^{d_r \times d_r}$ is the *connectivity matrix*, which is an Erdos-Renyi matrix with average degree 6; d_r indicates the dimension of the reservoir. The non-null elements are drawn from a uniform distribution taking values in the interval $(-1, 1)$. $\boldsymbol{W}$ is re-scaled so that its *SR* equals the user-defined hyper-parameter $\rho > 0$. We emphasize that having a SR smaller than 1 is not a sufficient nor necessary condition for the ESP to hold Yildiz et al. [2012]. The elements $\boldsymbol{w} \in \mathbb{R}^{d_r}$ of the *input vector* are drawn from a uniform distribution taking values in $(-\omega, \omega)$; we refer to ω as the *input scaling* hyper-parameter. $\boldsymbol{b} = [b, b, \dots, b]$ is a constant bias term, which is useful to control the non-linearity of the system and to break the symmetry with respect to the origin Lu et al. [2017]. Here, tanh stands for the hyperbolic-tangent function applied element-wise.

We use a linear readout, so that the predicted output is given by:

$$\hat{\boldsymbol{y}}_t = \boldsymbol{\psi}(\boldsymbol{r}_t) \equiv \boldsymbol{W}_{\text{out}}\boldsymbol{r}_t \tag{11.36}$$

where $\boldsymbol{W}_{\text{out}}$ is a $d_y \times d_r$ matrix, called readout matrix; d_y is the output dimension. We train the readout using ridge-regression with regularization parameter λ, but more sophisticated, offline optimization procedures can be designed as well [Gallicchio et al., 2017; Shi and Han, 2007; Løkse et al., 2017].

11.3.3 Reservoir observer

We test the hypothesis that learning in RC can happen only when the network is synchronized with the source system (2.1). To do so, we adopt the framework named *reservoir observer* Lu et al. [2017], which consists of setting $\boldsymbol{h}(\boldsymbol{s}) = s_1 = u$ and $\boldsymbol{y} = \boldsymbol{s}$. This means that the network is trained to reconstruct the full state of the source system by seeing only one component of it. Note that $\boldsymbol{k}$ in (2.10) is the identity for this task, and (11.15) reduces to finding the right inverse of the synchronization function (11.10). Practically, when using a linear readout (as in fact we do here), one implicitly assumes that $\boldsymbol{k} \circ \boldsymbol{\phi}^{\dagger}(\boldsymbol{r})$ in (11.15) can be expressed in linear form

$$\boldsymbol{\phi}^{\dagger}(\boldsymbol{r}) = \boldsymbol{W}_{\text{out}}\boldsymbol{r} \tag{11.37}$$

implying that

$$\boldsymbol{\phi}(\boldsymbol{s}) = \boldsymbol{W}^{*}_{\text{out}}\boldsymbol{s} \tag{11.38}$$

where $\boldsymbol{W}_{\text{out}}^{*}$ is the left pseudo-inverse of the readout matrix.

11.3.4 Results

As for the source system (2.1), we use the Lorenz model (see Appendix C.1 for details). In the listening phase, we use t in the interval of $T_{\text{train}} = (-100, 0)$. We discarded the first 1/10 of the data points for training, to account for transient effects in the synchronization process. The prediction phase was carried out for values of t in $T_{\text{test}} = (0, 80)$. The integration step was always set to $\tau = 0.05$. An example of this task is provided in Fig. 11.2.

For each hyper-parameter configuration, we repeated the experiment 10 times, generating a different realization of the source system (i.e., starting from distinct random initial conditions) and a different realization of the RCN (11.35). For each run, the MFNN between the driving system state $\boldsymbol{s}_t$ and the reservoir $\boldsymbol{r}_t$ was computed. As a performance measure for the prediction accuracy, we used the RMSE computed on the y and the z coordinate of the Lorenz system (since x is used as input). Following Rulkov et al. [1995], we plot the inverse of the MFNN. Accordingly, we plot the inverse of the RMSE, which can be interpreted as a form of accuracy. Unless differently stated, all hyperparameters are the ones reported in Tab. 11.1. All the plots refer to the *predicting phase*.

Table 11.1. Default hyper–parameters used in all experiments, unless differently stated.

ρ	1	T_{train}	$(-100, 0)$
ω	0.1	T_{test}	$(0, 80)$
d_r	300	τ	0.05
b	1	λ	10^{-6}

In Fig. 11.3, we show the reciprocals of RMSE and of the MFNN index when the SR of the reservoir connectivity matrix varies in a suitable range. For smaller values of SR the reservoir dynamics are really simple and close to linear (since $\tanh(x) \approx x$ when x is small), so that the network it is not able to correctly represent the Lorenz attractor in its state. Note that, as pointed out in Lymburn et al. [2019], in this regime the reservoir is synchronized with the source system, but the synchronization functiion is non-invertible as the reservoir cannot display a correct representation of the source system in its states. This condition has been correctly identified by the MFNN test and the RMSE (since the reconstruction of the state is not possible,see Theorem 6). As the SR approaches 1 we

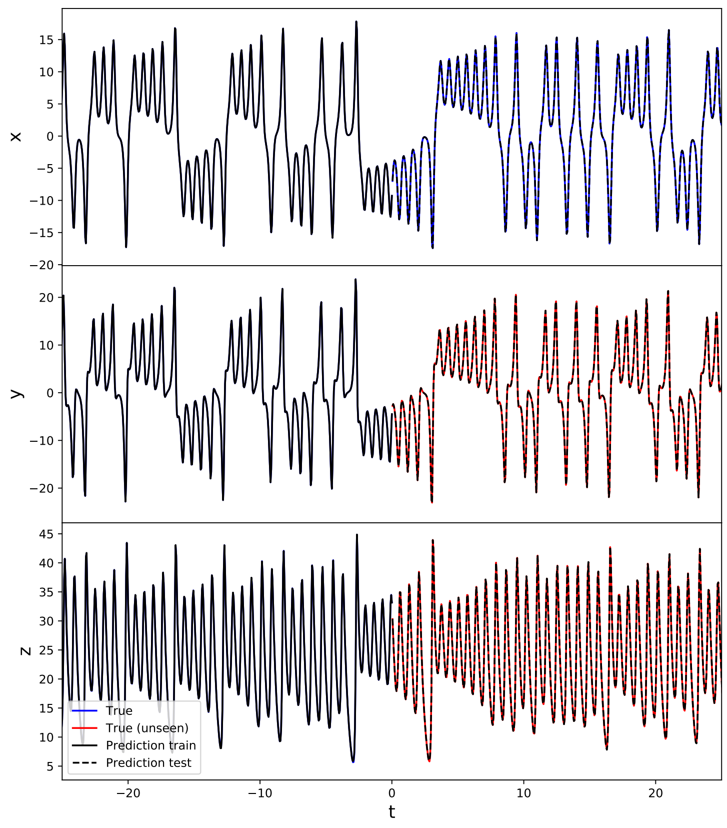


Figure 11.2. An example of the observer task using the Lorenz system. The top panel show the measurement $\boldsymbol{u}$ (blue), which is always available. The middle and the bottom panels represents the targets $\boldsymbol{y}$: in the training phase they are available (blue) while in the predicting phase (red) they cannot be accessed anymore. The predicted targets $\hat{\boldsymbol{y}}$ (black dashed lines) are generated by means of the RCN described in Sec. 11.3.2.

see that both measures improve, as a richer representation of the source system is present into the reservoir states, which leads to an invertible ϕ. When the SR started growing, the reservoirs becomes more and more unstable and it gradually de-synchronizes with the source system, such that the reconstruction of the coordinates becomes less precise.

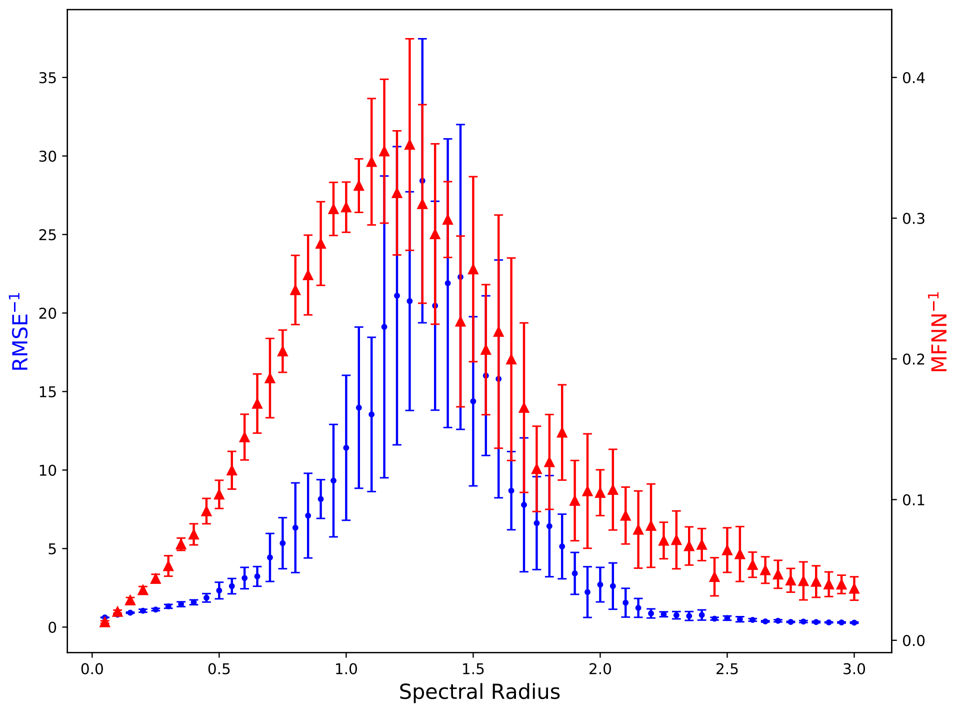


Figure 11.3. Results for the reservoir observer when varying the SR of the connectivity matrix, when using the Lorenz system as a source. Blue dots account for the reciprocal of the RMSE (left axis) while red triangles accounts for reciprocal of the MFNN (right axis).

In Fig. 11.4 we repeated the experiment, but this time varying the input scaling ω and holding ρ fixed to its default value. We see that the two quantities still correlates, with both the accuracy and the synchronization decreasing as the input scaling grows.

To assess the generality of our findings, we performed additional simulations by changing the source system (2.1). To this end, we consider now the Rössler system as a source system (see Appendix C.2 for details). Since the dynamics of the Rössler system are slower then the Lorenz ones, we set the integration step to $\tau = 0.5$, $T_{\text{train}} = (-200, 0)$ and $T_{\text{text}} = (0, 160)$. The remaining hyper-parameters are set as shown in Tab. 11.1. Again, we use the x-coordinate as input and the tasks consists of learning how to reproduce y and z. The results are displayed in Fig. 11.5 and look similar to the one obtained for the Lorenz system (Fig. 11.3).

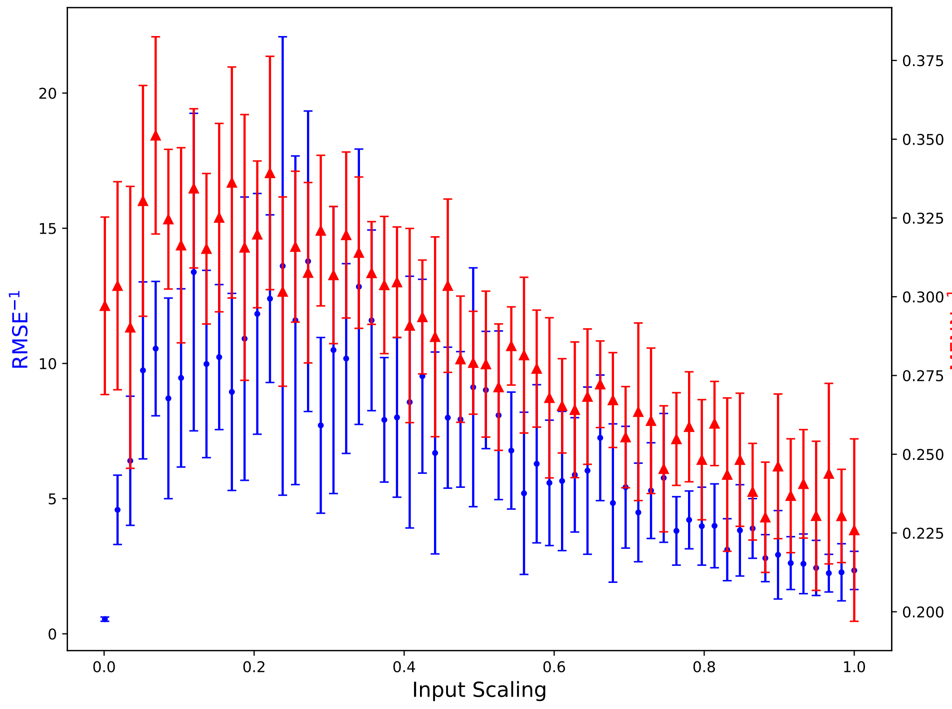


Figure 11.4. Results for the reservoir observer on Lorenz system when varying the input scaling ω. Blue dots account for the reciprocal of the RMSE (left axis) while red triangles accounts for reciprocal of the MFNN (right axis).

To show that GS plays an important role not only in the observer task, we also test our framework in a forecasting scenario (Fig. 11.6). To this end, we use the x-coordinate of the Lorenz system as the input ($u_t := x_t$, but this time the target $\boldsymbol{y}$ was chosen to be $y_t := u(t + 5\tau)$. This means that the RCN is required to correctly approximate the function $\boldsymbol{g}^5(\boldsymbol{s})$, which is highly nonlinear. The RMSE is computed between y_t and the network output $\hat{y}_t$. Notably, the MFNN here is almost identical to the one in Fig. 11.3: both experiments use the same hyperparameters, the same source system and construct $\boldsymbol{u}$ in the same way, so that the only difference (up to the particular realization) is the task they are trained to solve, which affects the readout and not on the dynamics. As in the other cases, we notice that the MFNN and the RMSE display a similar behavior.

These results confirm that the GS can be exploited to assess the quality of

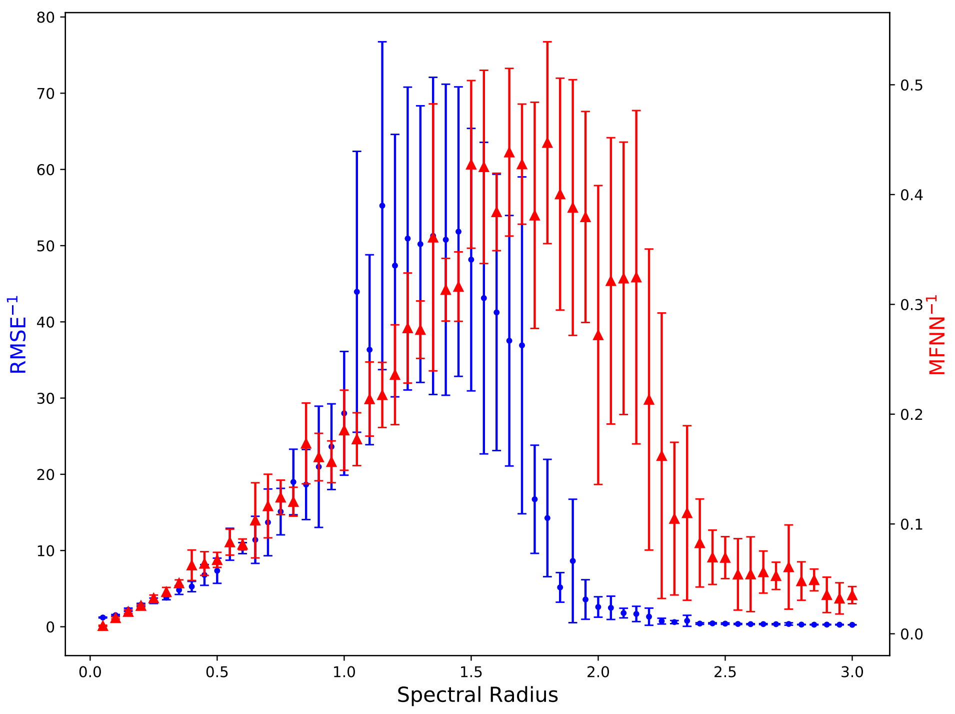


Figure 11.5. Results for the reservoir observer when varying the SR, when using the Rössler system as a source. Blue dots account for the reciprocal of the RMSE (left axis) while red triangles accounts for reciprocal of the MFNN (right axis).

the source system representation encoded in the reservoir states: in order to correctly solve the task at hand, the reservoir and the the source system should be synchronized.

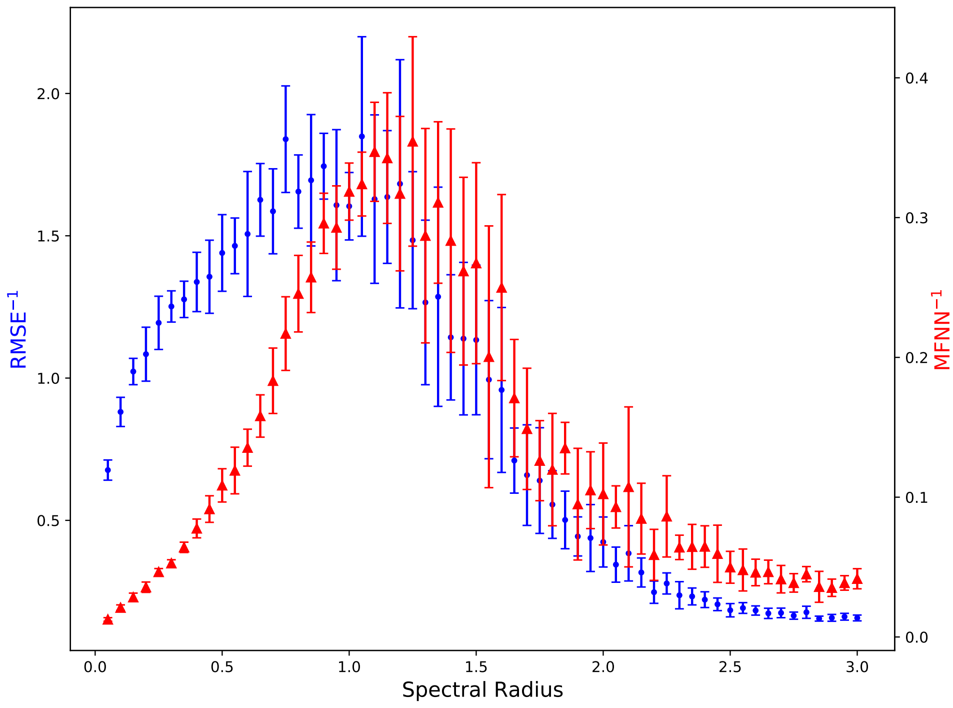


Figure 11.6. Results for the forecasting task on Lorenz system when varying the SR. Blue dots account for the reciprocal of the RMSE (left axis) while red triangles accounts for reciprocal of the MFNN (right axis).

Chapter 12

Conclusions

In this work, we studied the RC framework in the context of dynamical systems. We aimed at analyzing RC systems in order to understand their properties: using our findings to propose novel ways of improving the field, we exploit theoretical results to both enhance the models used in the literature and to propose new ones. Our analysis consists in the fact that the RC are inherently dynamical systems: hence, they can be used to study dynamical systems but, *vice versa*, also dynamical systems can be used to study RC. We divided our analysis into three parts.

Part I deals with how the *learning* occurs in RC. We reviewed the main procedure used for training in RC, which we named Reservoir Learning Algorithm (RLA). We also mentioned some relevant variations to it. This simple procedure for training displays a critical dependence on hyper-parameters, that need to be carefully tuned and cross-validated. We analyzed these hyper-parameters under the light of the state of the art, discussing how they affect the behaviors of such systems. On top of that, we proposed a model for RCNs whose performance is provably stable with regard to the most important hyper-parameter, the Spectral Radius (SR). It is based on the projection of the system states into a hyper-spherical surface, using a novel activation function designed for this scope. We first proved that the proposed model is a universal approximator just like regular RCNs and gave sufficient conditions to support this claim. Our theoretical analysis showed that the behavior of the resulting network is never unstable, regardless of the setting of the main hyper-parameters affecting its dynamics in phase space. This claim was supported both analytically and experimentally by showing that the maximun Lyapunov exponent is always zero, implying that the proposed model always operates on the Edge of Criticality (EoC). This leads to networks which are (globally) stable for any hyper-parameter configuration, regardless of the particular input signal driving the network, hence solving possible stability problems that might affect RCNs.

Part II is concerned with the *representation* of the input signal in the states of the reservoir. First we dealt with the memory of RC systems, showing experiments comparing different models. We also examined the *memory-nonlinearity trade-off*. The self-normalizing model proposed in the previous section displays a competitive performance. We then moved to the analysis of ESP, a fundamental property of RCN addressing their signal-encoding capability. Since we studied signals that are generated by dynamical systems, we discussed how this property can be linked to the celebrated Takens's theorem. Finally, we proposed a method to study RCNs which explicitly addresses their finite capacity. Starting from the Cayley-Hamilton (CH) theorem, we expressed the system state in terms of the *controllability matrix* $\mathscr{C}$ and the *network encoded input* $\boldsymbol{v}$. The matrix properties of $\mathscr{C}$ allowed us to compare different connectivity patterns for the reservoir in a quantitative way by studying its rank. Results show that reservoirs with a cyclic topology give the richest possible encoding of input signals, yet they also offer one of the most parsimonious reservoir parametrization.

In Part III we addressed the study of the *dynamics* of the reservoir states as a computational tool. At first, we briefly described the relationship between computation and dynamics, with a particular emphasis on the topic of EoC. We later introduced the binary RCNs which are in principle similar to regular RCNs, sharing their fundamental features. However, their simplicity allowed a theoretical analysis of some important aspects of the transition to instability. The expression we derived for the autonomous case perfectly matches the experimental results. Our analysis of the noise applied to the neurons activations showed how the network stability increases linearly with the mean degree of the recurrent connections. The effects of input signals on the network dynamics are more complex to understand, since they introduce correlations among neurons. Our analysis partially explained the role that the signal magnitude and the mean degree play in shaping the EoC of the non-autonomous case. At last, we laid down the groundwork for establishing and analyzing working principles of RC within the theoretical framework of synchronization between dynamical systems. We showed that the presence of a synchronization function permits to formally consider the reservoir states as an unsupervised, high-dimensional representation of an unknown source system that generates the observed data. We proved that the realizability of learning, defined as the possibility of perfectly solving the task, crucially depends on the existence of a function $\mathscr{F}$ connecting the reservoir states with the source system states: the presence of Generalized Synchronization (GS) implies the existence of a synchronization function $\boldsymbol{\phi}$ playing an analogous role, which is found in an unsupervised way by exploiting the RC dynamics. This formally demonstrates that it is possible to solve the task at hand by firstly creating an unsupervised representation of the source system (listening phase) and then using a suitable readout to correctly represent the target (fitting phase), thus justifying the RC training principle in a formal way. Moreover, the presence of such a synchronization function allows one to make use of the ergodicity of the source system to grant some results on the generalization error for a given task. Finally, we made use of an index (MFNN) to quantify the degree of synchronization and experimentally validate our claims. Results show that the more the reservoir is synchronized with the source, the better the system approximates the target, hence stressing that synchronization is paramount and plays a fundamental role within the RC framework.

The research in the field still faces numerous challenges. Among them, it is particularly impelling to address the integration of domain knowledge (e.g., phyical) into the RC, particularly in the training procedure. Some steps have been made toward this direction, like the hybrid-modeling presented in [Pathak, Wikner, Fussell, Chandra, Hunt, Girvan and Ott, 2018] or the Physics-informed ESP [Doan et al., 2020], but they rely on information about the source system generating the data which are barely available in real applications. Moreover, as previously discussed, theoretical and practical problems rise in the generating scenario when the system switches from the non-autonomous training phase to the autonomous testing phase. The theory of GS we presented is not suitable for the latter, which instead involves a *bidirectional* synchronization between the reservoir and the output. Therefore, the stability of the synchronization in the training phase might not necessarily imply the stability of the autonomous system during the testing. We are currently investigating this problem, studying the properties of the approximated autonomous system that the RC system learns. Another issue is related to the development of the theory of *Non-Autonomous Dynamical Systems* [Manjunath et al., 2012], which is still in its infancy. The exploitation of some relevant results from this field might lead to a more solid theoretical base for the theory of RC (and RNNs in general). Finally, a more advanced understanding of the role played by nonlinearity is needed in order to progress in the field, both practically and theoretically. Most of the theory, as we showed, relies on the linearization of the system around the fixed point, e.g., [Bollt, 2021; Inubushi and Yoshimura, 2017], basically neglecting the role of the nonlinear contributes. Currently, research is being carried to explore the representation properties of nonlinear reservoirs.

Appendices

Appendix A

Dynamical Systems

In this Appendix, we give a formal introduction to the topic of dynamical systems. Section A.1 presents the theory for Autonomous dynamical systems, both in continuous and discrete time. In our exposition, we mainly follow [Meiss, 2007]. In Section A.2 we briefly present the theory of Non-Autonomous Dynamical Systems, with a particular emphasis on Input-Driven Systems. We only discuss the discrete-time scenario. We follow [Manjunath et al., 2012]. A detailed treatment of the field can be found in [Kloeden and Rasmussen, 2011].

A.1 Autonomous Dynamical Systems

To define a Dynamical System we need two fundamental elements:

- a *set* X;
- a *map* $\boldsymbol{f} : X \to X$.

Usually X is called the *phase space*. Starting from these simple and quite general elements, we can define dynamical system. We note that for any point $\boldsymbol{x}_0 \in X$, called *initial condition*, the map $\boldsymbol{f}$ gives us the new state of the system $\boldsymbol{x}_1 := \boldsymbol{f}(\boldsymbol{x}_0)$. Since, by construction, $\boldsymbol{x}_1 \in X$ we can apply the map again and obtain $\boldsymbol{x}_2 = \boldsymbol{f}(\boldsymbol{x}_1) = \boldsymbol{f}(\boldsymbol{f}(\boldsymbol{x}_0))$, i.e., the state of the system starting from $\boldsymbol{x}_0$ after two application of the process. More generally, we can *iterate* the process any number of times, so we define a compact notation for the k-th iteration, which reads:

$$\boldsymbol{f}^k = \underbrace{\boldsymbol{f} \circ \cdots \circ \boldsymbol{f}}_{k\text{times}} \tag{A.1}$$

where $\circ$ denotes the composition of maps. We also write the state of the system starting from $\boldsymbol{x}_0$ after k iterations as:

$$\boldsymbol{x}_k := \boldsymbol{f}^k(\boldsymbol{x}_0). \tag{A.2}$$

With the notation we developed, we now give some definitions.

Definition 9 (Forward Orbit)**.** The forward orbit of $\boldsymbol{x}_0 \in X$ is the set

$$\mathcal{O} := \{\boldsymbol{x}_n\}_{n\in\mathbb{N}}$$

Definition 10 (Orbit)**.** If $\boldsymbol{f}$ is invertible, the *full* orbit of $x_0 \in X$ is the set

$$\mathcal{O}^+ := \{\boldsymbol{x}_n\}_{n\in\mathbb{N}}$$

Definition 11 (Fixed and Periodic Points)**.** If $\boldsymbol{f}(\boldsymbol{x}_0) = \boldsymbol{x}_0$ we call $\boldsymbol{x}_0$ a *Fixed Point*. If $\boldsymbol{f}^k(\boldsymbol{x}_0) = \boldsymbol{x}_0$ we call $\boldsymbol{x}_0$ a *Periodic Point* of period k. If $\boldsymbol{x}_j$ is periodic for some $j \geq 0$ we call $\boldsymbol{x}_0$ a *Pre-periodic Point*.

Definition 12 (Limit sets)**.** Let X be a topological space and let $\boldsymbol{f} : X \to X$. For $\boldsymbol{x}_0 \in X$ we define the *omega limit set* $\omega(\boldsymbol{x}_0)$ as

$$\omega(\boldsymbol{x}_0) := \{y \in X : x_{n_j} \to y \text{ for some } n_j \to -\infty \text{ as } j \to \infty\}.$$

If $\boldsymbol{f}$ is invertible, for $\boldsymbol{x}_0 \in X$ we define the *alpha limit set* $\alpha(\boldsymbol{x}_0)$ as

$$\alpha(\boldsymbol{x}_0) := \{y \in X : x_{n_j} \to y \text{ for some } n_j \to -\infty \text{ as } j \to \infty\}$$

Definition 13 (Transitive map)**.** We say that $\boldsymbol{f} : X \to X$ is *transitive* is there exists $\boldsymbol{x} \in X$ with $\omega(x) = X$.

Definition 14 (Attracting Fixed Point)**.** The point $p \in X$ is an *attracting fixed point* if there exists a neighbourhood U of p such that $\omega(x) = \{p\}$ for all $x \in U$.

Definition 15 (Repelling Fixed Point)**.** Assume $\boldsymbol{f}$ is invertible. The point $p \in X$ is a *repelling fixed point* if there exists a neighbourhood U of p such that $\alpha(x) = \{p\}$ for all $x \in U$.

A.1.1 Discrete Time Dynamical Systems

We are now ready to give the definition of a discrete time dynamical system,

Definition 16 (Dynamical System)**.** Given an arbitrary set X and an arbitrary map $\boldsymbol{f} : X \to X$, we call *Discrete Time Dynamical System* the family $\{\boldsymbol{f}^t\}_{t\in\mathbb{N}}$.

Let $Id : X \to X$ be the identity function. It is easy to see that $\{f^t\}_{t\in\mathbb{N}}$ satisfies the *semi-group* properties under composition, namely:

Closure $f^s \circ f^t = f^{s+t} \forall s, t \in \mathbb{N}$;

Identity $f^0 = Id$;

Associativity $f^r \circ (f^s \circ f^t) = (f^r \circ f^s) \circ f^t \forall r, s, t \in \mathbb{N}$.

Definition 17 (Invertible Discrete Time Dynamical System)**.** Given an arbitrary set X and an *invertible* map $f : X \to X$, we call *Discrete Time Invertible Dynamical System* the family $\{f^t\}_{t\in\mathbb{Z}}$.

When f is invertible, the family $\{f^t\}_{t\in\mathbb{Z}}$ is a *group*, i.e., it satisfies:

Closure $f^s \circ f^t = f^{s+t} \quad \forall s, t \in \mathbb{Z}$;

Identity $f^0 = Id$;

Associativity $f^r \circ (f^s \circ f^t) = (f^r \circ f^s) \circ f^t \quad \forall r, s, t \in \mathbb{Z}$;

Inverse For every f^n there exists f^{-n} such that $f^n \circ f^{-n} = f^{-n} \circ f^n = f^0$.

A.1.2 Continuous Time Dynamical Systems

Definition 18 (Semi-Flow)**.** Given a metric space X and an arbitrary map $f : X \to X$, we call *(Non-invertible) Continuous Time Dynamical System* or *Semi-Flow* the family $\{f^t\}_{t\in\mathbb{R}_+}$.

It is easy to see that $\{f^t\}_{t\in\mathbb{R}_+}$ satisfies the *semi-group* properties under composition, namely:

Closure $f^s \circ f^t = f^{s+t} \quad \forall s, t \in \mathbb{R}_+$;

Identity $f^0 = Id$

Associativity $f^r \circ (f^s \circ f^t) = (f^r \circ f^s) \circ f^t \forall r, s, t \in \mathbb{R}_+$.

Definition 19 (Flow)**.** Given an arbitrary set X and an *invertible* map $f : X \to X$, we call *(Invertible) Continuous Time Dynamical System* or *Flow* the family $\{f^t\}_{t\in\mathbb{R}}$.

When f is invertible, the family $\{f^t\}_{t\in\mathbb{R}}$ is a *group*, i.e., it satisfies:

Closure $f^s \circ f^t = f^{s+t} \forall s, t \in \mathbb{R}$;

Identity $\boldsymbol{f}^0 = Id$;

Associativity $\boldsymbol{f}^r \circ (\boldsymbol{f}^s \circ \boldsymbol{f}^t) = (\boldsymbol{f}^r \circ \boldsymbol{f}^s) \circ \boldsymbol{f}^t \quad \forall r, s, t \in \mathbb{Z}$;

Inverse For every $\boldsymbol{f}^n$ there exists $\boldsymbol{f}^{-n}$ such that $\boldsymbol{f}^n \circ \boldsymbol{f}^{-n} = \boldsymbol{f}^{-n} \circ \boldsymbol{f}^n = \boldsymbol{f}^0$.

All the definitions given above can be easily generalized to the continuous case. For example we can define the forward orbit as

$$\mathcal{O}^+ := \{\boldsymbol{x}_n\}_{n\in\mathbb{R}^+},$$

and, for a flow, the (full) orbit as

$$\mathcal{O} := \{\boldsymbol{x}_n\}_{n\in\mathbb{R}}$$

the same can be done for the $\omega-$ and $\alpha-$sets.

Discrete- and continuous-time systems can be related. We now list some ways in which this relation can be constructed.

Time-one Maps (Sampling) Given $\{\boldsymbol{f}^t\}_{t\in\mathbb{R}}$, one can construct the map $\boldsymbol{g} = \boldsymbol{f}^1$ which associates to each point $\boldsymbol{x}_0$ its position after one unit of time under the flow $\boldsymbol{g}(\boldsymbol{x}_0) = \boldsymbol{x}_1$. This is called the *time-one map* associated with the flow. It is easy to see that $\{\boldsymbol{g}^n\}_{n\in\mathbb{N}}$ constitutes a discrete-time dynamical system. In an exactly analogous way, is it possible to define *time-τ map* for any τ. The discrete trajectory obtained one obtains in this way can be seen as a *sampling* from the original flow, with a sample frequency of $\frac{1}{\tau}$. Notice the the time-τ map can be defined for any flow or semi-flow, but it is not true that any discrete-time system is a sampling of continuous-time one.

Poincaré Section Another possible way to construct a discrete-time system out of a continuous-time one is via the *Poincaré section*. Suppose that there exists a subset $\Sigma \subset X$ such that for all $x \in \Sigma$ the *first return time* function can be defined

$$\tau(\boldsymbol{x}) := \min\{t > 0 : \boldsymbol{f}^{(t)}(x) \in \Sigma\} \qquad \text{(A.3)}$$

such that $\boldsymbol{f}^t(\boldsymbol{x}) \notin \Sigma$ for all $t \in (0, \tau(\boldsymbol{x})$ and $\boldsymbol{f}^{\tau(x)} \in \Sigma$. We then cal Σ the *Poincaré section* for the flow which, in turn, defines the *Poincaré first return time map* $\boldsymbol{F} : \Sigma \to \Sigma$, given by:

$$\boldsymbol{F}(\boldsymbol{x}) := \boldsymbol{f}^{\tau(x)} \qquad \text{(A.4)}$$

One can recover properties of the flow by studying the discrete time dynamical systems generated by the map $\boldsymbol{F}$. For instance, note that any periodic point defines a periodic orbit for the flow. But, while every flow and semi-flow has a corresponding time-1 map, it is not the case that every flow or semi-flow has a Poincaré map.

Suspension Flow It is also possible to go on the other direction, and build a flow out of a discrete time dynamical systems. This is done by defining the *extended phase space*

$$\tilde{X} := X \times [0,1]/\sim \tag{A.5}$$

where the relation $\sim$ identifies the point $(\boldsymbol{x}, 1)$ with the point $(\boldsymbol{f}(\boldsymbol{x}), 0)$. We can then define a flow on X by a simple vertical translation with constant speed 1. Notice that the original map $\boldsymbol{f}$ is exactly the time-one map of this flow which in this special case also coincides with the first return Poincaré map on the cross section X.

A.1.3 Ordinary Differential Equations

Let $\{\boldsymbol{f}^t\}_{t\in\mathbb{R}}$ be a flow on $\mathbb{R}^n$ and suppose that the family $\boldsymbol{f}^t$ depends continuously on the point x and differentiably on the parameter t. In particular, this differentiability means that the orbit of each point is a differentiable curve and that for each t the derivative $\dot{\boldsymbol{x}}(t) := \frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t}$ is a well defined vector tangent to the curve of the orbit in $\boldsymbol{x}(t)$. Let $U \subseteq \mathbb{R}^n$ be an open set.

Definition 20 (Vector Field). A vector field on U is a function $\boldsymbol{V} : U \to \mathbb{R}^n$.

We briefly discussed above how flows defines vector fields. The opposite also holds, provided that the vector field is Lipschitz.

Definition 21 (Integral Curve). Let $I \subseteq \mathbb{R}$ be an open set. A function $\boldsymbol{x} : I \to U$ is a *integral curve* of a vector field $\boldsymbol{V}$ if

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{V}(\boldsymbol{x}(t)) \tag{A.6}$$

for every $t \in I$. A function which satisfies (A.6) is called a *local solution* of the differential equation. If moreover $I = \mathbb{R}$, it is called a *global solution*.

Let $\boldsymbol{V} : U \to \mathbb{R}^n$ be a vector field and a point $\boldsymbol{x}_0 \in U$. We write an *initial value problem* as:

$$\begin{cases} \dot{\boldsymbol{x}} = \boldsymbol{V}(\boldsymbol{x}) \\ \boldsymbol{x}(0) = \boldsymbol{x}_0 \end{cases} \tag{A.7}$$

The following theorem deals with the existence and uniqueness of the solutions of the system (A.7). To simplify its statement, we consider $U = \mathbb{R}^n$.

Theorem 7 (Fundamental Theorem of Ordinary Differential Equations). *Let $\boldsymbol{V} : \mathbb{R}^n \to \mathbb{R}^n$ be a continuous vector field. For every $\boldsymbol{x}_0 \in \mathbb{R}^n$:*

1. *There exists a local solution $\boldsymbol{x} : I \to \mathbb{R}$ for* (A.7)*;*
2. *if $\boldsymbol{V}$ is locally Lipschitz, this solution is unique;*
3. *if $\boldsymbol{V}$ is Lipschitz, the solution is global and it depends continuously on the initial condition.*

A.1.4 Attractors

Informally, an attractor can be defined a set towards all nearby trajectory move. To define it more formally, we generalize the concept of stability to a set.

Definition 22 (Trapping Region). A set N is a trapping region for $\boldsymbol{f}_t$ if it is compact and $\boldsymbol{f}_t(N) \subset \text{int}(N)$ for each t.

Here, $\text{int}(N)$ denotes the interior of N.

Definition 23 (Attracting Set). A set S is attracting if there is a compact trapping region N such that $S \subset N$ and

$$S = \bigcap_t f_t(N)$$

Any attracting set has a maximal trapping region, which is usually called the *Basin of Attraction*:

Definition 24 (Basin of Attraction). The basin of attraction $B(S)$ of an invariant set S is the set of all points $\boldsymbol{x}$ for which $d(\boldsymbol{f}^t(\boldsymbol{x}, S)) \to 0$ as $t \to \infty$.

Definition 25 (Attractor). A set $\mathscr{A}$ is an attractor if it is an attracting set and there is some point $\boldsymbol{x}$ such that $\mathscr{A} = \omega(\boldsymbol{x})$.

Note that not every ω-limit set is an attractor.

A.2 Non-Autonomous Dynamical Systems

Definition 26 (Non-Autonomous System). A (discrete-time) *Non-Autonomous System* is a family of maps $\{\boldsymbol{f}_n\}$ where each $\boldsymbol{f}_n : X \to X$ is a continuous map and the state of the system $\boldsymbol{x}_n$ at time n satisfies

$$\boldsymbol{x}_{n+1} = \boldsymbol{f}_n(\boldsymbol{x}_n) \tag{A.8}$$

Definition 27 (Input Driven System). An *Input Driven System (IDS)* comprises a continuous map $\boldsymbol{f} : X \times U \to X$ and a sequence $\{\boldsymbol{u}_n\}$ such that

$$\boldsymbol{x}_{n+1} = \boldsymbol{f}(\boldsymbol{x}_n, \boldsymbol{u}_n) \tag{A.9}$$

We would like to generalize the concept of attractor for IDSs. To do so, we first introduce the concept of *process*, which will ease the notation lately.

Definition 28 (Process). Let $\mathbb{Z}^2_{\geq} := \{(n,m) : n,m \in \mathbb{Z}, n \leq m\}$. A *Process* Φ on a state space X is a mapping $\Phi : \mathbb{Z}^2_{\geq} \times X \to X$ which satisfies the *evolution properties*:

1. $\Phi(m,m,x) = x$ for all $m \in \mathbb{Z}, x \in X$;

2. $\Phi(n,m,x) = \Phi(n,k,\Phi(k,m,x))$ for all $m,k,n \in \mathbb{Z}$ with $m \leq k \leq n$, $x \in X$;

3. for given n,m, the map $\Phi(n,m,\cdot)$ is continuous.

Processes and Non-Autonomous Systems have a one-to-one correspondence. It is straightforward to generate a process using a Non-Autonomous System $\{\boldsymbol{f}_n\}$. One simply sets $\Phi(m,m,\boldsymbol{x}) := \boldsymbol{x}$ and $\Phi(n,m,\boldsymbol{x}) := \boldsymbol{f}_{n-1} \circ \cdots \circ \boldsymbol{f}_m(\boldsymbol{x})$, which clearly possess the required properties. Instead, starting from a process Φ on X, a Non-Autonomous System $\{\boldsymbol{f}_n(\boldsymbol{x})\}$ can be defined as $\boldsymbol{f}_n(\boldsymbol{x}) := \Phi(n+1,n,\boldsymbol{x})$.

To define attractors, we need to define invariant sets and orbits and solutions converging to them.

Definition 29 (Entire Solution). Let Φ be a process on X. An *Entire Solution* of Φ is sequence $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_n\}_{n \in \mathbb{Z}}$ such that $\theta_m \in X$ for all m and $\Phi(n,m,\boldsymbol{\theta}_m) = \boldsymbol{\theta}_n$ for all $m \leq n$.

Definition 30 (Φ-invariant Set). A Non-Autonomous set $\mathscr{A} = \{A_n : A_n \subset X\}$ is said to be Φ-invariant if $A_m \subset X$ for all m and $\Phi(n,m,A_m) = A_n$ for all $n > m$. $\mathscr{A}$ is said to be *Φ-positively-invariant* or *Φ-+invariant* if $\Phi(n,m,A_m) \subset A_n$.

Definition 31 (Natural Association). Let Φ be a process on compact space X. We call *Natural Association* of Φ on X the sequence $\{X_n\}$ defined by

$$X_n := \bigcap_{m<n} \Phi(n,m,X) \tag{A.10}$$

Proposition 8. *Given a process Φ on a compact space X, let $\{X_n\}$ be its Natural Association. Then, it holds that:*

- *For each $n > m$, $\Phi(n,m,X_m) = X_n$;*
- *A Non-Autonomous Set $\mathscr{A} = \{A_k\}$ is Φ-invariant* if and only if *for each pair $k \in \mathbb{Z}$, $x \in A_k$ there exists an entire solution $\{\theta_n\}$ such that $\theta_k = x$ and $\theta_k \in A_k$ for each $k \in \mathbb{Z}$;*
- *A Φ-invariant set $\{Y_n\}$ is the natural association of Φ (i.e., $Y_n = X_n$ for each n)* if and only if *every entire solution $\{\theta_n\}$ is such that $\theta_k in Y_k$ for each $k \in \mathbb{Z}$.*

Definition 32 (Hausdorff semi-distance)**.** Let $A, B \subset X$ be non-empty sets. We define the *Hausdorff semi-distance* from A to B as:

$$d_H(A,B) := \sup d(x,B) : x \in A \tag{A.11}$$

where $d(x,B)$ is the distance between a point and a set.

It is called semi-distance as , in general, $d_H(A,B) \neq d_H(B,A)$. Note that if $A \subset B$ then $d_h(A,B) = 0$. The notion of approaching of two set sequences $\{A_k\}$ and $\{B_k\}$ in some space X can be captured by stating that $\lim_{n\to\infty} d_H(A_n, B_n) = 0$.

Let $A \subset X_k$ where $\{X_k\}$ is the natural association of a process Φ on X.

$$\mathscr{B}_\eta^{(i)}(A) := \{x \in X_i : d(x,A) < \eta\} \tag{A.12}$$

We can now define two distinct notion of attractor, namely the *Pull-Back Attractor* and the *Forward Attractor*.

Definition 33 (Non-Autonomous Attractors)**.** Let Φ be a process on X, with a natural association $\{X_n\}$ and let $\mathscr{A} = \{A_n\}$ be a Φ+-invariant set such that each A_n is compact and is a subset of X_n. Then

- We say that $\mathscr{A}$ is a *Local +Invariant Pull-Back Attractor* if for some $\eta > 0$:

$$\lim_{j\to\infty} d_H\Big(\Phi\big(n, n-j, \mathscr{B}_\eta^{(n-j)}(A_{n-j})\big), A_n\Big) \quad \forall n; \tag{A.13}$$

- We say that $\mathscr{A}$ is a *Local +Invariant Forward Attractor* if for some $\eta > 0$:

$$\lim_{j\to\infty} d_H\Big(\Phi\big(n+j, n, \mathscr{B}_\eta^{(n)}(A_n)\big), A_{n+j}\Big) \quad \forall n. \tag{A.14}$$

If $\mathscr{A}$ is Φ-invariant, then we say that they are *Local Invariant Attractors* of the respective kind.

In general, the two notions do not coincide and some sets might be attractors according to a definition but not according to the other one. We refer to [Manjunath et al., 2012] for further details.

Appendix B

Memory and Activation

B.1 Memory

Here, we theoretically analyze the ability of RCNs to retain memory of past inputs in the state. In order to perform such analysis, we define a measure of network's memory that quantifies the impact of past inputs on current state $\boldsymbol{r}_t$. This proposal shares similarities with a memory measure called Fisher memory, first proposed by Ganguli et al. [2008] and then further developed in [Tiňo and Rodan, 2013; Tiňo, 2018]. However, our measure can be easily applied also to non-linear networks like the one Eq. 5.1 proposed here, justifying the analysis discussed in the following.

Considering one time-step in the past for the Self-Normalizing activation function (5.1), we have:

$$\boldsymbol{r}_t = \frac{W}{N_t}\boldsymbol{r}_{t-1} + \frac{1}{N_t}\boldsymbol{x}_t \tag{B.1}$$

All past history of the signal is processed in $\boldsymbol{r}_{t-1}$. Note that (B.1) keeps the same form for every t. Going backward in time one more step, we obtain:

$$\boldsymbol{r}_t = \frac{W}{N_t}\left(\frac{W}{N_{t-1}}\boldsymbol{r}_{t-2} + \frac{1}{N_{t-1}}\boldsymbol{x}_{t-1}\right) + \frac{1}{N_t}\boldsymbol{x}_t \tag{B.2}$$

As $\|\boldsymbol{r}_t\| = \|\boldsymbol{r}_{t-1}\| = \|\boldsymbol{r}_{t-2}\| = 1$, we see that there is a sort of recursive structure in this procedure, where each incoming input is summed to the current state and then their sum is normalized. This is a key feature of the proposed architecture. In fact, it guarantees that each input will have an impact on the state, since the norm of the activations will not be too big or too small, because of the normalization. We now express this idea in a more formal way.

The norm of (5.24) can be written as

$$\|M^{(t,k)}\| = \frac{1}{\prod_{l=k}^{t} N_l}\|W^{t-k}\| \overset{t-k \gg 1}{\approx} \frac{1}{\prod_{l=k}^{t} N_l}\rho^{t-k} \tag{B.3}$$

where the approximation holds thanks to the Gelfand's formula[1]. If the input is null, we expect each N_l to be of the order of ρ as $\|x_l\| = 1$. So we write:

$$N_l = \|W\boldsymbol{r}_{l-1} + \boldsymbol{x}_l\| = \rho + \delta_l \tag{B.4}$$

where δ_l denotes the impact of the l-th input on the l-th normalization factor. Its presence is due to the fact that without any input N_l would be approximately equal to ρ, while the input modifies the state and so the normalization value will be modified accordingly. The value δ_l is introduced to account for this fact: $\delta_l := N_l - \rho$.

If we assume that each input produces a similar effect on the state (i.e. $\delta_l = \delta$ for every l), we finally obtain:

$$\|M^{(t,k)}\| \approx \frac{\rho^{t-k}}{(\rho+\delta)^{t-k}} \tag{B.5}$$

We note that such assumption is reasonable for inputs that are symmetrically distributed around a mean value with relatively small variance, e.g. for Gaussian or uniformly distributed inputs (as demonstrated by our results). However, our argument might not hold for all types of inputs and a more detailed analysis is left as future research.

We define the memory of the network $\mathscr{M}_\alpha(k|t)$ as the influence of input at time k on the network state at time t. More formally, having defined $\alpha := \rho/\delta$ (since (B.5) only depends on this ratio), we use (B.5) to define the memory as:

$$\mathscr{M}_\alpha(k|t) := \left(\frac{\alpha}{\alpha+1}\right)^{t-k} = \left(1 - \frac{1}{\alpha+1}\right)^{t-k} \tag{B.6}$$

Eq. B.6 goes to 0 (i.e., no impact of the input on the states) as $\alpha \to 0$ and tends to 1 for $\alpha \to \infty$, implying that for an infinitely large SR the network perfectly remembers its past inputs, regardless of their occurrence in the past. Note that (B.6) does not depend on k and t individually, but only on their difference (elapsed time): the larger the difference, the lower the value of $\mathscr{M}_\alpha(k|t)$, meaning that far away inputs have less impact on the current state.

[1]The Gelfand's formula Lax [2002] states that for any matrix norm $\rho(A) = \lim_{k\to\infty} \left\|A^k\right\|^{\frac{1}{k}}$.

B.2 Memory loss

By using (B.6), we define the memory loss between state $\boldsymbol{r}_t$ and $\boldsymbol{r}_m$ of the signal at time-step k (with $m > t > k$ and $m = t + \tau$) as follows:

$$\begin{aligned}\Delta\mathscr{M}(k|m,t) :=& \mathscr{M}_\alpha(k|m) - \mathscr{M}_\alpha(k|t) = \left(\frac{\alpha}{\alpha+1}\right)^{m-k} - \left(\frac{\alpha}{\alpha+1}\right)^{t-k} \\ =& \frac{(\alpha+1)^k}{\alpha^k}\frac{\alpha^t}{(\alpha+1)^t}\left(\frac{\alpha^\tau}{(\alpha+1)^\tau} - 1\right) \\ =& \left(1 - \frac{1}{\alpha+1}\right)^t\left(1 + \frac{1}{\alpha}\right)^k\left(\left(1 - \frac{1}{\alpha+1}\right)^\tau - 1\right) \leq 0\end{aligned} \tag{B.7}$$

For very large values of α, we have $\Delta\mathscr{M}(k|m,t) \to 0$, meaning that in our model (5.1b) larger spectral radii eliminate memory losses of past inputs. Now, we want to assess if inputs in the far past have higher/lower impact on the state more than recent inputs. To this end, we selected $t > k > l$ and defined $k = t - a$ and $l = t - b$, leading to $b > a > 0$ and $b = a + \delta$. Define:

$$\begin{aligned}\Delta\mathscr{M}(k,l|t) =& \mathscr{M}(t-a|t) - \mathscr{M}(t-a-\delta|t) = \frac{\alpha^a}{(\alpha+1)^a} - \frac{\alpha^{a+\delta}}{(\alpha+1)^{a+\delta}} = \\ =& \frac{\alpha^a}{(\alpha+1)^a}\left(1 - \frac{\alpha^\delta}{(\alpha+1)^\delta}\right) = \left(1 - \frac{1}{\alpha+1}\right)^a\left(1 - \left(1 - \frac{1}{\alpha+1}\right)^\delta\right) \geq 0\end{aligned} \tag{B.8}$$

We have that $\lim_{\delta\to\infty}\Delta\mathscr{M}(t-a, t-a-\delta|t) = \left(1 - \frac{1}{\alpha+1}\right)^a$, showing how the impact of an input that is infinitely far in the past is not null compared with one that is only $a < \infty$ steps behind the current state. This implies that the proposed network is able to preserve in the network state (partial) information from all inputs observed so far.

Linear networks

In order to assess the memory of linear models, we perform the same analysis for linear RNNs (i.e., $\boldsymbol{r}_t = \boldsymbol{a}_t$) by using the definitions given in the previous section. In this case, an expression for the memory can be obtained straightforwardly and reads:

$$\mathscr{M}_L(k|t) := \rho^{t-k} \tag{B.9}$$

It is worth noting that there is no dependency on δ and, therefore, on the input. Just like before, we have the memory loss of the signal at time-step k between state $\boldsymbol{r}_t$ and $\boldsymbol{r}_m$, as:

$$\begin{aligned}\Delta\mathscr{M}_L(k|m,t) &= \mathscr{M}_L(k|m) - \mathscr{M}_L(k|t) \qquad &\text{(B.10)}\\ &= \mathscr{M}_L(k|t+\tau) - \mathscr{M}_L(k|t) = (\rho^\tau - 1)\rho^{t-k} \leq 0 \qquad &\text{(B.11)}\end{aligned}$$

where we set $m > t > k$ and $m = t + \tau$. As before, we also discuss the case of two different inputs observed before time-step t. By selecting time instances $t > k > l$, $k = t - a$ and $l = t - b$, we have $b > a > 0$ allowing us to write $b = a + \delta$. As such:

$$\Delta\mathscr{M}_L(k,l|t) = \mathscr{M}_L(k|t) - \mathscr{M}_L(l|t) = (1-\rho^\delta)\rho^a \geq 0. \qquad \text{(B.12)}$$

We see that in both (B.10) and (B.12) the memory loss tends to zero as the spectral radius tends to one, which is the critical point for linear systems. So, according to our analysis, linear networks should maximize their ability to remember past inputs when their SR in close to one and, moreover, their memory should be the same disregarding the particular signal they are dealing with. We will see in the next section that both these claims are confirmed by our simulations.

Hyperbolic tangent

We now extend the analysis to standard ESNs, i.e., those using a tanh activation function. Define $\sigma :=$ tanh (applied element-wise). Then, for generic scalars a and b, when $|b| \ll 1$ the following approximation holds:

$$\sigma(\boldsymbol{a}+\boldsymbol{b}) = \frac{\sigma(\boldsymbol{a})+\sigma(\boldsymbol{b})}{1+\sigma(\boldsymbol{a})\sigma(\boldsymbol{b})} \approx \sigma(\boldsymbol{a}) + \sigma(\boldsymbol{b}) \qquad \text{(B.13)}$$

When $\boldsymbol{a}$ is the state and $\boldsymbol{b}$ is the input, our approximation can be understood as a small-input approximation [Verstraeten and Schrauwen, 2009]. Then, the state-update reads:

$$\begin{aligned}\boldsymbol{r}_t &= \sigma(W\boldsymbol{r}_{t-1} + \boldsymbol{x}_t) \approx \sigma(W\boldsymbol{r}_{t-1}) + \sigma(\boldsymbol{x}_t)\\ &= \sigma(W\sigma(W\boldsymbol{r}_{t-2} + u_{t-1})) + \sigma(\boldsymbol{x}_t) \qquad \text{(B.14)}\\ &\approx \sigma(W\sigma(W\boldsymbol{r}_{t-2})) + \sigma(W\sigma(\boldsymbol{x}_{t-1})) + \sigma(\boldsymbol{x}_t)\end{aligned}$$

Thus, applying the same argument used in the previous cases, it is possible to write:

$$\|\boldsymbol{r}_t\| = S_\rho^{(0)}(\sigma(\|\boldsymbol{x}_t)\|) + S_\rho^{(1)}(\sigma(\|\boldsymbol{x}_{t-1}\|)) + \cdots + S_\rho^{(t)}(\sigma(\|\boldsymbol{x}_0\|)) = \sum_{k=0}^{n} S_\rho^{(t-k)}(\sigma(\|\boldsymbol{x}_k\|)) \tag{B.15}$$

where we defined:

$$S_\rho^{(t)}(\xi) := \underbrace{\sigma(\rho \cdot \ldots \sigma(\rho \cdot \sigma(}_{t \text{ times}}\xi))) \tag{B.16}$$

We see that, differently from the previous cases, the final state $\boldsymbol{r}_t$ is a sum of nonlinear functions of the signal $\boldsymbol{x}_t$. Each signal element $\boldsymbol{x}_k$ is encoded in the network state by first multiplying it by ρ and then passing the outcome through the nonlinearity $\sigma(\cdot)$. This implies that, for networks equipped with hyperbolic tangent function, larger spectral radii favour the forgetting of inputs (as we are in the non-linear region of $\sigma(\cdot)$). On the other hand, for small spectral radii the network operates in the linear regime of the hyperbolic tangent and the network behaves like in the linear case.

Appendix C

Systems and Datasets

In this appendix, we briefly present the main systems that we used to conduct our experiments.

C.1 Lorenz System

The Lorenz system [Lorenz, 1963] is a 3-dimensional dynamical system characterizing a simple model for atmospheric convection. Its equations read:

$$\begin{aligned} \dot{x} &= \sigma(y - x) \\ \dot{y} &= (\rho - z)x - y \\ \dot{z} &= xy - \beta z \end{aligned} \tag{C.1}$$

where $x = x(t)$, $y = y(t)$, $z = z(t)$ are the variables, σ, ρ and β are the model parameters and the dot denotes the first-order derivative with respect to time t. In this work, we choose the commonly used values $\sigma = 10$, $\rho = 28$ and $\beta = 8/3$, for which the system is known to be a chaotic one and to have a strange attractor.

C.2 Rössler System

The Rössler system [Rössler, 1976] is a 3-dimensional chaotic dynamical system defined as follows:

$$\begin{aligned} \dot{x} &= -y - z \\ \dot{y} &= x + ay \\ \dot{z} &= b + z(x - c) \end{aligned} \tag{C.2}$$

where $x = x(t)$, $y = y(t)$, $z = z(t)$ are the variables and a, b, and c are the model parameters, which in our paper are set to $a = 0.1$, $b = 0.1$, and $c = 14$. The dot denotes the first-order derivative with respect to time t.

C.3 Mackey-Glass System

The Mackey-Glass system [Mackey and Glass, 1977] is given by the following delayed differential equation:

$$\dot{x}(t) = -\beta x(t) + \frac{\alpha x(t-\lambda)}{1 + x(t-\lambda)} \tag{C.3}$$

In our experiments, we simulated the system using standard parameters, that is, $\lambda = 17$, $\alpha = 0.2$ and $\beta = 0.1$.

C.4 Santa Fe Laser

The Santa Fe Laser [Weigend and Gershenfeld, 1993] is an experimental time-series obtained from an physical experiments measuring the intensity of a NH_3 laser. It is a standard benchmark for temporal tasks.

C.5 Multiple Superimposed Oscillators

In a standard benchmark task, the network is fed with Multiple Superimposed Oscillator (MSO) with incommensurable frequencies. In our experiments, we use 3 oscillators so that the input is generated according to:

$$u_t = \sin(0.2t) + \sin(0.311t) + \sin(0.42t) \tag{C.4}$$

using a integer t. This task is know to be difficult because of the multiple time scales characterizing the inputs [Jaeger and Haas, 2004].

Appendix D

Cayley-Hamilton Theorem

In this Appendix, we expose some facts regarding the Cayley-Hamilton (CH) Theorem and its implications for the theory developed in Chapter 8. In Section D.1 we briefly introduce and comment the theorem, while Section D.2 contains some calculations based on it, that lead to some results contained Chapter 8. Section D.3 briefly describes the *Frobenius companion matrix*.

D.1 The Theorem

The Cayley-Hamilton (CH) Theorem allows one to describe the n-th power of a matrix in term of the first $n-1$-powers (including the zero power, which is the identity).

Let $\boldsymbol{W} \in \mathbb{R}^{n \times n}$ be a square matrix. Its characteristic polynomial is defined as:

$$\det(\lambda \boldsymbol{I} - \boldsymbol{W}) = 0 \quad \rightarrow \quad \lambda^n + \alpha_{n-1}\lambda^{n-1} + \cdots + \alpha_1 \lambda + \alpha_0 = 0 \tag{D.1}$$

where λ is an eigenvalue of $\boldsymbol{W}$ and the α_k are the coefficients of the characteristic polynomial.

Theorem 9 (Cayley-Hamilton)**.** *Every real square matrix satisfies its characteristic equation, i.e.,*

$$\boldsymbol{W}^n + \alpha_{n-1}\boldsymbol{W}^{n-1} + \cdots + \alpha_1 \boldsymbol{W}^1 + \alpha_0 \boldsymbol{I} = \boldsymbol{0}. \tag{D.2}$$

where n is the matrix dimension and $\boldsymbol{I}$ *is the identity matrix.*

For more details about this fundamental result see for example [Bernstein, 2009].

D.2 Implications

Rearranging the terms in (D.2) it is possible to show that the n-th power of the matrix can be represented as a *linear combination* of its lower powers:

$$\boldsymbol{W}^n = -\alpha_{n-1}\boldsymbol{W}^{n-1} - \cdots - \alpha_1 \boldsymbol{W}^1 - \alpha_0 \boldsymbol{I} \tag{D.3}$$

To ease the notation, we define $\varphi_k := -\alpha_k$ so that (D.3) reads:

$$\boldsymbol{W}^n = \varphi_{n-1}\boldsymbol{W}^{n-1} + \varphi_{n-2}\boldsymbol{W}^{n-2} + \cdots + \varphi_1 \boldsymbol{W} + \varphi_0 \boldsymbol{I} \tag{D.4}$$

It holds true that

$$\boldsymbol{W}^m = \phi_{n-1}^{(m)}\boldsymbol{W}^{n-1} + \phi_{n-2}^{(m)}\boldsymbol{W}^{n-2} + \cdots + \phi_1^{(m)} \boldsymbol{W} + \phi_0^{(m)} \boldsymbol{I} \tag{D.5}$$

implying that any power $m \geq n$ of $\boldsymbol{W}$ can be specified by $\boldsymbol{W}$ and scalars $(\phi_{n-1}^{(m)}, \dots, \phi_0^{(m)})$. The apexes denote the fact that the n coefficients are those proper of the m-th power for the ϕ_j^m coefficients. Note that, for $m < n$, we have $(\phi_{n-1}^{(i)}, \dots, \phi_0^{(i)}) = (0, \dots, 0, 1, 0, \dots, 0)$, where the only non-zero term is the m-th one, i.e.,

$$\phi_j^{(m)} = \delta_{mj} \quad \text{for} \quad m < n \tag{D.6}$$

Moreover, note that for $m = n$, $(\phi_{n-1}^{(m)}, \dots, \phi_0^{(m)}) = (\varphi_{n-1}, \dots, \varphi_0)$.

For each $m \geq n$, we can derive the scalars in recursive way by noting that:

$$\begin{aligned}
&\boldsymbol{W}^{m+1} = \boldsymbol{W}^m \boldsymbol{W} && \text{(D.7)}\\
&= (\phi_{n-1}^{(m)}\boldsymbol{W}^{n-1} + \phi_{n-2}^{(m)}\boldsymbol{W}^{n-2} + \cdots + \phi_1^{(m)} \boldsymbol{W} + \phi_0^{(m)} \boldsymbol{I})\boldsymbol{W} && \text{(D.8)}\\
&= \phi_{n-1}^{(m)}\boldsymbol{W}^{n} + \phi_{n-2}^{(m)}\boldsymbol{W}^{n-1} + \cdots + \phi_1^{(m)} \boldsymbol{W}^2 + \phi_0^{(m)} \boldsymbol{W} && \text{(D.9)}\\
&= \phi_{n-1}^{(m)}\left(\varphi_{n-1}\boldsymbol{W}^{n-1} + \varphi_{n-2}\boldsymbol{W}^{n-2} + \cdots + \varphi_1 \boldsymbol{W} + \varphi_0 \boldsymbol{I}\right) \\
&+ \phi_{n-2}^{(m)}\boldsymbol{W}^{n-1} + \cdots + \phi_1^{(m)} \boldsymbol{W}^2 + \phi_0^{(m)} \boldsymbol{W} && \text{(D.10)}\\
&= \underbrace{(\varphi_{n-1}\phi_{n-1}^{(m)} + \phi_{n-2}^{(m)})}_{\phi_{n-1}^{(m+1)}}\boldsymbol{W}^{n-1} + \underbrace{(\varphi_{n-2}\phi_{n-1}^{(m)} + \phi_{n-3}^{(m)})}_{\phi_{n-2}^{(m+1)}}\boldsymbol{W}^{n-2} \\
&+ \cdots + \underbrace{(\varphi_1\phi_{n-1}^{(m)} + \phi_0^{(m)})}_{\phi_1^{(m+1)}}\boldsymbol{W} + \underbrace{(\varphi_0\phi_{n-1}^{(m)})}_{\phi_0^{(m+1)}}\boldsymbol{I} && \text{(D.11)}
\end{aligned}$$

which implies

$$\begin{cases} \phi_0^{(m+1)} & = \varphi_0 \phi_{n-1}^{(m)} \\ \phi_1^{(m+1)} & = \varphi_1 \phi_{n-1}^{(m)} + \phi_0^{(m)}) \\ & \dots \\ \phi_{n-2}^{(m+1)} & = \varphi_{n-2} \phi_{n-1}^{(m)} + \phi_{n-3}^{(m)} \\ \phi_{n-1}^{(m+1)} & = \varphi_{n-1} \phi_{n-1}^{(m)} + \phi_{n-2}^{(m)} \end{cases} \tag{D.12}$$

D.3 The Companion Matrix

Eq. D.12 can be thought as a linear system:

$$\begin{bmatrix} \phi_0^{(m+1)} \\ \phi_1^{(m+1)} \\ \vdots \\ \phi_{n-2}^{(m+1)} \\ \phi_{n-1}^{(m+1)} \end{bmatrix} = \boldsymbol{M} \begin{bmatrix} \phi_0^{(m)} \\ \phi_1^{(m)} \\ \vdots \\ \phi_{n-2}^{(m)} \\ \phi_{n-1}^{(m)} \end{bmatrix} \tag{D.13}$$

where $\boldsymbol{M}$ is defined as:

$$\boldsymbol{M} = \begin{bmatrix} 0 & \dots\dots\dots & & 0 & \varphi_0 \\ 1 & 0 & \dots & 0 & \varphi_1 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 1 & 0 & \varphi_{n-2} \\ 0 & 0 & \dots & 1 & \varphi_{n-1} \end{bmatrix} \tag{D.14}$$

Note that the characteristic polynomial of $\boldsymbol{M}$ is equal to the one of $\boldsymbol{W}$, so that they also share the same eigenvalues. In fact, $\boldsymbol{M}$ is also know as the *Frobenius companion matrix* of $\boldsymbol{W}$.

Bibliography

Ando, H. and Chang, H. [2021]. A model of computing with road traffic dynamics, *Nonlinear Theory and Its Applications, IEICE* **12**(2): 175–180.

Antonik, P., Gulina, M., Pauwels, J. and Massar, S. [2018]. Using a reservoir computer to learn chaotic attractors, with applications to chaos synchronization and cryptography, *Physical Review E* **98**(1): 012215.

Appeltant, L., Soriano, M. C., Van der Sande, G., Danckaert, J., Massar, S., Dambre, J., Schrauwen, B., Mirasso, C. R. and Fischer, I. [2011]. Information processing using a single dynamical node as complex system, *Nature Communications* **2**: 468.

Ashwin, P. and Postlethwaite, C. [2018]. Sensitive finite-state computations using a distributed network with a noisy network attractor, *IEEE transactions on neural networks and learning systems* **29**(12): 5847–5858.

Barbosa, W. A. S., Griffith, A., Rowlands, G. E., Govia, L. C. G., Ribeill, G. J., Nguyen, M.-H., Ohki, T. A. and Gauthier, D. J. [2021]. Symmetry-aware reservoir computing, *Phys. Rev. E* **104**: 045307.
URL: *https://link.aps.org/doi/10.1103/PhysRevE.104.045307*

Basterrech, S. [2017]. Empirical analysis of the necessary and sufficient conditions of the echo state property, *2017 International Joint Conference on Neural Networks (IJCNN)*, IEEE, pp. 888–896.

Baydogan, M. G. and Runger, G. [2016]. Time series representation and similarity based on local autopatterns, *Data Mining and Knowledge Discovery* **30**(2): 476–509.

Bengio, Y., Simard, P. and Frasconi, P. [1994]. Learning long-term dependencies with gradient descent is difficult, *IEEE Transactions on Neural Networks* **5**(2): 157–166.

Bernstein, D. S. [2009]. *Matrix mathematics*, Princeton university press.

Berry, T., Giannakis, D. and Harlim, J. [2015]. Nonparametric forecasting of low-dimensional dynamical systems, *Physical Review E* **91**(3): 032915.

Berry, T., Giannakis, D. and Harlim, J. [2020]. Bridging data science and dynamical systems theory, *arXiv preprint arXiv:2002.07928* .

Bertschinger, N. and Natschläger, T. [2004]. Real-time computation at the edge of chaos in recurrent neural networks, *Neural computation* **16**(7): 1413–1436.

Bezruchko, B. P. and Smirnov, D. A. [2010]. *Extracting knowledge from time series: An introduction to nonlinear empirical modeling*, Springer Science & Business Media.

Bianchi, F. M., Maiorino, E., Kampffmeyer, M. C., Rizzi, A. and Jenssen, R. [2017]. Recurrent neural networks for short-term load forecasting: an overview and comparative analysis.

Bianchi, F. M., Scardapane, S., Løkse, S. and Jenssen, R. [2020]. Reservoir computing approaches for representation and classification of multivariate time series, *IEEE Transactions on Neural Networks and Learning Systems* .

Birkhoff, G. D. [1931]. Proof of the ergodic theorem, *Proceedings of the National Academy of Sciences* **17**(12): 656–660.

Bittanti, S. [2019]. *Model identification and data analysis*, John Wiley & Sons.

Boccaletti, S., Kurths, J., Osipov, G., Valladares, D. and Zhou, C. [2002]. The synchronization of chaotic systems, *Physics Reports* **366**(1-2): 1–101.

Bollt, E. [2021]. On explaining the surprising success of reservoir computing forecaster of chaos? the universal machine learning dynamical system with contrast to var and dmd<? a3b2 show [feature]?>, *Chaos: An Interdisciplinary Journal of Nonlinear Science* **31**(1): 013108.

Bompas, S., Georgeot, B. and Guéry-Odelin, D. [2020]. Accuracy of neural networks for the simulation of chaotic dynamics: precision of training data vs precision of the algorithm, *arXiv preprint arXiv:2008.04222* .

Bouvrie, J. and Hamzi, B. [2017]. Kernel methods for the approximation of nonlinear systems, *SIAM Journal on Control and Optimization* **55**(4): 2460–2492.

Brunton, S. L., Brunton, B. W., Proctor, J. L., Kaiser, E. and Kutz, J. N. [2017]. Chaos as an intermittently forced linear system, *Nature communications* **8**(1): 1–9.

Brunton, S. L., Budišić, M., Kaiser, E. and Kutz, J. N. [2021]. Modern koopman theory for dynamical systems, *arXiv preprint arXiv:2102.12086* .

Brunton, S. L., Noack, B. R. and Koumoutsakos, P. [2020]. Machine learning for fluid mechanics, *Annual Review of Fluid Mechanics* **52**: 477–508.

Brunton, S. L., Proctor, J. L. and Kutz, J. N. [2016]. Discovering governing equations from data by sparse identification of nonlinear dynamical systems, *Proceedings of the national academy of sciences* **113**(15): 3932–3937.

Caluwaerts, K., Wyffels, F., Dieleman, S. and Schrauwen, B. [2013]. The spectral radius remains a valid indicator of the echo state property for large reservoirs, *The 2013 International Joint Conference on Neural Networks (IJCNN)*, IEEE, pp. 1–6.

Carroll, T. [2020a]. Path length statistics in reservoir computers, *Chaos: An Interdisciplinary Journal of Nonlinear Science* **30**(8): 083130.

Carroll, T. [2021a]. Low dimensional manifolds in reservoir computers, *Chaos: An Interdisciplinary Journal of Nonlinear Science* **31**(4): 043113.

Carroll, T. L. [2020b]. Dimension of reservoir computers, *Chaos: An Interdisciplinary Journal of Nonlinear Science* **30**(1): 013102.

Carroll, T. L. [2020c]. Do reservoir computers work best at the edge of chaos?, *Chaos: An Interdisciplinary Journal of Nonlinear Science* **30**(12): 121109.

Carroll, T. L. [2021b]. Optimizing reservoir computers for signal classification, *Frontiers in Physiology* **12**: 893.

Carroll, T. L. and Pecora, L. M. [2019]. Network structure effects in reservoir computers, *Chaos: An Interdisciplinary Journal of Nonlinear Science* **29**(8): 083130.

Casey, M. [1996]. The dynamics of discrete-time computation, with application to recurrent neural networks and finite state machine extraction, *Neural computation* **8**(6): 1135–1178.

Ceni, A., Ashwin, P., Livi, L. and Postlethwaite, C. [2020]. The echo index and multistability in input-driven recurrent neural networks, *Physica D* **412**.

Chang, B., Chen, M., Haber, E. and Chi, E. H. [2019]. Antisymmetricrnn: A dynamical system view on recurrent neural networks, *arXiv preprint arXiv:1902.09689* .

Chattopadhyay, A., Hassanzadeh, P. and Subramanian, D. [2020]. Data-driven predictions of a multiscale lorenz 96 chaotic system using machine-learning methods: reservoir computing, artificial neural network, and long short-term memory network, *Nonlinear Processes in Geophysics* **27**(3): 373–389.

Chaudhari, P., Choromanska, A., Soatto, S., LeCun, Y., Baldassi, C., Borgs, C., Chayes, J., Sagun, L. and Zecchina, R. [2019]. Entropy-sgd: Biasing gradient descent into wide valleys, *Journal of Statistical Mechanics: Theory and Experiment* **2019**(12): 124018.

Chen, R. T., Rubanova, Y., Bettencourt, J. and Duvenaud, D. [2018]. Neural ordinary differential equations, *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 6572–6583.

Cho, K., Van Merriënboer, B., Bahdanau, D. and Bengio, Y. [2014]. On the properties of neural machine translation: Encoder-decoder approaches, *arXiv preprint arXiv:1409.1259* .

Cranmer, M., Greydanus, S., Hoyer, S., Battaglia, P., Spergel, D. and Ho, S. [2020]. Lagrangian neural networks, *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*.

Dambre, J., Verstraeten, D., Schrauwen, B. and Massar, S. [2012]. Information processing capacity of dynamical systems, *Scientific reports* **2**(1): 1–7.

Derrida, B. and Pomeau, Y. [1986]. Random networks of automata: a simple annealed approximation, *EPL (Europhysics Letters)* **1**(2): 45.

Di Gregorio, E., Gallicchio, C. and Micheli, A. [2018]. Combining memory and non-linearity in echo state networks, *International Conference on Artificial Neural Networks*, Springer, pp. 556–566.

Doan, N. A. K., Polifke, W. and Magri, L. [2020]. Physics-informed echo state networks, *Journal of Computational Science* **47**: 101237.

Doan, N. A. K., Polifke, W. and Magri, L. [2021]. Short-and long-term prediction of a chaotic flow: A physics-constrained reservoir computing approach, *arXiv preprint arXiv:2102.07514* .

Elman, J. L. [1990]. Finding structure in time, *Cognitive science* **14**(2): 179–211.

Engelken, R., Wolf, F. and Abbott, L. F. [2020]. Lyapunov spectra of chaotic recurrent neural networks, *arXiv preprint arXiv:2006.02427* .

Fan, H., Jiang, J., Zhang, C., Wang, X. and Lai, Y.-C. [2020]. Long-term prediction of chaotic systems with recurrent neural networks, *arXiv preprint arXiv:2004.01258* .

Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L. and Muller, P.-A. [2019]. Deep learning for time series classification: a review, *Data Mining and Knowledge Discovery* **33**(4): 917–963.

Fernando, C. and Sojakka, S. [2003]. Pattern recognition in a bucket, *European conference on artificial life*, Springer, pp. 588–597.

Ferreira, A. A., Ludermir, T. B. and De Aquino, R. R. [2013]. An approach to reservoir computing design and training, *Expert systems with applications* **40**(10): 4172–4182.

Flynn, A., Herteux, J., Tsachouridis, V. A., Räth, C. and Amann, A. [2021]. Symmetry kills the square in a multifunctional reservoir computer, *Chaos: An Interdisciplinary Journal of Nonlinear Science* **31**(7): 073122.

Gaedke-Merzhäuser, L., Kopaničáková, A. and Krause, R. [2020]. Multilevel minimization for deep residual networks, *arXiv preprint arXiv:2004.06196* .

Gallicchio, C. [2018]. Chasing the echo state property, *Proceedings of the 27th european symposium on artificial neural networks.* .

Gallicchio, C. [2020]. Sparsity in reservoir computing neural networks, *2020 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, IEEE, pp. 1–7.

Gallicchio, C. [2021]. Reservoir computing by discretizing odes, *Proceedings of the 29th european symposium on artificial neural networks. 2021.*

Gallicchio, C. and Micheli, A. [2011]. Architectural and markovian factors of echo state networks, *Neural Networks* **24**(5): 440–456.

Gallicchio, C., Micheli, A. and Antonio, S. [2020]. Simplifying deep reservoir architectures.

Gallicchio, C., Micheli, A. and Pedrelli, L. [2017]. Deep reservoir computing: A critical experimental analysis, *Neurocomputing* **268**: 87–99.

Gallicchio, C., Micheli, A. and Pedrelli, L. [2018]. Design of deep echo state networks, *Neural Networks* **108**: 33–47.

Gallicchio, C., Micheli, A. and Silvestri, L. [2021]. Phase transition adaptation, *arXiv preprint arXiv:2104.10132* .

Ganguli, S., Huh, D. and Sompolinsky, H. [2008]. Memory traces in dynamical systems, *Proceedings of the National Academy of Sciences* **105**(48): 18970–18975.

Gers, F. A., Schmidhuber, J. and Cummins, F. [2000]. Learning to forget: Continual prediction with lstm, *Neural computation* **12**(10): 2451–2471.

Gilpin, W. [2020]. Deep learning of dynamical attractors from time series measurements, *arXiv preprint arXiv:2002.05909* .

Gilpin, W. [2021]. Chaos as an interpretable benchmark for forecasting and data-driven modelling, *arXiv preprint arXiv:2110.05266* .

Gonon, L., Grigoryeva, L. and Ortega, J.-P. [2020a]. Approximation bounds for random neural networks and reservoir systems, *arXiv preprint arXiv:2002.05933* .

Gonon, L., Grigoryeva, L. and Ortega, J.-P. [2020b]. Memory and forecasting capacities of nonlinear recurrent networks, *Physica D: Nonlinear Phenomena* **414**: 132721.

Gonon, L., Grigoryeva, L. and Ortega, J.-P. [2020c]. Risk bounds for reservoir computing, *Journal of Machine Learning Research* **21**(240): 1–61.

González-García, R., Rico-Martìnez, R. and Kevrekidis, I. G. [1998]. Identification of distributed parameter systems: A neural net based approach, *Computers & chemical engineering* **22**: S965–S968.

Goudarzi, A., Marzen, S., Banda, P., Feldman, G., Teuscher, C. and Stefanovic, D. [2016]. Memory and information processing in recurrent neural networks, *arXiv preprint arXiv:1604.06929* .

Graves, A., Wayne, G. and Danihelka, I. [2014]. Neural turing machines, *arXiv preprint arXiv:1410.5401* .

Greydanus, S., Dzamba, M. and Yosinski, J. [2019]. Hamiltonian neural networks, *arXiv preprint arXiv:1906.01563* .

Grigoryeva, L., Hart, A. and Ortega, J.-P. [2020]. Chaos on compact manifolds: Differentiable synchronizations beyond takens, *arXiv preprint arXiv:2010.03218* .

Grigoryeva, L., Hart, A. and Ortega, J.-P. [2021]. Learning strange attractors with reservoir systems, *arXiv preprint arXiv:2108.05024* .

Grigoryeva, L. and Ortega, J.-P. [2018]. Echo state networks are universal, *Neural Networks* **108**: 495–508.

Haluszczynski, A. and Räth, C. [2019]. Good and bad predictions: Assessing and improving the replication of chaotic attractors by means of reservoir computing, *Chaos: An Interdisciplinary Journal of Nonlinear Science* **29**(10): 103143.

Hammer, B. [2000]. On the approximation capability of recurrent neural networks, *Neurocomputing* **31**(1-4): 107–123.

Hammer, B. and Tiňo, P. [2003]. Recurrent neural networks with small weights implement definite memory machines, *Neural Computation* **15**(8): 1897–1929.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C. and Oliphant, T. E. [2020]. Array programming with NumPy, *Nature* **585**(7825): 357–362.
URL: *https://doi.org/10.1038/s41586-020-2649-2*

Hart, A. G., Hook, J. L. and Dawes, J. H. [2021]. Echo state networks trained by tikhonov least squares are l2 (μ) approximators of ergodic dynamical systems, *Physica D: Nonlinear Phenomena* **421**: 132882.

Hart, A., Hook, J. and Dawes, J. [2020]. Embedding and approximation theorems for echo state networks, *Neural Networks* **128**: 234–247.

Hasani, R., Lechner, M., Amini, A., Rus, D. and Grosu, R. [2021]. Liquid time-constant networks, *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35, pp. 7657–7666.

Herteux, J. and Räth, C. [2020]. Breaking symmetries of the reservoir equations in echo state networks, *Chaos: An Interdisciplinary Journal of Nonlinear Science* **30**(12): 123142.

Hochreiter, S. and Schmidhuber, J. [1997]. Long short-term memory, *Neural computation* **9**(8): 1735–1780.

Inubushi, M. and Yoshimura, K. [2017]. Reservoir computing beyond memory-nonlinearity trade-off, *Scientific reports* **7**(1): 1–10.

Jaeger, H. [2001]. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note, *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report* **148**(34): 13.

Jaeger, H. [2002]. *Short term memory in echo state networks*, Vol. 5, GMD-Forschungszentrum Informationstechnik.

Jaeger, H. and Haas, H. [2004]. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication, *science* **304**(5667): 78–80.

Jaeger, H., Lukoševičius, M., Popovici, D. and Siewert, U. [2007]. Optimization and applications of echo state networks with leaky-integrator neurons, *Neural networks* **20**(3): 335–352.

Jordan, I. D., Sokół, P. A. and Park, I. M. [2021]. Gated recurrent units viewed through the lens of continuous time dynamical systems, *Frontiers in computational neuroscience* p. 67.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S. and Yang, L. [2021]. Physics-informed machine learning, *Nature Reviews Physics* pp. 1–19.

Kauffman, S. [1969]. Homeostasis and differentiation in random genetic control networks, *Nature* **224**(5215): 177–178.

Kerg, G., Goyette, K., Puelma Touzel, M., Gidel, G., Vorontsov, E., Bengio, Y. and Lajoie, G. [2019]. Non-normal recurrent neural network (nnrnn): learning long time dependencies while improving expressivity with transient dynamics, *Advances in Neural Information Processing Systems* **32**: 13613–13623.

Kingma, D. P. and Ba, J. [2014]. Adam: A method for stochastic optimization, *arXiv preprint arXiv:1412.6980* .

Kloeden, P. E. and Rasmussen, M. [2011]. *Nonautonomous dynamical systems*, number 176, American Mathematical Soc.

Kocarev, L. and Parlitz, U. [1996]. Generalized synchronization, predictability, and equivalence of unidirectionally coupled dynamical systems, *Physical Review Letters* **76**(11): 1816.

Koopman, B. O. [1931]. Hamiltonian systems and transformation in hilbert space, *Proceedings of the national academy of sciences of the united states of america* **17**(5): 315.

Krishnagopal, S., Girvan, M., Ott, E. and Hunt, B. R. [2020]. Separation of chaotic signals by reservoir computing, *Chaos: An Interdisciplinary Journal of Nonlinear Science* **30**(2): 023123.

Langton, C. G. [1990]. Computation at the edge of chaos: Phase transitions and emergent computation, *Physica D: Nonlinear Phenomena* **42**(1-3): 12–37.

Larger, L., Soriano, M. C., Brunner, D., Appeltant, L., Gutiérrez, J. M., Pesquera, L., Mirasso, C. R. and Fischer, I. [2012]. Photonic information processing beyond turing: an optoelectronic implementation of reservoir computing, *Optics Express* **20**(3): 3241–3249.

Laurent, T. and von Brecht, J. [2016]. A recurrent neural network without chaos, *arXiv preprint arXiv:1612.06212* .

Lax, P. D. [2002]. *Functional analysis*, Pure and Applied Mathematics: A Wiley-Interscience Series of Texts, Monographs and Tracts, Wiley.

Legenstein, R. and Maass, W. [2007]. Edge of chaos and prediction of computational performance for neural circuit models, *Neural networks* **20**(3): 323–334.

Liesen, J. and Strakos, Z. [2013]. *Krylov subspace methods: principles and analysis*, Oxford University Press.

Liu, G.-H. and Theodorou, E. A. [2019]. Deep learning theory review: An optimal control and dynamical systems perspective, *arXiv preprint arXiv:1908.10920* .

Liu, Y., Kutz, J. N. and Brunton, S. L. [2020]. Hierarchical deep learning of multiscale differential equation time-steppers, *arXiv preprint arXiv:2008.09768* .

Livi, L., Bianchi, F. M. and Alippi, C. [2017]. Determination of the edge of criticality in echo state networks through fisher information maximization, *IEEE transactions on neural networks and learning systems* **29**(3): 706–717.

Løkse, S., Bianchi, F. M. and Jenssen, R. [2017]. Training echo state networks with regularization through dimensionality reduction, *Cognitive Computation* **9**(3): 364–378.

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

Lu, Z. and Bassett, D. S. [2020]. Invertible generalized synchronization: A putative mechanism for implicit learning in neural systems, *Chaos: An Interdisciplinary Journal of Nonlinear Science* **30**(6): 063133.

Lu, Z., Hunt, B. R. and Ott, E. [2018]. Attractor reconstruction by machine learning, *Chaos: An Interdisciplinary Journal of Nonlinear Science* **28**(6): 061104.

Lu, Z., Pathak, J., Hunt, B., Girvan, M., Brockett, R. and Ott, E. [2017]. Reservoir observers: Model-free inference of unmeasured variables in chaotic systems, *Chaos: An Interdisciplinary Journal of Nonlinear Science* **27**(4): 041102.

Luh, K. and O'Rourke, S. [2021]. Eigenvectors and controllability of non-hermitian random matrices and directed graphs, *Electronic Journal of Probability* **26**: 1–43.

Lukoševičius, M. [2012]. A practical guide to applying echo state networks, *Neural networks: Tricks of the trade*, Springer, pp. 659–686.

Lukoševičius, M. and Uselis, A. [2019]. Efficient cross-validation of echo state networks, *International Conference on Artificial Neural Networks*, Springer, pp. 121–133.

Lymburn, T., Khor, A., Stemler, T., Corrêa, D. C., Small, M. and Jüngling, T. [2019]. Consistency in echo-state networks, *Chaos: An Interdisciplinary Journal of Nonlinear Science* **29**(2): 023118.

Maass, W., Natschläger, T. and Markram, H. [2002]. Real-time computing without stable states: A new framework for neural computation based on perturbations, *Neural Computation* **14**(11): 2531–2560.

Mackey, M. C. and Glass, L. [1977]. Oscillation and chaos in physiological control systems, *Science* **197**(4300): 287–289.

Manjunath, G. and Jaeger, H. [2013]. Echo state property linked to an input: Exploring a fundamental characteristic of recurrent neural networks, *Neural Computation* **25**(3): 671–696.

Manjunath, G., Tino, P. and Jaeger, H. [2012]. Theory of input driven dynamical systems, *dice. ucl. ac. be, number April* pp. 25–27.

Marzen, S. [2017]. Difference between memory and prediction in linear recurrent networks, *Physical Review E* **96**(3): 032308.

Meiss, J. D. [2007]. *Differential dynamical systems*, SIAM.

Neftci, E. O. [2018]. Data and power efficient intelligence with neuromorphic learning machines, *Iscience* **5**: 52–68.

Neftci, E. O., Augustine, C., Paul, S. and Detorakis, G. [2017]. Event-driven random back-propagation: Enabling neuromorphic deep learning machines, *Frontiers in Neuroscience* **11**: 324.

O'Rourke, S. and Touri, B. [2015]. Controllability of random systems: Universality and minimal controllability, *arXiv preprint arXiv:1506.03125* .

Ott, E. [2002]. *Chaos in dynamical systems*, Cambridge university press.

Parlitz, U. [2012]. Detecting generalized synchronization, *Nonlinear Theory and Its Applications, IEICE* **3**(2): 113–127.

Pascanu, R., Gulcehre, C., Cho, K. and Bengio, Y. [2013]. How to construct deep recurrent neural networks, *arXiv preprint arXiv:1312.6026* .

Pascanu, R., Mikolov, T. and Bengio, Y. [2013]. On the difficulty of training recurrent neural networks, *International conference on machine learning*, PMLR, pp. 1310–1318.

Pathak, J., Hunt, B., Girvan, M., Lu, Z. and Ott, E. [2018]. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach, *Physical review letters* **120**(2): 024102.

Pathak, J., Wikner, A., Fussell, R., Chandra, S., Hunt, B. R., Girvan, M. and Ott, E. [2018]. Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model, *Chaos: An Interdisciplinary Journal of Nonlinear Science* **28**(4): 041101.

Pecora, L. M. and Carroll, T. L. [1990]. Synchronization in chaotic systems, *Physical Review Letters* **64**(8): 821.

Plackett, R. L. [1950]. Some theorems in least squares, *Biometrika* **37**(1/2): 149–157.

Platt, J. A., Wong, A. S., Clark, R., Penny, S. G. and Abarbanel, H. D. [2021]. Forecasting using reservoir computing: The role of generalized synchronization, *arXiv preprint arXiv:2103.00362* .

Prokopenko, M., Harré, M., Lizier, J., Boschetti, F., Peppas, P. and Kauffman, S. [2019]. Self-referential basis of undecidable dynamics: From the liar paradox and the halting problem to the edge of chaos, *Physics of life reviews* **31**: 134–156.

Pyragas, K. [1996]. Weak and strong synchronization of chaos, *Physical Review E* **54**(5): R4508.

Qi, D. and Majda, A. J. [2020]. Using machine learning to predict extreme events in complex systems, *Proceedings of the National Academy of Sciences* **117**(1): 52–59.

Racca, A. and Magri, L. [2021]. Robust optimization and validation of echo state networks for learning chaotic dynamics, *Neural Networks* .

Rajan, K., Abbott, L. and Sompolinsky, H. [2010]. Stimulus-dependent suppression of chaos in recurrent neural networks, *Physical Review E* **82**(1): 011903.

Regazzoni, F., Dede, L. and Quarteroni, A. [2019]. Machine learning for fast and reliable solution of time-dependent differential equations, *Journal of Computational physics* **397**: 108852.

Rivkind, A. and Barak, O. [2017]. Local dynamics in trained recurrent neural networks, *Physical Review Letters* **118**(25): 258101.

Rodan, A. and Tino, P. [2010]. Minimum complexity echo state network, *IEEE transactions on neural networks* **22**(1): 131–144.

Rohlf, T. and Bornholdt, S. [2002]. Criticality in random threshold networks: annealed approximation and beyond, *Physica A: Statistical Mechanics and its Applications* **310**(1-2): 245–259.

Rössler, O. E. [1976]. An equation for continuous chaos, *Physics Letters A* **57**(5): 397–398.

Rudelson, M. and Vershynin, R. [2010]. Non-asymptotic theory of random matrices: extreme singular values, *Proceedings of the International Congress of Mathematicians 2010 (ICM 2010) (In 4 Volumes) Vol. I: Plenary Lectures and Ceremonies Vols. II–IV: Invited Lectures*, World Scientific, pp. 1576–1602.

Rulkov, N. F., Sushchik, M. M., Tsimring, L. S. and Abarbanel, H. D. [1995]. Generalized synchronization of chaos in directionally coupled chaotic systems, *Physical Review E* **51**(2): 980.

Schmid, P. J. [2010]. Dynamic mode decomposition of numerical and experimental data, *Journal of fluid mechanics* **656**: 5–28.

Schmidhuber, J. and Hochreiter, S. [1996]. Guessing can outperform many long time lag algorithms, *IDSIA technical report* .

Schrauwen, B., Verstraeten, D. and Van Campenhout, J. [2007]. An overview of reservoir computing: theory, applications and implementations, *Proceedings of the 15th european symposium on artificial neural networks. p. 471-482 2007*, pp. 471–482.

Shalev-Shwartz, S. and Ben-David, S. [2014]. *Understanding machine learning: From theory to algorithms*, Cambridge university press.

Shawe-Taylor, J. and Cristianini, N. [2004]. *Kernel Methods for Pattern Analysis*, Cambridge University Press, Cambridge, UK.

Shi, Z. and Han, M. [2007]. Support vector echo-state machine for chaotic time-series prediction, *IEEE Transactions on Neural Networks* **18**(2): 359–372.

Siegelmann, H. T. [2003]. Neural and super-turing computing, *Minds and Machines* **13**(1): 103–114.

Siegelmann, H. T. and Sontag, E. D. [1995]. On the computational power of neural nets, *Journal of computer and system sciences* **50**(1): 132–150.

Sompolinsky, H., Crisanti, A. and Sommers, H.-J. [1988]. Chaos in random neural networks, *Physical review letters* **61**(3): 259.

Sontag, E. D. [2013]. *Mathematical control theory: deterministic finite dimensional systems*, Vol. 6, Springer Science & Business Media.

Stark, J. [1999]. Regularity of invariant graphs for forced systems, *Ergodic theory and dynamical systems* **19**(1): 155–199.

Sussillo, D. and Abbott, L. F. [2009]. Generating coherent patterns of activity from chaotic neural networks, *Neuron* **63**(4): 544–557.

Takens, F. [1981]. Detecting strange attractors in turbulence, *Dynamical Systems and Turbulence*, Springer, pp. 366–381.

Tallec, C. and Ollivier, Y. [2018]. Can recurrent neural networks warp time?, *International Conference on Learning Representation 2018*.

Tanaka, G., Yamane, T., Héroux, J. B., Nakane, R., Kanazawa, N., Takeda, S., Numata, H., Nakano, D. and Hirose, A. [2019]. Recent advances in physical reservoir computing: A review, *Neural Networks* **115**: 100 – 123.

Tao, T., Vu, V. and Krishnapur, M. [2010]. Random matrices: Universality of esds and the circular law, *The Annals of Probability* **38**(5): 2023–2065.

Thiede, L. A. and Parlitz, U. [2019]. Gradient based hyperparameter optimization in echo state networks, *Neural Networks* **115**: 23–29.

Tiňo, P. [2018]. Asymptotic fisher memory of randomized linear symmetric echo state networks, *Neurocomputing* **298**: 4–8.

Tiňo, P. [2020]. Dynamical systems as temporal feature spaces., *Journal of Machine Learning Research* **21**(44): 1–42.

Tiňo, P. and Dorffner, G. [2001]. Predicting the future of discrete sequences from fractal representations of the past, *Machine Learning* **45**(2): 187–217.

Tiňo, P. and Rodan, A. [2013]. Short term memory in input-driven linear dynamical systems, *Neurocomputing* **112**: 58–63.

Verstraeten, D., Dambre, J., Dutoit, X. and Schrauwen, B. [2010]. Memory versus non-linearity in reservoirs, *The 2010 international joint conference on neural networks (IJCNN)*, IEEE, pp. 1–8.

Verstraeten, D. and Schrauwen, B. [2009]. On the quantification of dynamics in reservoir computing, *International Conference on Artificial Neural Networks*, Springer, pp. 985–994.

Verstraeten, D., Schrauwen, B., d'Haene, M. and Stroobandt, D. [2007]. An experimental unification of reservoir computing methods, *Neural Networks* **20**(3): 391–403.

Verzelli, P., Alippi, C. and Livi, L. [2019]. Echo state networks with self-normalizing activations on the hyper-sphere, *Scientific reports* **9**(1): 1–14.

Verzelli, P., Alippi, C. and Livi, L. [2021]. Learn to synchronize, synchronize to learn, *Chaos: An Interdisciplinary Journal of Nonlinear Science* **31**(8): 083119.

Verzelli, P., Alippi, C., Livi, L. and Tiňo, P. [2021]. Input-to-state representation in linear reservoirs dynamics, *IEEE Transactions on Neural Networks and Learning Systems* .

Verzelli, P., Livi, L. and Alippi, C. [2018]. A characterization of the edge of criticality in binary echo state networks, *2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP)*, IEEE, pp. 1–6.

Vlachas, P. R., Byeon, W., Wan, Z. Y., Sapsis, T. P. and Koumoutsakos, P. [2018]. Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks, *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* **474**(2213): 20170844.

Vlachas, P. R., Pathak, J., Hunt, B. R., Sapsis, T. P., Girvan, M., Ott, E. and Koumoutsakos, P. [2020]. Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics, *Neural Networks* .

Vogt, R., Touzel, M. P., Shlizerman, E. and Lajoie, G. [2020]. On lyapunov exponents for rnns: Understanding information propagation using dynamical systems tools, *arXiv preprint arXiv:2006.14123* .

Weigend, A. S. and Gershenfeld, N. A. [1993]. Results of the time series prediction competition at the santa fe institute, *IEEE international conference on neural networks*, IEEE, pp. 1786–1793.

Weinan, E. [2017]. A proposal on machine learning via dynamical systems, *Communications in Mathematics and Statistics* **5**(1): 1–11.

Weng, T., Yang, H., Gu, C., Zhang, J. and Small, M. [2019]. Synchronization of chaotic systems and their machine-learning models, *Physical Review E* **99**(4): 042203.

Williams, R. J. and Zipser, D. [1989]. A learning algorithm for continually running fully recurrent neural networks, *Neural computation* **1**(2): 270–280.

Xue, Y., Yang, L. and Haykin, S. [2007]. Decoupled echo state networks with lateral inhibition, *Neural Networks* **20**(3): 365–376.

Yildiz, I. B., Jaeger, H. and Kiebel, S. J. [2012]. Re-visiting the echo state property, *Neural Networks* **35**: 1–9.

Yule, G. U. [1927]. On a method of investigating periodicities disturbed series, with special reference to wolfer’s sunspot numbers, *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* **226**(636-646): 267–298.

Zhang, B., Miller, D. J. and Wang, Y. [2011]. Nonlinear system modeling with random matrices: echo state networks revisited, *IEEE Transactions on Neural Networks and Learning Systems* **23**(1): 175–182.

Zimmermann, R. S. and Parlitz, U. [2018]. Observing spatio-temporal dynamics of excitable media using reservoir computing, *Chaos: An Interdisciplinary Journal of Nonlinear Science* **28**(4): 043118.