



**Universitat**  
de les Illes Balears



Instituto de Física Interdisciplinar y Sistemas Complejos

# DOCTORAL THESIS

## 2015

# COMPUTATIONAL PROPERTIES OF DELAY-COUPLED SYSTEMS

**Miguel Angel Escalona Morán**



**Universitat**  
de les Illes Balears



Instituto de Física Interdisciplinar y Sistemas Complejos

# DOCTORAL THESIS

## 2015

### Doctoral Program of Physics

# COMPUTATIONAL PROPERTIES OF DELAY-COUPLED SYSTEMS

## Miguel Angel Escalona Morán

Thesis Supervisor: Claudio R. Mirasso Santos  
Thesis Co-Supervisor: Miguel Cornelles Soriano

## Doctor by the Universitat de les Illes Balears

## COMPUTATIONAL PROPERTIES OF DELAY-COUPLED SYSTEMS

Miguel Angel Escalona Morán

Tesis realizada en el Instituto de Física Interdisciplinar y Sistemas Complejos (IFISC) y presentada en el Departamento de Física de la Universitat de les Illes Balears.

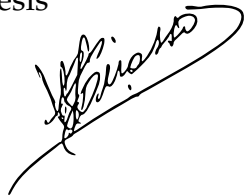
Supervisor: Prof. Claudio R. Mirasso Santos  
Co-Supervisor: Dr. Miguel Cornelles Soriano



---

Tesis Doctoral presentada por Miguel Angel Escalona Morán para optar al título de Doctor, en el Programa de Física del Departamento de Física de la Universitat de les Illes Balears, realizada en el IFISC bajo la dirección de Claudio Mirasso, catedrático de universidad y Miguel Cornelles Soriano, contratado postdoctoral CAIB.

Visto bueno  
Director de la tesis



Prof. Claudio Mirasso

Co-Director de la tesis



Dr. Miguel Cornelles Soriano

Doctorando



Miguel Angel Escalona Morán

Palma, Junio de 2015

---



---

# Summary of the research work

In this research work we study the computational properties of delay-coupled systems. In particular, we use a machine learning technique known as [reservoir computing](#). In machine learning, a computer *learns* to solve different tasks using examples and without knowing explicitly their solution.

For the study of the computational properties, a numerical toolbox, written in Python, was developed. This toolbox allows a fast implementation of the different scenarios described in this thesis.

Using a reservoir computer, we studied several computational properties, focusing on its kernel quality, its ability to separate different input samples and the intrinsic memory capacity. This intrinsic memory is related to the delayed-feedback of the reservoir.

We used a delay-coupled system as reservoir to study its computational ability in three different kinds of tasks: system's modeling, time-series prediction and classification tasks.

The system's modeling task was performed using the Nonlinear Autoregressive Moving Average (of ten steps), NARMA10. The NARMA10 model creates autoregressive time series from a set of normally distributed random sequences. The reservoir computer learns how to emulate the system using only the sequence of random numbers and the autoregressive time series, but without knowing the equations of the NARMA10. The results of our approach are equivalent to those published by other authors and show the computational power of our method.

For the time-series prediction tasks, we used three kinds of time series: a model that gives the variations in temperature of the sea surface that provoke El Niño phenomenon, the Lorenz system and the dynamics of a chaotic laser. Different scenarios were explored depending on the nature of the time series. For the prediction of the variation in temperature of the sea surface, we perform estimations of one, three and six months in advance. The error was measured as

the Normalized Root Mean Square Error ([NRMSE](#)). For the different prediction horizons, we obtained errors of 2%, 8% and 24%, respectively.

The classification tasks were carried out for a Spoken Digit Recognition ([SDR](#)) task and a real-world biomedical task. The [SDR](#) was used to illustrate different scenarios of a machine learning problem. The biomedical task consists on the automatic classification of heartbeats with cardiac arrhythmias. We use the MIT-BIH Arrhythmia database, a widely used database in cardiology. For comparison purposes, we followed the guidelines of the Association for the Advancement of Medical Instrumentation for the evaluation of arrhythmia-detector algorithms. We used a biostatistical learning process named logistic regression that allowed to compute the probability that a heartbeat belongs to a particular class. This is in contrast to the commonly used linear regression. The results obtained in this work show the versatility and efficiency of our implemented reservoir computer. Our results are equivalent and show improvement over other reported results on this problem under similar conditions and using the same database.

To enhance the computational ability of our delay-coupled system, we included a multivariate scheme that allows the consideration of different variables of a system. We evaluated the influence of this multivariate scenario using a time-series prediction and the classification of heartbeat tasks. The results show improvement in the performance of the reservoir computer in comparison with the same tasks in the univariate case.



---

# Resumen del trabajo de investigación

En esta tesis se estudian las propiedades computacionales de sistemas acoplados con retraso. En particular la técnica de machine learning, conocida como [reservoir computing](#), es utilizada. En esta técnica el ordenador aprende a resolver tareas a partir de ejemplos que se han dado previamente pero sin indicarle de forma explícita la forma de resolver estos problemas.

El desarrollo de este trabajo incluye la creación de una herramienta computacional, escrita en lenguaje Python para la ejecución de los diferentes escenarios presentados en este trabajo.

Con la implementación de un sistema acoplado con retraso, hemos estudiado las propiedades de cómputo de este tipo de sistemas, interesándonos principalmente en la calidad del sistema acoplado, su habilidad de separación de elementos distintos y su capacidad intrínseca de memoria, la cual está asociada a la presencia de una retroalimentación retrasada.

El sistema se ha usado para demostrar el poder de cálculo que ofrecen los sistemas acoplados con retraso. Se utilizaron tres tipos de tareas: modelado, predicción de series de tiempo y clasificación.

El modelado se realizó utilizando el sistema Nonlinear Autoregressive Moving Average de 10 pasos (NARMA10). Este sistema, construye series temporales autoregresivas a partir de series de números aleatorios. El ordenador basado en reservoir aprende a emular este sistema (sin conocer de forma explícita las ecuaciones del mismo) a partir de las secuencias de números aleatorios y las series temporales autoregresivas. Los resultados obtenidos son equivalentes a los publicados por otros autores, demostrando el poder computacional de este método.

Para la predicción de series temporales se usaron modelos de variación de temperatura que provocan la aparición del fenómeno de El Niño, el sistema de Lorenz en régimen caótico y la dinámica de un laser caótico. Las estimaciones de series temporales se realizaron bajo diversas circunstancias dependiendo de la naturaleza de las series. Para el caso de El Niño, se realizaron predicciones a uno,

tres y seis meses con errores de estimación, determinados por el Normalized Root Mean Square Error (NRMSE) de 2%, 8% y 24%, respectivamente.

Como primera tarea de clasificación, se utilizó la tarea de Spoken Digit Recognition y se utilizó para ilustrar diferentes escenarios posibles de un sistema acoplado con retraso. La segunda tarea de clasificación y la más exhaustiva, se realizó en un problema real de origen biomédico: la clasificación de latidos cardíacos para el caso de arritmias. Se utilizó la base de datos MIT-BIH Arrhythmia la cual ha sido ampliamente usada en cardiología. Por motivos de comparación de resultados, se siguieron las recomendaciones dadas por la Association for the Advancement of Medical Instrumentation para la evaluación de algoritmos detectores de arritmias. Se utilizó un método de entrenamiento del *reservoir computer* llamado regresión logística en lugar del comúnmente usado: la regresión lineal. La regresión logística nos permite obtener como resultado la probabilidad de que un latido cardíaco pertenezca a una clase u a otra. Los resultados obtenidos demuestran la versatilidad y eficacia de nuestro método de cálculo, ya que son equivalentes e incluso mejores a los resultados publicados por otros trabajos bajo circunstancias similares de evaluación y utilizando la misma base de datos.

Para mejorar la capacidad de computación del sistema con retraso, se incluyeron variables dinámicas adicionales en nuestro sistema para evaluar el efecto en la predicción de series de tiempo y la clasificación de latidos cardíacos. Los resultados mostraron una mejora sustancial en comparación con el caso en que sólo una variable o canal del electrocardiograma fue usado para realizar la tarea dada.

---

# Resum del treball de recerca

En aquesta tesi s'estudien les propietats computacionals de sistemes acoblats amb retard. En particular, hem utilitzat la tècnica de "machine learning" coneguda com [reservoir computing](#). En aquesta tècnica, l'ordinador aprèn a resoldre tasques a partir d'exemples que s'han donat prèviament però sense indicar-li de forma explícita la forma de resoldre aquests problemes.

El desenvolupament d'aquest treball inclou la creació d'una eina computacional, escrita en llenguatge Python per a l'execució dels diferents escenaris presentats en aquest treball.

Amb la implementació d'un sistema acoblat amb retard, hem estudiat les propietats de còmput d'aquest tipus de sistemes, interessant-nos principalment en la qualitat del sistema acoblat, la seva habilitat de separació d'elements diferents i la seva capacitat intrínseca de memòria, la qual està associada a la presència d'una retroalimentació retardada.

El sistema s'ha fet servir per demostrar el poder de càlcul que ofereixen els sistemes acoblats amb retard. Es van utilitzar tres tipus de tasques: modelatge, predicció de sèries de temps i classificació.

El modelatge es va realitzar utilitzant el model "Nonlinear Autoregressive Moving Average" de 10 passos (NARMA10). Aquest model, construeix sèries temporals autoregressives a partir de sèries de nombres aleatoris. L'ordinador basat en "reservoir computing" aprèn a emular aquest model (sense conèixer de forma explícita les equacions del mateix) a partir de les seqüències de nombres aleatoris i les sèries temporals autoregressives. Els resultats obtinguts són equivalents als publicats per altres autors, demostrant el poder computacional d'aquest mètode.

Per a la predicció de sèries temporals es van usar models de variació de temperatura que provoquen l'aparició del fenomen de El Niño, el sistema de Lorenz en règim caòtic i la dinàmica d'un làser caòtic. Les estimacions de sèries temporals es van realitzar sota diverses circumstàncies depenent de la naturalesa de les sèries. Per al cas d'El Niño, es van realitzar prediccions a un, tres i sis mesos

amb errors d'estimació, determinats pel "Normalized Root Mean Square Error" (NRMSE) de 2%, 8% i 24%, respectivament.

Com a primera tasca de classificació, es va utilitzar la tasca de "Spoken Digit Recognition" i s'han il·lustrat diferents configuracions possibles d'un sistema acoblat amb retard. La segona tasca de classificació i la més exhaustiva, es va realitzar en un problema real d'origen biomèdic: la classificació de batecs cardíacs per al cas d'arítmies. Es va utilitzar la base de dades "MIT-BIH Arrhythmia", la qual ha estat àmpliament usada en cardiologia. Per motius de comparació de resultats, es van seguir les recomanacions donades per la "Association for the Advancement of Medical Instrumentation" per a l'avaluació d'algoritmes detectors d'arítmies. Es va utilitzar un mètode d'entrenament del reservoir computer anomenat regressió logística en lloc del comunament usat: la regressió lineal. La regressió logística ens permet obtenir com a resultat la probabilitat que un batec cardíac pertanyi a una classe o a una altra. Els resultats obtinguts demostren la versatilitat i eficàcia del nostre mètode de càlcul, ja que són equivalents i fins i tot millors als resultats publicats per altres treballs sota circumstàncies similars d'avaluació i fent servir la mateixa base de dades.

Per millorar la capacitat de computació del sistema amb retard, es van incloure variables dinàmiques addicionals en el nostre sistema per avaluar l'efecte en la predicció de sèries de temps i la classificació de batecs cardíacs. Els resultats van mostrar una millora substancial en comparació amb el cas en que només una variable o canal de l'electrocardiograma va ser usat per realitzar la tasca donada.

---

# List of Publications

## Published

1. M.C. Soriano, D. Brunner, M. Escalona-Morán, C. R. Mirasso and I. Fischer, *Minimal approach to neuro-inspired information processing*. *Frontiers in Computational Neuroscience*, 9:68, (2015).
2. M.A. Escalona-Morán, M.C. Soriano, I. Fischer, C.R. Mirasso. *Electrocardiogram classification using reservoir computing with logistic regression*. *IEEE J. Biomedical and Health Informatics*, 19, issue 3, pp. 892-898, (2015).
3. M. Escalona-Morán, M.C. Soriano, J. García-Prieto, I. Fischer, C.R. Mirasso. *Multivariate nonlinear time-series estimation using delay-based reservoir computing*. *European Physical J. Special Topics*, 223, issue 13, pp. 2903-2912, (2014).
4. K. Hicke, M.A. Escalona-Morán, D. Brunner, M.C. Soriano, I. Fischer and C.R. Mirasso. *Information processing using transient dynamics of semiconductor lasers subject to delayed feedback*. *IEEE J. Selected Topics in Quantum Electronics*, 19, issue 4, 1501610, (2013).

## Accepted

5. Miquel L. Alomar, Miguel C. Soriano, Miguel Escalona-Morán, Vincent Canals, Ingo Fischer, Claudio R. Mirasso, José L. Rosselló. *Digital Implementation of a single dynamical node reservoir computer*. *IEEE Transactions on Circuits and Systems II*, vol. Accepted, April 2015.



---

# Contents

Cover	i
Second cover	ii
Summary of the research work	vii
Resumen del trabajo de investigación	ix
Resum del treball de recerca	xi
List of Publications	xiii
Contents	xv
<b>I Introduction and purpose of research</b>	<b>1</b>
<b>1 Introduction</b>	<b>3</b>
1.1 Machine Learning . . . . .	4
1.1.1 Kernels to process information . . . . .	6
Artificial Neural Networks . . . . .	6
Reservoir Computing . . . . .	7
1.1.2 Learning process . . . . .	9
Linear regression models . . . . .	9
Normal equations . . . . .	10
Regularization . . . . .	10
1.2 Diagnosing learning algorithms . . . . .	11
1.2.1 Model complexity, bias and variance . . . . .	11
1.2.2 Cross-validation . . . . .	12
1.2.3 Learning curves . . . . .	15
Small note about floating-point precision of a machine . . .	17
1.2.4 Course to follow for high bias or high variance problems .	17
	xv

1.3	Evaluation of performance . . . . .	19
1.3.1	Confusion matrix . . . . .	19
	Sensitivity . . . . .	20
	Precision . . . . .	21
	Specificity . . . . .	21
	Accuracy . . . . .	21
1.3.2	The trade-off between sensitivity and specificity . . . . .	22
1.3.3	Receiver operating characteristics curve . . . . .	24
	Area under a ROC curve . . . . .	25
1.4	Object of the research work . . . . .	25
1.5	Structure of this thesis . . . . .	26
 <b>II Methodology, results and discussion</b>		<b>29</b>
 <b>2 Delay-based reservoir computing</b>		<b>31</b>
2.1	Reservoir computing based on delay-dynamical systems . . . . .	32
2.1.1	The input layer: pre-processing of the input signal . . . . .	34
2.1.2	The reservoir layer: A closer look to the reservoir dynamics . . . . .	36
2.1.3	The output layer . . . . .	37
2.1.4	Mackey-Glass delayed feedback oscillator as a reservoir . . . . .	38
2.2	Task-independent properties of reservoirs . . . . .	40
2.2.1	Kernel quality . . . . .	41
2.2.2	Generalization property . . . . .	42
2.2.3	Computational ability . . . . .	43
2.2.4	Memory capacity and quality . . . . .	44
2.3	Typical machine learning tasks using Reservoir Computing . . . . .	46
2.3.1	Spoken digit recognition (SDR) . . . . .	47
2.3.2	El Niño time-series prediction . . . . .	53
2.3.3	Modelling: NARMA10 . . . . .	55
2.4	Exploration of Reservoir's features . . . . .	57
2.4.1	Initializing reservoir's states . . . . .	58
2.4.2	Quorum sensing: subsampling reservoir's states . . . . .	60
 <b>3 Classification of heartbeats using RC</b>		<b>63</b>
3.1	Motivation to the classification of heartbeats . . . . .	63
3.2	Some generalities about the cardiovascular system . . . . .	65
3.2.1	The cardiac cycle and the ECG . . . . .	67
3.2.2	Registration of the electrical activity of the heart . . . . .	68
	Cardiac electrogenesis and the origin of the vectocardiogram . . . . .	69
	Aumented and precordial leads . . . . .	70
	Modifications of leads . . . . .	71
3.2.3	Cardiac arrhythmia . . . . .	72



3.3	Data base description . . . . .	73
3.4	The learning process: Logistic regression . . . . .	77
3.4.1	Estimation of logistic regression coefficients . . . . .	78
3.5	Classification of heartbeats using logistic regression . . . . .	79
3.5.1	Results and discussion . . . . .	79
3.6	Conclusions . . . . .	87
<b>4</b>	<b>Multivariate delayed reservoir computers</b>	<b>89</b>
4.1	Time series estimation: the Lorenz system . . . . .	89
4.1.1	One-step prediction . . . . .	91
4.1.2	Multistep prediction . . . . .	95
4.2	Multivariate arrhythmic heartbeat classification with logistic regression . . . . .	97
4.3	Conclusions . . . . .	99
<b>5</b>	<b>Reservoir computing using semiconductor laser dynamics</b>	<b>103</b>
5.1	Semiconductor laser rate equations . . . . .	104
5.1.1	Input signal injection into the laser dynamics . . . . .	106
5.2	Numerical results . . . . .	107
5.2.1	Spoken digit recognition task . . . . .	108
5.2.2	Santa Fe time series prediction task . . . . .	110
5.2.3	Influence of spontaneous emission noise . . . . .	113
5.3	Conclusion . . . . .	114
<b>III</b>	<b>Conclusions</b>	<b>117</b>
<b>6</b>	<b>Conclusions</b>	<b>119</b>
6.1	Future perspectives . . . . .	123
<b>A</b>	<b>Supplementary material for Chapter 3</b>	<b>125</b>
A.1	MIT-BIH arrhythmia database . . . . .	125
A.1.1	Annotations . . . . .	126
	<b>Glossary</b>	<b>129</b>
	<b>Acronyms</b>	<b>131</b>
	<b>Bibliography</b>	<b>135</b>



# **Part I**

## **Introduction and purpose of research**



# Introduction

Evolution of computing is facing nowadays what is called the *information age*, an era where the amount of digital information is huge and individuals are free to transfer information and have instant access to knowledge. Managing and processing big amounts of digital information has become a problem in computer science demanding processing techniques and novel computational concepts that go beyond those implemented in traditional computers [1, 2]. Even if processing a large amount of information has changed the way computers work, this is not new for our brain. In a quiet place, the brain is receiving information about the environment. Without thinking specifically on that, we are aware of the temperature of the room, surfaces in contact with us, objects in front, sounds and smells. The brain processes all this information almost simultaneously and it is able to produce a response in fractions of a second if needed. Recognizing a common object can happen without us even noticing we performed that task. The case of computers is not that straightforward. Building a traditional piece of code to recognize objects can be a difficult task and could take a long time to the computer to deliver an answer. Again the brain proceeds in a different way. Instead of only studying particular features of objects, it analyzes the full concept by learning from examples of the object. Traditional Von Neumann computers or Turing approaches [3] are very efficient when executing basic mathematical instructions. These approaches are usually much faster than the human brain. However, for highly complex computational tasks, such as face or handwritten-digit recognition, traditional computers run into trouble and the brain shows to be more efficient.

The networks of neurons that constitute our brain are in a constant activity categorizing patterns, making predictions, and silencing stimuli. At this moment, the reader is not only recognizing the symbols of these words but giving a meaning to the sentence. At the same time the brain knows that there are shoes on the reader's feet however the reader did not notice them until it was mentioned. This is an example of silencing stimuli that are not needed for the

## CHAPTER 1. INTRODUCTION

task of reading. Teaching a computer to know what is important for a task is not an easy endeavor.

An alternative to traditional computers is a neuro-inspired scheme of processing information. Using an [artificial neural network \(ANN\)](#), a computer can *learn* how to solve a problem without executing the traditional set of preprogrammed instructions. The scientific discipline that focuses on designing and implementing algorithms to optimize the learning of machines is called Machine Learning. The idea behind machine learning is to let the computer extract the rules to perform a task by *showing* characteristic examples of the elements to study.

### 1.1

---

## Machine Learning

The machine learning concept is a branch of artificial intelligence ([AI](#)) that focuses on the construction and study of systems that can learn from examples. For instance, if we want a computer to recognize alphabetic handwritten characters, we could provide the computer with some handwritten samples. The machine will learn the patterns during a training process. Then we can provide unseen samples of handwritten characters and take the answer of the machine during a testing process. The unseen characters can be interpreted to belong to one of the categories of characters that was presented during the training process. The machine has to be able to generalize to samples that were not present during the training process in order to be useful.

Machine learning algorithms are also used in data mining. For example, in medical records learning algorithms can transform raw data related to specific tests and patient history into *medical knowledge* where trends of a patient can be detected improving medical practice. Learning algorithms are currently used in many applications. During electronic transactions in major webpages, there is usually a learning algorithm that checks whether a credit card is not being used in fraudulent transactions by comparing the habits of the user. When using online stores or movie services on internet, commonly there is a learning algorithm, named *recommender*, that learns the preferences of the user and recommends possibly interesting products to the user. The core of machine learning deals with: representation and generalization. Representation of data samples into different spaces with particular properties and the functions to evaluated these samples are part of every machine learning algorithm. Generalization is the property that the algorithm will perform well in unseen data instances; the conditions under which this can be guaranteed are a key object of study in the subfield of computational learning theory.

## 1.1. MACHINE LEARNING

There are two major classes of learning for these kind of algorithms. **Supervised learning** is the task of inferring the underlined rules of a particular problem from data that is already labeled. A simple example is given by a house seller who wants to know the approximate price of a house. Let us suppose that we have access to a database of houses of a certain area where prices are available according to their size in square meters. We could simply generate a regression (linear, polynomial, etc.) to estimate the price of a new house. This is a supervised learning process because we already know the label (prices) of the data (size of a house) for some samples. This kind of problem falls in what is called a *regression problem*. There is another kind of problem named *classification problem* where usually the variable we want to predict is discrete rather than continuous. Imagine the case we collect data about breast cancer and we want to relate the size of a tumor to its kind, malignant or benign. In this case we are trying to classify a tumor according to its size making this a classification task. However, other variables can also be used. For instance, one may choose to include the age of the patient or other characteristics of the patient. Then, our data can lie in a 2, 3, or larger finite dimensional space. The reader might wonder: what if the set of features lie in an infinite dimensional space? Some methods in machine learning actually expand the finite set of features of the data into an infinite dimensional space in order to discover the patterns that describes best the data.

As mentioned above, in supervised learning algorithms the labels or real answers of a problem are known. In the **Unsupervised learning** class, we have access to a database that has not been labeled, i. e. the right answers of our problem are unknown. Then we ask our algorithm to find structure in the data. This class of learning algorithm is useful, for instance, in image segmentation. For this example the algorithm can cluster pixels that have similar properties and, as an answer, it can return a contour map of objects in the image without or with little intervention of a user.

A machine learning algorithm is usually divided as follows:

- A feature selection process to select a set of representative data. This step is extremely dependent on the problem to be solved. The set of features must be representative of the underlying phenomenon. Here, the old computer science adage "garbage in, garbage out" could not apply more strongly. If the training data is not representative the learning algorithm might be useless.
- A kernel method to process the data. Sometimes a simple linear transformation will be enough. In other cases, more complicated methods are required to capture the patterns in the set of features. This step helps the algorithm to separate the features in order to be classified. The most typical transformation is given by placing a neural network with a sigmoid

## CHAPTER 1. INTRODUCTION

transformation or a self organized feature map (SOFM) network. Some of these methods include a dimensional expansion of the features set.

- A learning process that represents an optimization problem. It takes the transformed data and tries to infer the underlying dynamics that describes it. There are many types of these algorithms, such as linear classifiers (e.g. linear or logistic regression, naïve Bayes classifier, perceptrons, support vector machines (SVM), among others), quadratic classifiers, K-means clustering, genetic algorithms, decision trees, neural and bayesian networks, etc.

These three ingredients determine the properties of a learning algorithm. In the next section we focus the attention on two kernel methods: Artificial Neural Networks (ANN) and Reservoir computing. The latter derives from the former when recurrence is added to the network. Then, we will focus on the last of these ingredients: the learning process. Finally, some ways of evaluating the performance of a learning algorithm will be discussed.

### 1.1.1 Kernels to process information

#### Artificial Neural Networks

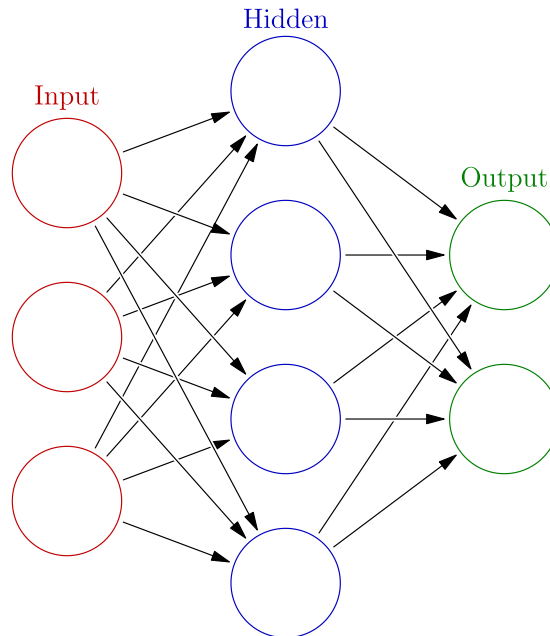
This kind of networks is inspired by biological neural networks, e.g. based on the function of the central nervous system of animals and, in particular, the brain. An artificial neural network is composed of a large number of interconnected elements that are called neurons (to sustain the analogy). There is no single formal definition of an ANN. However, we could call ANN to the class of statistical models that consists of sets of adaptive weights, i.e. numerical parameters that are tuned by a learning process, and are capable of approximating non-linear functions of their inputs. Recognizing faces, handwriting characters, trends and patterns are typical tasks for ANNs thanks to their adaptive nature (plasticity). Importantly, neural networks and conventional computing are not in competition but in complement with each other, e.g. arithmetic operations are more suited to conventional computing and normally conventional computing is used to supervise neural networks. There is a large number of tasks that requires algorithms that use a combination of these two approaches in order to perform at maximum efficiency. Fully automated ANNs have a disadvantage: their results can be sometimes unpredictable because they find out how to solve the problem by themselves.

An ANN is usually represented by a set of inputs connected to some processor elements (neurons) which are also connected to an output set of neurons. The set of inputs are known as the *input layer* and the set of outputs as the *output layer*.



## 1.1. MACHINE LEARNING

The set of connected neurons that are not in any of these layers are organized in what is called the *hidden layer* as represented in Figure 1.1. The aim of a learning process is thus to compute the importance (weights) of the links among neurons.



**Figure 1.1:** Schematic representation of an ANN by [4].

A change of paradigm came with the introduction of the idea of connecting neurons among themselves in the hidden layer. These neurons could create a cycle introducing recurrence in the network and therefore they are named **recurrent neural networks (RNN)**. RNNs suffer from training difficulties since they are highly nonlinear, require a lot of computational power, and the training algorithm not necessarily converges. Exactly this problem is avoided in the recently introduced concept of **reservoir computing**, where only the connections from the network to the output layer are trained and computed. Using this procedure the training problem can be solved by a linear learning process.

### Reservoir Computing

A neuro-inspired concept of machine learning named **reservoir computing (RC)** has changed the way ANNs are implemented. In 1995, Buonomano and Merzenich [5] presented a framework for neural computation of temporal and spatiotemporal information processing. Their approach included a hidden random recurrent network, which was left untrained. Then the problem was solved by a simple classification/regression technique. Even though the term of **reservoir computing** was not introduced, this work contains the main ideas behind it.

## CHAPTER 1. INTRODUCTION

The concept of reservoir computing was developed independently under two approaches: [Echo State Network \(ESN\)](#) [6] and [Liquid State Machine \(LSM\)](#) [7]. Based on these approaches, we can understand a reservoir computer as a [RNN](#) where the connections among neurons are fixed and the weights of the output neurons are the only part of the network that can change and be trained.

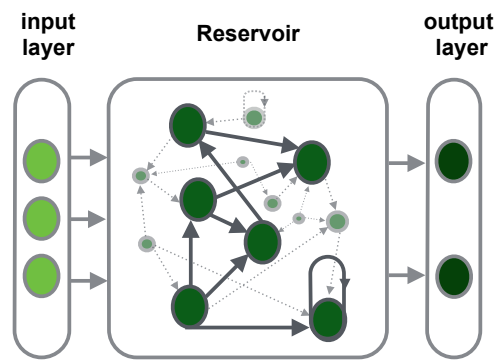
Usually this kind of [RNNs](#) or reservoir computer consists of a large number ( $10^2 - 10^3$ ) of randomly connected nonlinear dynamical nodes, giving rise to a high-dimensional state space of the reservoir. The dynamical nodes or artificial neurons usually have a transfer function with a hyperbolic tangent shape. However, in recent years, novel approaches are being considered, using different nonlinearities and coupling configurations, such as delay-coupled nonlinear systems [8], photonic crystal cavities [9], or the Mackey-Glass oscillator [10]. In all cases, the reservoir serves as a core (machine learning kernel) element for processing information.

The procedure is as follows: input signals, usually low-dimensional, are injected into the reservoir through an input layer, as illustrated in [Figure 1.2](#). The connections from the input layer and the nodes of the reservoir are assumed to have random weights. Via the reservoir, the dimension of the signal is expanded proportionally to the number of nodes. The input signal provokes transient dynamics in the reservoir that characterizes the state of the neurons. The readout process, i.e. the process that reads the response of the network to the input signal, is usually evaluated through a linear weighted sum that connects the reservoir nodes to the output layer. The evaluation of the processed data in the reservoir is possible due to the nonlinear projection of the input signal onto a high-dimensional state space created by the multiple nodes of the reservoir.

A reservoir has to fulfill some properties in order to perform a task properly. One of the most important properties is consistency [11], where the system responses must be sufficiently similar for almost identical input signals. This is also known as the approximation property. However, if the input signals belong to different classes, their transient states must sufficiently differ (separation property). These two properties are complemented by a short-term (fading) memory, created by the recurrence of the network, that becomes handy when the input information is processed in the context of past information, like in a time series estimation task [8].

[Reservoir computing](#) is generally very suited for solving temporal classification, regression or prediction tasks where very good performance can usually be attained without having to care too much about the reservoir parameters [12, 13].

## 1.1. MACHINE LEARNING



**Figure 1.2:** Schematic representation of a reservoir. The information to be processed is received by the input layer and sent to the reservoir where it is projected onto a high-dimensional state space. The dynamical response of the reservoir is readout in order to set the weights of the connections between the reservoir and the output layer, which is in charge of performing the estimation task.

### 1.1.2 Learning process

There are many ways of *teaching* an algorithm to perform an estimation task. Some methods can be used for supervised learning while others are used in unsupervised learning only. This section does not pretend to give an exhaustive list of learning methods. Instead, it illustrates the general idea behind a learning process.

Learning can be defined as the acquisition, modification or reinforcement of knowledge or skills through experience, study or by being taught. Not only humans have the ability to learn, but also animals and most recently machines.

For machines, a learning process can be translated to an optimization problem. The aim is to find the best fit and most parsimonious, efficiently interpretable model that describes the relationship between a dependent variable (labels) and a set of independent variables (the input) [14].

There are many mathematical methods to solve this kind of problems, and the choice depends on the knowledge we can have about the relationship between the variables. Most of machine learning methods are based on statistics. In the following section we will introduce a simple but yet powerful method.

### Linear regression models

The most simple but yet powerful learning process is a linear regression analysis. In a regression model (machine learning kernel), one solves the problem of

## CHAPTER 1. INTRODUCTION

relating a set of independent variables  $\mathbf{X}$  to a dependent variable  $Y$  through a function  $g$  and parameters  $\boldsymbol{\beta}$ . Mathematically, it can be expressed as

$$Y \approx g(\mathbf{X}, \boldsymbol{\beta}), \quad (1.1)$$

where the bold letters refer to vectors. The above approximation is formalized as  $E(Y|\mathbf{X}) = f(\mathbf{X}, \boldsymbol{\beta})$ , where the form of  $f$  must be specified. The selection of the function  $f$  is usually based on *a priori* knowledge about the relationship between dependent and independent variables.

A good estimation of the regression model depends on the length  $k$  of the parameter vector  $\boldsymbol{\beta}$ . If the number of observed data points  $N$ , of the form  $(Y, \mathbf{X})$ , are less than  $k$  the system is underdetermined. In case that  $N = k$ , and  $f$  is linear, then equations 1.1 can be solved exactly (via normal equations) rather than approximately. Mathematically the problem is solvable and the solution is unique. In the last case where  $N > k$  the system is overdetermined. This means that there are several solutions what leads to estimate a unique value for  $\boldsymbol{\beta}$  that best fits the data.

### Normal equations

Let  $f$  be a linear function. We could solve the equations 1.1 by rewriting the system as

$$(\mathbf{X}^T \mathbf{X}) \hat{\boldsymbol{\beta}} = \mathbf{X}^T \mathbf{y}, \quad (1.2)$$

where  $\hat{\boldsymbol{\beta}}$  represents the vector of the optimal parameter values. If the matrix  $\mathbf{X}^T \mathbf{X}$  is invertible, the problem can be solved explicitly. In numerical computation, if the determinant of the  $\mathbf{X}^T \mathbf{X}$  matrix is close to zero, then the problem would be probably not solvable. We can introduce some methods to make it solvable. They are known as regularization methods.

### Regularization

In mathematics a regularization method is the process of introducing information in a system in order to solve an ill-posed problem. In machine learning the main idea of regularization is to prevent overfitting of the data (see Section 1.2.1). Besides, some regularization methods can also be used as a feature selection process.

First introduced in statistics by Hoerl and Kennard [15], ridge regression is a regularization method that helps in the solution of normal equations when the matrix  $\mathbf{X}^T \mathbf{X}$  is singular. It is written as

$$\hat{\boldsymbol{\beta}}^{ridge} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y}, \quad (1.3)$$

## 1.2. DIAGNOSING LEARNING ALGORITHMS

where  $\lambda \geq 0$  is the regularization parameter: the larger the value of  $\lambda$ , the greater the regularization.  $\mathbf{I}$  is the  $k \times k$  identity matrix. Ridge regression can also solve problems when matrix  $\mathbf{X}^T\mathbf{X}$  is not of full rank, i.e. a square matrix. Another use of ridge regression, more relevant in machine learning, is as a feature selector. This regularization method imposes a penalty on regression coefficients size forcing them to be small. We can write the residual sum-of-squares in ridge regression as

$$RSS(\lambda) = (y - \mathbf{X}\beta)^T(y - \mathbf{X}\beta) + \lambda\beta^T\beta. \quad (1.4)$$

The rightest term in Eq. 1.4 is a quadratic penalty, denoted by  $\beta^T\beta$ , and it is the responsible of the penalization. The penalization term is generalized in a regularization method named Lasso ( $L_q$ ) that includes a parameter  $q$  to control the degree of the penalty. For  $q = 1$  the penalty is linear and it is known as L1 regularization. For  $q = 2$  or L2 regularization, the penalty is quadratic, as in Eq. 1.4. The parameter  $q$  can take non-integer values producing what is called an elastic net [16]. An intuitive and mathematical-elegant explanation about regularization and feature selection given by Hastie et al. can be found in ref. [17, p. 57-93].

1.2

---

## Diagnosing learning algorithms

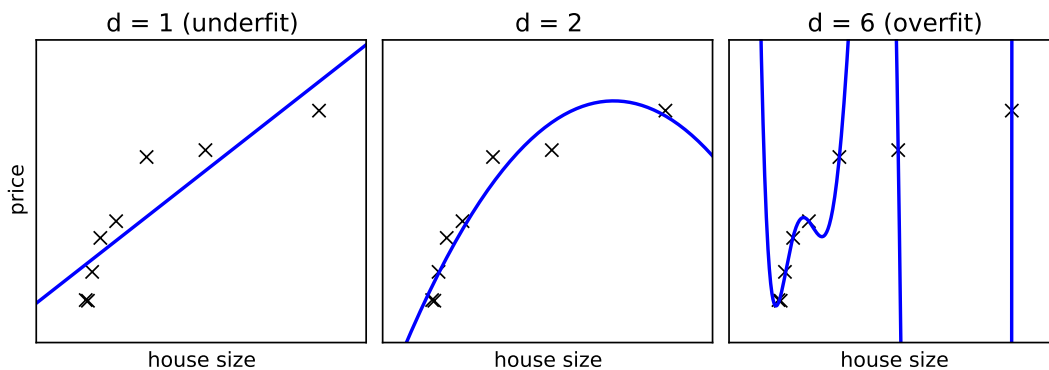
In practice, a big task in [reservoir computing](#) and more general in machine learning involves selecting nonlinearities, parameters and sets of data to optimize the overall results. These features may affect the quality of the results. For example, when having an error rate larger than expected, one can think that increasing the training set will lead to better results. However, if the model is suffering a high variance problem more samples will not improve results. In this section an overview of different scenarios and measurements are presented for the recognition of problems in the model. The Scikit-Learn Documentation [18] follows an interesting approach to explain diagnosing. In this section we will follow some parts of their approach.

### 1.2.1 Model complexity, bias and variance

In this thesis we will refer to the complexity of a model (machine learning kernel) as its degree of nonlinearity. Thus the simplest model is a linear model. From there we can build more complex estimators using polynomial models, trigonometric models, and other nonlinear functions. For clarity, let us illustrate the effect of the model function over the results using polynomial models. Let us recall the example mentioned in the introduction (Section 1.1) regarding the

## CHAPTER 1. INTRODUCTION

price of houses. For this example a house seller wants to know the estimated price of a new house according to its size in square meters. In order to know the estimated price of a new house we perform a linear regression using different models. We will use here a linear, a quadratic and a degree-6 polynomial function. The dataset is shown in Figure 1.3 as crosses and the continuous lines represent the best fit of each model.



**Figure 1.3:** Linear regression for polynomial models. On the top of each panel, the value of  $d$  shows the degree of the polynomial. The data is represented as crosses and is the same in each plot. Continuous lines represent the best fit for each model.

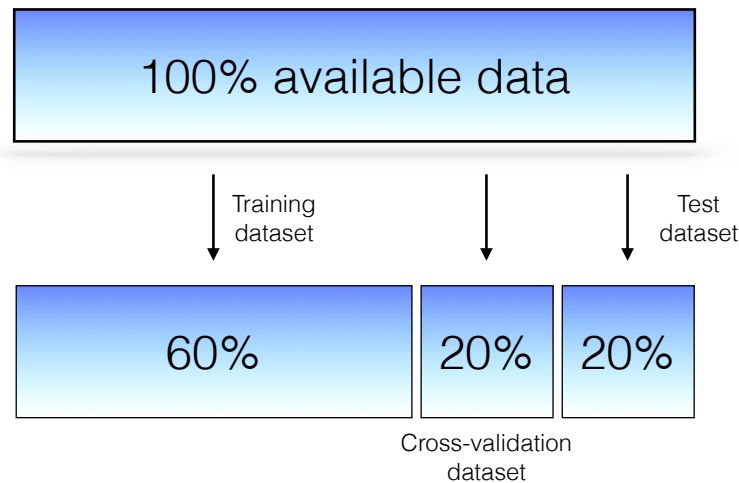
In this figure we notice that a linear model is not the best option for the prediction. This model under-fits the data meaning that the model is too simple. In machine learning vocabulary this is known as a **high bias** problem. The model itself is biased and this is reflected as a poor fit of the data. At the other extreme is the degree-6 polynomial function exhibiting a very accurate fit. This model *touches* each sample fitting the data perfectly and causing a problem known as overfitting. Thus, this is not a good feature of the model since it lacks the ability to generalize to new samples. This problem is known as **high variance**. The middle panel seems to be the mid-point between these two types of problems. One may wonder how to recognize these problems and the mid-point where a model can perform best. To quantitatively identify bias and variance and therefore optimize parameters we need to go through a process called cross-validation.

### 1.2.2 Cross-validation

In the previous section we recognized the problem of high variance (overfitting) in a dataset. This set of data is known as the *training data* because it was the data used by the model to build the estimator. If we now compute the error rate of

## 1.2. DIAGNOSING LEARNING ALGORITHMS

the overfit model, i.e. the degree-6 polynomial function, we will get zero error. However if we want to know the price of a new house using panel c of Fig. 1.3, the new price can be very different of the rest of the dataset. Therefore the training error is not a good indicator of the performance. To avoid this problem, the dataset should be divided into three smaller datasets that we will call: the training dataset, the cross-validation dataset and the test dataset, see Figure 1.4.



**Figure 1.4:** Scheme of a possible data partitioning to avoid optimistic results.

The reason to split the data into three datasets come from the fact that we can overfit the data at different levels. The model parameters are learned using the training data. In our example, these parameters are the coefficients of the polynomial function. Once the model is trained, we evaluate the error using the cross-validation dataset and we choose the meta parameters, i.e. the degree of the polynomial function. As with the training, we could be overfitting the data respect to the meta parameters, thus the minimal cross-validation error tends to under-estimate the expected error on a new set of data. Then, the test dataset is the only one that the model has not seen and is the one used to evaluate the final model. Note that the test dataset is not used to optimize any parameter of the model but to evaluate its performance once all parameters are fixed.

By partitioning the available data into three different sets, we drastically reduce the number of samples which can be use for training the model. However, the results can depend on a particular random choice for a pair of (train, validation) dataset. A solution to this problem is the basic approach of **cross-validation (CV)**, known as **k-fold CV** where the training set is split into  $k$  smaller sets. The procedure is as follows:

1. The learning algorithm is trained using  $k - 1$  folds,

## CHAPTER 1. INTRODUCTION

2. then, the resulting model is evaluated using the rest of the data.
3. The process is repeated by taking out a different fold of the  $k$  folds for the testing.

The performance of the  $k$ -fold CV is the average of the performance measure computed in the loop. This approach can be computationally expensive, but does not waste much data.

There are other approaches to CV, however they follow the same principles. For example **Leave-One-Out (LOO)** is a simple cross-validation where each learning set is created by taking all the samples except one, the test set being the sample left out. Thus, for  $n$  samples, we have  $n$  different training sets and  $n$  different tests set. This cross-validation procedure does not waste much data as only one sample is removed from the training set. When we compare LOO with  $k$ -fold CV, one builds  $n$  models from  $n$  samples instead of  $k$  models, where  $n > k$ . Moreover, each model is trained on  $n - 1$  samples rather than  $\frac{(k-1)n}{k}$ . In both ways, assuming that  $k$  is not too large and  $k < n$ , LOO is computationally more expensive than  $k$ -fold cross-validation.

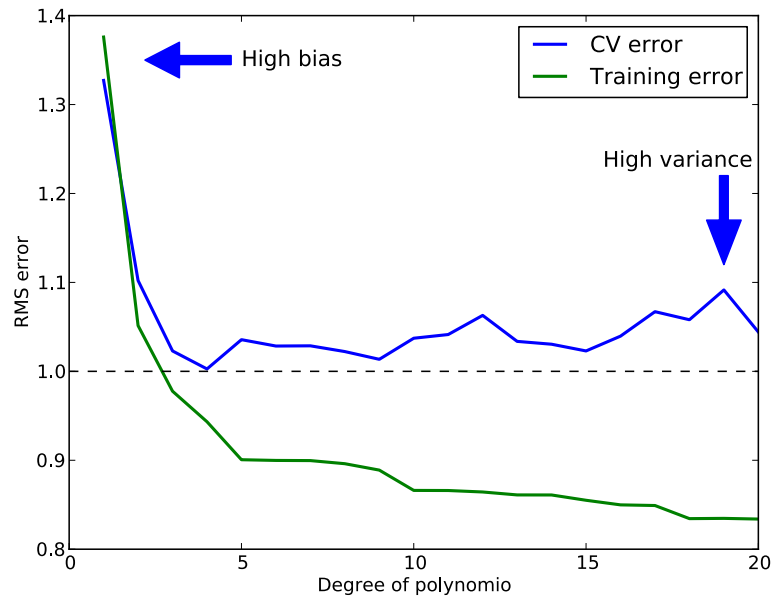
A version of leave-one-out CV is the leave- $p$ -out CV that creates all the possible training-testing datasets by removing  $p$  samples from the complete set.

With a cross-validation method, we can now choose the complexity of our model. Bringing back the example of the house seller from the previous section, the 20-fold cross-validation error of our polynomial classifier can be plotted as a function of the polynomial degree  $d$ . Figure 1.5 shows the reason why cross-validation is important. On the left side of the plot, we have very low-degree polynomial, which under-fits the data. This leads to a very high error for both the training set and the cross-validation set. On the far right side of the plot, we have a very high degree polynomial, which over-fits the data. This can be seen in the fact that the training error is very low, while the cross-validation error is very high. Choosing  $d = 6$  in this case gets very close to the optimal error.

It might seem that something is amiss here: in Figure 1.5,  $d = 6$  gives good results, but in Figure 1.3, we found that  $d = 6$  vastly overfits the data. The difference is the number of training points used. For Figure 1.3, there were only eight training points. In contrast for Figure 1.5, we have 100. As a general rule of thumb, the more training points used we use, the more complicated the model can be. But how can we determine for a given model whether more training points will be helpful? A useful diagnostic tool for this are the so called learning curves.



## 1.2. DIAGNOSING LEARNING ALGORITHMS



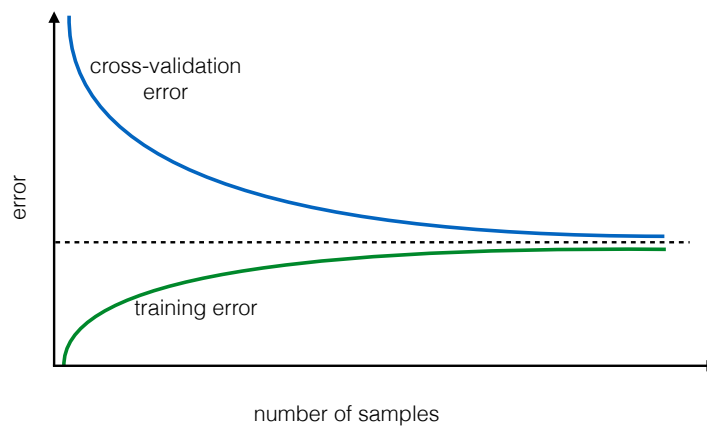
**Figure 1.5:** Model complexity. This figure shows the 20-fold cross-validation error (blue line) and the training error (green line) as a function of the polynomial degree  $d$  of the model. For low degree, the system suffers high bias problem. In contrast, for high degree the model suffers high variance problem. The horizontal dashed line serves as a guide to a possible intrinsic error.

### 1.2.3 Learning curves

A learning curve is a graphical representation of the increase in learning as a function of experience. The concept was first used in psychology of learning by Ebbinghaus in 1885 [19, 20], although the name was not used until 1909. The plot of a learning curve depicts improvement in performance on the vertical axis when there are changes in another parameter (on the horizontal axis), such as training set size (in machine learning) or iteration/time (in both machine and biological learning). Then in machine learning, a typical learning curve shows training and cross-validation (CV) error as a function of the number of training samples. Note that when we train with a small subset of the training data, the training error is computed using this subset, not the full training set. These plots can give a quantitative view into how beneficial will be to add training samples.

Let us describe what is happening and how to interpret a learning curve. Regarding the training error, when the number of samples is one or very small, any model (linear or nonlinear) can fit the data (almost) perfectly. This fact causes the training error to be zero or very small. As the number of samples in the training

## CHAPTER 1. INTRODUCTION



**Figure 1.6:** Typical learning curve.

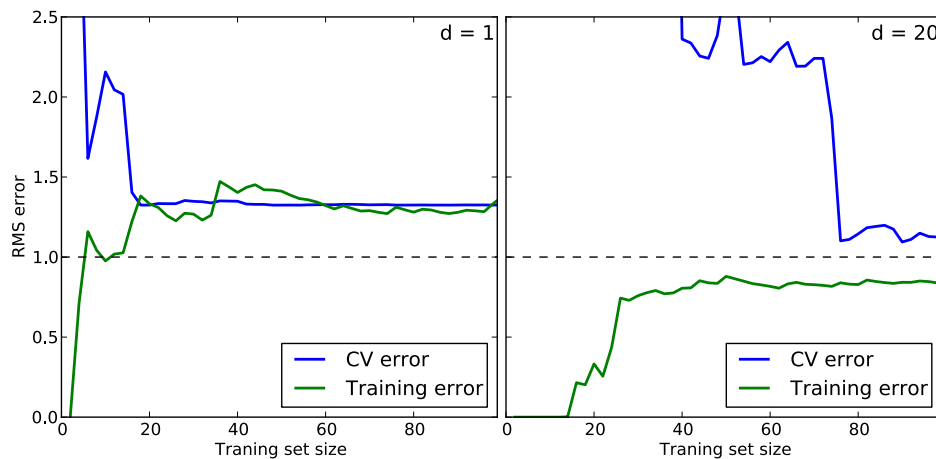
set increases it becomes more difficult to fit all the points in the training set raising the training error. Eventually the training error will flatten once the number of training samples is enough to learn the patterns in the training dataset. In contrast, the cross-validation error is expected to be big for small number of samples because the parameters of the model are very inaccurate (they were trained using only one or few samples). As the number of samples increases, the parameter set of the model gets more accurate and the cross-validation error decreases until it flattens as the training error does.

Figure 1.6 shows the expected shape of a learning curve. For small number of samples, the training error is minimal while the cross-validation error is maximal. As the number of samples in the training set increases, the two errors tend to flatten at a certain value that is determined by the task and the bias and variance of the model.

To elaborate this last point, we come back to our example of the house seller. In Figure 1.5 we showed that using a hundred samples, a linear model shows high bias (underfitting) problems. In the learning curve this problem is recognized because the training and cross-validation errors converge very rapidly at a relative high error. We can see this behavior in the left panel of Figure 1.7. If we continue adding samples to the training set, it is unlikely that the situation changes. The two errors have converge to a certain value and they become independent of the number of samples.

In the right panel of the same figure, we have the opposite case, a high variance problem given by a degree-20 polynomial model. The characteristic features of high variance (overfitting) is the gap that exists between the training and the cross-validation errors. If we increase the number of training samples it is likely that the gap reduces causing the errors to converge in the middle point.

## 1.2. DIAGNOSING LEARNING ALGORITHMS



**Figure 1.7:** Learning curves depicting high bias (left panel) and high variance (right panel).

### Small note about floating-point precision of a machine

The theory of machine learning is based on statistics and basic mathematical operations and most of them can be explicitly solved. However, in practice we usually end up performing those operations using a computer. But there are limitations that we have to be aware of when using a numerical algorithm.

For a training set with  $N$  samples, if  $N \leq d$ , being  $d$  the degrees of freedom of a model, the system of equations is perfectly solvable. This means that we can expect to have a perfect fit (zero error) between the data and the predicted model. The right panel of Figure 1.7 shows a degree-20 polynomial function. For  $N \leq 20$  the model should be solved exactly, however we can see non-zero errors before we reach the number of degrees of freedom of the model. This does not mean that the figure is wrong, in fact, the resulting fit has small residuals because it needs very large oscillations to fit all the points perfectly, similar to  $d = 6$  case in Figure 1.3.

### 1.2.4 Course to follow for high bias or high variance problems

We have seen in this section that there are several tools to diagnose a learning algorithm. All these tools can be applied to the particular case of [reservoir computing](#) in order to evaluate, diagnose and improve performance. Here we present some actions that can be taken when a high bias or a high variance problem is found. When having high bias problems, we can:

## CHAPTER 1. INTRODUCTION

- Add more features. In our example of predicting house prices, including not only the size of the house but the year it was built, the neighborhood, and other features may help to a high-biased estimator.
- Increase the complexity of the model. As we studied in section 1.2.1, for polynomial models we can increase the degree of the polynomial function to add complexity. Other kind of models will have their own methods for adding complexity.
- Decrease regularization. Regularization is a technique used to impose simplicity in some machine learning models by adding a penalty term that depends on the characteristics of the parameters. If a model has high bias, decreasing the effect of regularization can lead to better results. Refer to Section 1.1.2 for more information.
- Use a smaller training set. This is more an advice than a guidance. Reducing the number of training samples will probably not improve the performance of the estimator since a high-biased model will keep the same error for smaller training datasets. However, for computationally expensive algorithms, reducing the training set can lead to improvements in computational speed.

In contrast, if an algorithm is suffering a high variance problem, some steps we can follow are:

- Use fewer features. Using a feature selection method may decrease the overfitting of the estimator.
- Increase the training set. Adding samples can help to reduce a high variance problem as mentioned in section 1.2.3.
- Increase regularization. Increasing the influence of the regularization parameter on the model may help to reduce overfitting. This term is intended to avoid exactly this problem. See Section 1.1.2.

Up to this point of this thesis we have studied how to *teach* a learning algorithm such as artificial neural network or [reservoir computing](#), to perform a task. We have also seen what we can do to optimize and diagnose problems in our model. We may wonder now how to quantify the *goodness* of a model, and how its performance is compared to other approaches. In the following section, we explore different measures to compare a model.

## Evaluation of performance

There are many measures of performance that have been traditionally used in machine learning. None of them being *the* measure that correctly shows how good or bad a model performs. These measures highlight the goodness or badness of some of model's features. In this section, we introduce and discuss the pros and cons of some of the available measures of performance. This is not an exhaustive list since each kind of problem may have a particular set of measures that fits better for its evaluation. For example, measuring the performance of a time series estimator will use a different set of measurements than evaluating a classification algorithm.

We will now describe measurements commonly used for classification tasks. Most of these measurements are based in a statistical tool named contingency table that can display the uni- or multi-variate frequency distribution of the variables. In machine learning this table is known as error matrix or confusion matrix. The latter will be the preferred term used in this thesis and it comes from the fact that this matrix makes it easy to recognize if a model is confusing two classes.

### 1.3.1 Confusion matrix

In a classification task, a confusion matrix is a square matrix that allows to visualize the performance of an algorithm. In supervised learning we know the true labels of the test data, so a model has to estimate to which class each test sample belongs to. Having these two sets of labels, the real labels and the estimated ones, we can build a matrix that in its columns contains the estimation of the model while in the rows it has the ground truth.

Figure 1.8 depicts a confusion matrix. To understand how it works, let us consider a 2-class classification task. One of the classes will be named the positive class, while the other the negative class. In a confusion matrix when a positive sample is classified by a model as such, it is counted as a **True Positive (TP)**. In contrast, if the same sample is classified as negative then we have a **False Negative (FN)**. A similar reasoning applies for a negative sample. If classified as positive, it is a **False Positive (FP)**, and if classified as negative, it becomes a **True Negative (TN)**. As a mnemonic technique we can read these terms using: *it is \_\_\_ that is \_\_\_*, i.e. a false negative can be read as "it is *false* that is *negative*", meaning that the real class of that sample is the positive class.

False positive and false negative are, respectively, what we call type I and type II errors in statistical test theory.

## CHAPTER 1. INTRODUCTION

		Predicted values	
		positive	negative
Real values	positive	True positive	False negative (Type II error)
	negative	False positive (Type I error)	True negative

**Figure 1.8:** Confusion matrix for a binary classification task.

Thus, from a confusion matrix we can derive a set of measurements that take into account different aspects of a classifier. Let us study some of these measures in the following sections.

### Sensitivity

Sensitivity ( $Se$ ), also known as recall or true positive rate ( $TPR$ ), represents the proportion of real positive cases correctly classified as positive. It is defined as

$$Se = \frac{\text{true positive}}{\text{true positive} + \text{false negative}} \times 100. \quad (1.5)$$

To give some intuition about the meaning of sensitivity, let us consider a population of subjects. Some of them are infected with a virus, and the rest are completely healthy. Imagine that a company designs a blood test to identify the infected subjects. The new blood test may report a sensitivity of 100% meaning that all infected subjects are correctly identified. If the test has zero sensitivity, it means it is unable to recognize any infected subject. This measure is very important in medicine because it allows to correctly identify medical conditions in patients. However it has to be taken carefully: If the test identifies not only all infected subjects but also some (or even all) of the healthy subjects as infected it will still report 100% sensitivity. This is because sensitivity accounts for the positive cases disregarding the errors of classifying a healthy subject as infected, i.e. the false positive cases. Then another measure is needed to quantify those healthy subjects incorrectly classified as infected.

### 1.3. EVALUATION OF PERFORMANCE

#### Precision

Precision ( $Prec$ ) or confidence denotes the proportion of predicted positive cases that are actually real positives. It can be expressed as

$$Prec = \frac{\text{true positive}}{\text{false positive} + \text{true positive}} \times 100. \quad (1.6)$$

This measure solves the problem of misclassifying healthy subjects as infected. It complements sensitivity in the characterization of a classifier. So, a test report should at least show two performance measures to be meaningful. Note that precision still suffers the same drawback as sensitivity. It does not take into account those infected subjects that were not detected by the test, i.e. the false negatives.

#### Specificity

Specificity ( $Sp$ ), also known as inverse sensitivity, inverse recall or true negative rate (TNR), is the proportion of real negative cases that are correctly classified. Specificity can be written as

$$Sp = \frac{\text{true negative}}{\text{false positive} + \text{true negative}} \times 100. \quad (1.7)$$

Intuitively, if all the healthy subjects are correctly classified as not infected, the blood test will have specificity of 100%. Note that if some (or all) of the infected subjects are classified as healthy, the blood test still will be a 100% specific.

There is a complementary measure from specificity called **False Positive Rate (FPR)** that is defined as

$$FPR = \frac{\text{false positive}}{\text{false positive} + \text{true negative}}. \quad (1.8)$$

#### Accuracy

Accuracy ( $Acc$ ) is the combination of the previous mentioned measures. It is defined as,

$$Acc = \frac{\text{true positive} + \text{true negative}}{\text{sum of all samples}}, \quad (1.9)$$

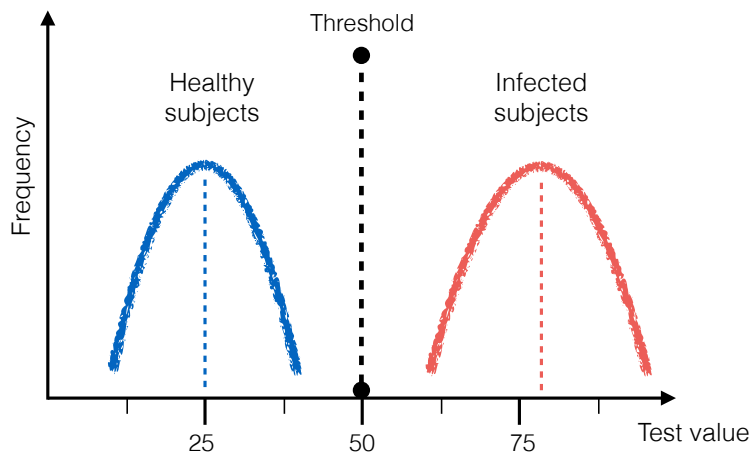
It can be seen as a weighted average between sensitivity and specificity [21].

Sensitivity and specificity are linked such that changing the sensitivity affects the specificity of a test. In the following sections we will describe how they interact.

## CHAPTER 1. INTRODUCTION

### 1.3.2 The trade-off between sensitivity and specificity

We will illustrate the trade-off between these measures using an example. We have again a blood test to discriminate healthy from infected subjects. Imagine that the result of the test is a counter, such as the concentration of a certain substance in a certain volume of blood. We decide to evaluate the test over a population that is healthy. Thus we are going to have a certain distribution of test values for those healthy subjects. If we now perform the test over an infected population, we will then have the corresponding distribution of test values for infected subjects. Figure 1.9 depicts this situation.



**Figure 1.9:** Distribution of test results in a healthy (blue) and infected (red) population.

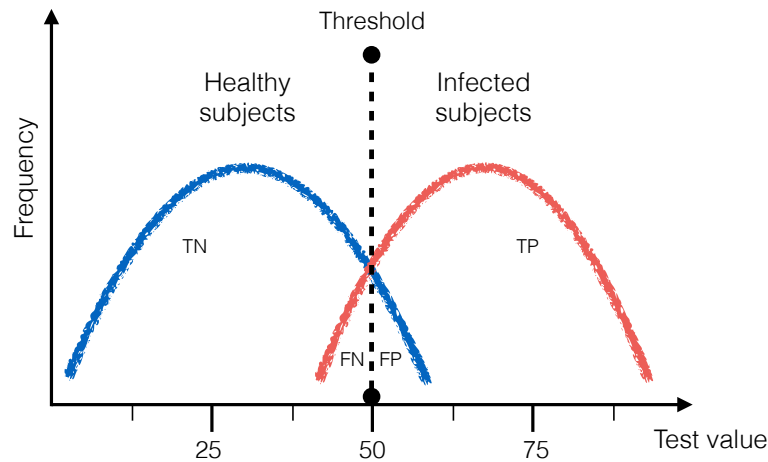
Let us say that small concentrations of our substance is a healthy indicator, and a high concentration means infection. Now, the problem we are facing is to decide from up to which point the test is going to be considered negative and from which point it is going to be considered positive. This can be done by setting a threshold value that separates the two distributions. In this example, it looks natural to set that threshold at a value of 50 since our two distributions do not overlap.

Now consider a more interesting scenario where the two distributions overlap, as shown in Figure 1.10. For this case, setting a threshold is a more difficult task. To begin with, let us keep the threshold in the middle, as we did in the previous figure. Now, to identify the true/false positive/negative values in this scheme we consider one distribution curve at a time. We start considering only the distribution of the infected subjects. On the right of the threshold line are those concentrations that we consider positive and come from infected subjects, so these are true positives (TP). On the left of the threshold line are those concentrations that we consider negative (because they are in the left side of the



### 1.3. EVALUATION OF PERFORMANCE

division line) but are actually positive, i.e. they are in the positive distribution curve. Then these cases represent the false negatives (FN). If we now consider the distribution for the healthy subjects, we can identify the true negatives (TN) as those concentrations in the left of the threshold line and as false positives (FP) those on the right of the threshold line.



**Figure 1.10:** Overlapping distribution of test results in a healthy (blue) and infected (red) population.

So sensitivity is the proportion of the area under the positive distribution curve that is at the right of the threshold line with respect to the area under the full positive distribution curve. Note that if we move the threshold line to the left, the sensitivity will increase but it also will include more false positives. In the extreme case, if we set the threshold line all the way to the left, we will get a 100% sensitivity but our test will lack the ability to distinguish healthy from infected subjects, i.e. it will classify every subject as infected.

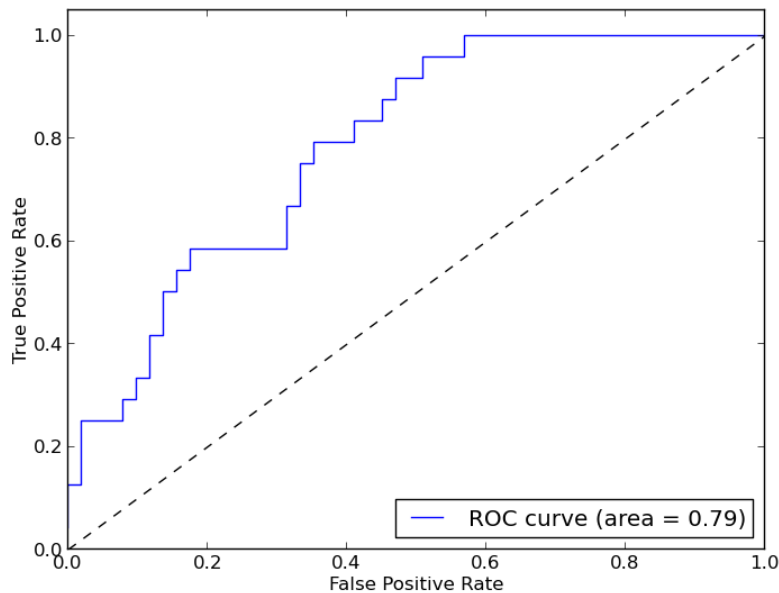
On the opposite side, specificity is the proportion of the area under the negative distribution curve to the left of the threshold line with respect to the area under the full negative distribution curve. Again, we can increase this proportion by setting the threshold line to the right. However this will increase the amount of false negatives.

A similar relationship between sensitivity and precision, or any confusion matrix derived measure can be established. In conclusion, one has to set a threshold value for a given test in order to characterize it. This threshold value is going to define the sensitivity and specificity of the test. There is no general rule to choose this threshold. It depends on the task and the peculiarities of the estimator. One way to visualize the general properties of an estimator using the above mentioned trade-off is the so-called receiver operating characteristics (ROC) curve.

## CHAPTER 1. INTRODUCTION

### 1.3.3 Receiver operating characteristics curve

A **Receiver Operating Characteristics (ROC)** curve is a statistical tool to visualize the performance of a classifier when the threshold that defines the classes is varied. It is based on the trade-off between sensitivity and specificity (section 1.3.2). To build a **ROC** curve, we need to plot the sensitivity of the classifier as a function of the false positive rate (**FPR**, shown in equation 1.8) for different values of the discriminating threshold.



**Figure 1.11:** A ROC curve with the line of nondiscrimination. Each step in the curve corresponds to a value of the threshold that separates one class from the other. Image by [18].

Figure 1.11 depicts the **ROC** space. The best classifier will have coordinates  $(0,1)$ , representing a 100% sensitivity and a 100% specificity. This point is also called *perfect classification*. Departing from this point, the performance of our classifier describes a curve. The identity line, a straight line that connects the  $(0,0)$  point with the  $(1,1)$  point, is called the *line of nondiscrimination* and represents a random guess. Curves above this line are considered to exploit information from the data for a good classification. Curves overlapping the line of nondiscrimination do not contain information about the classes and make the classification randomly. Any curve under this line performs worse than a random classifier. In this case, the classifier contains information about the classes but it uses this information incorrectly [22]. Another good property of **ROC** curves is that they are insensitive to changes in class distribution (class skew) [23], i.e. they do not depend on the number of instances in each class (data balance).

## 1.4. OBJECT OF THE RESEARCH WORK

### Area under a ROC curve

ROC curves provide a lot of information for characterizing classifiers. However one can reduce this information in order to give an overall measure of performance. A common quantity that fulfills this criterion is the so-called **Area Under the Curve (AUC)** [24]. A property of ROC curves is that they are drawn in a unitary square, so that the AUC ranges from 0 to 1. If the AUC of a classifier is less than 0.5, it is said that the classifier is unrealistic [23]. If the AUC is 1, it represents the perfect classifier. From this, we could say that the higher the AUC the better the classifier. In the example ROC curve in Figure 1.11 the AUC = 0.79. The AUC should be used to summarize the information on a ROC curve but it should not substitute it since there are classifiers with high AUC that can perform worse in a specific region of the ROC space than a classifier with low AUC [23].

1.4

---

### Object of the research work

The object of this thesis is the study of the computational properties of delay-coupled systems and their applications. To this end, several tasks are tackled:

1. Build a computational toolbox for the numerical study of delay-coupled systems.
2. Implement a delay-based reservoir computer for the study of the computational properties that are intrinsic to this kind of system and are independent of the task to be performed.
3. Describe the working principles of a reservoir computer by studying typical machine learning tasks that are difficult to solve by using conventional computing.
4. Apply the computational power of delay-coupled systems to a real-world open biomedical problem.
5. Study the influence on performance of multivariate inputs in a reservoir computer.
6. Development of a computational framework that allows the tuning of a hardware implementation based on a semiconductor laser dynamics.

The different parts of our main objectives are developed along this manuscript. A description of the structure of this research work is given in the next section.

## Structure of this thesis

This thesis is based on the research developed at the Instituto de Física Interdisciplinar y Sistemas Complejos (IFISC) during the period 2011-2015. The results of the different studies have been published in several international recognized scientific journals. The related articles are enumerated in the *List of Publications*.

This manuscript is divided in three parts. The first part called *Introduction* is composed of one chapter, Chapter 1, and gives an overview of machine learning by focusing on a particular kernel type known as *reservoir computing*. Standard methods for the diagnostics and evaluation of learning algorithms are introduced in this part.

The second part, called *Methodology, results and discussion* contains the core of this thesis. It is composed of 4 chapters. The first one, Chapter 2, describes the principles of information processing using delay-coupled systems. This chapter describes the computational properties of the delay-coupled approach. After this description, results are presented using the proposed scheme for *reservoir computing*, tackling typical machine learning tasks, such as a classification task (spoken digit recognition), a time-series estimation task and a modeling task. Each task's section is self-contained, including its description, aim of the task, results and discussion. In this way, the reader might jump among the different tasks without getting lost. The end of this chapter, presents a short exploration of the reservoir's performance in different conditions other than the typical ones. In particular, the spoken digit recognition task is used in the evaluation of performance of the reservoir with different initial conditions. An evaluation of the performance when reading out a portion of the dynamical states of the system, i.e. reading some virtual nodes, is also presented.

The evaluation of a especially designed reservoir computer for the application in a medical task is presented in Chapter 3. This chapter contains a brief description of the physiology of the cardiovascular system and, in particular, of the heart. The reservoir computer utilizes electrocardiographic signals for the classification of cardiac arrhythmias. The origin of these signals and how they are prepared to be included in our designed computer is described. A particular feature of our approach is the learning method. We exploit the statistical properties of the logistic regression method. This learning procedure is particularly suitable in biostatistical tasks where binary solutions are expected, such as normal or abnormal heartbeat. Results in internationally recognized format for the evaluation of arrhythmia classifiers and for clinically relevant scenarios are presented and discussed at the end of this chapter.

## 1.5. STRUCTURE OF THIS THESIS

The computational power found in the delay-coupled system described in the previous chapters is evaluated using multivariate inputs in Chapter 4. Two tasks are utilized to demonstrate that a reservoir computer might utilize beneficially information from different variables of the system under study. We use as examples: a time-series estimation task based on the Lorenz system in a chaotic regime and the classification task presented in Chapter 3 when using two electrocardiographic signals recorded simultaneously from different sites of the chest.

Numerical implementations of the reservoir computer described in this thesis might be implemented in hardware using optical nonlinearities. Chapter 5 describes the numerical implementations of a semiconductor laser serving as the nonlinear node of a reservoir computer. This chapter shows the expected performance of such reservoir under different conditions allowing to tune specific parameters for a hardware implementation. Only numerical computations are shown in this chapter. The interested reader in the hardware implementation is refer to the corresponding article [25] and references there in.

The final part of this thesis is *Conclusions* part. It is composed of one chapter, Chapter 6. In this part we summarize what we have accomplished along the different parts of this research work. We also present some potential changes in the proposed scheme that we believe are important to understand might be useful for future applications.

At the end of the manuscript, an appendix related with technical descriptions of the electrocardiographic database is presented. Also, a small glossary and a list of the acronyms used in this thesis are described. The bibliography can be found at the very end of this thesis, showing references in order of appearance.



## **Part II**

### **Methodology, results and discussion**





## Delay-based reservoir computing

**Reservoir computing (RC)** [6, 7, 13, 26, 27, 28] is a recently introduced paradigm in machine learning. It is inspired by our understanding of how the brain processes information. For example, Rabinovich et al. [29] stated that the brain would process information by the generation of patterns of transient states in the dynamical activity of neurons when they are excited by a sensory stimulus. Those transients, that come from billions of neurons, organize in a way that the brain can deliver a response to the stimulus in fractions of seconds. The aim is then to extract some information about the stimulus by using the transient dynamics that it provokes. For example, a **doppler radar** can determine the location and velocity of storms by measuring the perturbations that they provoke in the environment. These radars use the Doppler effect to produce velocity data about objects at a distance.

As mentioned in the introduction, traditional **RC** is a kind of **recurrent neural network** whose configuration is divided into three layers as shown in Fig 1.2. Typical **RNNs** require the weights of the connections between the nodes to be computed and adjusted for different tasks, which usually is a very time-consuming computational task. Therefore **RNNs** were not very attractive, from the computational perspective, until the advent of **reservoir computing** in which the weights of the reservoir connections are kept fixed and only those weights from the reservoir layer to the output layer are computed.

The experimental implementation of **RC** brings, however, a key challenge with it. Complex networks composed of a large number of randomly connected nonlinear dynamical elements, as required in traditional **RC**, define strong limitations in what can, and cannot, be implemented in hardware. Therefore, until recently, mostly software realizations were considered. To overcome this restriction, the use of delay-coupled systems has been recently proposed and proven to be as efficient as the traditional reservoirs in certain tasks [8, 30, 31]. The simplest delay system consists of a nonlinear node subject to feedback, i.e. the system's dynamics is influenced by its own output at a certain time  $\tau$  in the past.

## CHAPTER 2. DELAY-BASED RESERVOIR COMPUTING

According to the characteristics of the nonlinear function, this kind of systems can be considered for RC.

In the following sections we will study in depth the characteristics and computational properties of the above mentioned simple delay-based reservoir computing system.

### 2.1

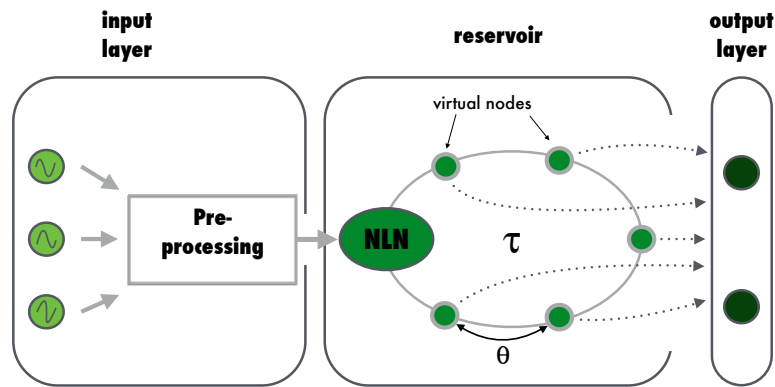
## Reservoir computing based on delay-dynamical systems

In 2011, Appeltant et al. [8] succeeded in replacing an entire network of connected nonlinear nodes by one single nonlinear node subject to delayed feedback. This approach demonstrated the computational power hidden in simple delay dynamical systems providing, at the same time, a drastic simplification of the experimental implementation of artificial neural networks for computing purposes. The approach is depicted in Figure 2.1 as the equivalent to the regular RC implementation illustrated in Figure 1.2. In the latter, the connections among nodes are usually chosen randomly while in the former the connections are fixed due to a particular coupling (ring) topology. The nonlinear node is depicted as NLN and a set of important positions in the delay line are known as *virtual nodes*. These virtual nodes do not perform any transformation to the input signal but they provide a state of the transient dynamics at every time interval  $\theta$ . This interval is determined by the number of nodes  $N$  in the reservoir and the length of the delay line  $\tau$  as  $\theta = \tau/N$ . Thus the position of each virtual node is kept fixed and equally separated from the other nodes.

When an input is injected into the reservoir, it perturbs the quiescent state of the nonlinear node (NLN) producing a transient dynamics. The virtual nodes reflect the evolution of this transient dynamics with a certain resolution. The resolution of observation is given by the time-interval  $\theta$ , which is a very relevant parameter in this configuration. If  $\theta$  is very long, the perturbations provoked by the input signal will vanish and the reservoir will go back to a steady state. On the contrary, if  $\theta$  is very short the reservoir will not have time to react to the input signal. As a general principle, we typically choose  $0.1T < \theta < T$ , where  $T$  represents the characteristic time scale of the nonlinear node. Appeltant et al. [8] found that using  $\theta = 0.2T$  usually leads to good results. Connecting nodes in the above mentioned manner emulates a network that fulfills the minimum requirements for information processing and serves as a reservoir [8, 32].

The reservoir depicted in Fig. 2.1 constitutes a single-node dynamical system with delayed feedback in which the transient caused by one point of a given

## 2.1. RESERVOIR COMPUTING BASED ON DELAY-DYNAMICAL SYSTEMS



**Figure 2.1:** Schematic representation of a reservoir as a single (NLN) node dynamical system. All the information is pre-processed to be sent to the reservoir through a single channel. The response of the nonlinearity is *observed* by the virtual nodes (that do not perform any transformation of the signal). The dynamical response observed by the virtual nodes is readout in order to set the weights of the connections between the reservoir and the output layer.

input is re-injected when processing the subsequent point. Mathematically, by introducing delayed feedback, dynamical systems become infinite dimensional. This is because its state at time  $t$  depends on the output of the nonlinear node during the continuous time interval  $[t-\tau, t]$  [33], with  $\tau$  being the delay time. The dynamics of the delay system remains finite dimensional in practice [34], but exhibits the properties of high dimensionality and short term memory, properties that are required for RC. From the point of view of hardware implementation, delay systems are very attractive since only few components are required to build them, e.g. a nonlinear node and a delay loop suffice [8, 30, 31, 35, 25, 36].

Up to this point we have described the reservoir internal dynamics, and we refer to an input signal as the input. However, when using a ring configuration (Fig. 2.1), we require a uni-dimensional information stream that is injected through a single channel to the NLN. To achieve this goal, we pre-process every input signal using a masking process that combines time-multiplexing of the inputs with imprinting different scaling factors on the input ensuring that the NLN always resides in the transient regime [37]. The entire process is explained in detail below in the upcoming sections. After pre-processing the input signals and injecting them into the NLN, the transient responses are used for training an estimator. The estimator might be a classifier or a predictor.

In the next sections, a detailed description of the information processing using delay-based reservoir computing is given.

### 2.1.1 The input layer: pre-processing of the input signal

In the approach presented by Appeltant et al. [8], the aim is to keep every single sample of an input signal  $u_t$  into the delay line for a certain time interval  $\tau$ . The delay line of length  $\tau$  contains the virtual nodes. For sufficiently enough long  $\tau$ , the dynamical state of the last virtual node might be the steady state of the nonlinearity because the transient provoked by each sample of  $u_t$  might have disappeared. This is not a desirable feature in the reservoir. To avoid this situation, each single sample of the input signal  $u_t$  is modulated to keep transient dynamics during the whole length of the delay line. This modulation is achieved by a masking process.

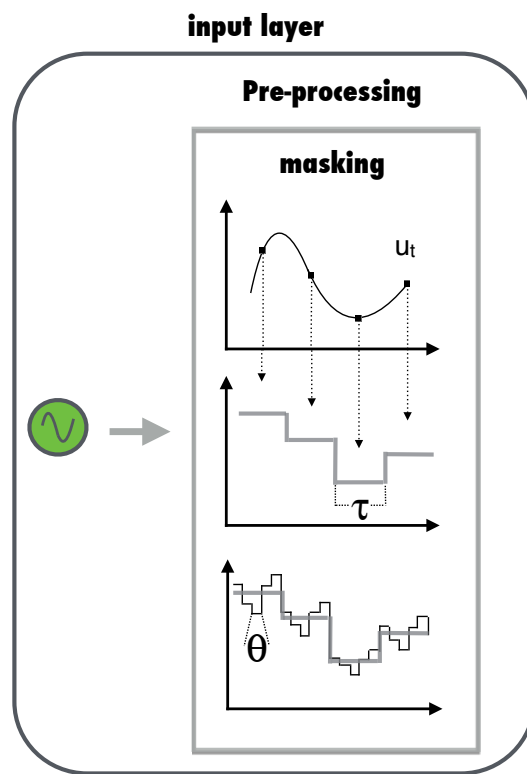


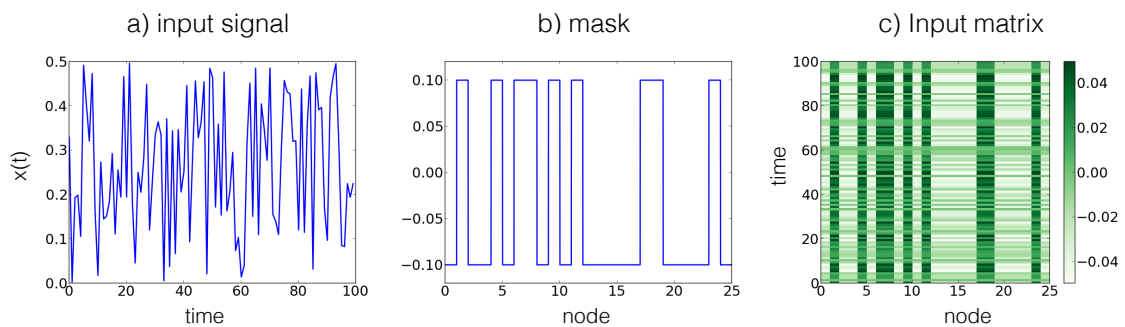
Figure 2.2: Schematic representation of the masking process.

The masking process takes each sampled point of  $u_t$  and multiplies it by a certain sequence of numbers that modulates each sampled value of the input signal. In Fig. 2.2 we show a scheme of this process. In this scheme, the value of each sample in the input signal is kept fix for an interval of length  $\tau$  (grey horizontal lines) and then it is modulated. Since there are virtual nodes separated by a time interval  $\theta$ , where  $\theta < \tau$ , the sample point is modulated to change its value every interval  $\theta$  and therefore to provoke a transient in the response of the NLN.

## 2.1. RESERVOIR COMPUTING BASED ON DELAY-DYNAMICAL SYSTEMS

We will elaborate about the properties of the mask later on. Let us concentrate now in the practical issues of the process.

For a number  $N$  of virtual nodes (*vn*) and an input signal  $u_t$  of dimensions  $M \times C$ , where  $M$  denotes the number of observations (sampled points) and  $C$  the characteristics observed, the mask matrix should have dimensions  $C \times N$ . Then performing a matrix multiplication between  $u_t$  and  $M$  results in an input matrix  $\mathbf{I}$  of dimensions  $M \times N$ . Note that the original input is  $u_t$  while matrix  $\mathbf{I}$  represents the expanded version of  $u_t$  and serves as the input to the nonlinear node.



**Figure 2.3:** a) Input signal of  $M = 100$  samples. b) Binary mask. The multiplication of a) and b) results in the input matrix shown in c).

Let us illustrate the pre-processing with the following example: Consider a reservoir with  $N = 25$  nodes and a time series of  $M = 100$  sample points. This represents the simple case where there is only one feature, i. e.  $C = 1$ , see Fig. 2.3(a). Then the mask, in Fig. 2.3(b), has dimension  $25 \times 1$  to hold each sampled point for an interval  $\tau$ . This results in an input matrix  $\mathbf{I}$  of dimension  $100 \times 25$ , Figure 2.3(c). One row of the input matrix  $\mathbf{I}$  corresponds to one modulated (masked) value in the input signal  $u_t$ . In this example, we have employed a two-level (binary) mask that consists of 0.1 and  $-0.1$  distributed in a random fashion, see Fig. 2.3(b). Therefore, if one plots the first column of matrix  $\mathbf{I}$ , the original signal multiplied by  $-0.1$  would be recovered.

We can highlight two basic requirements to configure a mask: the level of the mask and the sequence. In the previous example we illustrated a two-level (binary) mask. These levels enhance the separation of classes in the hyper-space created by the mask and the nonlinear transformation [35, 38]. This choice usually depends on the type of the data to process. The other requirement to take into account is how the sequence of symbols (mask levels) are organized. Random masks are widely used in [reservoir computing](#) perhaps motivated by the random connectivity of neurons in [RNNs](#). This randomness of the mask may cause results to vary across different realizations of the same task. The variability due to the mask randomness can be eliminated using a technique that

## CHAPTER 2. DELAY-BASED RESERVOIR COMPUTING

allows to design the mask. Appeltant et al. [39] presented a way to create the shortest possible mask that leads to maximum variability in the reservoir states. Their method utilizes maximum length sequences to define the combination of symbols in a mask such that sequences are not repeated. In the binary case, this method produces masks of minimum length of  $2^m + m - 1$  where  $m$  is the length of the basic-sequence to include in the mask. The approach can be extended to multi-level masks. Designing the mask limit the variability of results to come is from the [cross-validation](#) process, the training samples and the training algorithm. Controlling these variabilities, it is possible to identify the training samples that do not significantly contribute to solve a particular task. To define the levels of a mask and maximize the variability of the transient dynamics of the reservoir, aperiodic time series, e.g. the values of a deterministic chaotic time series, can also be used [40].

### 2.1.2 The reservoir layer: A closer look to the reservoir dynamics

The reservoir is composed by a nonlinear node subject to delay feedback. In a compact form, the dynamics of the node can be described as [32]

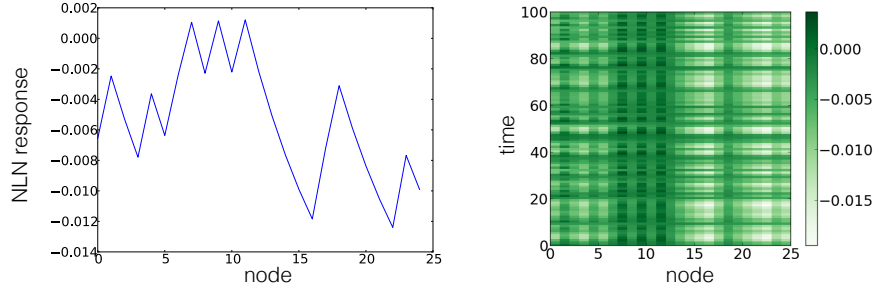
$$\dot{x}(t) = -x(t) + f(x(t - \tau), I(t, n)), x \in \mathbb{R}. \quad (2.1)$$

In this system  $x$  represents the dynamical variable,  $x(t - \tau)$  its delayed version at a certain past time  $\tau$ , and  $f$  is a smooth-real function. Equation 2.1 might exhibit a wide range of dynamical behaviors. In particular, if equation 2.1 exhibits chaotic behavior, the system would have a very good separability property since it would be very sensitive to the initial conditions. However, the approximation property will not be fulfilled for long time periods, i. e.  $\tau \gg 0$ . It will also lack of memory due to the exponential decay of auto-correlations in the attractor.

The input to be processed by the nonlinear node (Eq. 2.1) is represented by  $I(t, n)$  with  $n$  denoting the virtual node. This is the serial version of the input matrix  $\mathbf{I}$  in the previous section. The elements of  $\mathbf{I}$  are fed into the nonlinear node sequentially. For a certain time  $t \in M$ , where  $M$  is the number of sampled points in the input signal  $u_t$ , the elements  $n \in N$  are processed by the nonlinearity one after the other. Each element of  $I(t, n)$  creates a transient response on the [NLN](#). A full row of  $\mathbf{I}$  fills the delay line (Left panel of Fig. 2.4). Concatenating the responses of the [NLN](#) to each row of  $\mathbf{I}$ , another matrix, named state matrix  $\mathbf{S}$ , is created as the one shown in the right panel of Fig. 2.4. The state matrix  $\mathbf{S}$  has the same dimension as the input matrix  $\mathbf{I}$ .

In this scheme we can clearly see the recurrence or intrinsic memory of the system. When the first element of a row in  $\mathbf{I}$  is processed by the [NLN](#), it is affected by the transient dynamics of the first element of the previous row.

## 2.1. RESERVOIR COMPUTING BASED ON DELAY-DYNAMICAL SYSTEMS



**Figure 2.4:** The left panel shows the **NLN** response to sample 50 of  $u_t$  along the 25 virtual nodes. The right panel shows the state matrix **S** built out of all the virtual nodes.

Finally, note that for each input signal  $u_t$  injected into the reservoir there is associated a state matrix **S**.

Figure 2.4 was made by substituting function  $f$  of Eq. 2.1 by

$$f(x(t - \tau), I(t, n)) = \frac{\eta(x(t - \tau) + \gamma I(t, n))}{1 + (x(t - \tau) + \gamma I(t, n))^p}, \quad (2.2)$$

with  $\eta$ ,  $\gamma$  and  $p$  being parameters for the feedback strength, input scaling and degree of nonlinearity, respectively.

It is expected that the degree of nonlinearity of a system is related to the capability of linearly separate different input signals [41]. The memory of the system can be detected in the autocorrelation function through the peaks at the multiples of the delay time  $\tau$ . The degree of the nonlinearity can cause the autocorrelation values to approach to zero after few delays, limiting the memory capacity of the reservoir [42]. The nonlinearity of the **NLN** has to be chosen according to the kind of task to be performed, considering memory requirements and separation property. The limitation on the memory is not strict since there are different ways of increasing the memory of the system, e.g. by including additional delay loops [37, 43].

### 2.1.3 The output layer

This layer receives all the state matrices **S**. It can be configured in two modes: training or evaluation mode. For training purposes, the state matrices are combined in order to compute the weights of the connections between the reservoir layer and the output layer. In practice, we want to know the contribution of each virtual node, i.e. the weights. This is achieved by using a learning process as described in Section 1.1.2. The learning process assigns an output weight  $\omega_{jk}$

## CHAPTER 2. DELAY-BASED RESERVOIR COMPUTING

to each virtual node  $vn_j$ , such that the state matrices approximate to the desired output or target value  $y_k$  as close as possible fulfilling the condition that

$$y_k = \sum_{j=1}^N \omega_{jk} vn_j. \quad (2.3)$$

The usual practice in RC to estimate the values of the  $\omega_{jk}$  is a linear regression model [26] as those described in Section 1.1.2.

During the evaluation mode, one can process *unseen* input signals and evaluate the performance of the reservoir. In this case, the output layer utilizes the weights  $\omega_{jk}$  of the virtual nodes to assign a class to the new input signal  $u_t$  using, for instance, the so-called **winner-take-all** criterion.

Before moving forward, let us introduce a nonlinearity that fulfills the characteristics of the reservoir layer described in Section 2.1.2.

### 2.1.4 Mackey-Glass delayed feedback oscillator as a reservoir

The Mackey-Glass oscillator [10] is a nonlinear function that has proven to serve as a reservoir [8]. The original oscillator described in [10] is modified to include an external input signal. Its new form can be expressed as

$$\dot{x}(t) = -x(t) + \frac{\eta(x(t-\tau) + \gamma I(t, n))}{1 + (x(t-\tau) + \gamma I(t, n))^p}. \quad (2.4)$$

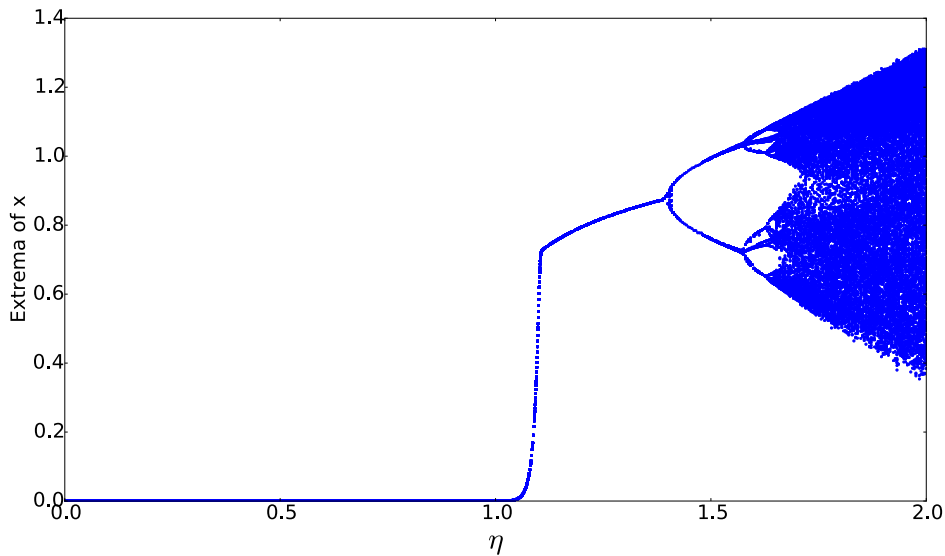
The solutions of this equation are real numbers. Here,  $x$  denotes the dynamical variable,  $t$  is a dimensionless time, and  $\tau$  is the delay-feedback loop. Parameters  $\eta$ ,  $\gamma$  and  $p$  represent the feedback strength, the input strength and the degree of the nonlinearity, respectively. The Mackey-Glass modified oscillator (Eq. 2.4) follows the standard representation of a node subject to delay feedback (Eq. 2.1) with function  $f$  as in Eq. 2.2. The Mackey-Glass oscillator can exhibit different kind of dynamical behaviors that can be explored by switching off the input signal ( $\gamma = 0$ ) and making its corresponding bifurcation diagram.

Figure 2.5 shows the bifurcation diagram where we can see a fix-point dynamics for  $0 < \eta < 1.39$ . This fix point is zero for  $\eta < 1$ . Beyond the fix-point dynamics, limit cycles develop a deterministic chaotic dynamics. We can then adjust the values of  $\eta$  to make the system to operate in a stable fixed-point when no input is injected ( $\gamma = 0$ ). However, when an input is injected into the oscillator, the system might exhibit complex transient dynamics.

The Mackey-Glass oscillator has two important advantages: The first one is that it can be easily implemented in hardware [37, 44, 45] and the second is that it



## 2.1. RESERVOIR COMPUTING BASED ON DELAY-DYNAMICAL SYSTEMS

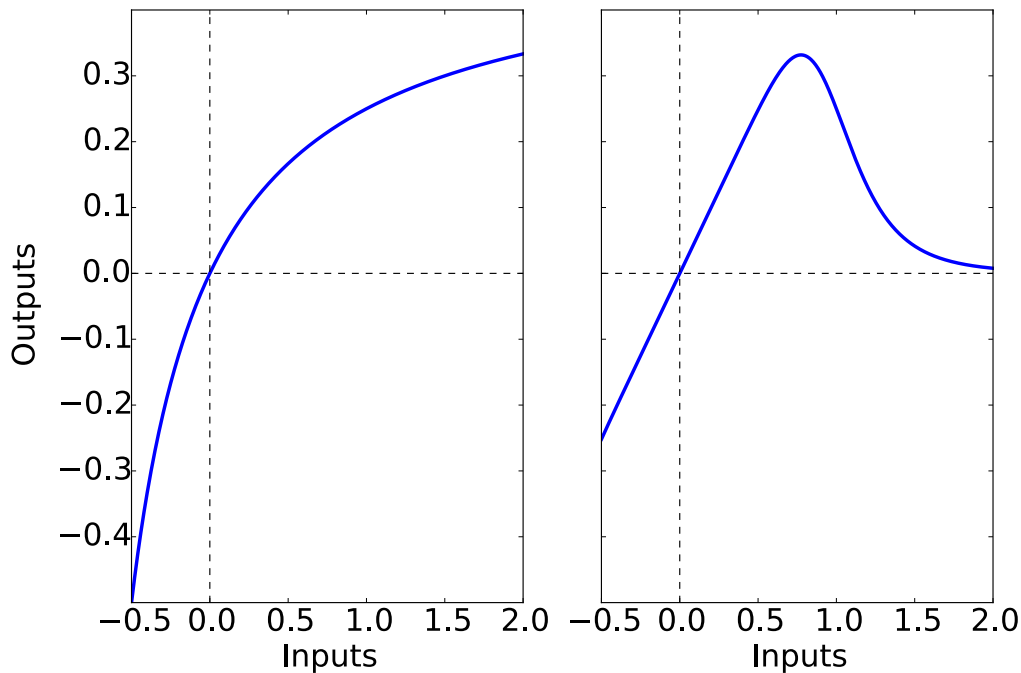


**Figure 2.5:** Bifurcation diagram for the Mackey-Glass delayed feedback oscillator (Eq. 2.4) with  $\gamma = 0$ ,  $p = 7$ , and delay time  $\tau = 80$ . The strength of the feedback  $\eta$  is varied to exhibit the different dynamics. Only extreme values are plotted.

allows to modify its degree of nonlinearity. This allows to change the operational point as the reservoir changes from strongly nonlinear to a *weak* nonlinearity. The shape of the nonlinearity is illustrated in Fig. 2.6. The left panel shows the transfer function of the Mackey-Glass oscillator (as in Eq. 2.2) for  $p = 1$  while the right panel shows the same function for  $p = 7$ , both functions with  $\eta = 0.5$ .

For  $p = 1$  the transfer function is a monotonically increasing function while for  $p = 7$  it has an extreme value. For inputs close to zero, the right figure looks more linear than the left one. Changing the values of the input scaling  $\gamma$ , one can explore different regions of the transfer function. The drawback here is that for odd values of  $p$ , the Mackey-Glass oscillator presents a discontinuity at  $-1$  (see Eq. 2.4).

Other nonlinearities can be used for the purpose of [reservoir computing](#) performing similarly to eq. 2.4, for example an optoelectronic oscillator or a semiconductor laser [8, 30, 31, 35, 25, 36]. In biology, it has been shown that cortical areas of the brain exhibit properties of [reservoir computing](#) [46], as well as the DNA as it was recently showed by Goudarzi [47]. These implementations indicate that the exact shape of the nonlinearity is not as crucial as it was assumed before [48].



**Figure 2.6:** Mackey-Glass transfer function. The left panel depicts the shape for  $p = 1$  whereas the right panel shows it for  $p = 7$ . In both cases we use  $\eta = 0.5$ .

## 2.2

### Task-independent properties of reservoirs

In Section 1.1.1, we mentioned some properties that a **RC** has to fulfill in order to obtain good performances. In this section we will extend these concepts. We mentioned **consistency** as the property of obtaining similar reservoir transient dynamics for similar input signals. In practice, this is achieved by setting the system to operate in an asymptotically stable quiescent state (fixed point) in the absence of input [8]. The **separation property** states that for two different enough stimuli, the corresponding transients must differ sufficiently. A system with these two properties can therefore served as a **reservoir computing** system. These two properties are complemented by a short-term (fading) memory [49, 50] due to the recurrent nature of the reservoir. This memory can be exploited by a reservoir to process information based on past events, which is particularly useful in time-series estimation tasks.

The separation property can be quantified by computing the distance between two reservoir states that are driven with different inputs. This measure was

## 2.2. TASK-INDEPENDENT PROPERTIES OF RESERVOIRS

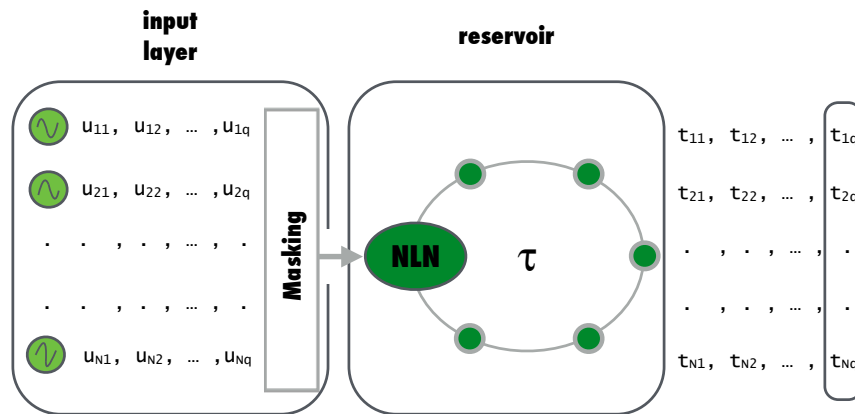
firstly introduced by Maass *et al.* (2002) [7] using neural microcircuits and spike trains. This property can be generalized using the so-called **kernel quality**.

### 2.2.1 Kernel quality

This quantity measures the ability of the reservoir to separate instances of different classes using a linear decision-hyperplane [51]. It is computed by injecting  $N$  random time series of length  $q$ ,  $U = \{u_1, u_2, \dots, u_N\}$  where  $N$  is the number of nodes in the reservoir. To compute the kernel quality, it is necessary to build a kernel matrix using the following algorithm:

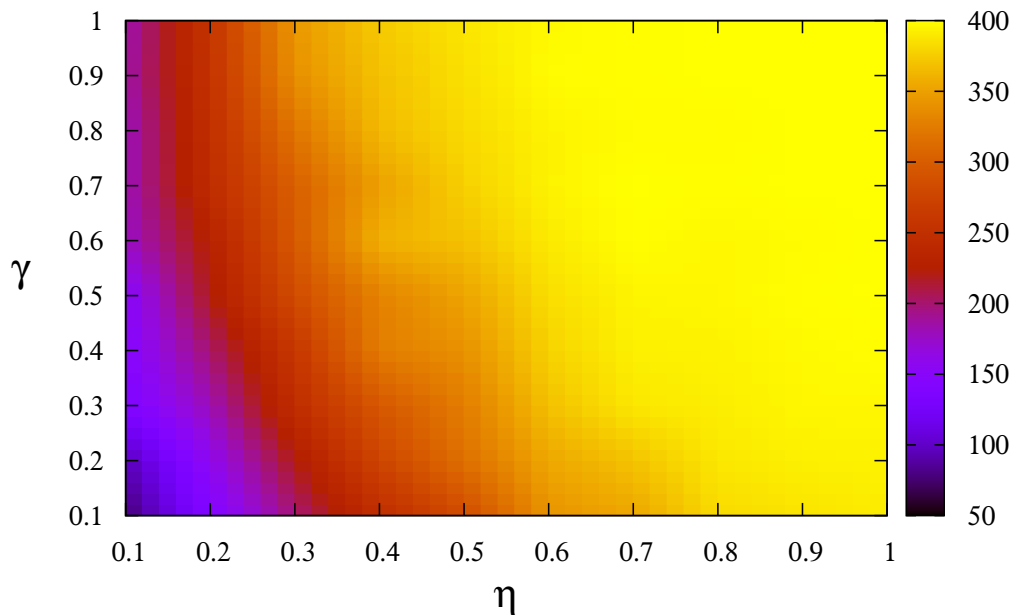
1. First, input  $u_i$ , ( $i = \{1, \dots, N\}$ ), is injected into the NLN, but only the transient of the last sample of  $u_i$ , i.e.  $u_{iq}$ , is kept for readout. This transient has dimensions  $1 \times N$ . Processing the  $N$  time series results in  $N$  vectors of  $1 \times N$ .
2. Then, the transient vectors are concatenated to obtain a  $N \times N$  kernel matrix.

A scheme of this procedure is shown in Fig. 2.7.



**Figure 2.7:** Scheme for building a kernel matrix.

To know the kernel quality, one has to compute the number of linearly independent eigenvectors (rank of the matrix) of the kernel matrix, i.e. how many input responses are distinguished using the linear readout. If the rank  $r_q = N$  then the reservoir is able to assign the correct target to each input  $u_i$  through the readout process. Thus, the more independent eigenvectors the better the quality of the reservoir. Smaller values of  $r_q$  indicate the degrees of freedom that are actually available to the reservoir to assign its states to the correct targets, i.e. a measure of the computational performance of the reservoir.



**Figure 2.8:** Parameter space  $\eta - \gamma$  showing the kernel quality rank. The Mackey-Glass oscillator served as the NLN with  $N = 400$  and  $p = 1$ . For this plot a sequence of  $q = 800$  samples was used.

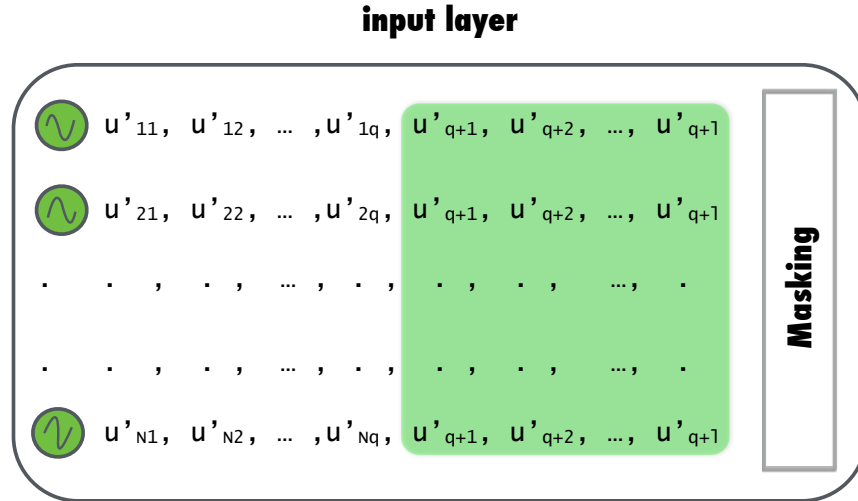
For illustration purposes, we compute the kernel quality for a reservoir with a Mackey-Glass NLN. The parameter space  $\eta - \gamma$  is explored when  $p = 1$ . The size of the reservoir is  $N = 400$  which represents the maximum rank that can be found. The rank of the kernel matrices are shown in Fig. 2.8. We can see a large region with a good kernel quality specially for high values of the feedback strength and the input scaling.

### 2.2.2 Generalization property

This property refers to the ability of the reservoir to generalize the learning, i.e. the reservoir learns from a particular set of data, but it is expected to work with a different set of data of the same kind. In other words, different input signals not necessarily must be assigned to different targets if the differences are not significant. The reservoir should be able to recognize those small differences and neglect them while keeping the general patterns of the signal. This property is quantified by the generalization rank which is computed in a similar way than the kernel quality rank.

## 2.2. TASK-INDEPENDENT PROPERTIES OF RESERVOIRS

In this case, we need  $N$  time-series  $U' = \{u'_1, u'_2, \dots, u'_N\}$  of random numbers with length  $q + l$  each, in which  $l$  is a fix value and its sequence is equal for all the  $u'_i$  time series. Thus, we are evaluating the ability of the reservoir to classify the same inputs while modifying the initial state of the reservoir. A scheme of the inputs  $u'_i$  is shown in Fig. 2.9. The shaded area is equal for all inputs. As the reader might already expect, the choice of  $q$  and  $l$  are important factors. The generalization rank  $r_g$  is then given by the rank of the  $N \times N$  matrix, as explained in the previous section. If the generalization rank is close to  $N$ , the system is not



**Figure 2.9:** Scheme for building a generalization matrix. The green-shaded area is an equal sequence to all the inputs.

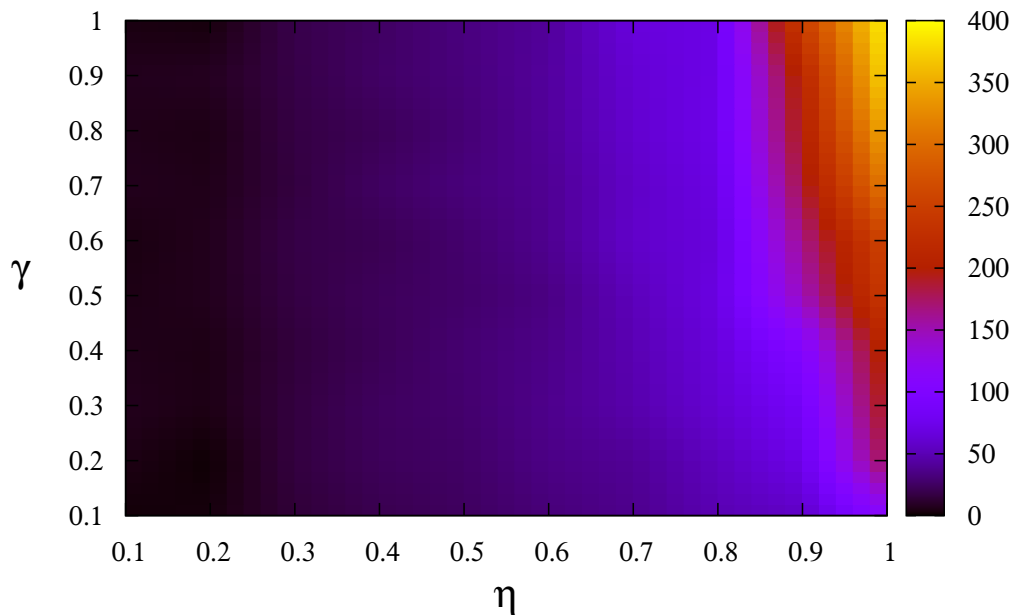
able to neglect the  $q$  random time steps, and classifies each signal as different. So, the smallest the generalization rank the better the generalization capabilities. This is opposite to the kernel quality rank. Fig. 2.10 shows the generalization rank in the parameter space  $\eta - \gamma$  using the same conditions as in Section 2.2.1.

In this case we observed a wide range where the evaluated reservoir has a good generalization property. A tradeoff of kernel quality and generalization rank is given by the computational ability measure.

### 2.2.3 Computational ability

For a reservoir to have a good performance, we desire a high kernel quality rank  $r_q$  and a low generalization rank  $r_g$ . When using a highly NLN, the separation property of the system improves, reducing the generalization capabilities of the system. We can find an optimal degree of nonlinearity using the computational ability defined as

$$r_c = r_q - r_g. \quad (2.5)$$



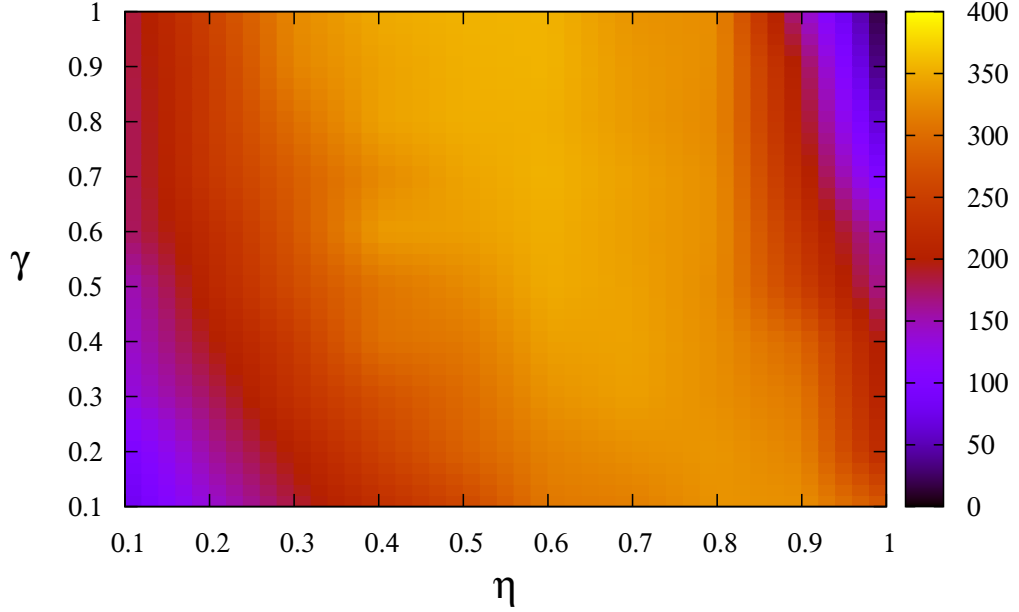
**Figure 2.10:** Generalization rank for the reservoir used in Section 2.2.1. The length of the  $u'_i$  signals was 801, with  $q = 800$  and  $l = 1$ .

The computational ability is shown in Fig. 2.11. Only when  $\eta - \gamma$  pairs are close to zero or close to 1, the reservoir does not have a good computational ability. The rest of the parameter values seem to be suitable to perform computational tasks. For an extended range of parameters of the above mentioned properties, the reader is referred to [37].

## 2.2.4 Memory capacity and quality

**Recurrent neural networks** include an intrinsic fading memory due to its characteristic recurrence. However, this memory might not be enough for certain tasks, e.g. time series prediction, degrading the performance of the reservoir significantly. To quantify the amount of memory available in the system, Jaeger (2002) [42, 52] proposed a test called memory capacity. In this test a set of input time series  $u(t)$  with samples from an uniform distribution in the interval  $[-0.8, 0.8]$  are injected into the reservoir with the aim to map those inputs to an infinite number of output time series  $y_k^i$ . The output signals  $y_k^i$  are the  $u(t)$  time series shifted by  $i$  steps, such that  $y_k^i = u(t - i) \forall i = \{1, \dots, \infty\}$ . Thus memory

## 2.2. TASK-INDEPENDENT PROPERTIES OF RESERVOIRS



**Figure 2.11:** Computational ability of the reservoir. Same parameters as in Fig. 2.8 and 2.10

capacity is defined as

$$\mu_c = \sum_{i=1}^{\infty} m_i, \quad (2.6)$$

where  $m_i$  is the normalized correlation between the  $\hat{y}_k^i$  and  $y_k^i$ , given by

$$m_i = \text{corr}[y_k^i, \hat{y}_k^i] = \text{corr}[y_k^i, u(t - i)]. \quad (2.7)$$

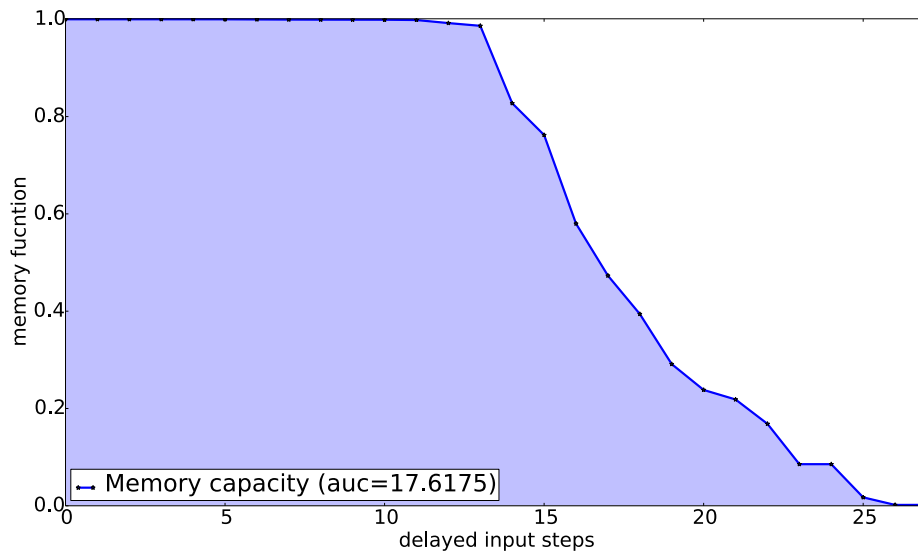
In practice, the sum in Eq. 2.6 is stopped when the scores are very close to zero. An example of the memory function  $m_i$  is shown in Fig. 2.12.

The memory capacity is obtained by integrating the curve in Fig. 2.12. This measure is complemented with a memory quality measure that takes into account the shape of the curve. This measure was introduced by Hermans and Schrauwen (2010) [49] as

$$\mu_q = \frac{1}{\mu_c} \sum_{i=1}^{\mu_c} m_i. \quad (2.8)$$

It indicates how rectangular is the shape of the memory curve which can be important in tasks that need a minimum of memory to perform well. The

## CHAPTER 2. DELAY-BASED RESERVOIR COMPUTING



**Figure 2.12:** Memory function of the reservoir for different delayed steps of the input signal. The memory capacity is given by the **AUC**. The function is computed with parameter  $p = 1$  and  $\eta = 0.5$  of the Mackey-Glass oscillator.

maximum value of the  $\mu_q$  is one. The closest  $\mu_q$  is to one, the more rectangular the shape of the memory function is. The reservoir used to compute Fig. 2.12 has a  $\mu_q = 0.8595$  indicating a fair rectangular shape.

In the following section we introduce some examples in order to illustrate the essence of **reservoir computing**.

### 2.3

## Typical machine learning tasks using Reservoir Computing

In this section we illustrate some benchmark tasks used in **reservoir computing**. We make use of the concepts explained in the Introduction and in the current chapter. For this purpose, we have chosen three different tasks that serve to show the computational power in a delay-coupled system. The tasks include:

- A classification task, where the system has to learn the differences of pronounced numbers and recognize which number was mentioned.
- A time series prediction task, where the system needs to use its own memory for the prediction of events in the future, and



## 2.3. TYPICAL MACHINE LEARNING TASKS USING RESERVOIR COMPUTING

- an emulation of a model, where the system uses the inputs and outputs of a model to infer its dynamics, without explicitly knowing the equations of the model.

These three tasks are described in the same ordered as mentioned above.

### 2.3.1 Spoken digit recognition (SDR)

This task consists on the classification of isolated spoken digits. The database contains ten spoken digits, from 0 to 9, and it is part of the NIST TI-46 corpus [53, 54]. Each digit was recorded ten times by five female speakers. Thus, our dataset contains 500 samples. To extract the best features for the classification each sample was preprocessed using a cochlear ear model [55] resulting in samples of the form of matrices of 86 columns (frequency channels) and as many rows as time steps in each acoustic signal. Using the same nomenclature as introduced in section 2.1.1, each digit  $u_t$  contains  $C = 86$  characteristics and  $M$  observations. The number of observations changes between digits due to the time it takes to pronounce them. As a kernel method we use a delay-based reservoir computer and as learning process a linear regression model with a ridge regression (L2) regularization (Section 1.1.2).

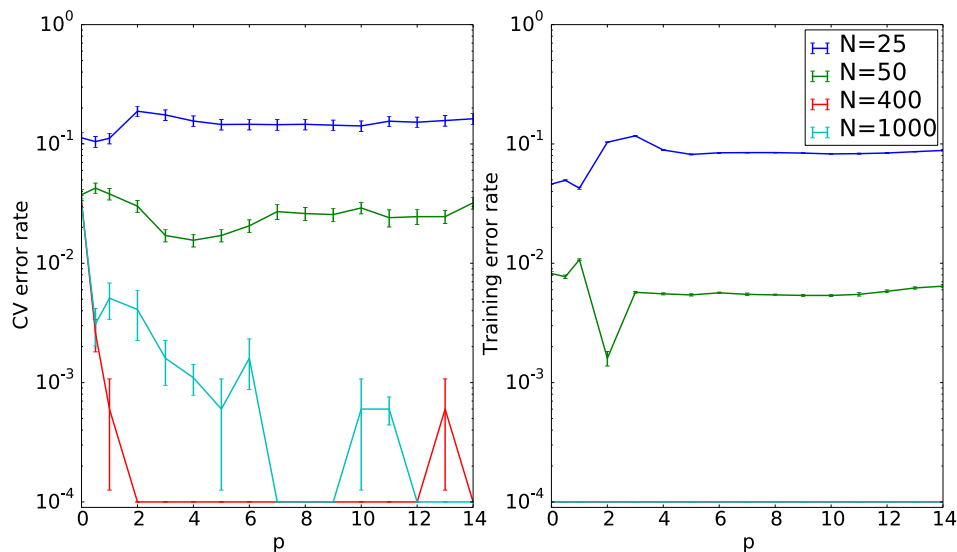
We will follow the scheme presented in section 1.2 for the diagnosis of learning algorithms. We should not forget that tuning an ANN is an optimization process that requires several iterations. In the following paragraphs we show one or few of them. We begin by building a cross-validation data set by dividing the full dataset into 5 non-overlapping sets each of 100 samples, i.e. 10 digits, 1 speaker, 10 recordings. Two sets are used to determine the complexity of the model, two more to estimate model's parameters, and the last set for evaluation. In all steps we use a k-fold cross-validation with  $k = 10$ . Note this is the most difficult scenario for the reservoir to estimate the correct targets. Setting reservoir parameters is performed by using different speakers than those used for the evaluation, so not only we include samples that are not used while optimizing the reservoir but also we introduce a different speaker.

**Model Complexity (Section 1.2.1)** The first step is to select a model and to determine the degree of complexity that best fits the data. Very simple models can lead to high bias problems whereas too elaborated models might suffer high variance problems. We can determine an optimal degree of complexity by building a model complexity curve. As model, we choose the Mackey-Glass nonlinearity (Eq. 2.4) with parameter  $\eta = 0.8$  and  $\gamma = 0.5$  which ensures the system to be in a steady state in the absence of inputs. This equation contains a parameter  $p$  that controls the degree of nonlinearity of the system. Note that

## CHAPTER 2. DELAY-BASED RESERVOIR COMPUTING

the size of the reservoir also reflects the complexity of the model since each virtual node represents a delayed-nonlinear transformation of the input in the new high-dimensional space. So, the size of the reservoir can be considered as both: a feature (for feature selection) and the level of complexity.

In order to include information from these two variables: nonlinearity degree and reservoir's size, the model complexity is split into two plots, one with the training errors and another with the cross-validation errors. The errors are computed as error rates, i.e. for a predicted digit different than the original (a mistake) the system scores 1, otherwise it scores 0. The error rate is then the average of these scores. The prediction of the reservoir is based on the  $y_k$  values from Eq. 2.3. The reservoir selects the class  $k$  of the predicted digit using the **winner-take-all (WTA)** criterion. In this particular context this error is called **Word Error Rate (WER)**. The lower the error rate, the better the model. Figure 2.13 shows the training error rate (right panel) and the cross-validation error rate (left panel) for different reservoir sizes and nonlinearity degrees  $p$ . Note that we have added a constant value of  $10^{-4}$  to the errors to make a semi-logarithmic scale possible. Vertical lines show the **standard error of the mean (s.e.m.)**. The previous figure shows the errors for four different reservoir sizes. We have chosen the more representative ones among all the simulated sizes.



**Figure 2.13:** Model complexity curves for different reservoir sizes  $N$ . The left panel shows the cross-validation error while the right panel shows the training error. Vertical lines denote the **s.e.m.** In order to draw errors in logarithmic scale, we added a constant value of  $10^{-4}$  to all curves.

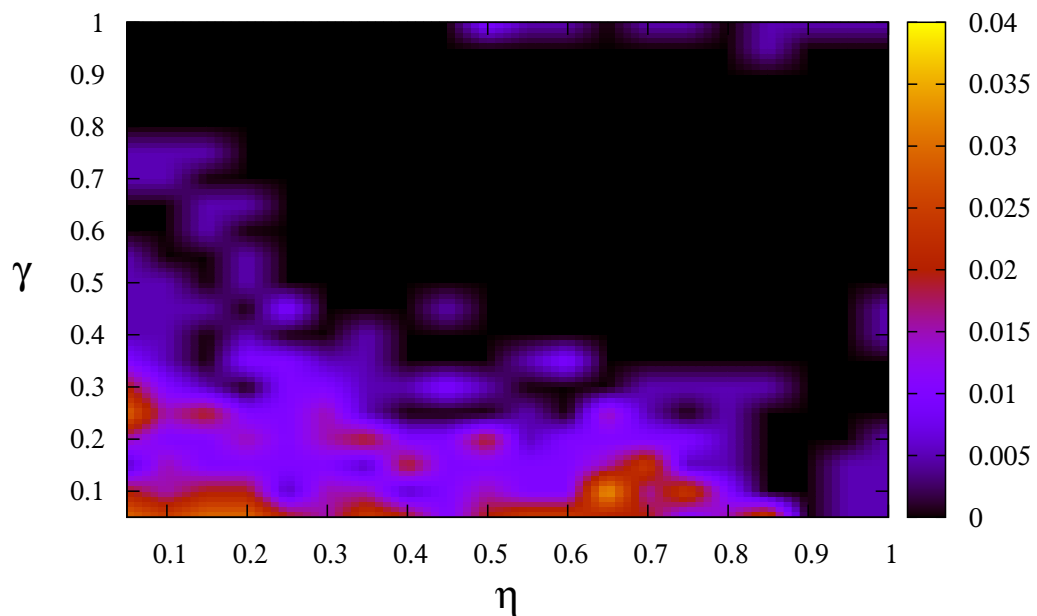
Results in Fig. 2.13 suggest that  $N = 400$  is the best size for this task. The two error rates converge to zero for a wide range of  $p$ . Note that if the size of the

### 2.3. TYPICAL MACHINE LEARNING TASKS USING RESERVOIR COMPUTING

reservoir is much larger ( $N \sim 10^3$ ) then the error rate curves do not converge due to the ability of the reservoir to capture too many features of the inputs that overfits the data, i.e. the reservoir becomes specialized in the training set but it is not able to generalize.

From now on, we take  $p = 7$  and  $N = 400$  which implies a delay time  $\tau = 80$  since  $\theta = 0.2$ . As an example, the previous computations were performed using Mackey-Glass parameters  $\eta$  and  $\gamma$  fixed to 0.8 and 0.5, respectively. To evaluate the influence of these parameters, we explore the parameter space  $\eta - \gamma$  in the next section.

**Exploring parameter space  $\eta - \gamma$ .** The Mackey-Glass oscillator in Eq. 2.4 includes parameters that account for the scaling  $\gamma$  of the input matrix  $\mathbf{I}$  and the dynamical state of the oscillator, i.e. parameter  $\eta$ . Note that an input  $u_t$  is always pre-processed with a mask before reaching the nonlinear node. This causes the original scales of the input to be modified. Thus, the values of the mask play an important role in this case. For this example, we are using a randomly generated mask with dimensions  $400 \times 86$  that contains values of  $\{0, 0.41, 0.59\}$  in a proportion of  $\{90, 5, 5\}\%$  respectively.



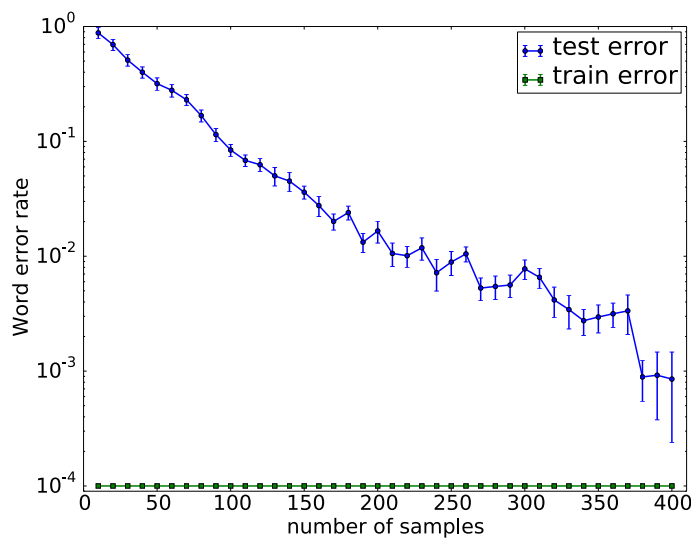
**Figure 2.14:** Mackey-Glass oscillator parameter space  $\eta - \gamma$  for  $p = 7$ . Colors show the CV error rate for each set of parameter.

## CHAPTER 2. DELAY-BASED RESERVOIR COMPUTING

The parameter space is shown in Figure 2.14 in a color scale. Darker colors represent low [cross-validation](#) error rate. In this figure, we see that there is a wide range of parameters that are optimal to perform this task. Note that basically the full parameter space is useful since the highest [CV](#) error rate is smaller than 4%. Since we reach an average error very close to zero, further computations are performed using  $\eta = 0.8$  and  $\gamma = 0.5$ .

We have set the complexity of the model and explored the parameter space of the nonlinear node. Our system is now ready to classify spoken digits. The question is: how well does the system recognize the digits? Let us plot a learning curve for a first diagnosis and then evaluate our designed system.

**Learning curve (Section 1.2.3)** Using a learning curve we can determine if the reservoir designed in the previous steps suffers a bias or variance problem. To build a learning curve, we plot the train and test ([CV](#)) errors as a function of the number of samples (digits) in the dataset. Both sets of data: model complexity and parameter space databases (beginning of Section 2.3.1), can be used to build the learning curve since no parameters are being estimated in this step yet optimistic results are expected because these datasets were used for the setting of parameters. Therefore our data set includes a total of 400 samples.



**Figure 2.15:** Learning curve. The testing [WER](#) is represented in blue and shows the dependency of the number of samples. The curves show the average over ten realizations shuffling randomly the database (vertical lines represent the [s.e.m.](#)). Training error is always  $0 \pm 0$ , so a constant value of  $10^{-4}$  was added to both curves to plot them in a semi-logarithmic scale.

### 2.3. TYPICAL MACHINE LEARNING TASKS USING RESERVOIR COMPUTING

Figure 2.15 shows the learning curve of the designed reservoir. The curve represents the average of 10 executions of the algorithm including samples in different order each time. Vertical bars represent the *s.e.m.* For all realizations, the training error is  $0 \pm 0$ . We have added a constant value of  $10^{-4}$  to both curves in order to plot the curves in a semi-logarithmic scale.

The decreasing trend of the test error indicates that the reservoir improves the classification when more samples are introduced for training. When 270 digits are used for training, the reservoir makes 1 mistake in average and keeps performing similarly until the full dataset is used. We can exclude a high variance problem because the difference between error rates is very small (an average of less than 1 mistake in a 10-fold *CV*).

The final step is the real proof to the designed reservoir. We evaluate its ability to generalize to samples that have not been included for training or setting of any parameter. The evaluation dataset consists of 100 samples from a different speaker.

**Evaluation of performance (Section 1.3)** Training is achieved using the same 400 samples used for the learning curve, i.e. 80% of the total available data (4 speakers), whereas evaluation of the reservoir is performed with a new speaker (20% of data) via the *winner-take-all* (*WTA*) criterion. In this case we find a *WER* of 9.7%. In comparison with previous results using delay-based reservoir computing (*WER*<1%) [8, 25] this *WER* is worse. It is worth to remember that the scheme followed in this chapter represents one of the most difficult for this classification task. Previous results utilized a *cross-validation* scheme using the same data utilized for the setting of parameters that yields to better, although less useful results.

In this classification task there are 10 classes (digits from 0 to 9). Thus the resulting confusion matrix, shown in Table 2.1, is a  $10 \times 10$  matrix. The confusion matrix identifies a problem in the classification of the new speaker. The reservoir *confuses* digits 1, 3 and 5 with digits 9, 8, and 1, respectively. For the rest of the digits, the reservoir performs correctly.

In order to compute confusion matrix derived measures, the matrix in Table 2.1 is binarized via *one-vs-all* (*OvA*) criterion. Results per class are presented in Table 2.2. This table shows sensitivity, precision, specificity and accuracy when only samples of the new speaker are considered.

Using this kind of table one can identify those problematic classes in a classification task. As an example, let us go through the measures for class 5. Recall that sensitivity represents the proportion of positive samples correctly identified. For digit 5 (the positive class) we have 80% sensitivity because the reservoir is able to recognize 80% (8 out of 10 samples) of the digits as digit 5. From those 8

## CHAPTER 2. DELAY-BASED RESERVOIR COMPUTING

**Table 2.1:** Confusion matrix for the [SDR](#) classification task.

Predicted digit → Real digit ↓	0	1	2	3	4	5	6	7	8	9
0	10	0	0	0	0	0	0	0	0	0
1	0	9	0	0	0	0	0	0	0	1
2	0	0	10	0	0	0	0	0	0	0
3	0	0	0	4	0	0	0	0	6	0
4	0	0	0	0	10	0	0	0	0	0
5	0	2	0	0	0	8	0	0	0	0
6	0	0	0	0	0	0	10	0	0	0
7	0	0	0	0	0	0	0	10	0	0
8	0	0	0	0	0	0	0	0	10	0
9	0	0	0	0	0	0	0	0	0	10

samples predicted as digit 5, all of them were actually digit 5 (Precision=100%). There are not false positives for digit 5 (in Table 2.1, column 5 is always zero for any digit different than 5), then this classifier is 100% specific. Finally accuracy shows the sum of the true predictions (positive and negatives) with regards to all the samples in the dataset that for digit 5 is 97.99%. Digits 8 and 9 represent the opposite case to digit 5 where sensitivity is maximum but precision and specificity are less than 100%. Finally, digit 1 is the combination of these two cases.

To finish this example, we build a [ROC](#) curve (Section 1.3.3). To build such a curve we need to express digits as scores ( $y_k$  values). These values can be any real number.

We can set a threshold value to differentiate the score distributions and in this way build the corresponding [ROC](#) curve. Figure 2.16 shows the [ROC](#) curve for the multiclass [SDR](#) task. The points of the curve in the left of the space, represent conservative models, while the points on the right prioritize the sensitivity of the classifier. The selection of a classifier will require to average [ROC](#) curves in a [CV](#) fashion, however it mainly depends on the kind of task that is performed.

The area under the [ROC](#) curve on Figure 2.16 is 0.9916. Any value over 0.9500 is considered a good value. The [AUC](#) of a classifier is equivalent to the probability that the classifier prefers a positive class over a negative one when these classes are randomly ordered [23]. The particular [AUCs](#) per digit are shown in Table 2.3.

## 2.3. TYPICAL MACHINE LEARNING TASKS USING RESERVOIR COMPUTING

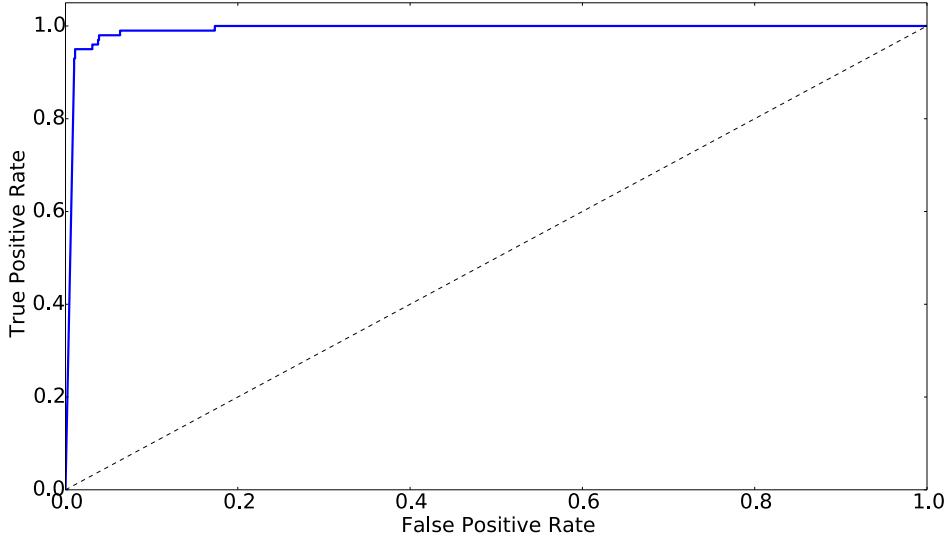
Table 2.2: Confusion matrix derived measures.

Measure → Digit ↓	Sensitivity	Precision	Specificity	Accuracy
0	100.0	100.0	100.0	100.0
1	90.00	81.81	97.77	96.99
2	100.0	100.0	100.0	100.0
3	40.00	100.0	100.0	93.99
4	100.0	100.0	100.0	100.0
5	80.00	100.0	100.0	97.99
6	100.0	100.0	100.0	100.0
7	100.0	100.0	100.0	100.0
8	100.0	62.50	93.33	93.99
9	100.0	90.90	98.88	98.99
<b>Average</b>	90.99	93.52	98.99	98.20

### 2.3.2 El Niño time-series prediction

The phenomenon of El Niño is an excellent example of the interaction between the ocean and the atmosphere evidencing their combined effect on global climate. El Niño disrupts the ocean-atmosphere system when strong westward blowing trade winds subside and warm water slowly moves back eastward across the Pacific. This results in a redistribution of rains, the interruption of upwelling of cool and nutrient-rich waters bringing consequences as flooding and droughts, the death of fishes and climatic changes in many parts of the world.

The phenomenon is not completely periodic. It develops in a range of two to seven years. Therefore the forecast of El Niño is a challenging and open problem among climatologists. What is known as El Niño is one part of the [El Niño Southern Oscillation \(ENSO\)](#) phenomenon. The Southern Oscillation refers to the variations in temperature of the tropical eastern Pacific ocean and in air surface pressure in the tropical western Pacific. A warm oceanic phase comes with a high air surface pressure in the western Pacific (El Niño), whereas a cold phase accompanies low air surface pressure (La Niña). The [Southern Oscillation Index \(SOI\)](#) shows the mean sea-level pressure difference between Tahiti and Darwin, Australia [56], and it is commonly used for the detection and prediction of El Niño.



**Figure 2.16:** Multiclass ROC curve for the SDR classification task.

For illustration purposes, we have chosen a simple model described by a scalar delay differential equation that mimics most of the observed dynamics of the ENSO phenomenon [57]. The model considers irregular fluctuations of the sea-surface temperature, and incorporates the full Navier-Stokes dynamics of El Niño events by a suitable chosen nonlinearity. This task was developed in collaboration with the Potsdam Institute for Climate Impact (PIK), Germany.

Using an equivalent procedure as shown in Section 2.3.1, we set a reservoir computer to predict the ENSO time series using the Mackey-Glass nonlinearity with parameters  $\eta = 0.5$ ,  $\gamma = 0.05$ ,  $\tau = 80$ ,  $p = 1$ , and  $N = 400$ . We have scaled the signal (zero mean and unit variance) and averaged the data monthly. To perform cross-validation we split the resulting time-series in 4-noncontinuous smaller segments, i.e. three for training and one for testing or 4-fold cross-validation.

To measure the goodness of prediction, we use the Normalized Root-Mean-Square Error (NRMSE) which shows how effective a mathematical model predicts a time-series. The NRMSE is written as

$$NRMSE = \sqrt{\frac{1}{m} \frac{\sum_{i=1}^m (\hat{y}_i - y_i)^2}{\sigma^2(y_i)}}, \quad (2.9)$$

where  $m$  is the number of time steps in the time series,  $\hat{y}$  is the predicted time series and  $y$  is the original time series.

We show in the upper panel of Figure 2.17 a sample raw-signal of the modeled SOI. The lower three panels show two curves each. In blue, the normalized



## 2.3. TYPICAL MACHINE LEARNING TASKS USING RESERVOIR COMPUTING

Table 2.3: Area under the ROC curve for each class.

Digit	AUC
0	0.9999
1	0.9878
2	0.9999
3	0.9967
4	0.9999
5	1.0000
6	0.9999
7	0.9999
8	0.9667
9	0.9944
<b>average</b>	0.9916

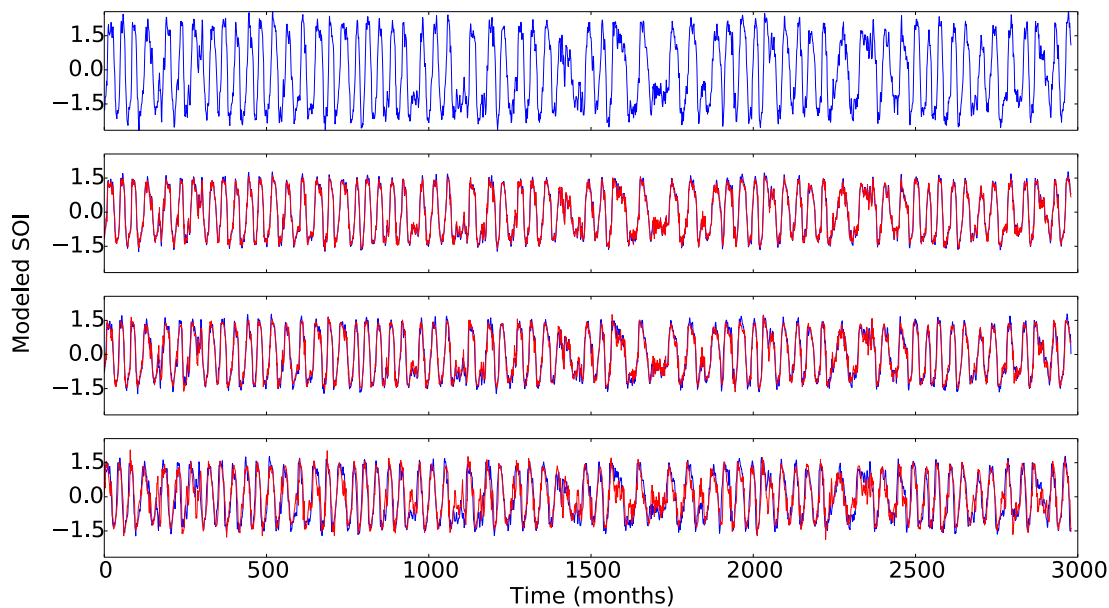
time-series of the delay-action model [57] and in red, the [reservoir computing](#) predicted (one-point ahead) time-series. From top to bottom, we present one-, three- and six-month predictions, respectively.

For one-month prediction, i.e. the reservoir is predicting the value of the variation in temperature for the next month, we computed a ([NRMSE](#)  $\pm$  [SD](#)) of  $(0.02551 \pm 0.00001)$ . As the prediction horizon increases, the error rises due to its accumulation during previous predictions. We found a  $(0.07542 \pm 0.00034)$  and  $(0.24 \pm 0.000129)$  [NRMSE](#) for predictions of 3 and 6 months, respectively. The errors reported are mostly caused by the fast oscillations in the time-series. This indicates that the occurrence of El Niño, characterized by the slow oscillation, can be predicted with a lower error rate.

### 2.3.3 Modelling: NARMA10

In time series prediction, the dependence of a current value to previous observations is commonly assumed. This dependence is very well captured in the autoregressive-moving average (ARMA) models [58]. For the autoregressive (AR) part, ARMA considers a time series that depend on its lags values. These models also take into account that an event happening at time  $t$  causes modifications in the dynamical state of the system at time near to instant  $t$ . This accounts for the moving-average (MA) part of the models. These models have been used as benchmark tasks in machine learning for prediction tasks. The nonlinear version of ARMA models (NARMA) [59] has been used in reservoir

## CHAPTER 2. DELAY-BASED RESERVOIR COMPUTING



**Figure 2.17:** El niño estimation for 1,3, and 6 months.

computing [40, 60]. The NARMA receives an input time series  $u_k$  that consists of scalar random numbers from a uniform distribution in the interval  $[0, 0.5]$ . Usually a number at the end of the model identifies the degree of autoregression, e.g. NARMA10 or NARMA30 refer to the number of steps that the model is recursive. For this example, we employ a NARMA10 model which is defined as

$$y_{k+1} = 0.3y_k + 0.05y_k \left[ \sum_{i=0}^9 y_{k-i} \right] + 1.5u_k \cdot u_{k-9} + 0.1 \quad (2.10)$$

In this equation,  $y_{k+1}$  represents the target of the reservoir. The time series  $u_k$  is preprocessed with a mask as described in Section 2.1.1 and then fed to the nonlinear node. The mask in this task consists of a binary mask with values of  $\{-0.1, 0.1\}$ . To perform this task correctly, the reservoir needs to have a memory of at least 10 steps due to the presence of the term  $u_{k-9}$  in the above equation. To perform this task, we provide the reservoir with the input signals  $u_k$  and the target signals  $y_{k+1}$ . The reservoir needs to learn the internal dynamics of the NARMA10 model having access only to the input and output.

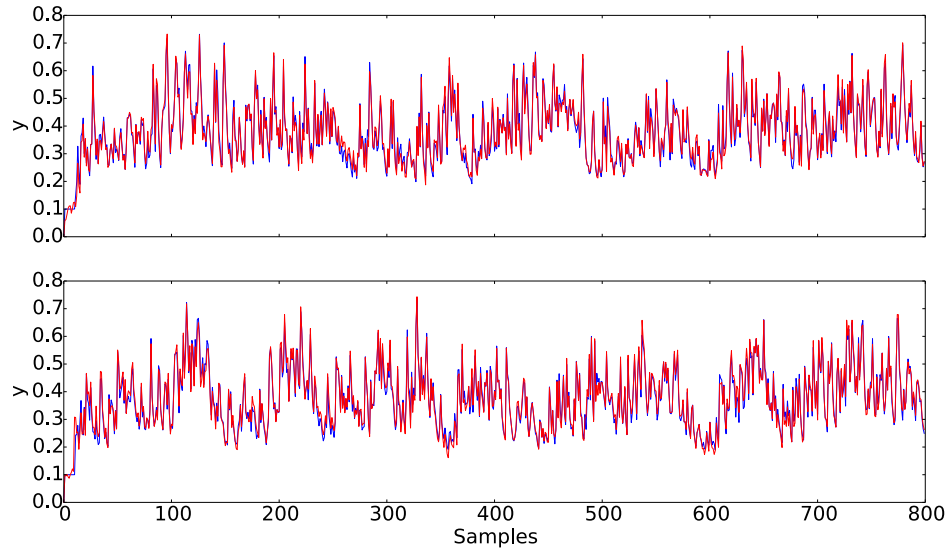
**Table 2.4:** Comparing NARMA10 performance with results in literature.

Ref.	[37]	[60]	This thesis
<b>NRMSE</b>	0.120	0.099	$0.0991 \pm 0.0056$

Performance is measure using the **NRMSE** between the output signal and the estimated signal.

## 2.4. EXPLORATION OF RESERVOIR'S FEATURES

The optimal parameters of the Mackey-Glass oscillator to perform this task are:  $\eta = 0.5$ ,  $\gamma = 0.05$  and  $p = 1$  with  $N = 400$  nodes and therefore a delay time of  $\tau = 80$ . With these parameters, the reservoir exhibits a good computational ability (Fig. 2.11). As mentioned before this task needs at least a memory of 10 time steps to be performed correctly. The memory capacity of this particular configuration of the reservoir was shown in Fig. 2.12 where more than 15 time steps are kept in the memory of the reservoir.



**Figure 2.18:** NARMA10 time-series. In blue is the original time-series while in red is the estimated one. The upper panel shows an estimation during training and the lower is computed during evaluation (4-fold CV).

Figure 2.18 shows the original NARMA10 time trace (in blue) and the estimated trace (in red) during two phases of the learning. The upper panel shows a signal used during training whereas the lower panel shows the signal used for evaluation in a 4-fold cross-validation scheme.

We achieve a competitive performance, as indicated by the [NRMSE](#) values in Table 2.4

2.4

---

### Exploration of Reservoir's features

In this section we explore some of the characteristics of a delay-coupled system when serving as reservoir. The way we perform the tasks described before followed a typical path but, what happens when we change this path? Are there

## CHAPTER 2. DELAY-BASED RESERVOIR COMPUTING

important consequences from these changes? Let us start with the initial state of the reservoir.

### 2.4.1 Initializing reservoir's states

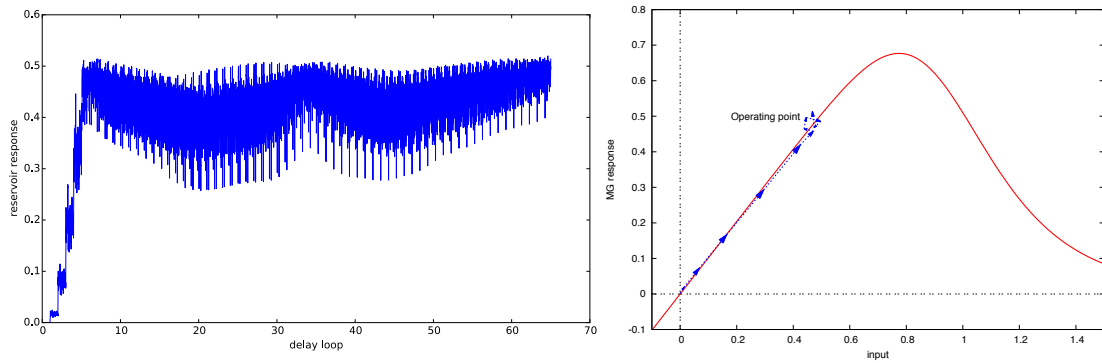
When a reservoir starts to receive an input signal the typical configuration is to initialize the delayed states to a fix point of the function. This ensures a steady behavior of the system. Although we initialize the reservoir states at a stable point when an input is injected, it causes a jump in the dynamics, followed by a period of time (transient time) in which the dynamics is dominated by the input sampled points. This can be seen as a kind of *transient of the transients* caused by the input. This means that there are some cycles of length  $\tau$  that might be not useful for the task. One may consider to disregard a few rows of every state matrix  $\mathbf{S}$  to avoid this issue losing some information from the beginning of the input signal. This drawback is illustrated in Figure 2.19 for one digit of the SDR task described in Sec. 2.3.1. The digit contains 65 time steps, each one expanded to a 400 hyper-space. The left panel of the figure shows the response of the reservoir. In its leftmost part, one can see how the response of the nonlinearity rises from the initialization state (zero state) to the real dynamics of the perturbations induced by the input. The first  $\tau$  corresponds to the initializing vector that in this case is a zero vector. The right panel of Fig. 2.19 shows the transfer function of the Mackey-Glass system for  $p = 7$ . The arrows show the transition from one delay time  $\tau$  to the next one. As in the left panel, we can observe how the dynamical state of the reservoir is modified by the input signal until it reaches a point where the reservoir operates for the classification task.

After processing the last sample of the input signal, the reservoir dynamics relaxes back to a steady point. Then it is ready to start to process a new input. The phenomenon observed in Figure 2.19 is then repeated in every state matrix.

Going back to brain dynamics, the brain does not go to a steady state before processing a new sensory signal. We do not pretend to emulate the exact brain dynamics but some questions can be posed in this matter: what happens when the reservoir is initialized at a different state?, e.g. the transient dynamics of the previous input signal, and how this change affects the performance?

Let us illustrate this using the spoken digit recognition task described in Section 2.3.1. We use the optimal set of parameters previously found for this task, except for  $\eta = 0.65$  and  $\gamma = 0.1$  to avoid a zero error rate. As a database, we utilize the 500 samples with a 20-fold CV. This will result in an optimistic absolute performance, but we are not interested in the overall performance of the reservoir but in the relative performance. As before, we employ the Mackey-Glass nonlinearity.

## 2.4. EXPLORATION OF RESERVOIR'S FEATURES



**Figure 2.19:** Left panel: Reservoir response to a spoken digit of 65 time steps. The delayed states of the reservoir were initialized at the zero fix-point. The dynamics induced by the input is thus found after 6 delay loops. Right panel: The transfer function of the Mackey-Glass system. The arrow show the transition from one delay time to the next one. For visualization purposes not all the delays are shown.

We executed the task 10 times for both cases: the typical initial condition at the steady state for each input and the initial conditions based on the transient of the previous input.

**Table 2.5:** Comparison of the WER between a reservoir initializing at a steady point and a reservoir initializing at the previous input transient state.

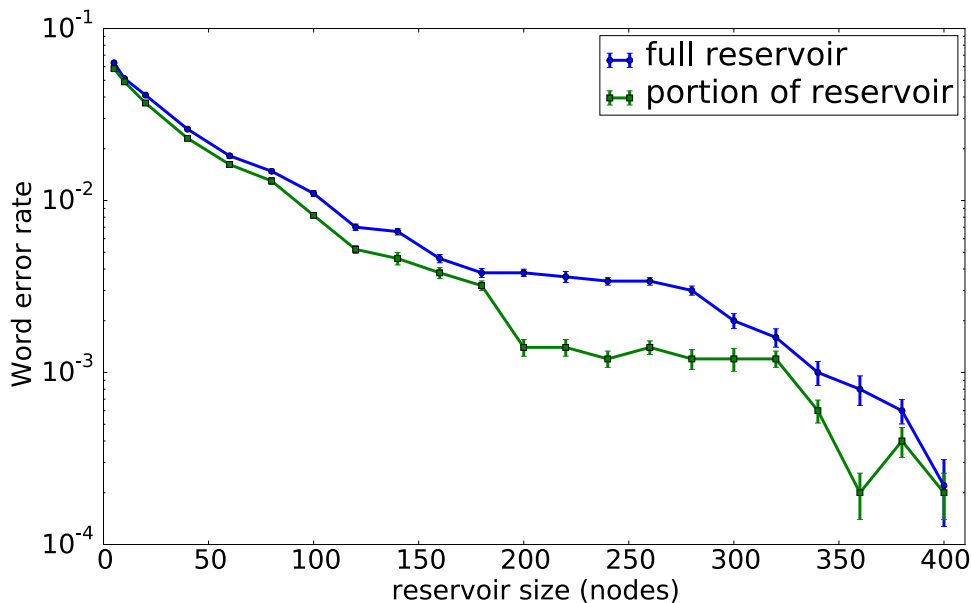
	Steady point	Previous input transient
WER $\pm$ s.e.m.	0.019 $\pm$ 0.001	0.019 $\pm$ 0.001

Results in Table 2.5 show no statistical changes among the evaluated cases. Results are presented as the word error rate (WER) with the standard error of the mean (s.e.m.). This could mean that the reservoir is able to extract information from these *transients of transients* or that the information of the input signal contained in the beginning can be disregarded. The consistency in the results is related to the generalization property (Section 2.2.2). This property modifies the delayed dynamics of the NLN using random numbers to identify a fix sequence that is common for all the inputs. Although the way of measuring the generalization property and the way the reservoir operates to build the states matrices are different, in both cases, we are studying the statistical differences when the system is initiated at a point different than the fix point.

## 2.4.2 Quorum sensing: subsampling reservoir's states

What occurs when, for a certain reason, we can not access all the nodes of the reservoir to make the readout? Will the implicit redundancy of information of the network be enough to perform a task?

Let us consider the scenario of an experiment where we have a very large reservoir but we have limitations to read the state of every single node, i.e. we only have access to a portion of the reservoir. In 2009, Nikolić et al. [46] made multi-electrode recordings from a cat primary cortex and analyzed the evolution of stimulus-related information in the spiking activity of an ensemble of 100 neurons. They used different stimulus signals constituted by letters of the alphabet and were able to decode them from the recordings using machine learning algorithms. In this case it is clear they can not have access to the whole primary visual cortex network. So, in the case of our reservoir, how this scenario affects the performance?



**Figure 2.20:** Quorum sensing of the reservoir. The blue curve represents a reservoir with the indicated amount of nodes while the green curve represents a reservoir of 400 nodes where the portion of nodes indicated in the x-axis is readout.

Utilizing the optimal configuration for the SDR task, we evaluate the performance of the reservoir due to a certain number of the total amount of virtual nodes. Results are shown in Figure 2.20 as the average of ten executions of the task. In each execution a certain number of virtual nodes is randomly chosen to be part of the state matrix. Vertical bars denote the s.e.m. The blue curve

## 2.4. EXPLORATION OF RESERVOIR'S FEATURES

represents the [WER](#) when all virtual nodes are accessed and used for readout. The green curve represents the [WER](#) of the portion of the reservoir that is used in the readout. In this case, the full reservoir is constituted by  $N = 400$  virtual nodes. The blue curve is systematically higher than the green one, i.e. it is better to sample a portion of a large reservoir than having a smaller reservoir. Note that the green curve makes in average 5 misclassifications (out of 500) when it reads 200 virtual nodes, which is a good result considering that it is reading half of the total information in the reservoir. But why a reservoir of 200 nodes performs worse? This might be due to the redundancy of information in the reservoir. The system contains all the information of the input signals, but this information might be reducing the degrees of freedoms of the state matrices. Partially eliminating the information of some nodes (still memory on the system) can solve this problem allowing the reservoir to perform better. A solution for the blue case is to optimize a mask that reduces this dependency. Coming back to our question in the beginning, a reservoir that can not access the total amount of its nodes, performs worse than the one that does. However, it still can have a good performance.

In the next chapter we explore the performance of [reservoir computing](#) for the solution of a time-consuming real-life medical task.





# Classification of heartbeats using reservoir computing

The implementation of theoretical concepts that work well with real data always represents a challenge. In this chapter a real-life time-consuming problem is presented as an attempt to use [reservoir computing](#) in the classification of heartbeats. International guidelines are followed for the evaluation and comparison of classification algorithms in cardiology. The chapter starts motivating the problem followed by a simple description of the cardiovascular system. The database, a well-known dataset in cardiology, is described in the upcoming section. The logistic regression learning process, different than what is usually utilized in [RC](#), is explained and then results and conclusions are presented.

## 3.1

---

### Motivation to the classification of heartbeats

[Cardiovascular disease \(CVD\)](#) is the first cause of death and disability all around the world. In the Western world, it accounts for 24% of all deaths globally [61]. Only in the EU the economic cost is about 196 billion euros per year [62]. However, the term [CVD](#) accounts for many heart-related diseases, some of them can threaten the life of the patient such as some kind of arrhythmias, ischemia and strokes are among the most dangerous. The risk of suffering [CVDs](#) increases with age, smoking, obesity, low physical activity, high blood lipid levels, hypertension and diabetes [63, p.18-19]. Therefore, there is a strong positive relationship between reduction of low density lipoprotein cholesterol ([LDL-C](#)) and the frequency of [CVDs](#) [64]. The most widely used test to study heart-related diseases, as a first diagnose, is the [electrocardiogram \(ECG\)](#). During more than one half of a century, the [ECG](#) has been a powerful and irreplaceable tool in the exploration and diagnostic of [cardiovascular diseases](#). Its acquisition requires

## CHAPTER 3. CLASSIFICATION OF HEARTBEATS USING RC

only simple and low-cost devices with a minimum impact on the patient. The ECG is an essential diagnostic tool for common pathologies such as myocardial ischemia [65, 66], arrhythmia and other rare pathologies as cardiac muscular dystrophy or Brugada syndrome [67]. These advantages have made the ECG a preferred tool used in clinical environments and outside by the use of portable devices such as the Holter monitor. Continuous monitoring in medicine has given physicians the ability to collect hours and days worth recordings of physiological signals, involving physicians in a time-consuming analyzing process<sup>1</sup>, where the time for diagnosis increases with the amount of available data. Realistically, a physician can hardly study each heartbeat, but rather gives a look to the whole time series searching for anomalies in the waveform of the ECG. Therefore, a reliable automatic classifier of heartbeats can reduce the time of diagnosis considerably and serve as a relevant tool in a clinical environment.

Several computational algorithms have been proposed to automate the process of ECG classification. The usual ECG classification solution includes a multistep procedure. A first step of detecting the heartbeat, usually done by a QRS detector algorithm, such as the Pan Tompkins method [68]. A second step requires features extraction, transforming the raw signal to meaningful quantities, such as the ECG morphology [69], the duration of different intervals on the ECG wave [70], the separation of landmarks on the waveform among several heartbeats [69, 70], coefficients of transformation of the original signal, etc. Most of the current algorithms rely on a good feature extraction to achieve good performance. However, this step can be time consuming due to the different waveforms and the fact that pathological beats not always clearly show the landmarks needed for the extraction of features. The last step of the processing represents the classification itself. This step consists of distinguishing the different types of beats to be classified. Many authors have already studied the heartbeat classification problem using several different techniques, such as self-organizing networks (SON) [71], self-organizing maps with learning vector quantization (SOMLVQ) [72], linear discriminants (LD) [73, 74], signal modeling (SM) [74], support vector machine (SVM) [75, 76], discrete wavelet transformation (DWT) [77], Bayesian artificial neural networks (BANN) [78], local fractal dimension [79] and delay differential equations (DDE) [80], obtaining different performance measures. Comparing results is difficult though, because of the different measures that were used, as well as the different partitions of the available data into training, testing and validation subsets. In 1987, the Association for the Advancement of Medical Instrumentation (AAMI) published a guideline for grouping heartbeat types into classes [81] and for evaluating the performance of algorithms [82]. Unfortunately, very few investigators have utilized these standards, making the direct comparison of results and the identification of pros and cons of the different methods difficult.

---

<sup>1</sup>In 24 hours, a Holter monitor can record over 100 000 heartbeats from one patient.

## 3.2. SOME GENERALITIES ABOUT THE CARDIOVASCULAR SYSTEM

In the following sections, we present a especially designed reservoir computer for the classification of cardiac arrhythmia that adopts the guidelines of the [AAMI](#).

### 3.2

---

## Some generalities about the cardiovascular system

The main function of the cardiovascular system is to guarantee a continuous flux of life-sustaining blood to organs and cellular tissue in the body in order to: *i*) provide oxygen and nutrients, *ii*) evacuate the metabolic products generated during organ's activity, and *iii*) transport hormones. This system is constituted by a pump organ, *the heart*, and a continuous and closed network of pipes that allows the transport of blood, *the vascular system*. Figure 3.1 shows the heart and the vascular system in what is known as the cardiovascular system.

The heart sits in the middle of the chest (slightly to the left), behind the breastbone and between the lungs, in a moistened chamber protected all round by the rib cage. The exact location of the heart can vary from one individual to another due to the shape of the diaphragm and the heart's size. The organ speeds up and slow downs automatically in response to nerve signals from the brain according to how much the body is being exerted. Normally, the human heart contracts and relaxes between 70 to 80 times per minute.

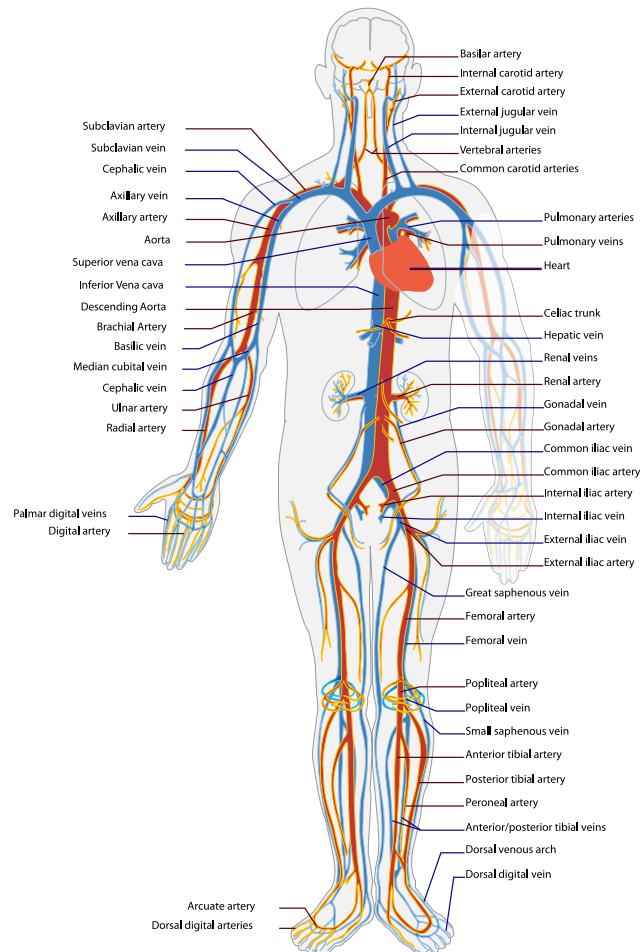
The human heart is divided in four chambers (see Figure 3.2). The two superior chambers are known as the atria, right and left atrium<sup>2</sup>. They are responsible of receiving blood from the vascular system and ejecting it to the ventricles. The two inferior cavities, the right and left ventricle, are in charge of ejecting blood to the vascular system. The right ventricle pumps blood to the lungs for the exchange of gases, while the left ventricle pumps most forcefully, pumping blood to every corner of the body. The ventricles are separated of the atria by the atrioventricular valves. The *tricuspid valve* separates the right atrium from the right ventricle, while the *mitral valve* separates the left cavities. The function of these atrioventricular valves is to avoid the return of the blood to the atria once the blood has been ejected to fill the ventricles. In a similar way, the sigmoïde (pulmonary and aortic) valves avoid the return of blood to the ventricles once it has been ejected to the pulmonary arteries and the aorta.

The mechanics that characterizes the pumping function of the heart can be divided in two intervals: *systole* and *diastole*. The first refers to the ejection of blood from the cavities and the second to the interval of relax of the heart when the cavities are being filled of blood. This mechanic function of the

---

<sup>2</sup>Positions are named from the point of view of the heart and not from the observer.

## CHAPTER 3. CLASSIFICATION OF HEARTBEATS USING RC

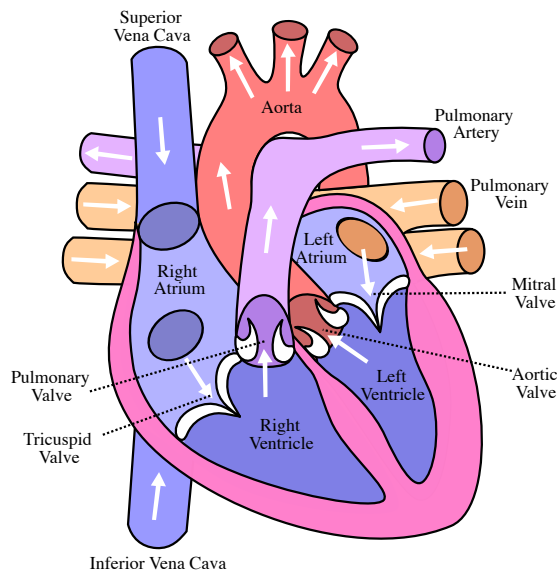


**Figure 3.1:** The cardiovascular system. In red it is the blood that goes out from the heart to other organs, usually carried by arteries. In blue, it is the blood that turns back to the heart. It is usually carried by veins. *Image by LadyofHats, Mariana Ruiz Villareal licensed under Public Domain via Wikimedia Commons.*

heart is the consequence of the electrical activation of the myocardial muscle or myocardium. To fulfill its function as a pump, the myocardium is composed of two different tissues:

1. **The conduction or nodal tissue** is constituted of specialized cells with properties of excitation, conductance and automatism. These properties allow a regular and spontaneous generation of electrical impulses and their organized transmission throughout the myocardium.
2. **The contractible tissue** occupies most of the cardiac muscle and also has excitation and conductance properties. However, as a difference to the nodal tissue, it is composed of muscle fibers that are able to contract.

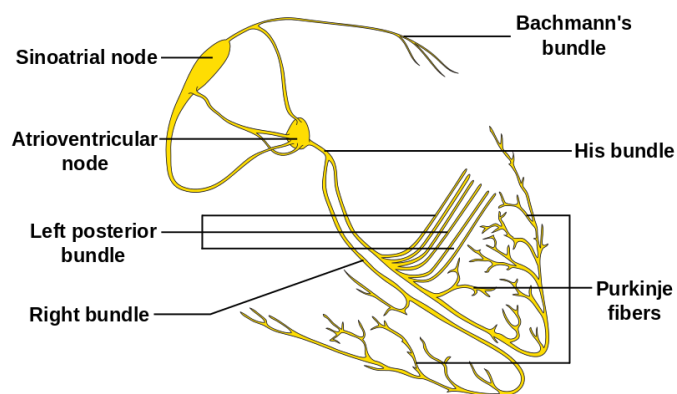
### 3.2. SOME GENERALITIES ABOUT THE CARDIOVASCULAR SYSTEM



**Figure 3.2:** Anatomical structure of the heart. The two atria and the two ventricles are the four chambers of the heart to fulfill its role as a pump.  
*Image licensed under CC BY-SA 3.0 via Wikimedia Commons.*

#### 3.2.1 The cardiac cycle and the ECG

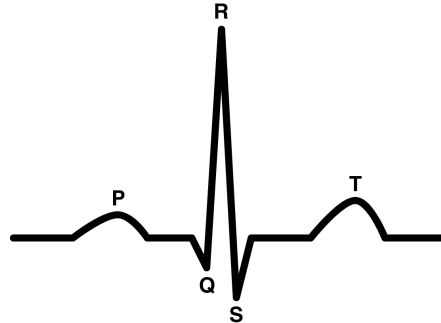
The depolarization and re-polarization processes of the cardiac structures are represented on the ECG as a sequence of deflections or superpose waves to a zero potential line, named isoelectric line. The morphology of these waves depends on two fundamental aspects: *i*) the anatomical structure that gives origin to the electrical impulse, i.e. the sinoatrial node, and *ii*) the conduction sequence through the myocardium.



**Figure 3.3:** Isolated heart conduction system. *Image created by Angelito7, distributed under the Creative Commons Attribution-Share Alike 3.0 Unported license.*

## CHAPTER 3. CLASSIFICATION OF HEARTBEATS USING RC

The cardiac cycle has its origin when the autonomous nodal tissue at the sinoatrial node (located on the upper wall of the right atrium, see Figure 3.3) depolarizes giving rise to a wavefront. This wavefront travels along the atria via the Bachmann's bundle provoking their contraction (atrial systole) and the ejection of the contained blood. The electrical atrial systole is detected in the ECG as the P-wave shown in Fig. 3.4. The atrial re-polarization is represented by the Ta-wave (not shown), and its direction is opposite to the P-wave. In general the Ta-wave is not visible on the ECG because it coincides with the QRS complex which has a higher amplitude. The wavefront then reaches the atrioventricular node, the unique electrical connection between the atria and the ventricles, which delays the impulse for about  $100ms$  and then the wavefront travels through heart muscle cells specialized for electrical conduction known as the Bundle of His. The Bundle of His goes until the lowest part of the heart (the apex) where the electrical path bifurcates in two branches, right and left. Each branch activates a ventricle throughout the Purkinje fibers, what causes the depolarization of the ventricles (ventricular systole) at a paced interval. Electrical systole of the ventricles begins at the beginning of the QRS complex (Figure 3.4). The ventricles have a large muscle mass compared to the atria, so the QRS complex usually has a much larger amplitude than the P-wave. Using the same electrical paths, the heart recovers (re-polarizes) starting from the ventricles, what is shown in the ECG as the T-wave. For the interested reader on ECG analysis, we refer to [83].



**Figure 3.4:** Typical ECG trace. *P* wave represents the atria depolarization, *QRS* complex takes place at ventricular depolarization and *T* wave represents ventricular repolarization. *Image created by Agateller for Wikipedia, Public Domain.*

### 3.2.2 Registration of the electrical activity of the heart

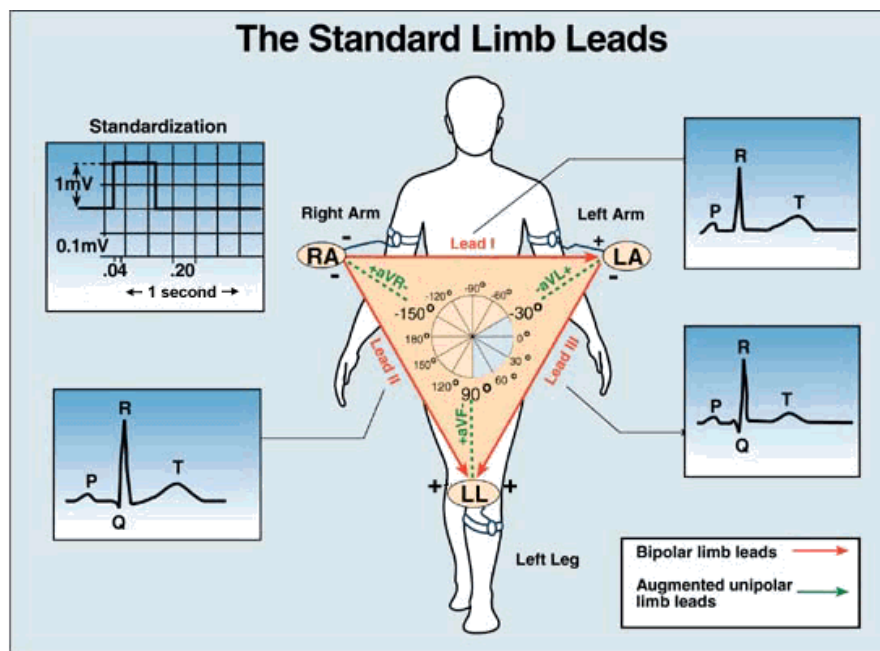
The flux direction and the amplitude of the electrical currents generated by the depolarization and re-polarization processes of the myocardial cells can be registered by electrodes positioned at the surface of the thorax. The analysis

### 3.2. SOME GENERALITIES ABOUT THE CARDIOVASCULAR SYSTEM

of this electrical activity is considered the main technique for the diagnosis of [cardiovascular diseases](#) and constitutes a fundamental tool for monitoring cardiac activity. In this section a brief introduction about the relation between the potentials registered at the surface of the body and the cellular action potentials, and the way to register them, is given.

#### Cardiac electrogenesis and the origin of the vectocardiogram

In the external surface of the membrane of each excited cell, two different polarities can be observed in each side of the activation front: a depolarized zone, negatively charged, and an adjacent positive zone (ready to be depolarized). These two zones constitute a dipole that can be represented, at a time  $t$ , by a vector of an electric field  $\vec{v}$ . The direction of the vector depends on the anatomical axis of the cardiac fiber and its amplitude depends on the action potential associated to the concerned cell. The same applies to the re-polarization process, where the dipole has similar direction, but reverse, and amplitude.



**Figure 3.5:** The vectocardiogram with Einthoven limb leads and Einthoven triangle. The Einthoven triangle is an approximate description of the lead vectors associated with the limb leads. *Image under CC license.*

Since the thorax is a conductor volume, the electrical potentials are propagated from the surface of the heart to the surface of the body. However, the thorax is an anisotropic, irregular, and dynamical conductor volume that avoids a uniform propagation and causes distortions on the superficial electrical field.

## CHAPTER 3. CLASSIFICATION OF HEARTBEATS USING RC

The approach of Einthoven [84] represents a simplification of the problem. It is based on the notion of an equivalent cardiac dipole and it supposes that an instantaneous cardiac vector can be estimated at any time  $t$  by the vectorial sum of all instantaneous elementary vectors associated to the active cells in the myocardium. It is also suppose that this vector is equivalent to the vector observed at the surface of the body. The alignment and direction of the instantaneous cardiac vector represent those of the cardiac activation front and its module depends on the quantity and type of active cells at time  $t$ . This scheme is known as the vectocardiogram (Fig. 3.5). A representation of the mean electrical activity in the time interval  $[t_1, t_2]$  (or mean cardiac vector) can be computed by a simple vectorial sum of instantaneous cardiac dipoles on the interval.

The Einthoven limb leads or standard leads can be defined as (see Fig. 3.5):

$$\text{Lead I: } V_I = \Phi_{LA} - \Phi_{RA} \quad (3.1a)$$

$$\text{Lead II: } V_{II} = \Phi_{LL} - \Phi_{RA} \quad (3.1b)$$

$$\text{Lead III: } V_{III} = \Phi_{LL} - \Phi_{LA}, \quad (3.1c)$$

where  $\Phi$  represents the potential at the particular location and  $V$  the voltage of the lead. Following Kirchhoff's law the lead voltages are related by

$$V_I + V_{III} = V_{II}, \quad (3.2)$$

hence only two of the three are independent.

### Aumented and precordial leads

By averaging the measurements of electrodes  $RA$ ,  $LA$  and  $LL$ , one gets the so-called Wilson's central terminal  $V_W$  [85]. The augmented limb leads  $aVR$ ,  $aVL$  and  $aVF$  ( $F$  refers to the left leg) use the same electrodes for the limb leads but they use Wilson's central terminal as the negative pole, such that

$$aVR = RA - \frac{1}{2}(LA + LL) = \frac{3}{2}(RA - V_W) \quad (3.3a)$$

$$aVL = LA - \frac{1}{2}(RA + LL) = \frac{3}{2}(LA - V_W) \quad (3.3b)$$

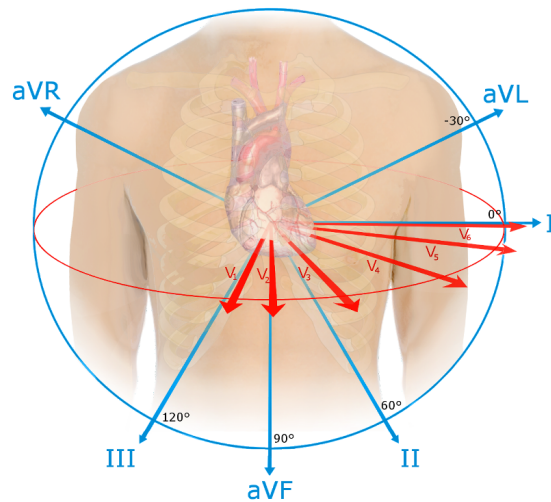
$$aVF = LL - \frac{1}{2}(RA + LA) = \frac{3}{2}(LL - V_W). \quad (3.3c)$$

The augmented leads are represented in blue in Fig. 3.6.

In the transverse (horizontal) plane Wilson et al. [86] introduced 6 electrodes ( $V_1, V_2, \dots, V_6$ ) located in the left part of the chest. These six electrodes act as positive pole and are referenced with the Wilson's central terminal as the negative pole. Their positions are shown in red in Fig. 3.6.



### 3.2. SOME GENERALITIES ABOUT THE CARDIOVASCULAR SYSTEM



**Figure 3.6:** Augmented (blue) and precordial (red) leads. *Image by Npatchett licensed under CC BY-SA 3.0 via Wikipedia.*

Currently the 12-lead **ECG** is the standard way of registering and reporting these signals for an international interpretation. The 12-lead **ECG** is composed by the limb leads (*I, II, III*), the augmented leads (*aVR, aVL, aVF*) and the precordial leads (*V<sub>1</sub>, V<sub>2</sub>, V<sub>3</sub>, V<sub>4</sub>, V<sub>5</sub>, V<sub>6</sub>*). A standard 12-lead **ECG** report (Fig. 3.7) shows 2.5 seconds of the trace in each lead. These traces are commonly arranged in a 3 × 4 grid. In the first column are the limb leads whereas augmented leads are represented in the second column. The last two columns show the six precordial leads. An extra row in the end is sometimes added to include a longer trace of one particular case. In Fig. 3.7, limb lead II is shown during 10 seconds.

#### Modifications of leads

The 12-lead configuration is the most used in clinical environments, however some modifications are used for particular applications. When recording an **ECG**, the signal is usually distorted by muscular movements, respiration and electrode artifacts, e.g. the transpiration of the patient. Mason and Likar (1966) [87] suggested to reduce the distortion due to muscular activation by placing electrodes in the shoulder and hips instead of the arms and leg. This is the most important modification to the limb leads and are known as Modified limb (ML) leads. In ambulatory recordings, as well as Holter monitoring, the limb electrodes are usually placed in the surface of the thorax instead of the extremities.

Additional electrodes might be added to the 12-lead configuration for specific diagnostic purposes. For instance, right sided precordial leads might help to study pathologies of the right ventricle.

## CHAPTER 3. CLASSIFICATION OF HEARTBEATS USING RC

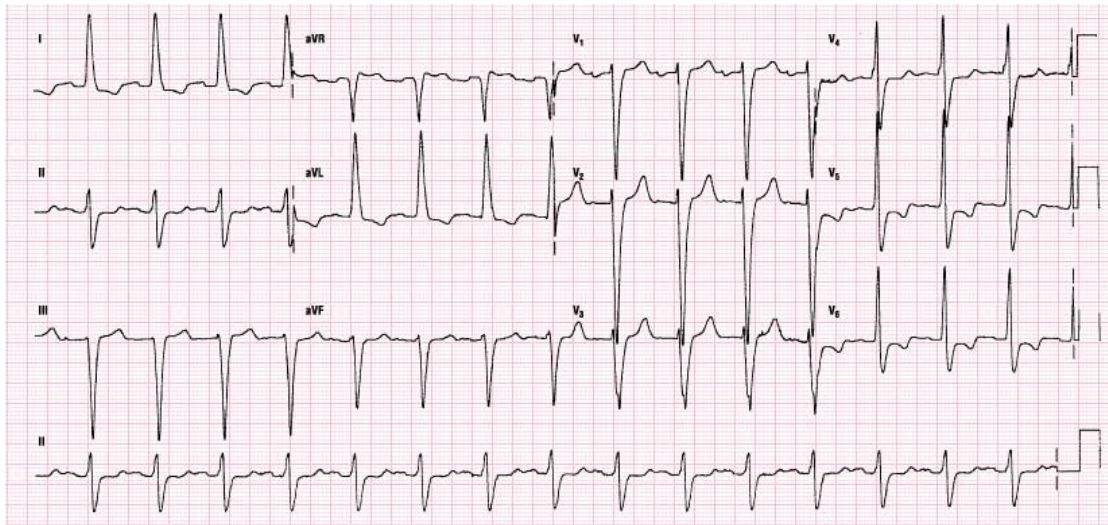


Figure 3.7: Standard 12-lead ECG report.

### 3.2.3 Cardiac arrhythmia

Problems in the normal rhythm of the heart, known as cardiac arrhythmias, occur when the electrical impulses originated at the sinoatrial node are transmitted faster or slower than normal or they do not follow the usual electrical path. Fast beating of the heart (more than 100 beats per minute) are called tachycardia whereas slow beating (less than 60 beats per minute) is known as bradycardia. These numbers always depend on the physical activity and fitness status of the subject.

There are four main types of arrhythmias. Two of them are named according to the location where they occur: ventricular arrhythmias occur in the ventricles while supra-ventricular usually occur in the atria. The other two are extra beats and bradyarrhythmias.

Arrhythmias are due to problems with the electrical conduction system of the heart [88] (Section 3.2.1) and the usual but not unique test for diagnosis include the ECG and the Holter monitor.

Ventricular fibrillation or tachycardia, two kinds of arrhythmias, are life-threatening and require immediate therapy with a defibrillator. Other type of arrhythmias, the most common ones, are not life-threatening but require therapy to prevent further problems such as cardiac failure or cardiac arrest [89]. These are the kind of arrhythmias we are interested to detect in this chapter since the early diagnosis might prevent serious damage to the myocardium.

In the next section, a description of the arrhythmia database used in this application is presented.

## Data base description

We employ the MIT-BIH Arrhythmia Database [90] available at [Physionet](#) which contains 48 ambulatory ECG recordings of half hour each, obtained from 47 subjects studied by the Massachusetts Institute of Technology (MIT) and the Beth Israel Hospital<sup>3</sup> (BIH) Arrhythmia Laboratory between 1975 and 1979. Twenty-three recordings were selected at random from the whole BIH database including a mix population of inpatients (about 60%) and outpatients (about 40%). The remaining 25 recordings were selected to include less common but clinically significant arrhythmia that would not be well-represented in a small random sample. The recordings were digitized at 360 samples per second with 11-bit resolution over a 10mV range. Two cardiologists independently annotated every heart beat in each record.

Each record was obtained using two leads, denoted as lead A and B. In 45 recordings lead A is the modified limb lead *II* (MLII) described in Section 3.2.2, and for the other three is *V5* (See Section 3.2.2). Lead B is represented by *V1* in 40 recordings, and is either lead *II*, *V2*, *V4* or *V5* for the rest of recordings.

The database includes 15 different heartbeat types that are shown in table 3.1. This table also shows the mapping of the 15 heartbeat types into the 5 classes recommended by the AAMI guidelines. In agreement with these guidelines, the four recordings containing paced beats were removed from the database. See Tables A.2 and A.3 of Appendix A for the direct mapping of heartbeats annotations in the MIT-BIH arrhythmia database into the AAMI recommended classes. This appendix contains technical information of the database.

For this study, we followed the database division used by De Chazal, et al. [73] in which the database was divided into two datasets containing approximately the same amount of heartbeats and a mixture of routine and complex arrhythmia recordings. The first dataset (DS1) is used to train the classifier, while the second dataset (DS2) is used for testing. Neither record can belong to both datasets, i.e. they are mutually exclusive. The corresponding identification of records included in each dataset can be consulted in Table A.1 (Appendix A).

To perform heartbeat classification, the ECG signals were down sampled to half of its sampling rate (360Hz) and divided into heartbeats using a fixed-length window of 170 samples around the R-peak (see Fig. 3.4). This particular point of the ECG is annotated in the database. The window was positioned around the maximum peak of the QRS complex to extract the waveform. Seventy samples before the R-peak were extracted to include P waves, and 100 samples after the R-peak were also included to have information about the T wave and the

---

<sup>3</sup>Currently Beth Israel Deaconess Medical Center

Table 3.1: Heartbeat types included in the MIT-BIH arrhythmia database and their mapping into the AAMI heartbeat classes

AAMI label	N	S	V	F	Q
<b>Description</b>	Any heartbeat not in S, V, F or Q classes	Supraventricular ectopic beat	Ventricular ectopic beat	Fusion beat	Unknown beat
	normal beat (Nor)	atrial premature beat (aP)	premature ventricular contraction (PVC)	fusion of ventricular and normal beat (FVN)	paced beat (P)
	left bundle branch block beat (LBBB)	aberrated atrial premature beat (aAP)	ventricular escape (VE)		fusion of paced and normal beat (FPN)
<b>MIT-BIH heartbeat types</b>	right bundle branch block beat (RBBB)	nodal (junctional) premature beat (nP)			Unclassified beat (U)
	atrial escape beats (Ae)	supraventricular premature beat (SP)			
	nodal (junctional) escape beat (NE)				

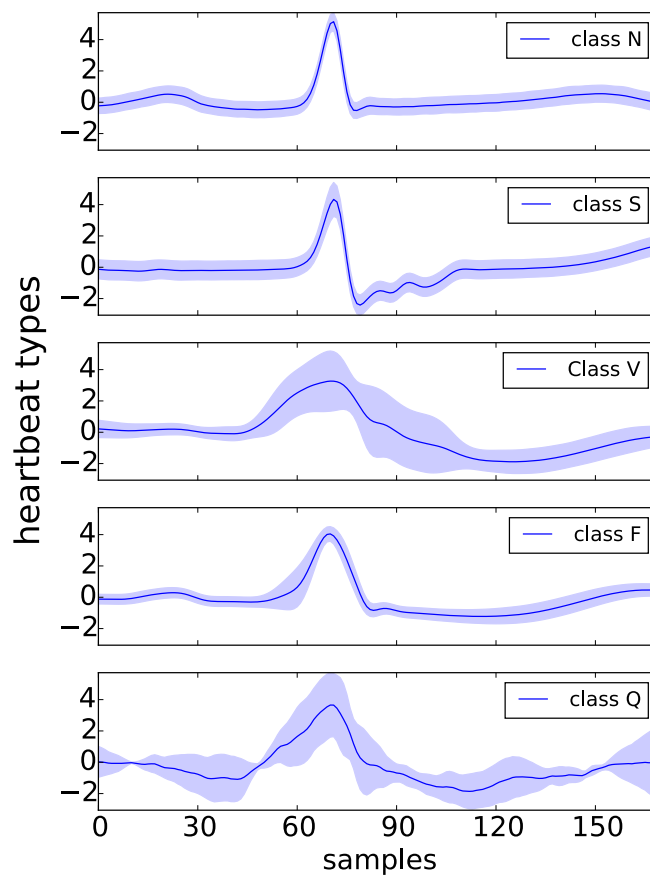
### 3.3. DATA BASE DESCRIPTION

**Table 3.2:** Heartbeat types included in each of the datasets used for training and testing.

MIT-BIH heartbeat type		Nor	LBBB	RBBB	RBBB	Ae	NE	aP	aaP	nP	SP	PVC	VE	fVN	P	fPN	U	
AAMI class		N	N	N	N	N	N	S	S	S	S	V	V	F	Q	Q	Q	total
<b>DS1</b>	number	38104	3949	3783	16	16	808	100	32	2	3682	105	415	0	0	0	8	51020
	% of total	74.1	7.7	7.4	0.0	0.0	1.6	0.2	0.1	0.0	7.2	0.2	0.8	0.0	0.0	0.0	0.0	100.0
	Present in # records	18	2	3	1	3	14	3	3	1	17	1	10	0	0	0	3	—
<b>DS2</b>	number	36444	4125	3476	0	213	1736	50	51	0	3220	1	388	0	0	0	7	49711
	% of total	73.3	8.3	7.0	0.0	0.4	3.5	0.1	0.1	0.0	6.5	0.0	0.8	0.0	0.0	0.0	0.0	100.0
	records	19	2	3	0	2	13	4	2	0	16	1	7	0	0	0	2	—

## CHAPTER 3. CLASSIFICATION OF HEARTBEATS USING RC

duration of the heartbeat. Each heartbeat time trace was then normalized with respect to the full ECG signal to have mean zero and variance one. Since a normal heartbeat lasts about 700ms, using a time window of 170 samples include these kind of heartbeats and other arrhythmic heartbeats that can last longer than the normal. Figure 3.8 shows the mean of the heartbeats of random subjects per AAMI class.



**Figure 3.8:** ECG traces of the mean of the heartbeats of random subjects. In all cases the R-point is located at sample 70. The SD is shown as a shade of the average heartbeat.

In the next section we describe the particularities of our approach. We start by describing the learning process and then we provide a description of the setting of the reservoir.

## The learning process: Logistic regression

The learning process (Section 1.1.2) is the mechanism of the reservoir to learn how to perform a task. In standard RC, it is common to use linear regression methods for this purpose. However, for this particular task we employ a logistic regression learning process. The logistic regression (LR) [91] is a widely used learning technique in biostatistical applications in which binary responses occur quite frequently, e.g. in questions such that a condition is present or absent. LR is specified in terms of *logit* transformations, defined as

$$\text{logit}(P) = \ln(\text{odds}) = \ln\left(\frac{P}{1-P}\right), \quad (3.4)$$

where the odds represent the ratio of the probability  $P$  that an event will occur to the probability that the same event will not occur. In the logistic regression, the aim is to linearly relate the *logit* function with the data  $\mathbf{D}$  finding the values of parameters  $\mathbf{a}$  and  $\mathbf{b}$  that satisfy

$$\text{logit}(\mathbf{P}) = \mathbf{a} + \mathbf{b}\mathbf{D}. \quad (3.5)$$

Consequently, the results can be directly interpreted as the probability of a condition to be true or false according to the following equations

$$\mathbf{P} = \frac{e^{\mathbf{a}+\mathbf{b}\mathbf{D}}}{1 + e^{\mathbf{a}+\mathbf{b}\mathbf{D}}}. \quad (3.6)$$

The shape of this function is depicted in Fig. 3.9 and it is known as the logistic function.

Note that *logit* functions are linearly related to the data  $\mathbf{D}$  (Eq. 3.5), but the probabilities are nonlinearly related to the data  $\mathbf{D}$  (Eq. 3.6). This is an advantage since in classical linear models it is usually assumed that the outcomes are independent and normally distributed with equal variance. These assumptions are often inadequate in medical applications due to redundancy in the data.

The LR model is well distinguished from linear regression and other types of binary regression analysis by the way the probability of a particular sample is linked to the classifier (Eq. 3.5). Besides, the conditional probability  $p(y|x)$  of the classifier is a Bernoulli distribution rather than a Gaussian distribution (linear regression), because the dependent variable is binary. In this way, we recognize the logistic regression as a type of **generalized linear model (GLM)**<sup>4</sup> [92] with the *logit* function as its *link* function.

<sup>4</sup>A possible point of confusion is the general linear model that may be viewed as a special case of the GLM with identity link function and normally distributed responses.

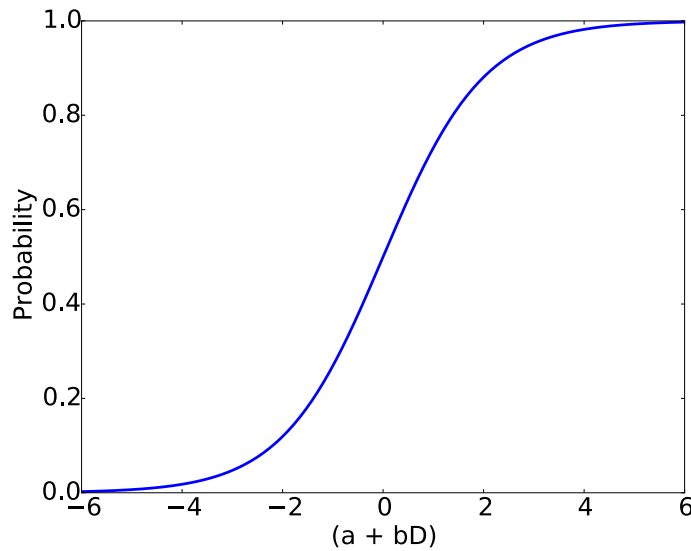


Figure 3.9: Logistic distribution function.

### 3.4.1 Estimation of logistic regression coefficients

Note that both the probabilities and the regression coefficients are unobserved, and the means of determining them is not part of the model itself.

The regression coefficients are usually estimated using maximum likelihood estimation [93]. This is done using an iterative process that searches the coefficient values that best fit the data. Some numerical iterative methods include: iterative reweighted linear least squares (IRLS) [94] or more commonly a quasi-Newton method such as the limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) [95, 96]. The problem with iterative processes is that they might not converge. If this is the case, then the coefficients are not meaningful. Failure to converge may indicate a large ratio of predictors to cases, multicollinearity, sparseness or complete separation [93]. The iterative process is usually subject to regularization conditions (Section 1.1.2) that seek to exclude unlikely values. Due to the iterative nature of these methods, there is a control parameter to be adjusted that refers to the tolerance to errors of the method and how good a new solution has to be in order to be accepted as an improvement.

To ensure that the learning process fully uses all the information included in the input, we reshaped the state matrices (Section 2.1.2) with dimensions  $M \times N$  to another matrix with dimensions  $(M \cdot N) \times 1$ . This modification provides a fine adjustment of weights. In the previous chapters when we used linear regression, the total number of weights was the same number of nodes ( $N$ ). In our modification of the LR, the total number of weights to be computed is the number of observations times the number of nodes of the reservoir ( $M \cdot N$ ). This



### 3.5. CLASSIFICATION OF HEARTBEATS USING LOGISTIC REGRESSION

is not a requirement, but we have chosen this methodology for a fine adjustment of each feature of the input signal.

3.5

---

## Classification of heartbeats using logistic regression

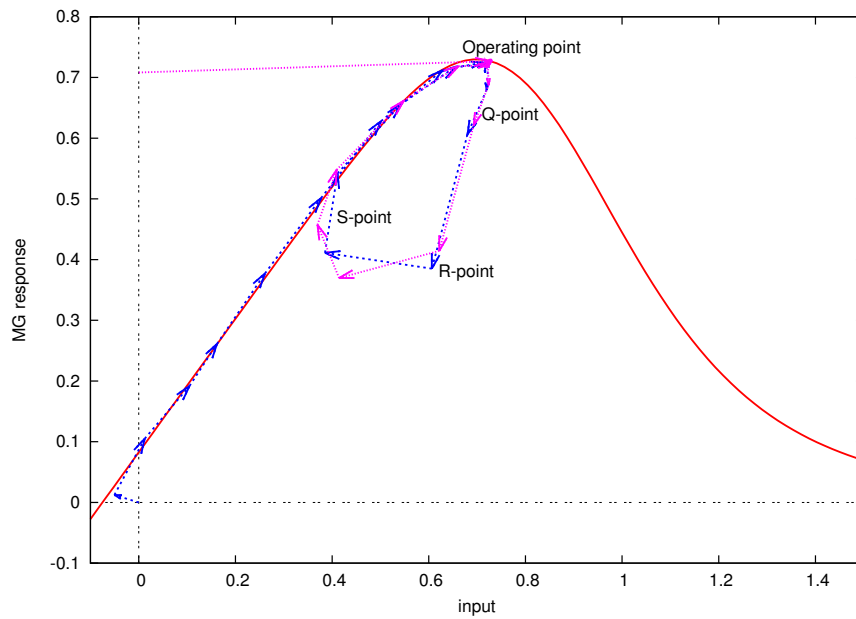
To perform this classification task, the Mackey-Glass nonlinearity is used as the [NLN](#) of the reservoir. The exponent  $p$  is chosen to be  $p = 7$ , since we expect that a high nonlinearity will be beneficial for separating classes while not much memory is required (See Section [2.3.1](#)). We follow a similar procedure than the one described before for the diagnosis of learning algorithms (Section [1.2](#)) with some adaptations to be suited to medical applications.

### 3.5.1 Results and discussion

The heartbeats to be analyzed in this task (see for instance Fig. [3.8](#)) contains an abrupt change of trend at the QRS complex. The transitions caused by the QRS complex can be observed using a similar plot to the one utilized for the [SDR](#) task (Right panel of Fig. [2.19](#)). In Figure [3.10](#), the Mackey-Glass transfer function for  $p = 7$  is depicted in red. Arrows show the path followed by the input to reach the operating point. In blue, we show the path when the reservoir is initialized at the zero state, whereas in magenta we show the path when the reservoir starts using the transient of a previous heartbeat. The operating point lies around  $\sim 0.72$ . This figure shows an approximate position of Q, R and S points. It can be observed that the QRS complex causes the [NLN](#) to create transients that go outside the transfer function curve during few delay loops. When the [NLN](#) goes back to the transfer function, it is far from the operating point and needs several delay loops to go back to that point. Similar to what we observed in Section [2.4.1](#) we found that these transients dynamical paths are used by the reservoir as a feature to distinguish instances in a classification task.

We now proceed to build the model complexity curve. Since we have already set the degree of the nonlinearity, we only need to evaluate the number of virtual nodes needed for a proper heartbeat classification using [DS1](#). To do so, we compute the error rate ([ER](#)), i. e. the number of misclassified heartbeats over the total number of heartbeats, for different number of virtual nodes. We utilize logistic regression as the learning process with a tolerance to errors of  $10^{-6}$ . As usual, two error rates are computed: the training and the testing [ER](#). To perform [cross-validation](#), we used a patient oriented scheme consisting in

## CHAPTER 3. CLASSIFICATION OF HEARTBEATS USING RC

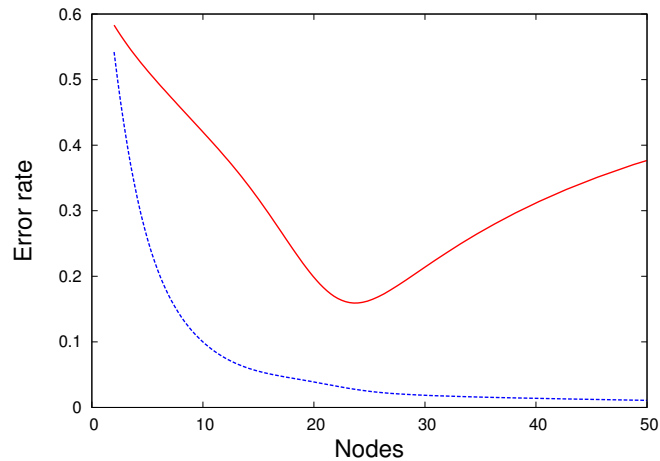


**Figure 3.10:** Mackey-Glass transfer function and the ECG path. Arrows show the path followed by the input signal. In blue, the path when the reservoir initializes at zero, whereas in magenta when it initializes at the transient of a previous input signal.

training with  $k - 1$  patients, where  $k$  is the total number of patients in DS1, and leaving one out. This is equivalent to the leave-one-out (Section 1.2.2) CV but considering patients instead of samples. The training error rate is the rate of misclassifications over  $k - 1$  patients used for training. The testing error is then the ER over the patient that was taking out during the training phase. Figure 3.11 shows the ER as a function of the number of virtual nodes.

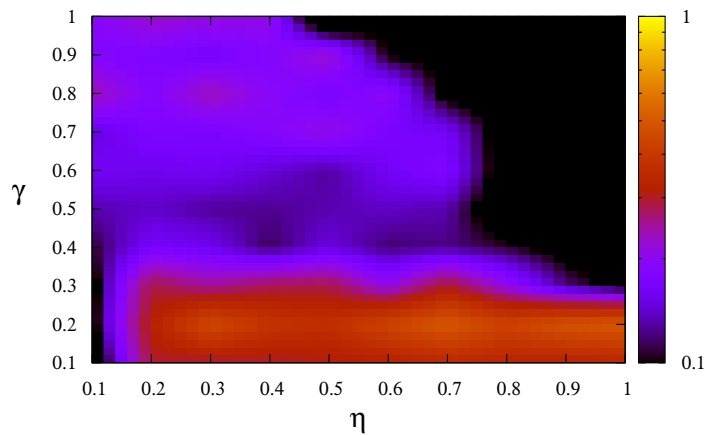
For a number of nodes less than 25 the classifier suffers a high bias problem. This scenario is recognized because both ERs, training (dash line) and testing (continuous line), are high. In contrast, for a number of nodes greater than 25 the classifier suffers a high variance problem noticeable by the low training ER and the high testing ER. High variance problem means that the reservoir is overfitting the training data and it is not capable to generalize. For 25 virtual nodes, we found a trade-off between these two problems indicating the optimal number of nodes. The gap between the training and testing ERs at this point indicates a high variability of ECG morphology that could be, in principle, reduced by increasing the number of samples. Looking at Table 3.2 however, we see that the number of samples is sufficient. Therefore, the gap between the two ER curves is due to the variability in the morphology of the heartbeats that it is different for different subjects. The DS1 dataset contains only 22 recordings from different patients causing a gap between the ERs at the optimal point.

### 3.5. CLASSIFICATION OF HEARTBEATS USING LOGISTIC REGRESSION



**Figure 3.11:** Model complexity. Dependence of the training (dashed line) and testing (continuous line) error rates as a function of the number of virtual nodes in the reservoir.

This indicates that our approach is sensitive to the number of subjects. In what follows, a total number of 25 virtual nodes are used for the reservoir.

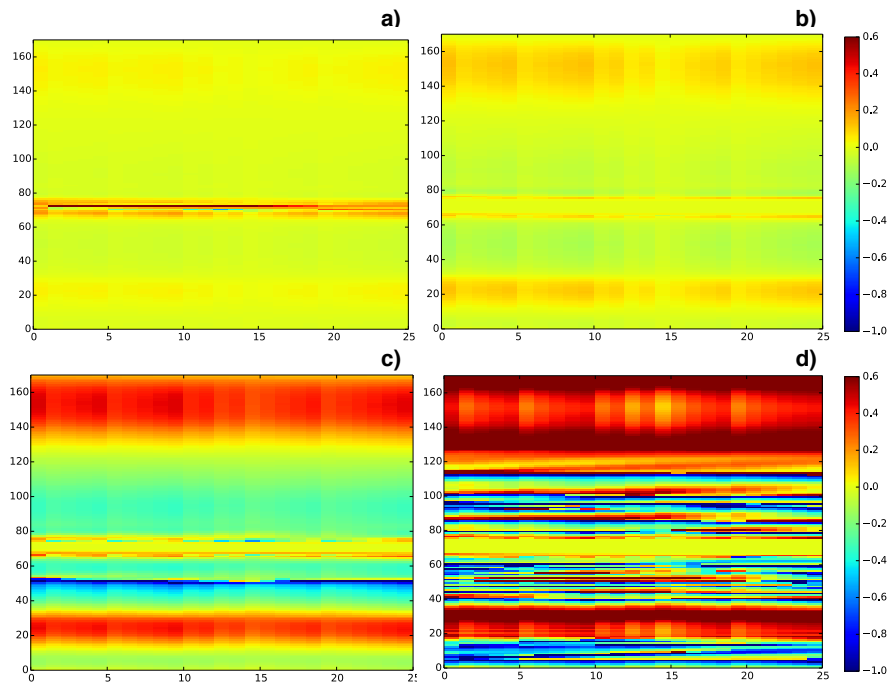


**Figure 3.12:** Test Error Rate as a function of parameters  $\eta$  and  $\gamma$ . The exponent  $p$  was chosen to be  $p = 7$ . We choose as optimal parameters to run the reservoir  $\eta = 0.8$  and  $\gamma = 0.5$ .

We now explore the dependence of the testing ER with the Mackey-Glass (Eq. 2.4) system's parameters  $\eta$  and  $\gamma$  using DS1 with patient-oriented CV. The parameter space  $\eta - \gamma$  is shown in Figure 3.12 in a logarithmic color scale.

## CHAPTER 3. CLASSIFICATION OF HEARTBEATS USING RC

In this figure it can be noted that a good performance of the reservoir is obtained, for a high value of the feedback strength  $\eta$  and a relative high value of the input scaling. When using parameters  $\eta = 0.8$  and  $\gamma = 0.5$ , it results in an error rate of  $\sim 8\%$ . In order to illustrate the influence of parameters  $\eta$  and  $\gamma$ , we have plotted a state matrix at different values of  $\eta$  and  $\gamma$  for comparison. Results are shown in Fig. 3.13. All state matrices correspond to the average normal



**Figure 3.13:** State matrix for different values of  $\eta$  and  $\gamma$ . In a),  $\eta = 0.4$  and  $\gamma = 0.2$ . In part b),  $\eta = 0.3$  and  $\gamma = 0.8$ . In part c),  $\eta = 0.8$  and  $\gamma = 0.5$ , and finally part d) has values of  $\eta = 1.2$  and  $\gamma = 1.2$ .

heartbeat (Class N) shown in Fig. 3.8. To interpret these matrices, it must be remembered that the vertical axis represents the time steps of the ECG signal with the R-point at sample 70 and the horizontal axis represents the state of the virtual nodes in the reservoir. Color scale is the same for the four matrices. Figure 3.13.a) illustrates the fact that for this combination of parameters the reservoir highlights the QRS complex skipping the information in the rest of the ECG signal. In part b), the reservoir starts to extract information around the QRS complex and still highlights the points around Q and S while the R-point has low meaning. Waves P and T start to have significance in the state matrix. Figure 3.13.c) is at the optimal configuration of the reservoir to perform this task. In this case, the R-point is not highlighted but the surroundings of the QRS complex receive higher intensities than before. In addition, another important point (shown in blue), corresponding to the inflection point between the P-wave and the Q-point of the ECG. The last state matrix, Fig. 3.13.d), shows a saturation

### 3.5. CLASSIFICATION OF HEARTBEATS USING LOGISTIC REGRESSION

on the values of the [ECG](#) features. Error rates for this kind of matrices are high. In the upcoming computations, the parameters  $\eta = 0.8$  and  $\gamma = 0.5$  were used.

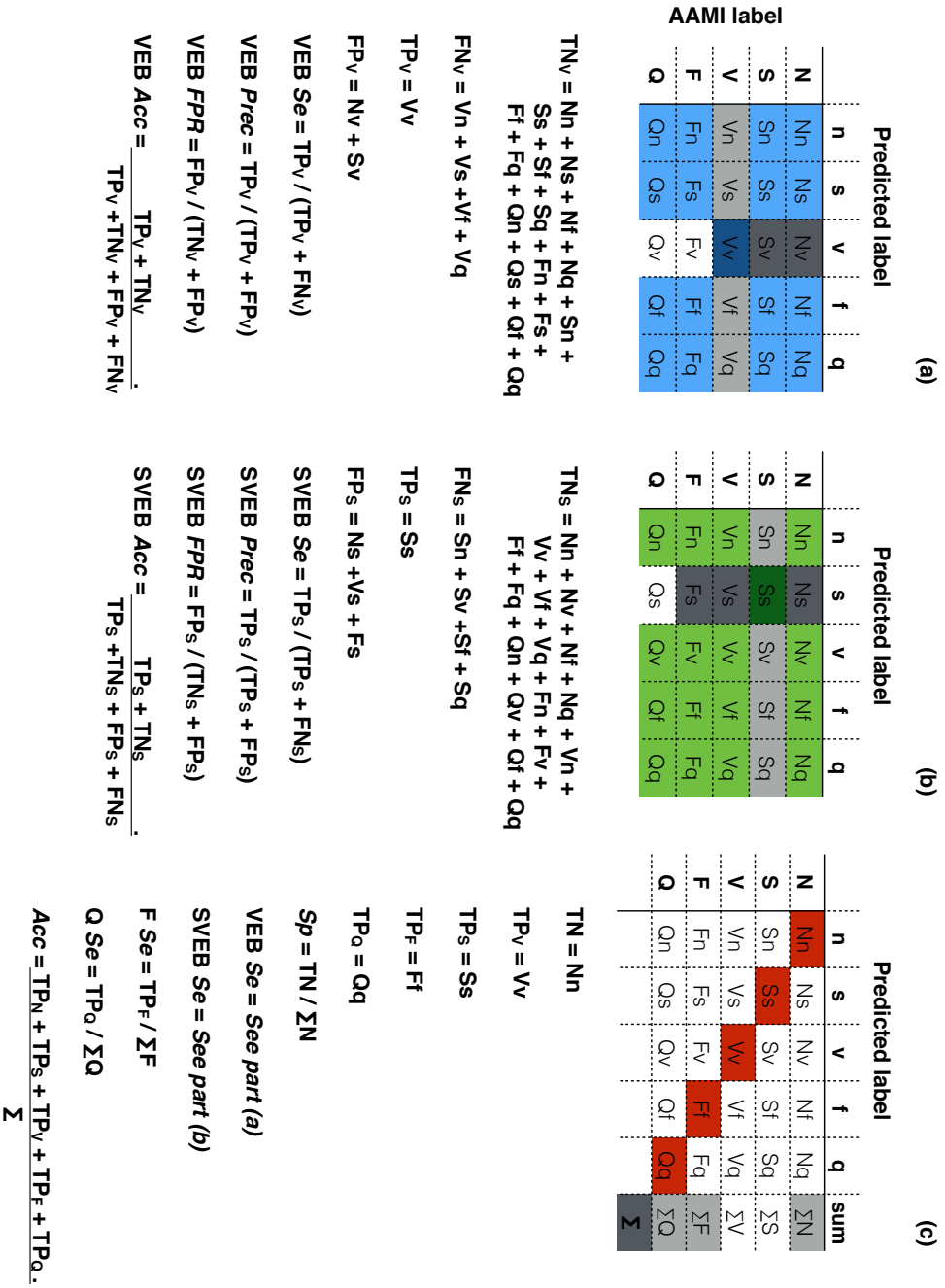
To compare the performance with other published results is a difficult task. One of the difficulties is due to the definition of classes without considering the recommendations of the [AAMI](#). Another problem is related to the measures of performance computed and the equivalence among them. Here, we compute the most common ones for comparison purposes.

In our approach, we use the [AAMI](#) guidelines which prompt to combine the heartbeat types (see table 3.1) into five classes, namely, N, S, V, F and Q. Classes can contain more than one heartbeat type. Class N refers to normal beats, class S is the supra-ventricular ectopic ([SVEB](#)) class. A ventricular ectopic ([VEB](#)) class is known as class V, while class F represents fusion of ventricular and normal beats. The last class, class Q, contains the unclassified beat in the database. For performance reasons, the [AAMI](#) recommends to compute some measures that focus on the ability of the algorithm to distinguish [VEB](#) from non-[VEB](#), and [SVEB](#) from non-[SVEB](#). In Fig. 3.14 we show the measures as they are defined by the [AAMI](#) for the evaluation of arrhythmia detectors. In the remaining of the chapter, the measures are computed using [DS2](#). The results, shown in Table 3.3, are giving as recommended by the [AAMI](#) and characterized by a high accuracy ( $Acc > 97\%$ ) and low false positive rate ( $FPR < 2\%$ ). Some of the performance measures were not possible to determine due to zeros in the denominator. In the last row of this table, we compare our results with the best performance reported by De Chazal et al. [73] using feature extraction and linear discriminants ([LD](#)). Although their approach is different to the one described here, they utilize the same database configuration.

Note that the [AAMI](#) recommended measures focus the attention on the classification of ventricular ([VEB](#)) and supra-ventricular beats ([SVEB](#)). Table 3.4 shows the results obtained using the [AAMI](#) guidelines for class definition but considering a multi-class scenario, i.e. considering the 5 possible classes which is a more clinically significant scenario.

In order to compute performance measures for each class, we separate the multi-class confusion matrix into binary confusion matrices. Then, measures such as sensitivity, specificity, accuracy and precision are computed using the typical definitions, see Section 1.3.

Table 3.4 shows the performance of our system per class. In our approach all measures are higher than 84%. Sensitivity is particularly significant for clinical use since it accounts for the percentage of true positive samples that were classified correctly, i.e. the percentage of arrhythmic heartbeats that were actually detected by the reservoir computer. It is worth to remember that for this database no preprocessing or feature extraction were made over the signal, so a raw, noisy [ECG](#) waveform is being used for training and classifying. Table



**Figure 3.14:** Binarized measures of performance to (a) discriminate VEB from non-VEB, (b) SVEB from non-SVEB and (c) the five heartbeat classes. The measures correspond to those described in Section 1.3. The colored cells represent the groups that are important in each measure.

### 3.5. CLASSIFICATION OF HEARTBEATS USING LOGISTIC REGRESSION

**Table 3.3:** Performance measures (in %) recommended by the AAMI for the evaluation of classifiers. Measures are calculated using [DS2](#).

Rec	number of beats					SVEB				VEB			
	N	S	V	F	Q	Acc (%)	Se (%)	Prec (%)	FPR (%)	Acc (%)	Se (%)	Prec (%)	FPR (%)
100	2239	33	1	0	0	99.6	93.9	79.5	0.4	100.0	100.0	100.0	0.0
103	2082	2	0	0	0	99.6	50.0	12.5	0.3	100.0	–	–	0.0
105	2526	0	41	0	5	97.2	–	0.0	2.8	99.9	95.1	100.0	0.0
111	2123	0	1	0	0	99.7	–	0.0	0.3	100.0	100.0	100.0	0.0
113	1789	6	0	0	0	97.1	83.3	8.9	2.8	100.0	–	–	0.0
117	1534	1	0	0	0	97.3	100.0	2.4	2.7	100.0	–	–	0.0
121	1861	1	1	0	0	98.5	100.0	3.6	1.5	100.0	100.0	100.0	0.0
123	1515	0	3	0	0	99.6	–	0.0	0.4	99.9	66.7	100.0	0.0
200	1743	30	826	2	0	99.3	74.4	20.9	0.6	99.6	98.8	100.0	0.0
202	2061	55	19	1	0	98.9	80.0	80.3	0.5	99.8	73.7	100.0	0.0
210	2423	22	195	10	0	98.0	81.8	28.6	1.8	99.4	92.3	100.0	0.0
212	2748	0	0	0	0	99.5	–	0.0	0.5	100.0	–	–	0.0
213	2641	28	220	362	0	97.3	78.6	22.7	2.5	99.7	95.5	100.0	0.0
214	2002	0	256	1	2	99.8	–	0.0	0.2	99.5	95.5	100.0	0.0
219	2082	7	64	1	0	93.9	85.7	4.5	6.0	99.4	81.2	100.0	0.0
221	2031	0	396	0	0	97.8	–	0.0	2.2	99.3	96.0	100.0	0.0
222	2274	209	0	0	0	93.7	59.8	61.6	3.3	100.0	–	–	0.0
228	1688	3	362	0	0	94.6	100.0	3.2	5.4	99.3	96.1	100.0	0.0
231	1568	1	2	0	0	98.0	100.0	3.1	2.0	99.9	50.0	100.0	0.0
232	398	1382	0	0	0	87.3	81.3	100.0	0.0	100.0	–	–	0.0
233	2230	7	831	11	0	97.2	98.4	48.4	2.8	99.0	96.4	100.0	0.0
234	2700	50	3	0	0	98.5	84.0	56.0	1.2	100.0	66.7	100.0	0.0
sum	44258	1837	3221	388	7								
average						97.4	84.5	24.4	1.8	99.8	87.7	100.0	0.0
<b>Ref</b>													
[73]						94.6	75.9	38.5	4.7	97.4	77.7	81.9	1.2

[3.4](#) also shows the average performance reported by De Chazal et al. [\[73\]](#) and Llamedo et al. [\[74\]](#) using [DS1](#) for training and [DS2](#) for testing. Reported performance was computed using their confusion matrices and separating them into binary matrices to be consistent with our way of computing performance. De Chazal et al. [\[73\]](#) used feature extraction and linear discriminants ([LD](#)) to construct an automated heartbeat classifier. Although several configurations of the database were considered, we compare our results with those of the best performance classifier. Unlike De Chazal et al., Llamedo et al. [\[74\]](#) utilized feature selection and signal modeling in order to build their classifiers. They

## CHAPTER 3. CLASSIFICATION OF HEARTBEATS USING RC

**Table 3.4:** Results for the multi-class classification problem using the AAMI guidelines for class grouping. Comparison with other methods using DS2.

Class	Sensitivity (%)	Specificity (%)	Accuracy (%)	Prec (%)
N	96.82	91.89	96.28	98.98
S	79.37	96.93	96.28	49.80
V	96.06	99.97	99.71	99.49
F	92.26	99.97	99.91	95.47
Q	57.14	100.00	99.99	100.00
average				
<b>This thesis</b>	<b>84.83</b>	<b>97.75</b>	<b>98.43</b>	<b>88.75</b>
LD [73]	65.95	96.04	94.35	45.57
LD/SM [74]	83.33	93.67	89.00	58.25

used several databases, including the arrhythmia MIT-BIH database, however modifying the AAMI guidelines. They discarded AAMI class Q arguing that it is marginally represented in the database. They also merged AAMI classes F and V into a ventricular class (V'). Llamedo's work is the first attempt to classify heartbeats across databases.

In this case and using the approach described in this chapter, the resulting performance of our designed reservoir computer is higher than those in literature using the AAMI guidelines and the MIT-BIH Arrhythmia database.

**Table 3.5:** Confusion matrix for the AAMI-classes classification problem using DS2

		Predicted samples				
		N	S	V	F	Q
Known samples	N	42852	1406	0	0	0
	S	379	1458	0	0	0
	V	53	60	3094	14	0
	F	10	4	16	358	0
	Q	0	0	0	3	4

Table 3.5 shows the multi-class confusion matrix of DS2 using the 5 AAMI classes. This table provides information on the misclassification of the different classes and serves as the base for future comparisons.



---

## Conclusions

In this chapter, the RC technique was adapted to study the open problem of automatic classification of heartbeats with cardiac arrhythmia. The adaptation includes a learning process known as logistic regression that is capable of providing the probability that a heartbeat belongs to a particular class. For the MIT-BIH arrhythmia database, no additional pre-processing or feature extraction of the ECG signal was needed. Only the usual masking was applied to the input signal to build the input matrix  $I$ . Our approach requires an approximate position of the R-point to lock the time-window that serves to separate the heartbeats. An extension to a different database that do not have annotations relative to the R-point will require a computationally inexpensive QRS detector. Since there is no feature selection step in our algorithm, the classification is based only on the morphology of the heartbeat.

The drawback of our approach is that the modifications to the way of training the logistic regression (transformation of the state matrices into vectors), that provide a fine setting of the weights, also come with a constrain: the heartbeats must be time-locked one respect to the other. In our approach we did this by locking the R-point at a particular position of the time window. A generalization of our approach to consider for instance heartbeats of people at rest and while doing exhausting physical activity will need to deal with the problem of adapting the size of the time window. The solution to this issue might come from registering the R-R interval, which is the interval of time between two consecutive R-points in the ECG.

The final configuration used in this classification task uses a reservoir computer that still suffers high variance problems due to the variability in the heartbeat wave morphology among the different subjects. This high variance problem might be diminished including more subjects in the database.

In this chapter, we presented how reservoir computing can identify significant features in the input signal for the classification of ECG signals with cardiac arrhythmia. Our results show that the QRS complex is not the most important feature to consider, but the information in its surroundings. Supra-ventricular arrhythmia occurs before the QRS complex while ventricular occurs during and/or soon after it.

We employed international standard recommendations to group heartbeat types into five classes and several performance quantities were computed. Despite the different criteria mentioned in this work to facilitate comparisons with other studies, we consider that using the AAMI guidelines and balanced datasets for training and testing are essential for comparing algorithm performance among

## CHAPTER 3. CLASSIFICATION OF HEARTBEATS USING RC

research teams. Our results on classifying **VEB** from non-**VEB** and **SVEB** from non-**SVEB** are very promising and are equivalent to those reported in literature.

For a more clinically relevant application, we performed a multi-class classification with results that indicate an average specificity of 97.75% with an average accuracy of 98.43%. Sensitivity and precision reach an average of 84.83% and 88.75%, respectively. Giving the significance of sensitivity in clinical applications, our results are equivalent to the best previously reported results. Part of the results presented in this chapter was recently published in the article:

M.A. Escalona-Morán, M.C. Soriano, I. Fischer, C.R. Mirasso. *Electrocardiogram classification using reservoir computing with logistic regression*. IEEE J. Biomedical and Health Informatics, 19, issue 3, pp. 892-898, (2015).

The implementation described in this chapter highlights the potential of the reservoir computing technique in the heartbeat classification problem and the hidden computational power contained in a single node reservoir subjected to delay feedback. The results reported in this chapter were obtained using a single **ECG** lead. The analysis of additional leads could provide a better understanding of the relevance of each lead for a proper heartbeat classification.

# Multivariate delayed reservoir computers

Estimation and classification of multivariate time series has become a mandatory task in many fields of science including neuroscience, genetics, economy, communications technology, social sciences and others. If the generating system is deterministic, it may be possible to reproduce or approximate the dynamics of the system with a constructed model. According to the Takens embedding theorem [97], a single variable of a multivariate time series is sufficient to recover the underlying dynamics, given that the variables are coupled. However, due to noise and other factors, this might not apply for real data and time series estimation and classification might benefit from the use of additional measurements [98]. In this chapter, we explore the benefits of using multiple data measurements for time-series prediction of the Lorenz attractor and for the classification of electrocardiographic signals. We present numerical simulations showing that the use of multiple variables or data measurements can significantly improve prediction and classification, when using a delayed reservoir computer.

## 4.1 ---

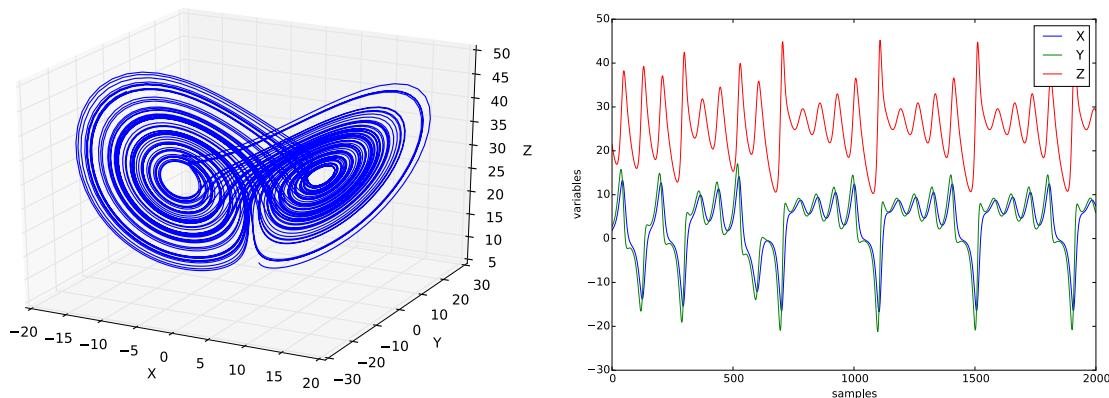
### Time series estimation: the Lorenz system

The prediction of chaotic time series is a demanding task due to the sensitive dependence on initial conditions and the intricate geometric structures of the corresponding chaotic attractors. In this example, we consider time series generated by the Lorenz system [99] described by

$$\begin{aligned}\dot{x} &= \sigma(y - x), \\ \dot{y} &= x(\rho - z) - y, \\ \dot{z} &= xy - \beta z.\end{aligned}\tag{4.1}$$

## CHAPTER 4. MULTIVARIATE DELAYED RESERVOIR COMPUTERS

In our particular case, we choose parameters  $\sigma = 10$ ,  $\rho = 28$ , and  $\beta = 8/3$ . Figure 4.1 shows the dynamics of the Lorenz system. The left panel shows its 3D representation while the right panel shows a portion of the time series used in the left panel. Notice that when the system is in one lobe of the 3D attractor  $x$  and  $z$  are positively correlated and when the system is in the other lobe  $x$  and  $z$  are negatively correlated.



**Figure 4.1:** The left panel shows the Lorenz attractor. The right panel shows part of the time series used in the left panel.

In order to perform the time series estimation, different numerical realizations of the Lorenz system are computed using random initial conditions and an integration time step of  $\Delta t = 0.01$ . Then the time series are down-sampled by a factor 20, resulting in a final sampling step of 0.2. For our analysis, each time series has a final length of  $10^3$  points. It is worth mentioning that results shown in this chapter are for the above mentioned length of the time series since results are dependent on the signal's length.

To build the estimator, 25 time series were generated from Eq. system 4.1 using different initial conditions and evaluated in a 25-random-fold [cross-validation](#).

As it is typical in [RC](#), a linear regression method is used to compute the weights of the virtual nodes. For this task, 400 virtual nodes were used to capture the dynamics of the system. We consider two cases: univariate and multivariate estimators. In the univariate case, the mask contains two values  $\{-0.01, 0.01\}$  assigned randomly to the corresponding virtual nodes. In the case of time series prediction using two variables of the Lorenz system as input to the reservoir, the mask then contains three different values  $\{-0.01, 0.0, 0.01\}$  assigned randomly following a proportion of  $[30, 40, 30]\%$  respectively. Note that the mask is a matrix of dimensions  $2 \times N$  in the case of 2-dimensional input signals.

We use the Mackey-Glass nonlinearity (Eq. 2.4) with exponent  $p = 1$  since it allows for a long fading memory in the reservoir (described in Sections 2.2.4 and 2.3.2), which is important in the context of time series prediction.

## 4.1. TIME SERIES ESTIMATION: THE LORENZ SYSTEM

For the prediction of the Lorenz time-series, the parameter  $\eta$  is chosen as  $\eta = 0.45$ , after exploring the parameter space  $\eta - \gamma$  and finding a good performance for this parameter value. To measure the accuracy of our prediction, we compute the Normalized Mean Square Error (NMSE) which is an estimator of the overall deviations between predicted and measured values. The NMSE is defined as,

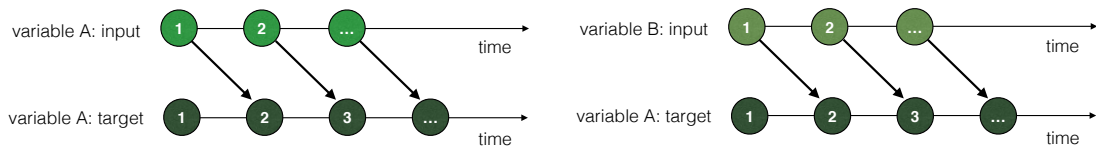
$$NMSE = \frac{1}{m} \frac{\sum_{k=1}^m (target_k - input_k)^2}{\sigma^2(t_k)}, \quad (4.2)$$

where  $m$  is the number of samples in the time series: *input* represents the original input signal and *target* is the predicted time series.  $\sigma$  denotes the standard deviation (SD).

Time series prediction is often performed using a single variable since other variables are usually hidden or not accessible. However, we will show below that if one has access to other variables of the system, the prediction could be significantly improved. For the Lorenz system, we concentrate on the variables  $x$  and  $y$  since they move between the two wings of the Lorenz strange attractor.

### 4.1.1 One-step prediction

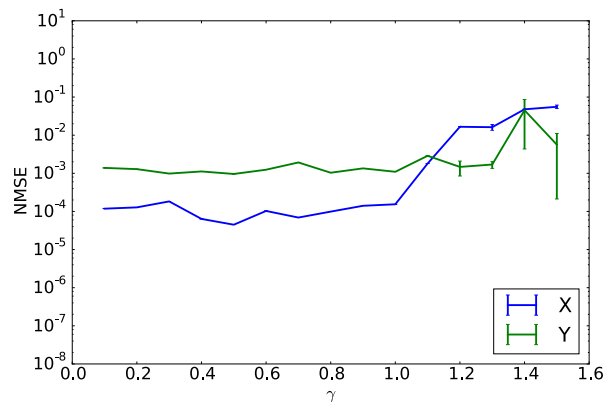
In this section, we estimate the upcoming value of a time-series A based on two cases: In the first case, we use variable A to estimate its own evolution, i.e. the target of the reservoir is the same time series shifted, in this case, by one time step (see left panel of Fig. 4.2). The second case is to use variable B to estimate variable A (right panel of Fig. 4.2). By shifting the target time series by one, the reservoir learns how to predict the next (future) point in the time series given the current (present) point. We use the scheme depicted in Fig. 4.2 to perform



**Figure 4.2:** Univariate one-step prediction scheme. The target time series is shifted by one time step. The input and target time series might belong to different variables of the system (right).

univariate one-step prediction of the Lorenz system. As variable A we take the  $x$  variable and as variable B we take the  $y$  variable.

We study two scenarios: the first one being the case in which variable- $x$  time trace is used for training and testing. The second scenario considers the time series of  $y$  to predict  $x$ . It is worth to remember that during the testing phase the time series of variable  $x$  are different than the time series used for training, since they



**Figure 4.3:** NMSE as a function of  $\gamma$  for the estimation of variable  $x$ . Green curve corresponds to the NMSE when using only  $y$  variable. Blue curve corresponds to the NMSE of estimating  $x$  when using only  $x$  variable. Error bars were computed as the SD of the 25 different folds.

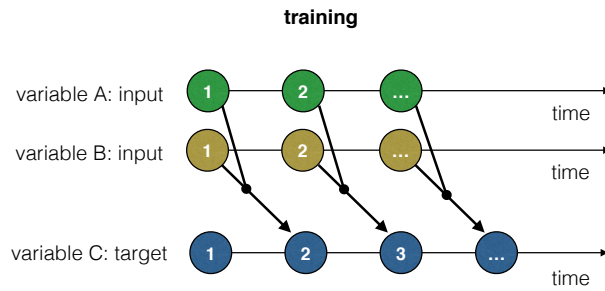
are produced using different initial conditions. To evaluate the performance, we use the NMSE, Eq. 4.2. In Figure 4.3 we plot the NMSE of the one-step prediction task as a function of the input scaling  $\gamma$  for the prediction of variable  $x$ . The first case, when variable  $x$  is used to predict its own dynamics, is shown in blue. The second case, where variable- $y$  time series is used to predict variable- $x$  dynamics, is shown in green. Vertical lines denote the SD.

The first observation is that the results do not depend much on the values of  $\gamma$  ( $< 1$ ). This highlights the robustness of the results to changes in parameter values. For  $\gamma > 1$ , results degrade since the inputs go far away from the fix-point (vertical section in Fig. 2.5), provoking dynamical transients that take too long to return to the resting state of the NLN. This fact could be compensated by reducing the values of the mask, i.e. re-scaling the input matrix  $\mathbf{I}$ . As it is expected, predicting variable  $x$  using its own dynamics performs better than using a different variable of the system,  $y$  in this case.

**One-step multivariate prediction** Let us now consider the multivariate time-series prediction. In this case, several variables can be used to estimate the values of a time series. The two-variable case is depicted in Fig. 4.4. Despite we explore the case of two variables in this chapter, the extension of this problem to several variables is straightforward. In this section, we study the case in which variables  $A$  and  $C$  are equal.

In 1981, Floris Takens showed that one can generically reconstruct a shadow version of the original manifold of a coupled system simply by looking at one of its time series projection [97]. The reconstructed manifold preserves essential mathematical properties of the original system, e.g. the topology of the mani-

#### 4.1. TIME SERIES ESTIMATION: THE LORENZ SYSTEM



**Figure 4.4:** Multivariate one-step prediction scheme. Information from variables A and B are used by the reservoir to estimate the future value of variable C. The inputs and target time series might belong to different variables of the system.

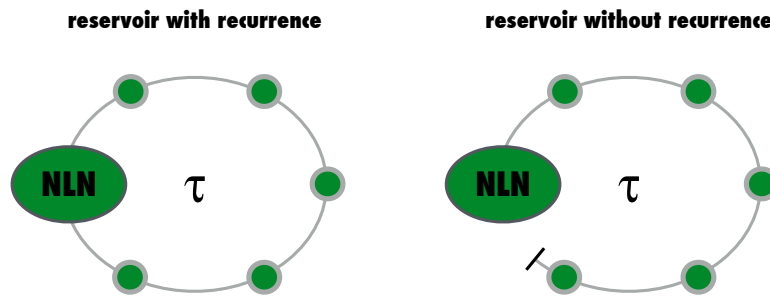
fold and its Lyapunov exponent. The associated method of the Takens theorem represents a one-to-one mapping between the original manifold and the reconstructed one allowing us to recover states of the original system by using a single time series. Then, our first approach of the multivariate estimation task is based on using several delayed versions of one of the variables of a coupled system. In this case, variable  $x$  of the Lorenz system and its delayed versions, denoted by  $x(t - T)$ , are used as input of the reservoir. In this case  $T$  represents the delay time associated to the Lorenz strange attractor.

In the reservoir scheme that we are using, i.e. ring topology with delay feedback, the dynamical degrees of freedom for delay systems are distributed along the reservoir's delay line. Due to the feedback of the reservoir, the delay line of an input sample affects the nonlinear node response of the following input sample creating an intrinsic memory in the system, the so-called fading memory (Section 2.2.4).

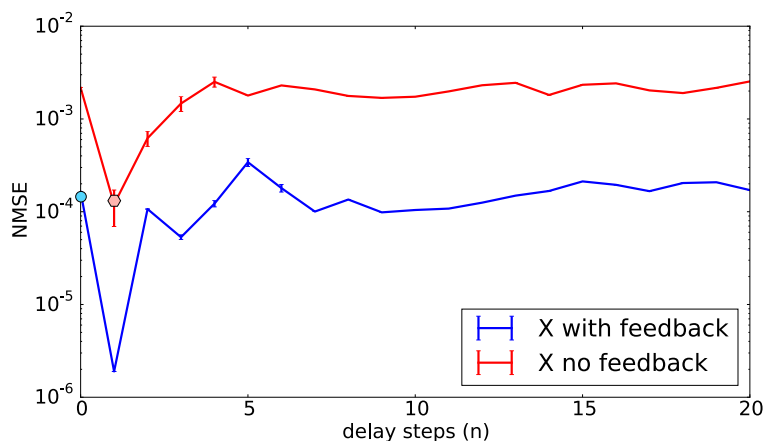
According to Takens theorem, we could reconstruct the dynamical states of a coupled system using one of its variables and its delay versions. Could we get better performance in the estimation of a chaotic time series by using delay versions of variable  $x$ ? Is the intrinsic fading memory of the reservoir playing an important role in the estimation task? In Fig. 4.5 we show a scheme of a reservoir with recurrence (left), e.g. ring topology with feedback, and a reservoir without recurrence (right), i.e. no feedback. This means that the transients generated by previous input samples are not re-injected into the nonlinear node eliminating the intrinsic memory of the reservoir.

We perform numerical simulations under two different conditions: our normal configuration of the NLN as stated in Eq. 2.4 (Fig. 4.5, left), and a reservoir with no feedback, i.e.  $x(t - \tau) = 0$  in Eq. 2.4 (Fig. 4.5, right).

## CHAPTER 4. MULTIVARIATE DELAYED RESERVOIR COMPUTERS



**Figure 4.5:** Left: reservoir with recurrence, i.e. as it has been used in all this thesis. Right: reservoir with no recurrence, i.e. the transient dynamics of the previous input samples do not affect the dynamics of new input samples.



**Figure 4.6:** **NMSE** for different delay steps for a reservoir with feedback (blue line) and another reservoir with no feedback (red line). Error bars were computed as the **SD** of the 25 different folds.

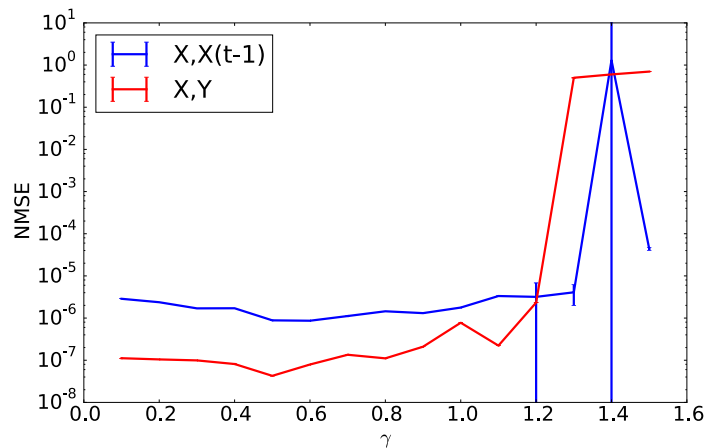
Figure 4.6 shows the **NMSE** for the prediction of variable  $x$  with and without feedback using  $x$  and a delayed version of  $x$ , i.e.  $x(t - nT)$  with  $T$  being the delay time associated to the Lorenz system. Note that the simulations are always using two variables ( $x(t)$  and  $x(t - nT)$ ) and not the accumulation of all the delay versions of  $x$ . The horizontal axis represents the number of points,  $n$ , taken for delay, i.e. 0 means  $x(t - 0T)$ , 1 means  $x(t - 1T)$ , and so on. In the case of  $x(t - 0T) = x(t)$  a univariate estimation is performed to avoid the determinant of the observable matrix to be zero in the learning process (Section 1.1.2). In red we plot the results for a reservoir with no feedback, whereas in blue is the **NMSE** for our standard configuration of the reservoir, i.e. with feedback. We observe that the **NMSE** in the standard configuration (blue line) is consistently lower than the reservoir without feedback. Interestingly, it can be seen that the **NMSE** when using  $x(t)$  with delay feedback (blue curve at  $n = 0$ ) is comparable to the



## 4.1. TIME SERIES ESTIMATION: THE LORENZ SYSTEM

NMSE when using  $x(t)$  and  $x(t - T)$  without feedback (red curve at  $n = 1$ ). This means that the intrinsic fading memory caused by the feedback of the reservoir already includes equivalent information than the one carried out by  $x(t - T)$ .

**Prediction with a different variable** Could the introduction of variable  $y$  add essential information to the prediction of  $x$ ? In Fig. 4.3, we showed that  $y$  predicting  $x$  yields relatively large prediction errors. But what if we estimate variable  $x$  from the combination of  $x$  and  $y$ ? In Figure 4.7, it is shown that the NMSE significantly reduces with respect to Fig. 4.3, reaching values of around  $10^{-7}$  for  $\gamma < 1$ . When comparing the results of predicting variable  $x$  using  $x$  time trace (Fig. 4.3, blue) and using  $x$  and  $y$  traces (Fig. 4.7, red), it can be seen that the NMSE is reduced by about three orders of magnitude. It is worth noting that the prediction shown in Fig. 4.3(blue) can also be improved if a delayed version of variable  $x$  is given as an input to the reservoir (blue line in Fig. 4.7). However, in this case, including  $x$  and  $y$  variables is more significant than including the variable  $x$  and its delayed version.

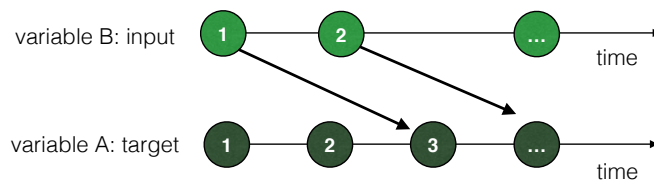


**Figure 4.7:** NMSE as a function of  $\gamma$  for the estimation of variable  $x$ . Red curve corresponds to the NMSE when using  $x$  and  $y$  variables. Blue curve corresponds to the NMSE of estimating  $x$  when using  $x$  and  $x(t-1)$  variables. Error bars were computed as the SD of the 25 different folds.

### 4.1.2 Multistep prediction

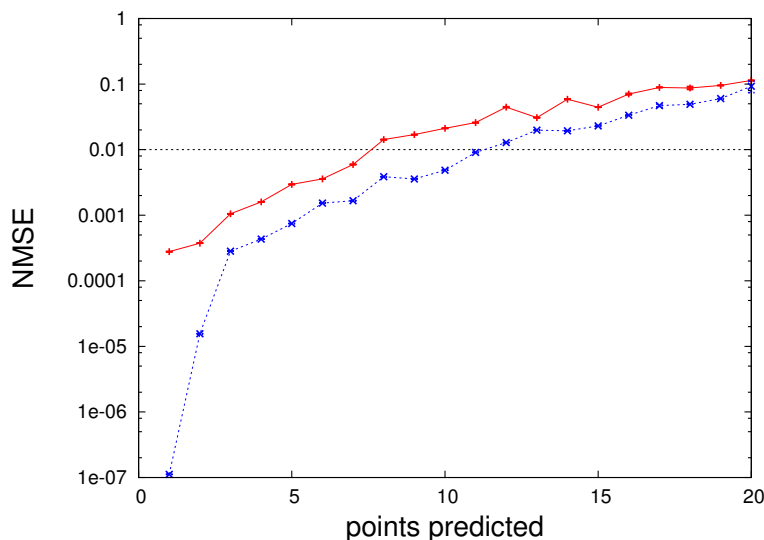
To substantiate our results further, we compute the prediction error of  $x$  for more than one time step ahead. An scheme of the two-step prediction task is shown in Fig. 4.8. Predicting more than two steps ahead means to shift the target signal by the amount of steps to be predicted.

## CHAPTER 4. MULTIVARIATE DELAYED RESERVOIR COMPUTERS



**Figure 4.8:** Univariate two-step prediction scheme. The target time series is shifted by two time steps. The input and target time series might belong to different variables of the system.

Figure 4.9 shows the **NMSE** as a function of the number of predicted points. Two curves are shown: the univariate and the multivariate curve. It can be observed that the **NMSE** remains below 1% (horizontal-dashed line) up to 11 predicted points ahead for the multivariate case (blue curve), i.e. when using both  $x$ - and  $y$ -variable time series to estimate the  $x$  variable. When using only the time series of variable  $x$  (red curve), i.e. the univariate case, only 7 points ahead can be predicted within the same error bounds. Therefore, the prediction of  $x$  is clearly improved when both  $x$  and  $y$  variables are presented as input to the reservoir.



**Figure 4.9:** **NMSE** as a function of the predicted steps for the variable  $x$ . Red curve (plus signs) corresponds to the **NMSE** of  $x$  when using only its own time series. Blue curve (stars) corresponds to the **NMSE** of  $x$  when using both  $x$  and  $y$  time series. Error bars were computed as the **SD** of the 25 different folds.

Time series prediction is one of the most common tasks in machine learning as well as classification tasks. In the next section we present an extension to the

## 4.2. MULTIVARIATE ARRHYTHMIC HEARTBEAT CLASSIFICATION WITH LOGISTIC REGRESSION

**Table 4.1:** MIT-BIH Arrhythmia database recording names used for training (DS1) and testing (DS2) datasets. Records in gray do not contain precordial lead V1 and are extracted from the datasets.

Dataset	MIT-BIH Arrhythmia record names
DS1	101, 106, 108, 109, 112, 114, 115, 116, 118, 119, 122, 124, 201, 203, 205, 207, 208, 209, 215, 220, 223, 230.
DS2	100, 103, 105, 111, 113, 117, 121, 123, 200, 202, 210, 212, 213, 214, 219, 221, 222, 228, 231, 232, 233, 234.

real-world classification problem described in Chapter 3. We apply a similar multivariate analysis to study the classification of ECG signals when a single or two channels are taken into account to classify healthy and pathological subjects with a delay-based reservoir.

4.2

### Multivariate arrhythmic heartbeat classification with logistic regression

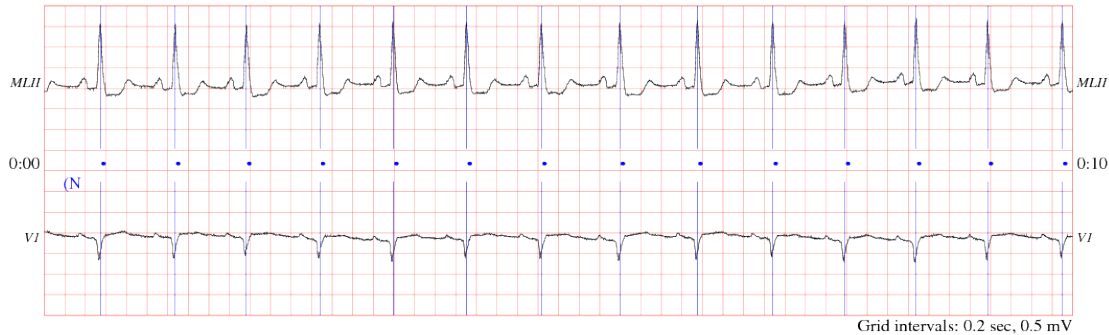
This task consists on the classification of heartbeats from the MIT-BIH Arrhythmia database. The univariate classification problem is described in detail in Chapter 3.

In the following, we choose the MIT-BIH Arrhythmia Database [90] which contains 48 ambulatory ECG recordings of half hour each, in order to test our methodology with this real-world problem. For this multivariate classification task, at least two channels of the ECG are required. The most common channel in the database is the modified limb lead II (MLII), present in 41 recordings. This is the lead used in Chapter 3 for the univariate case. The second most common channel is the precordial lead V1 (See Section 3.2.2) present in 37 recordings. Thus, we constrain the database to 37 recordings in order to include these two channels for our multivariate analysis. Table 4.1 shows the records of the training and testing datasets. Records in gray do not contain the precordial lead V1 and were extracted from the databases. These records are part of the univariate case developed in Chapter 3.

As before, the ECGs were down-sampled and divided into heartbeats using a fixed-length window of 170 samples locking the R-peak at sample 70. Each heartbeat was normalized with respect to the full signal to have zero mean and variance one. Figure 4.10 shows the beginning of an ECG of a normal subject. Both, MLII and V1 channels are represented. Blue vertical lines represent the

## CHAPTER 4. MULTIVARIATE DELAYED RESERVOIR COMPUTERS

position of the R-peak. The  $N$  in the plot means that the heartbeats are normal. Note that in V1, the R-point causes a negative deflection because it lies in the perpendicular plane of the limb and augmented leads plane, see Fig. 3.6.



**Figure 4.10:** ECG traces of the two main derivations, namely MLII and V1, for a normal subject. Vertical lines represent the position of the R-peak. Figure taken from Physionet.org

In this application, we use logistic regression (Section 3.4) for the learning procedure.

For the classification of the heartbeats, we again utilize the Mackey-Glass non-linearity with delay feedback as the reservoir with the same configuration described in Chapter 3. We have numerically checked that, for this task, the optimal number of virtual nodes is 25 (see Fig. 3.11 and its analysis).

In Chapter 3, Fig. 3.12, we have also explored the parameter space of the Mackey-Glass model finding a richer dynamics on the state matrices for  $\eta = 0.8$  and  $\gamma = 0.5$ . In addition, we choose parameter  $p = 7$  in Eq. 2.4, because for this value the NLN exhibits a short memory and a high degree of the nonlinearity, which we believe is more convenient for a classification task.

For a clinically-relevant, univariate, multiclass, heartbeat classification problem of 44 subjects from the MIT-BIH Arrhythmia Database the average performance of the classifier is shown in Table 3.4. However, and in contrast to Chapter 3, we construct the classifier from two channels, namely the MLII and the V1 channels. These two channels are only available for a reduced set of patients (37 subjects), leading to a smaller usable database and a degradation in performance, due to the high variability problems of the classifier (see Fig. 3.11 and its analysis). Table 4.2 shows the typical measures of performance (average along all classes) of the classifier for the univariate and multivariate case for the restricted database containing both derivations. The classification for a single channel is performed using channel MLII alone (37 subjects). Then a combination of channels MLII and V1, denoted as MLII-V1, is used to build a multivariate classifier. It can be observed that the sensitivity increases about 9% when using two variables in comparison with the case of one variable. De Chazal et al. [73] reported an

### 4.3. CONCLUSIONS

improvement in accuracy of 7% in a multivariate configuration of their approach to solve this classification task.

**Table 4.2:** Performance of the classification of ECGs using one or two channels.

Channel	Sensitivity (%)	Specificity (%)	Accuracy (%)	Precision (%)
MLII	76.15	97.82	98.02	78.15
MLII-V1	84.87	97.92	98.36	88.67

The accuracy is a global estimator that was not specially affected by the inclusion of another channel in the inputs. However we can see that the sensitivity, i.e. the ability of the classifier to recognize the positive cases, was improved by the inclusion of channel V1.

As discussed in the analysis of Fig. 3.11, this task is particularly sensitive to the number of records in the database because the classifier suffers of a high variance problem. Reducing the number of records from 44 to 37 causes a degradation in the performance, e.g. sensitivity drops from 84.83 to 76.15. It is worth noting that when using the combination MLII-V1 the sensitivity rises to comparable values of the full usable database. For future comparisons, we provide the confusion matrices of the two cases discussed in this section. Table 4.3 shows the confusion matrix of the classifier when using MLII over 35 subjects. Table 4.4 shows the confusion matrix of the classifier when using the combination MLII-V1. Note how despite the improvements shown in Table 4.2, the associated confusion matrix for MLII-V1 performs worse in the recognition of the ventricular class (V).

More tests have to be done to verify the usefulness of including more channels into the reservoir. For example, maybe including only V1 would perform better than including MLII. We leave the completion of this task as a future work.

4.3

---

## Conclusions

In this chapter we have numerically shown the ability of reservoir computing, based on delay-coupled systems, to perform time series prediction and classification tasks following a multivariate analysis. We have concentrated on two tasks, the prediction of a chaotic time series, given by the Lorenz system, and the classification of heart beats, obtained from ECG derivations.

## CHAPTER 4. MULTIVARIATE DELAYED RESERVOIR COMPUTERS

**Table 4.3:** Confusion matrix for the AAMI-classes classification problem using [MLII](#) (37 patients) over DS2

		Predicted samples				
		N	S	V	F	Q
Known samples	N	35265	1623	0	0	0
	S	306	1465	26	0	0
	V	38	53	3114	11	1
	F	9	4	16	359	0
	Q	4	0	0	2	1

**Table 4.4:** Confusion matrix for the AAMI-classes classification problem using [MLII-V1](#) over DS2

		Predicted samples				
		N	S	V	F	Q
Known samples	N	35706	1182	0	0	0
	S	294	1503	0	0	0
	V	79	115	2991	32	0
	F	8	4	12	364	0
	Q	0	0	0	3	4

For the one-step prediction task of the Lorenz system, we found a significant reduction ( $\sim 3$  orders of magnitude) of the normalized mean square error ([NMSE](#)) when using two variables to predict one, than when using only one variable. Moreover, we found that the [NMSE](#) remains smaller than 1% when predicting 11 steps ahead when using the two variables compared to only 7 steps for using one variable. We expect the results obtained for the well-known Lorenz system in the chaotic regime to be valid for similar multivariate dynamical systems.

We have also applied the multivariate approach to the classification of heart beats. We found an improvement of 9% in sensitivity when using two channels of the [ECG](#) as compared to the case when the classification was performed using a single channel.

Our results highlight that the use of more than one variable can significantly improve predictions when using reservoir computing techniques [100]. More tests with real-world data are however needed to explore the full potential of our approach.

### 4.3. CONCLUSIONS

It is worth noting that the fading memory present in recurrent networks resembles the time-delay embedding in Takens theorem. This fading memory implies that information about previous inputs is still present in the reservoir after a number of delay times  $\tau$ . In the Lorenz time-series prediction case, adding explicitly a delayed version of the same input does not provide as much information as adding the  $y$  variable. Thus, we find a significant improvement in the prediction capabilities of reservoir computing when using an additional variable.





# Reservoir computing using semiconductor laser dynamics

The amount of information that is generated nowadays not only requires new paradigms for the solution of difficult tasks but also needs these solution to be efficient in time. In previous chapters we studied how [reservoir computing](#) is able to solve computational problems that are difficult for standard computational techniques. Optical computation is an interesting approach to increase the rates of information that can be processed by a reservoir computer since it has properties such as high speed, energy efficiency and true parallelism.

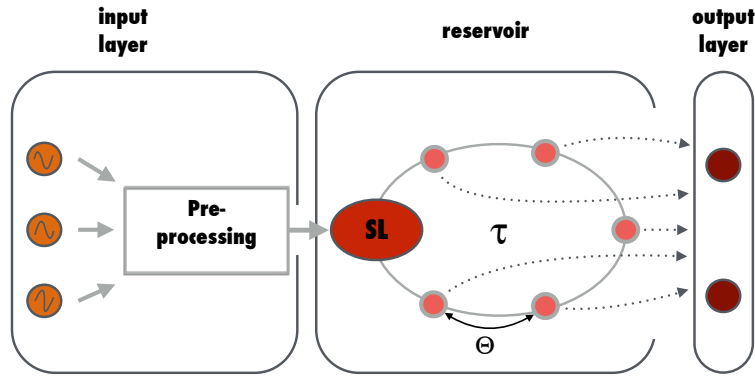
In this chapter we perform a thorough numerical study of the performance of a single-mode semiconductor laser subject to all-optical feedback and demonstrate how the rich dynamical properties of this delay system can be beneficially employed for processing time dependent signals. Parts of the content of this chapter is based on a collaboration published in the article:

K. Hicke, M.A. Escalona-Morán, D. Brunner, M.C. Soriano, I. Fischer and C.R. Mirasso. *Information processing using transient dynamics of semiconductor lasers subject to delayed feedback*. IEEE J. Selected Topics in Quantum Electronics, 19, issue 4, 1501610, (2013).

We have implemented the dynamical behavior of a semiconductor laser as the nonlinear node ([NLN](#)) of a reservoir computer for the processing of two tasks: a classification task and a time series prediction task. The implementation is depicted in Fig. [5.1](#). In this figure the general representation of the nonlinear node ([NLN](#)) is replaced by a semiconductor laser dynamics denoted as [SL](#).

Results presented in this chapter were obtained in close collaboration with Konstantin Hicke, specialist in the semiconductor laser dynamics. The numerical results were complemented by experimental results implemented in hardware, developed by Daniel Brunner in collaboration with Miguel C. Soriano, showing the robustness of the proposed scheme. This work was made under the super-

## CHAPTER 5. RESERVOIR COMPUTING USING SEMICONDUCTOR LASER DYNAMICS



**Figure 5.1:** Schematic representation of a reservoir computer using a semiconductor laser (SL) as the nonlinear node (NLN). This figure is equivalent to the general implementation of reservoir computing described in Chapter 2, Fig. 2.1.

vision of Ingo Fischer and Claudio R. Mirasso [8]. The current chapter presents only the numerical studies associated to the different tasks that were carried out by the author of this thesis.

### 5.1

## Semiconductor laser rate equations

We consider the following model [101, 102], describing the dynamics of a semiconductor laser subject to delayed feedback. It comprises equations for the slowly varying complex electric field amplitude (in both parallel  $\mathcal{E}_{\parallel}$  and perpendicular polarization direction  $\mathcal{E}_{\perp}$ , respectively) and the carrier number  $n$  in the laser cavity:

$$\begin{aligned} \dot{\mathcal{E}}_{\parallel}(t) &= \frac{1}{2}(1 + i\alpha) (\mathcal{G}_{\parallel}(\mathcal{E}_{\parallel}, n) - \gamma_{\parallel}) \mathcal{E}_{\parallel}(t) + \kappa_{\parallel} \mathcal{E}_{\parallel}(t - \tau_{ec}) \\ &+ \mathcal{E}_{inj}(t) e^{i\Delta\omega t} + F_{\mathcal{E}_{\parallel}}, \end{aligned} \quad (5.1)$$

$$\begin{aligned} \dot{\mathcal{E}}_{\perp}(t) &= -i\Delta\Omega \mathcal{E}_{\perp}(t) + \frac{1}{2}(1 + i\alpha) (\mathcal{G}_{\perp}(\mathcal{E}_{\parallel}, n) - \gamma_{\perp}) \mathcal{E}_{\perp}(t) \\ &+ \kappa_{\perp} \mathcal{E}_{\parallel}(t - \tau_{ec}) + F_{\mathcal{E}_{\perp}}, \end{aligned} \quad (5.2)$$

$$\begin{aligned} \dot{n}(t) &= \frac{I(t)}{e} - \gamma_e n(t) - \mathcal{G}_{\parallel}(\mathcal{E}_{\parallel}, n) |\mathcal{E}_{\parallel}(t)|^2 \\ &- \mathcal{G}_{\perp}(\mathcal{E}_{\perp}, n) |\mathcal{E}_{\perp}(t)|^2, \end{aligned} \quad (5.3)$$

with the gain functions

## 5.1. SEMICONDUCTOR LASER RATE EQUATIONS

$$\mathcal{G}_{\parallel}(\mathcal{E}_{\parallel}, n) = g_{\parallel} \frac{n(t) - n_T}{1 + \epsilon |\mathcal{E}_{\parallel}(t)|^2}, \quad (5.4)$$

$$\mathcal{G}_{\perp}(\mathcal{E}_{\perp}, n) = g_{\perp} \frac{n(t) - n_T}{1 + \epsilon |\mathcal{E}_{\perp}(t)|^2}. \quad (5.5)$$

Here,  $\alpha$  is the linewidth enhancement factor,  $\gamma_{\parallel, \perp}$  are the photon decay rates of the polarization modes,  $\kappa_{\parallel, \perp}$  the feedback rates,  $\tau_{ec}$  is the external cavity round-trip time,  $\Delta\omega$  denotes the detuning between the laser and the optical injection,  $\Delta\Omega$  is the detuning between  $\mathcal{E}_{\parallel}(t)$  and  $\mathcal{E}_{\perp}(t)$ ,  $I(t)$  is the time-dependent injection current,  $e$  is the elementary charge,  $\gamma_e$  denotes the electron decay rate,  $g_{\parallel, \perp}$  are the differential gains,  $n_T$  is the carrier number at transparency and  $\epsilon$  is the gain saturation coefficient.  $|\mathcal{E}_{\parallel}(t)|^2$  and  $|\mathcal{E}_{\perp}(t)|^2$  represent the number of photons in the parallel and perpendicular polarization direction, respectively. The output power is computed as  $P = [hc^2\alpha_m/(2\mu_g\lambda)]|\mathcal{E}|^2$  where  $h$  is the Planck constant,  $c$  the speed of light,  $\lambda$  the emission wavelength,  $\alpha_m$  the facet losses and  $\mu_g$  the group refractive index. The chosen values for these parameters used for the simulations are shown in Table 5.1. the model we consider, as well as the ration of the differential gains, are chosen according to [102].

The spontaneous emission noise is implemented as a complex Gaussian white noise term  $F_{\mathcal{E}}$  in the field equations:

$$F_{\mathcal{E}_{\parallel, \perp}} = F_1 + iF_2, \quad (5.6)$$

where the real and imaginary parts are independent random processes with zero mean

$$\langle F_{\mathcal{E}_{\parallel, \perp}}(t) \rangle = 0 \quad (5.7)$$

and a variance given by

$$\langle F_{\mathcal{E}_{\parallel, \perp}}(t) \overline{F_{\mathcal{E}_{\parallel, \perp}}(t')} \rangle = 2\beta_{\parallel, \perp} \gamma_e n(t) \delta(t - t'). \quad (5.8)$$

$\beta_{\parallel, \perp}$  are the spontaneous emission factors, describing the fraction of spontaneously emitted photons coupled into the respective lasing modes.

The delayed optical feedback is modeled for two configurations: polarization-maintained optical feedback (**PMOF**) and polarization-rotated optical feedback (**PROF**). For **PMOF**, with the polarization direction being defined by the axis of the laser cavity yielding a higher optical gain, the optical feedback goes from the dominant mode ( $\mathcal{E}_{\parallel}$ ) back to itself. For **PROF**, the feedback goes from the dominant polarization mode ( $\mathcal{E}_{\parallel}$ ) to the weaker polarization mode ( $\mathcal{E}_{\perp}$ ). For simplicity we assume that, due to the characteristics of edge emitting lasers, the

## CHAPTER 5. RESERVOIR COMPUTING USING SEMICONDUCTOR LASER DYNAMICS

Table 5.1: Laser parameter values used in the numerical simulations.

Parameter	Value
$\alpha$	3.0
$\gamma_{\parallel} = \gamma_{\perp}$	$200 \text{ ns}^{-1}$
$\beta_{\parallel} = \beta_{\perp}$	$10^{-6}$
$\kappa_{\parallel} = \kappa_{\perp}$	$10 \text{ ns}^{-1}$
$\tau_{ec}$	80 ns
$\Delta\omega$	0.0
$\Delta\Omega$	0.0
$\gamma_e$	$1 \text{ ns}^{-1}$
$g_{\parallel}$	$10^{-5} \text{ ns}^{-1}$
$g_{\perp}$	$8.4 \cdot 10^{-6} \text{ ns}^{-1}$
$\epsilon$	$10^{-7}$
$\lambda$	$1.5 \mu\text{m}$
$\alpha_m$	$45 \text{ cm}^{-1}$
$\mu_g$	4
$n_T$	$1.8 \cdot 10^8$
$\chi$	0.4
$\bar{P}_{inj}$	$436 \mu\text{W}$
$I_{thr}$	32.0 mA

dominant field component is  $\mathcal{E}_{\parallel}(t)$  and consequently only the delay term of the parallel component appears in the equation for  $\mathcal{E}_{\perp}(t)$ . In the case of **PMOF** the feedback rate  $\kappa_{\perp}$  is zero and in case of **PROF**  $\kappa_{\parallel}$  is zero. We also assume that both polarization components have the same frequency, i.e.  $\Delta\Omega = 0$ .

### 5.1.1 Input signal injection into the laser dynamics

For the injection of an input signal  $S(t)$  we consider two different methods: electrical injection and optical injection. In the case of electrical injection, the injection current is modulated with  $S(t)$  around a bias current, corresponding to

$$I(t) = I_b + \chi I_{thr} S(t), \quad (5.9)$$

## 5.2. NUMERICAL RESULTS

with  $S(t)$  being normalized, the bias current being  $I_b$  and the signal scaling  $\chi I_{thr}$ . Here,  $I_{thr}$  denotes the solitary laser threshold current. For electrical signal injection, the signal  $S(t)$  is positive for all  $t$ .

In case of optical injection the signal is injected via  $\mathcal{E}_{inj}(t)$ . For practical reasons, and to compare to the experimental implementations (see Ref. [25, 36]), we assume external modulation of the injected light via a Mach-Zehnder electro-optic modulator. The input is then modeled via an injected power  $P_{inj}$  modulated with a sine-square function around a mean value  $P_{inj}^0$ , yielding the signal:

$$P_{inj}(t) = P_{inj}^0 + P_{inj}^s \sin^2\left(a \frac{\pi}{4} S(t) + \Phi_0\right). \quad (5.10)$$

We set  $P_{inj}^0 = \bar{P}_{inj}/4$  and  $P_{inj}^s = 3/2\bar{P}_{inj}$  so that

$$P_{inj}(t) = \bar{P}_{inj} \left[ 1/4 + 3/2 \sin^2\left(a \frac{\pi}{4} S(t) + \Phi_0\right) \right], \quad (5.11)$$

which means that the injected power is modulated  $\pm 75\%$  around the average injected power  $\bar{P}_{inj}$ . For the optical injection, we distinguish between two different modulation methods. We consider symmetric modulation with  $S(t)$  normalized between  $\pm 1$ ,  $a = 1$  and  $\Phi_0 = \frac{\pi}{4}$  and asymmetric modulation with  $S(t)$  positive,  $a = 2$  and  $\Phi_0 = 0$ .

### 5.2

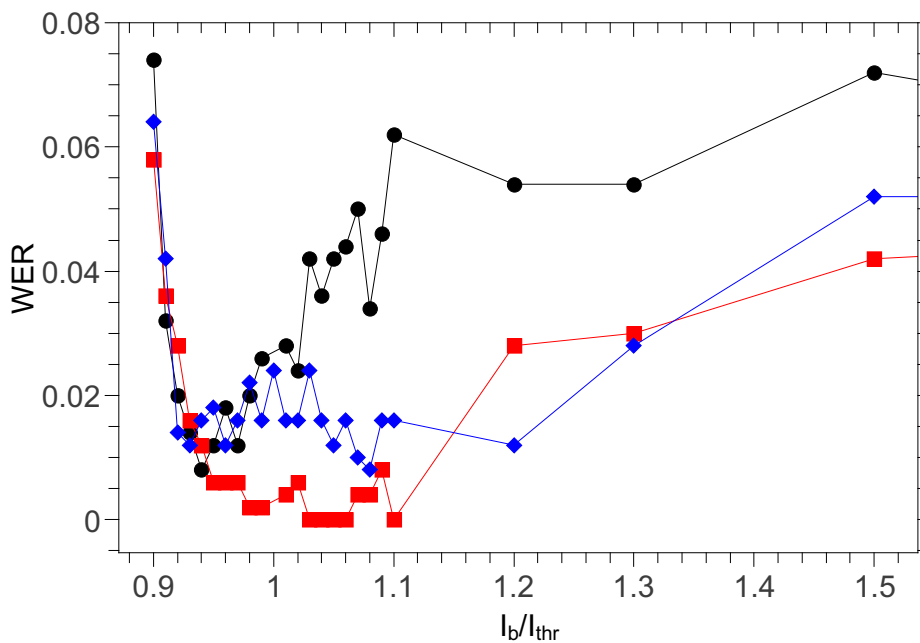
---

## Numerical results

In this section we present numerical results obtained from the simulations of a semiconductor laser subject to delayed optical feedback. We study both polarization-maintained and polarization-rotated feedback, respectively. For comparison, we use electrical and optical injection for the input signal. In order to compare our results with previously reported studies, we elaborate on two well accepted tasks in the machine learning community: spoken digit recognition (SDR) and time series prediction. While the former does not require much memory, and consequently the feedback is expected not to play an important role, the latter is memory dependent and feedback is expected to be essential. In our numerical analysis, the reservoir consisted of  $N = 400$  virtual nodes, resulting in a virtual node spacing of  $\Theta = 200$  ps ( $\tau_{ec}/N$ ).

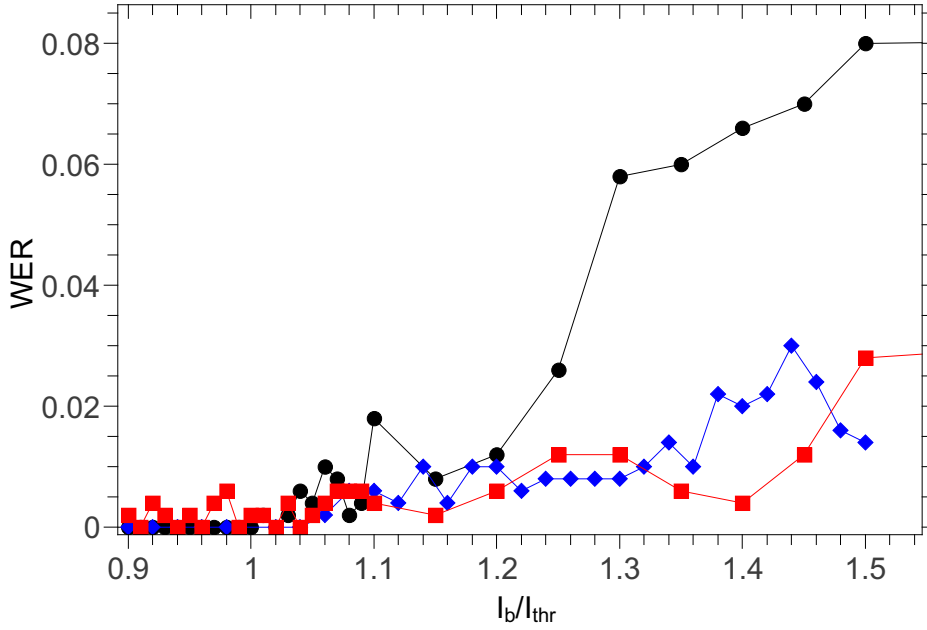
### 5.2.1 Spoken digit recognition task

We first evaluate the performance of the system for the spoken digit recognition task. The spoken digit data set consists of five female speakers uttering numbers from zero to nine with a tenfold repetition for statistics (500 samples in total) [53]. Before injecting the information into the laser, we perform standard preprocessing, creating cochleograms using the Lyon ear model [55]. The information injected into the laser ( $S(t)$ ) is given by the product of the cochleograms ( $R(t)$ ) and the mask matrix  $M$  [8, 30]. For the characterization of the classification performance, we evaluate the word error rate (WER) as a function of some key laser parameters and operating conditions. It is important to note that for the WER evaluation, we choose 20 random partitions of 25 samples each out of the 500 spoken digits, using 475 samples for training the readout weights, keeping the remaining 25 for testing. Following this procedure, each random partition and each sample are used exactly once for testing (20-fold-cross-validation).



**Figure 5.2:** Word error rate (WER) for the spoken digit recognition task versus the bias current  $I_b$  for electrical injection. Black circles denote the results for feedback with parallel polarization (PMOF) for a feedback rate of  $\kappa_{\parallel} = 10 \text{ ns}^{-1}$ . Blue diamonds are the results for polarization-rotated feedback (PROF) for a feedback rate of  $\kappa_{\perp} = 10 \text{ ns}^{-1}$ . Red squares represent the WER for the case without any feedback. The other parameters were set as in Table 5.1. Note that the lines are only guide to the eyes.

## 5.2. NUMERICAL RESULTS



**Figure 5.3:** Word error rate (WER) for the spoken digit recognition task versus bias current  $I_b$  for optical injection. Black circles denote the results for feedback with parallel polarization (PMOF) with a feedback rate of  $\kappa_{||} = 10 \text{ ns}^{-1}$ . (Red) squares are the results for polarization-rotated feedback (PROF) with a feedback rate of  $\kappa_{\perp} = 10 \text{ ns}^{-1}$ . (Blue) diamonds are the resulting word error rates for the case without any feedback. The other parameters were set as in Table 5.1. Note that the lines are only guide to the eyes.

Our numerical simulations allow us to consider two types of information injection. We can either inject the input information electrically by modulating the laser current or optically by injecting an externally modulated optical signal.

Figure 5.2 depicts the WER as a function of the laser bias current for the case of electrical input injection and for three different feedback conditions, in particular PMOF (circles in Fig. 5.2), PROF (diamonds in Fig. 5.2) and in the absence of feedback (squares in Fig. 5.2). We find that the best classification is found for bias currents around the solitary lasing threshold, independent of the feedback conditions. For the three feedback schemes, performance with WER of 0.008, 0.008 and 0 for the PMOF, PROF and the laser without feedback, respectively, is obtained. It is worth noting that bias currents around threshold ensure that the laser starts its dynamics from a steady state. Interestingly, we find that, for the semiconductor laser dynamics, the best performance is obtained in the absence of optical feedback due to the strong nonlinearity of the semiconductor laser. No significant differences are found between PMOF or PROF conditions in the case of electrical input injection for bias currents below and close to threshold.

## CHAPTER 5. RESERVOIR COMPUTING USING SEMICONDUCTOR LASER DYNAMICS

However, for larger bias currents  $I_b > 1.02I_{thr}$  the PROF configuration performs significantly better than PMOF.

The classification performance, as function of the bias current normalized to the threshold value is illustrated in Fig. 5.3 for the case of optical input injection. We again find that the WER depends on the laser bias current. We observe that the classification performance is qualitatively similar for PROF (squares in Fig. 5.3) and in the absence of feedback (diamonds in Fig. 5.3), with even better WER when compared to the case of electrical input injection shown in Fig. 5.2. In contrast, the classification error for PMOF (circles in Fig. 5.3) increases significantly above threshold. For optical injection, the classification error is minimum, reaching a 0 WER, for bias currents slightly below the solitary lasing threshold independent of the feedback conditions.

### 5.2.2 Santa Fe time series prediction task

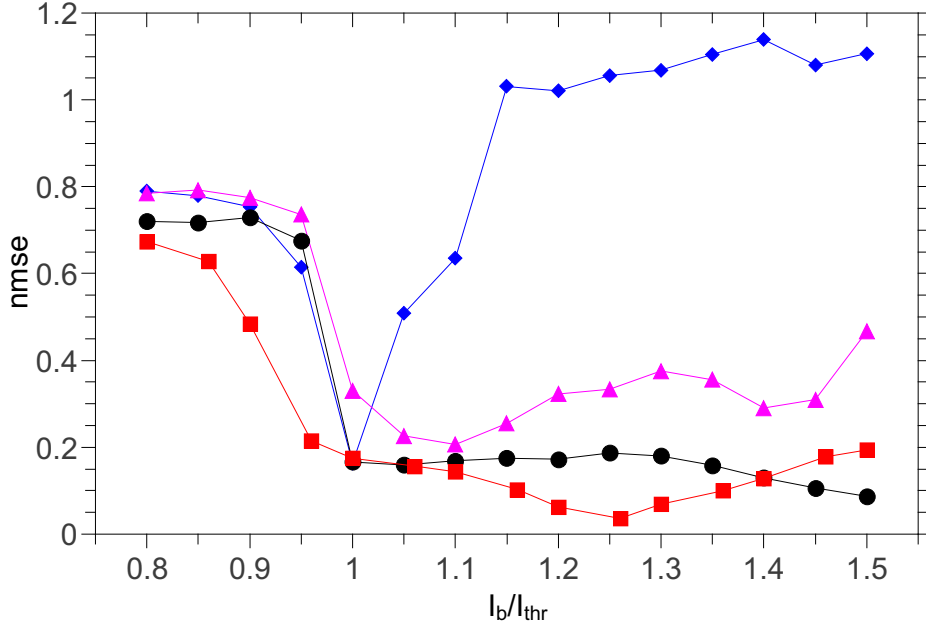
The second task that we tackle is time-series prediction. In this task, we evaluate the performance of our scheme in predicting the respective next point of a chaotic time series. We specifically employ data from the Santa Fe time series competition, data set A [103]. For the evaluation of the prediction error, we take 4,000 data points of this data set, created by a far-Infrared laser operating in a chaotic regime [104]. We use 75 % of the points for training and 25 % for testing. The information injected to the laser is given by the product of the samples in the Santa Fe time-series competition data set A and the one-dimensional mask matrix M [8, 38].

To characterize the performance of the system for this task, we compute the normalized mean square error (NMSE, Eq. 4.2) of the prediction, defined as the normalized difference between the predicted and its target value, and study its dependence with the laser bias current and the feedback rate. Similar to the case of spoken digit recognition, we find the performance of the system to be better over a wider range of laser bias currents when the information is optically injected. Therefore, we only concentrate on results for optical information injection in the time series prediction task. This prediction task requires the system to have memory, i.e. optical feedback is essential for this task.

In Fig. 5.4, we show the NMSE as a function of the laser bias current for two different values of the optical injection power. The first one corresponds to a large average power of the injected light, compared to the power of the laser subject to feedback,  $\bar{P}_{inj} = 436\mu W$ . In this case, the NMSE for the Santa Fe time-series prediction task is below 0.2 both for PMOF (see squares in Fig. 5.4) and PROF (see circles in Fig. 5.4) for laser bias currents above threshold, with a minimum NMSE for PMOF and  $I_b = 1.25I_{thr}$  of 0.036 and 0.087 for PROF at  $I_b = 1.5I_{thr}$ . We also present the results for a smaller average power of the optical



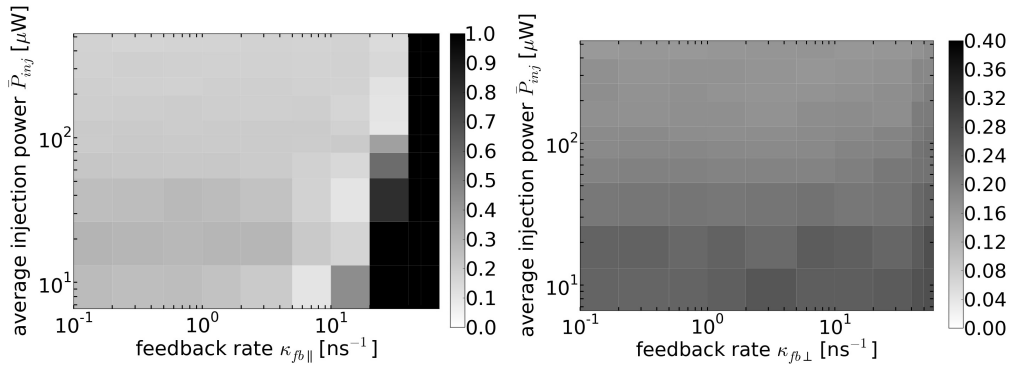
## 5.2. NUMERICAL RESULTS



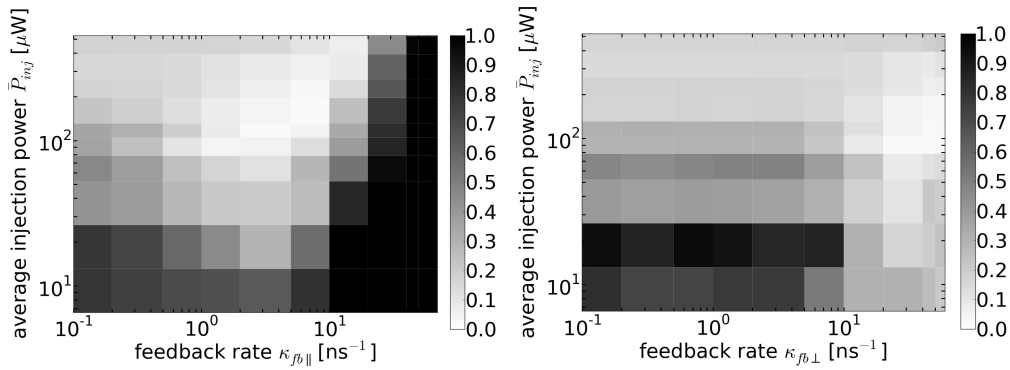
**Figure 5.4:** Normalized mean squared error (NMSE) for the Santa Fe time series prediction task versus the bias current  $I_b$  for polarization-rotated feedback with  $\bar{P}_{inj} = 436\mu W$  (black circles), polarization-maintained feedback with  $\bar{P}_{inj} = 436\mu W$  (red squares), polarization-maintained feedback with  $\bar{P}_{inj} = 11\mu W$  (blue diamonds) and polarization-rotated feedback with  $\bar{P}_{inj} = 11\mu W$  (magenta triangles), respectively. The feedback rates were set to  $\kappa_{\parallel} = 10 \text{ ns}^{-1}$  and  $\kappa_{\perp} = 10 \text{ ns}^{-1}$ , respectively, while the respective other was set to zero. The other parameters were chosen as in Table 5.1. Note that the lines are only guide to the eyes.

input injection,  $\bar{P}_{inj} = 11\mu W$ . In this case we find that for PMOF low prediction errors are restricted to laser bias currents close to the solitary lasing threshold (see diamonds in Fig. 5.4), with a minimal NMSE value of 0.164 for  $I_b = I_{thr}$ . The prediction error increases significantly for higher bias currents due to the onset of delayed feedback instabilities. For PROF and low injection power the bias current range for good performance is significantly broader (see triangles in Fig. 5.4) with the minimal NMSE being 0.206 for  $I_b = 1.1I_{thr}$ . The error increases less with increasing bias current for PROF. In the case of high injection power, however, the prediction hardly changes with the bias current (above threshold) for both feedback configurations. On the contrary, the prediction errors strongly increase for low injection current for both low and high injection powers. This is because we move far from the nonlinear region of operation of the laser. Nevertheless, in both cases, competitive prediction errors can be achieved. It is interesting to note, though, that a larger average optical injection power allows for a wider range of bias currents providing good performance.

## CHAPTER 5. RESERVOIR COMPUTING USING SEMICONDUCTOR LASER DYNAMICS



**Figure 5.5:** Colour-coded normalized mean squared error (NMSE) for the Santa Fe time series prediction task as a function of (left) the PMOF feedback rate  $\kappa_{\parallel}$  and (right) the PROF feedback rate  $\kappa_{\perp}$  and different average relative injected powers  $\bar{P}_{inj}$  (y-axis). The bias current was set to  $I_b = 1.01I_{thr}$ . The other parameters were chosen as in Table 5.1.



**Figure 5.6:** Colour-coded normalized mean squared error (NMSE) for the Santa Fe time series prediction task as a function of (left) the PMOF feedback rate  $\kappa_{\parallel}$  and (right) the PROF feedback rate  $\kappa_{\perp}$  and different average relative injected powers  $\bar{P}_{inj}$  (y-axis). The bias current was set to  $I_b = 1.18I_{thr}$ . The other parameters were set as in Table 5.1.

As the Santa Fe time-series prediction requires the presence of memory in the system, we also evaluate in detail the influence of the feedback strength on the prediction performance. Figures 5.5 and 5.6 present the NMSE, coded in gray scale, for two different laser bias currents, namely a current close to threshold ( $I_b = 1.01I_{thr}$ ) and a current clearly above threshold ( $I_b = 1.18I_{thr}$ ). Furthermore, we distinguish two different feedback conditions for each laser bias current. The left panels in Figs. 5.5 and 5.6 are calculated for polarization maintained feedback (PMOF), while the right panels are calculated for polarization rotated feedback (PROF).

## 5.2. NUMERICAL RESULTS

In the case of a bias current close to threshold ( $I_b = 1.01I_{thr}$ ) shown in Fig. 5.5(left), we find that PMOF yields low NMSE values for feedback rates below  $\kappa_{||} = 20 \text{ ns}^{-1}$  independently of the average power of the injected signal  $\bar{P}_{inj}$ , reaching the minimum value of 0.099. In addition, Fig. 5.5(right) shows that with PROF and a small bias current the error is almost independent of injection power and feedback strength. The minimum NMSE value is 0.161 while the overall average for the considered parameters is 0.201.

In the case of a current well above threshold ( $I_b = 1.18I_{thr}$ ) shown in Fig. 5.6(left), we find that PMOF yields low NMSE values for intermediate feedback rates and high average powers of the injected signal  $\bar{P}_{inj}$ , with a minimum value of 0.021. Interestingly, an increase in the laser bias current requires an increase in the average injection power and feedback rate to achieve a low prediction error in the case of PMOF. This result suggests that a balance between laser emission power and the average injection power is needed. As shown in Fig. 5.6, PROF yields low NMSE values (minimum value 0.022) for high average powers of the injected signal and for large feedback rates.

Overall, the results obtained with the delayed feedback scheme are very competitive compared to traditional reservoir computing techniques. All the so far presented results have been obtained considering spontaneous emission factor of  $10^{-6}$ . In order to evaluate the influence of different noise levels in more detail, we discuss in the following section the performance of our scheme for different spontaneous emission factors.

### 5.2.3 Influence of spontaneous emission noise

In this section we evaluate the influence of spontaneous emission on the performance of the system for the two tasks described previously. In the case of spoken digit recognition, we find that the task is extremely robust against spontaneous emission noise. As shown in Fig. 5.7, the classification error remains almost constant up to  $\beta = 0.0001$ , which is two orders of magnitude larger than a realistic value for the spontaneous emission, both for electrical (circles) and optical (squares) input injection.

In contrast, in the case of the Santa Fe time-series prediction, the NMSE degrades already for realistic values of the spontaneous emission. Figure 5.8 illustrates the degradation on the performance as a function of the spontaneous emission noise for polarization-maintained (circles) and polarization rotated (squares) optical feedback. The prediction error increases when the noise term is included in the numerical simulations, with a gradual degradation at realistic values ( $\beta = 10^{-6}$ ) and a sudden increase at  $\beta = 0.0001$ . The difference in sensitivity originates from the different nature of the two tasks. While spoken digit recognition is a classification task which only requires a winner-take-all criterion, time series

## CHAPTER 5. RESERVOIR COMPUTING USING SEMICONDUCTOR LASER DYNAMICS

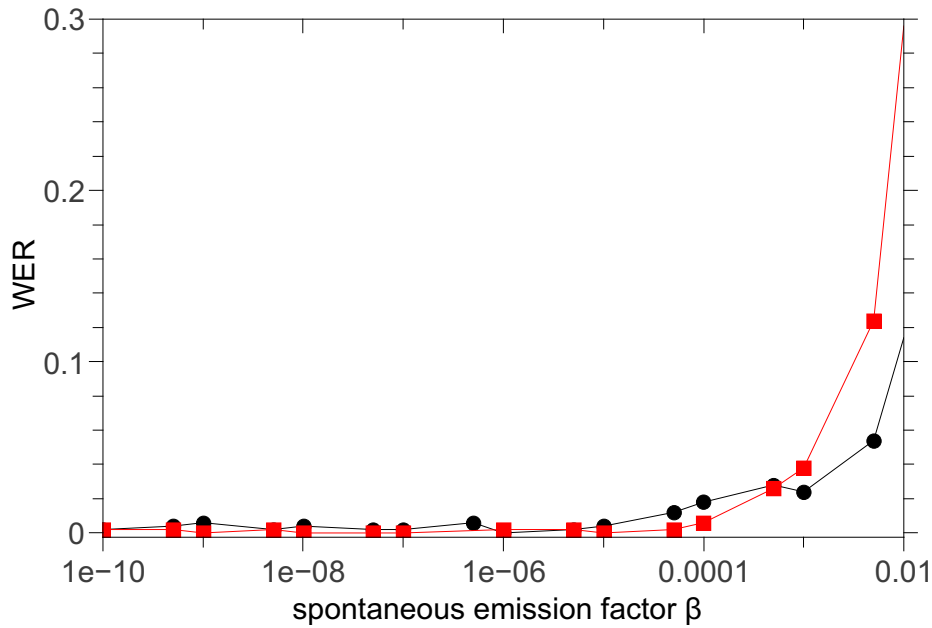
prediction actually requires the precise approximation of a nonlinear transformation. Furthermore, time series prediction tasks are more sensitive to noise than classification tasks [38].

5.3

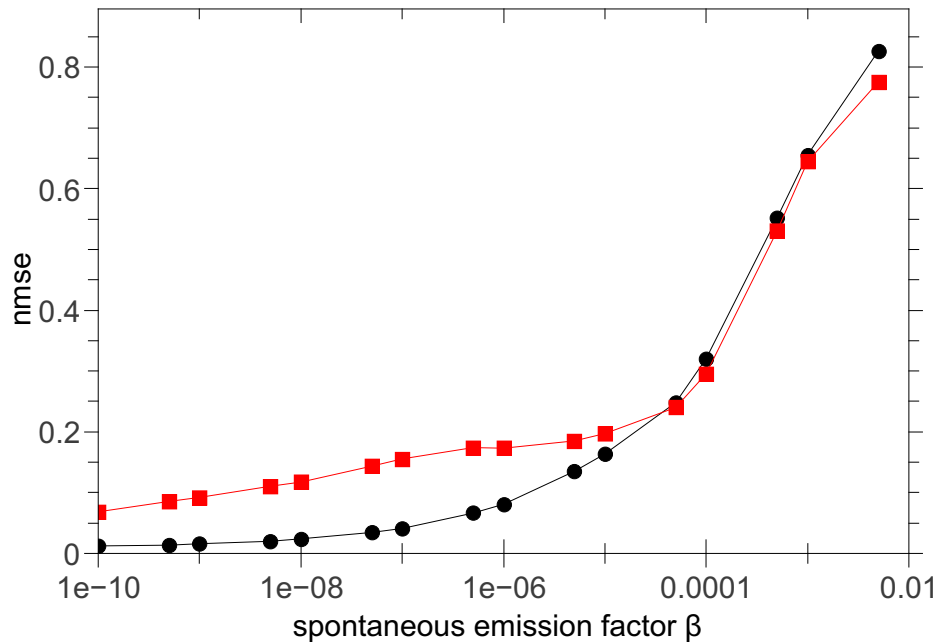
### Conclusion

In conclusion, we have studied the computational capabilities of a semiconductor laser subject to delayed optical feedback. Our numerical simulations highlight the potential and robustness of the proposed scheme. Moreover, the modeling provides guidelines for the experimental implementation of the scheme.

We find that this configuration offers, in a hardware implementation, excellent computational performance with low hardware requirements, high bandwidth, and low power consumption. The results obtained for the spoken digit recognition task ( $WER=0$ ) are better to those obtained with a system based on Hidden



**Figure 5.7:**  $WER$  for the spoken digit recognition task versus the spontaneous emission factor  $\beta$  for electrical (black circles) and optical injection (red squares), respectively. The feedback rate was set to  $\kappa_{\parallel,\perp} = 0$ . Bias current was set to  $I_b = 1.0I_{thr}$ . The other parameters were set as in Table 5.1. Note that the lines are only guide to the eyes.



**Figure 5.8:** NMSE for the Santa Fe time-series prediction task versus the spontaneous emission factor  $\beta$  for parallel polarization (black circles) and polarization-rotated feedback (red squares), respectively. The feedback rates were set to  $\kappa_{\parallel} = 10 \text{ ns}^{-1}$  and  $\kappa_{\perp} = 10 \text{ ns}^{-1}$ , respectively, while the respective other was set to zero. The bias current was  $I_b = 1.18I_{thr}$ . The other parameters were set as in Table 5.1. Note that the lines are only guide to the eyes.

Markovian Models (0.00168) [105] and with a traditional reservoir computing system (0.005) [106] under similar cross-validation configurations.

For the case of time series prediction, our numerical results of  $\text{NMSE}=0.02$  for both PMOF and PROF configurations, a bias current of  $I_b = 1.25I_{thr}$  and a high-power signal injection, are of the order of those obtained with more traditional techniques ( $<0.01$  [107]), although in the latter additional memory is artificially added into the input data.

Many parameter dependences and opportunities that this configuration offers remain unexplored so far. Nevertheless, the given examples represent a spotlight on the capabilities that this system provides. We expect that the full potential of this information processing scheme will become practical for photonics applications.

In addition to highlighting the potential of photonics for reservoir computing schemes, in this chapter we showed a different nonlinear function other than the Mackey-Glass system. This highlights the versatility of delay-based reservoir computing implementations.



## **Part III**

# **Conclusions**





## Conclusions

New trends in information processing require unconventional computing methods to solve challenging tasks. Along the different chapters of this thesis we have developed different aspects of delay-coupled systems in an information processing context.

This research work has served to implement a computational suite for the fast numerical implementation of reservoir computing. The suite, written in Python, is able to evaluate different nonlinear function dynamics and the many different scenarios presented in this thesis, e.g. search of optimal parameters, different measures of performance, cross-validation schemes, several learning procedures, etc. This computational tool is able to run sequentially or in parallel according to the needs of the task and the available computational resources.

Using the implemented Python module, we studied the computational properties of a delay-coupled system, emulating the functionality of a reservoir computer. This type of computational paradigm exploits the information contained in the transient dynamics when the dynamical state of the system is subject to an external input signal. Among the computational properties of the reservoir, we studied the intrinsic computational quality that such a computer has, independently of the realization of a particular task. These kind of properties highlight the computational power that a single node system subject to feedback delay contains. Among these properties, we studied the quality of the computational kernel and its capacity to generalize. A combination of these two properties was summarized as the computational ability that a reservoir computer has. We also explored the memory capacity of the delay-based system. Memory is an intrinsic characteristic of systems with recurrence. However, different tasks require different memory capacities. Not only the memory capacity of the system was evaluated but also its quality. We found wide range of parameters where the proposed reservoir computer exhibits good computational abilities. This allows us to predict the parameter regions where a good performance of the reservoir is expected. After characterizing the computational properties of the proposed

## CHAPTER 6. CONCLUSIONS

reservoir computer, we tested its ability to solve several typical machine learning tasks, e.g. classification tasks, time-series prediction and system's modeling.

In terms of classification, we evaluated two main tasks: the spoken digit recognition (SDR), a well-known task in machine learning community, and a world round medical problem, namely the automatic classification of heartbeats for the detection of cardiac arrhythmias. For the SDR task, we applied a standard procedure of machine learning in one of the most difficult classification scenarios. This standard procedure allows to detect possible problems of the reservoir computer, e.g. bias or variance problems. We used the word error rate (WER) as a measure of performance. We found a WER of 9.7% using the machine learning procedure to avoid high variance problems, in which the database was divided into subsets and optimal parameters were estimated using different non-overlapping subsets of the database. When neglecting the machine learning procedure and using the cross-validation method over the full database, our results (WER~ 0.2%) were equivalent to those reported in the literature under similar conditions.

We also utilized an especially designed reservoir computer for the automatic classification of arrhythmic heartbeats. Our approach differs from the typical learning procedure in reservoir computing, a linear regression, by using a logistic regression learning algorithm that provides the probability that a certain heartbeat belongs to a particular target class. This learning method is widely used in biostatistics where binary answers are expected, such as a condition is present or absent. The approach presented in this thesis requires computational inexpensive pre-processing of the ECG signals since it is based on the morphology of the heart's electrical activity. For comparison purposes, the guidelines of the Association for the Advancement of Medical Instrumentation for the evaluation of arrhythmia detector algorithms were followed. In order to avoid overfitting of the reservoir, we implemented a patient-oriented cross-validation method similar to the k-fold cross-validation but leaving out all the heartbeats from a patient instead of the k samples. Our results show improvements over previously reported performance under similar conditions of evaluation and using the same database. For instance, in a clinically relevant configuration of distinguishing the 5 classes of heartbeats in the database, we computed an average sensitivity of the reservoir to detect arrhythmias of 84.83%, more than one percent higher than the best configuration of previous reports.

By inspecting the state matrices that the reservoir produced, we were able to recognize which parts of the ECG were relevant for the classification. We realized that, for this particular task and dataset, the QRS complex was not as relevant as its surroundings for the classification of arrhythmias. This procedure, combined with the L1-regularization (that sets to zero the weights of the reservoir that are not relevant to perform the task), might be used as a feature selection method for this task. The use of an inexpensive pre-processing of heartbeat

signals combined with the fast processing of the reservoir computer leads to a real-time solution for the classification of cardiac arrhythmias.

The second widely explored family of tasks in this thesis was the time-series prediction tasks. We evaluated the performance of the reservoir for two different cases: the prediction of temperature variations that indicate the development of El Niño events, and the prediction of chaotic time-series given by the Lorenz system.

The first task consisted in the prediction of El Niño phenomenon using the Southern Oscillation Index (SOI) which gives an indication of the development and intensity of El Niño or La Niña events in the Pacific Ocean. The data for this task was obtained from a model described by a scalar delay differential equation that mimics most of the observed dynamics of the phenomenon. In collaboration with the PIK, a reservoir computer was set to estimate the temperature changes that provoke El Niño in a prediction horizon of 1, 3 or 6 months. We computed the NRMSE and found errors of about 3, 8 and 24%, respectively. The errors reported are given for the estimation of the full temperature changes that are described by fast oscillations in the temperature trends. However, the occurrence of El Niño phenomenon is described by slow oscillations that were predicted by our approach even for the 6-month prediction horizon.

We also evaluated the prediction capabilities of the reservoir computer by estimating chaotic time series. We used the well-known Lorenz system operating in a chaotic regime. We performed the estimation using two different inputs to estimate the future values of variable  $x$  of the Lorenz system. When the time trace of variable  $x$  is used to predict another time trace of the same variable from different initial conditions, we found a NMSE of about  $10^{-4}$ , whereas when using variable  $y$  to estimate variable  $x$  the error was of the order of  $10^{-3}$ . For this case, we see that using the same variable yields better results than using a different variable of the system. The introduction of several variables of a system as inputs in the reservoir computer was considered to evaluate its influence in the performance. For the time-series estimation task of the Lorenz system, the combination of the time traces from variables  $x$  and its delay version  $x(t - nT)$ , with  $n$  as the number of delay embedding steps and  $T$  as the time step of the Lorenz system, were used. In this case better results, than those obtained when using only the time series of  $x$ , were obtained. However, when estimating the values of variable  $x$  from the combination of variables  $x$  and  $y$ , the NMSE are about three orders of magnitude lower than when using the delay version of variable  $x$ . These two multivariate examples of time-series estimation already indicate that the reservoir computer is able to extract relevant information contained in the other variables of the system.

The multivariate scenario was also implemented for the electrocardiographic classification task. This presented some drawbacks since the database including two ECG channels is smaller (37 records) than the database used in the uni-

## CHAPTER 6. CONCLUSIONS

variate case (44 records). We also reported that the univariate case suffers from high variability, i.e. the reservoir has problems to generalize to new subjects. This problem is due to the few amount of records from different subjects in the database. The high variability problem causes a drop in performance of the univariate case when using the 37 records in common with the multivariate case. Finally, adding a different ECG channel, that lies in the perpendicular plane of the previous one, improves the results. The multivariate results rise to values that are comparable to the univariate case when using 44 records. A much patient-rich database is needed for the evaluation of this task. A bigger database should help to reduce the high variance problem of the reservoir.

To demonstrate the versatility of [reservoir computing](#), the nonlinear node used in the previous tasks, i.e. the Mackey-Glass oscillator, was replaced by the dynamics of a semiconductor laser. This allowed us to numerically study a system that can be implemented in photonics. Simulations under several different conditions of the laser parameters were carried out leading to similar results than those obtained with the Mackey-Glass oscillator.

The importance of the delay line in the reservoir, and thus its intrinsic memory, was studied along the different evaluated tasks. For instance, in the [SDR](#) task using the Mackey-Glass oscillator, we found that the reservoir does not need much memory to classify the digits (indicated by the operational point and the parameter value  $p = 7$ ). Due to the particularities of the semiconductor laser dynamics, simulations yield to the minimum error rate when the feedback of the reservoir is switched off. This is because the [SDR](#) task does not require considerable memory capacity in the reservoir and the transient dynamics already contains the relevant information.

The time-series estimation requires very different characteristics in the reservoir. For this kind of tasks, memory capacity in the reservoir plays an important role in the accuracy of the estimations. In the case of the chaotic laser dynamics, e.g. the Santa Fe time-series prediction, we found that in the absence or under small memory in the reservoir, the prediction errors rise. However, when the available memory was higher, the reservoir estimated the time series with low error rates.

The feedback of the reservoir was also switched off intentionally to study its influence in the multivariate prediction task of the Lorenz system. The results showed that the best performance was obtained when the fading memory of the reservoir was present. These results were better than the case in which there was no fading memory in the reservoir but the input was the multivariate input of  $x$  variable and its delay version  $x(t - T)$ , which is a way of introducing memory in the system via the input signal.

We briefly explored the effect of noise in the reservoir states using the specific integration method for delay equations subject to noise included in the computational suite. The influence of noise is important in real hardware im-

plementations. Using the semiconductor laser dynamics, we found that the noise degraded significantly the results for the Santa Fe estimation task. On the contrary, for the [SDR](#) task, the presence of noise did not play an important role. The sensitivity of the time-series prediction task comes from the fact that very precise transients are needed to accurately estimate future values of the time series. The [SDR](#) task, on the contrary, uses the [winner-take-all](#) criterion to make the classification.

The numerical simulations presented in this thesis shed some light on the understanding of the computational power contained in delay systems. Usually seen as a problem, delays in a system can be used beneficially for the execution of different tasks, e.g. chaos control and communications. This research work demonstrated, by means of real world applications, the usefulness of delay-coupled systems in the processing of information.

### 6.1

---

## Future perspectives

A promising continuation of this work is centered around the [ECG](#) classification task. Many challenges are still present, e.g. the acquisition of more data, the study of different features of the heartbeat, the use of multiple reservoirs, etc. For instance the inclusion of more features of the heartbeat might be considered beneficial from the fact that arrhythmic heartbeats occur faster or slower than the normal ones, then the interval of time that goes from the R-peak of one heartbeat to the R-peak of the immediately successive heartbeat, the R-R interval, might include relevant information for the classification, thus improving the results. The technique that we employed defines a general classifier that can be applied for many different users. However, a classifier that is patient specific might give better performance and could be achieved using a combination of the classification results of two different reservoirs: a general one trained with several patients and a specific reservoir of the patient under study.

Regarding the fundamental properties of the reservoir, future works should consider to study the design of mask matrices. There exist different methods to design masks that exploit the variability of the input signals, however it can sometimes happen that random masks perform better than deterministically-designed mask matrices. The reason for this remains unclear.

A different line of research to face in the future might be the study of different forms to add memory into the reservoir when the task requires more memory than the one the delay line can provide. This could be achieved by including multiple delay lines or an additional feedback connection from the output/readout layer into the nonlinear node.



---

# Supplementary material for Chapter 3

The information in this appendix was extracted directly from the documents of the MIT-BIH arrhythmia database from [Physionet](#).

## A.1

---

### MIT-BIH arrhythmia database

The database is constituted by 48 recordings from 47 subjects. The subjects were 25 men aged 32 to 89 years, and 22 women aged 23 to 89 years. The identifications of the [ECG](#) records is made by 3-digit non-continuous numbers ranging from 100 to 234. Records 201 and 202 come from the same male subject. There are two leads in each record. Lead A is modified limb lead II in most records, and lead B is usually modified lead V1 (occasionally V2 or V5, and in one instance V4). The modification to both leads A and B comes by placing the electrodes on the chest. A notable exception in the order of leads A and B is record 114 for which the signals are reversed. According to the [AAMI](#) recommendations, records including paced beats were removed from the study. Then the database

**Table A.1:** MIT-BIH Arrhythmia database recording names used for training and testing datasets

Dataset	MIT-BIH Arrhythmia record names
DS1	101, 106, 108, 109, 112, 114, 115, 116, 118, 119, 122, 124, 201, 203, 205, 207, 208, 209, 215, 220, 223, 230.
DS2	100, 103, 105, 111, 113, 117, 121, 123, 200, 202, 210, 212, 213, 214, 219, 221, 222, 228, 231, 232, 233, 234.

## APPENDIX A. SUPPLEMENTARY MATERIAL FOR CHAPTER 3

was divided into two datasets named: [DS1](#) for training and [DS2](#) for testing. The record identification used in each dataset are shown in [Table A.1](#).

### A.1.1 Annotations

Each heartbeat was initially labeled by a simple slope-sensitive QRS detector and marked as normal beat. The labeled recordings were then given to two cardiologists who worked on them independently adding additional beat labels where the detector missed beats, deleted false detections, and changed the labels for all abnormal beats. They also added rhythm labels, signal quality labels and comments. The annotations of the cardiologists were automatically compared beat-by-beat and return to them with the annotations in the margin and discrepancies highlighted. Each discrepancy was reviewed and resolved by consensus. Finally, cardiologists annotations were analyzed by an auditing program, which checked for consistency.

**Table A.2:** Physionet annotations: Beat annotations

MIT-BIH symbol	Meaning	AAMI label
N	Normal beat	N
.	Normal beat	N
L	Left bundle branch block beat	N
R	Right bundle branch block beat	N
e	Atrial escape beat	N
j	Nodal (junctional) escape beat	N
B	Bundle branch block beat (unspecified)	N
n	Supraventricular escape beat (atrial or nodal)	N
A	Atrial premature beat	S
a	Aberrated atrial premature beat	S
J	Nodal (junctional) premature beat	S
S	Supraventricular premature or ectopic beat (atrial or nodal)	S
V	Premature ventricular contraction	V
F	Fusion of ventricular and normal beat	V
E	Ventricular escape beat	V
r	R-or-T premature ventricular contraction	V
/	paced beat	F
f	Fusion of paced and normal beat	F
Q	Unclassified beat	Q
?	beat not classified during learning	Q



## A.1. MIT-BIH ARRHYTHMIA DATABASE

This database has suffered several modifications and adaptations to new definitions. For example the rhythm labels have been revised and now include notations for paced rhythm, bigeminy, and trigeminy. In October 2001, an error in the beat labels was discovered and corrected in record 209, among other modifications.

The usual annotations found in this database and their corresponding mapping to the [AAMI](#) 5 classes are shown in [Table A.2](#).

Also, some annotations that are not labels to particular beats but related to rhythms are shown in [Table A.3](#).

**Table A.3:** Physionet annotations: Non-beat annotations

MIT-BIH symbol	Meaning	AAMI label
[	start of ventricular flutter/fibrillation	Q
!	Ventricular flutter wave	Q
]	End of ventricular flutter/fibrillation	Q
x	Non-conducted P-wave	Q
p	Peak of P-wave	Q
	Isolated QRS-like artifact	Q
+	Rhythm change	Q
s	ST segment change	Q



---

# Glossary

## A

**artificial neural network** is a network of artificial neurons. An artificial neuron is a computational model inspired in biological neurons. The synapses of these artificial neurons are represented by the weight of the link among them. [4](#)

## C

**cardiovascular disease** is the term that refers to a broad class of diseases related with the heart and the cardiovascular system. Some of them include coronary heart disease causing arrhythmias, stroke, peripheral arterial and aortic disease. [63](#), [69](#)

**cross-validation** is a way of measuring the predictive performance of a statistical model. This is done by computing an error measure on a set of data not used in estimation of parameter. [12](#), [13](#), [36](#), [47](#), [50](#), [51](#), [54](#), [79](#), [90](#), [108](#), [115](#), [120](#)

## D

**doppler radar** is a radar tracking system using the Doppler effect to determine location and velocity. [31](#)

## E

**electrocardiogram** is a time series representing the electrical activity of the heart. [63](#)

## G

## Glossary

**generalized linear model** is an statistical tool that provides a flexible generalization of regular linear regression. It allows a linear model to be related to the outcome variable via a link function. [77](#)

## H

**Holter monitor** is a battery-operated portable device that measures and tape records of the heart's activity continuously for 24 hour or longer. The device is the size of a small camera with wires ending in electrodes that attach to the skin. [64](#)

## O

**one-vs-all** is a strategy used in multi-class classification machine learning algorithms that consists on training a single classifier per class considering the samples of that class as the first class and a mix of all other classes as the other class. [51](#)

## P

**Physionet** is a web page that concentrates many databases use in electrophysiology. It can be found at [www.physionet.org](http://www.physionet.org) [73](#), [125](#)

## R

**recurrent neural network** is a kind of artificial neural network where connections between neurons can form a loop introducing recurrence. [7](#), [31](#), [44](#)

**reservoir computing** is a type of RNN where the connections between neurons are kept fixed and only the connections to the output layer are trained. [vii](#), [ix](#), [xi](#), [6–8](#), [11](#), [17](#), [18](#), [26](#), [31](#), [35](#), [39](#), [40](#), [46](#), [55](#), [61](#), [63](#), [103](#), [122](#)

## S

**standard error of the mean** is the standard deviation of the sample means divided by the square root of the sample size. [48](#), [59](#)

## W

**winner-take-all** When comparing scores of a classifier, it refers to the competition among different classes. The resulting class will be the one with the highest score. [38](#), [48](#), [51](#), [113](#), [123](#)

---

# Acronyms

## A

**AAMI** Association for the Advancement of Medical Instrumentation [64](#), [65](#), [73](#), [76](#), [83](#), [86](#), [87](#), [125](#), [127](#)

**AI** Artificial Intelligence [4](#)

**ANN** Artificial Neural Network [4](#), [6](#), [7](#), [47](#), *Glossary:* [artificial neural network](#)

**AUC** Area Under the Curve [25](#), [46](#), [52](#)

## B

**BANN** Bayesian Artificial Neural Networks [64](#)

**BIH** Beth Israel Hospital [73](#)

## C

**CV** Cross-Validation [13–15](#), [50–52](#), [59](#), [80](#), [81](#), *Glossary:* [cross-validation](#)

**CVD** Cardio vascular disease [63](#), *Glossary:* [cardiovascular disease](#)

## D

**DDE** Delay Differential Equations [64](#)

**DS1** Data set 1 (use for training purposes) [73](#), [79–81](#), [85](#), [97](#), [126](#)

**DS2** Data set 2 (use for evaluation purposes) [73](#), [83](#), [85](#), [86](#), [97](#), [126](#)

**DWT** Discrete Wavelet Transformation [64](#)

## E

## Acronyms

**ECG** Electrocardiogram [63](#), [64](#), [68](#), [71–73](#), [76](#), [80](#), [82](#), [83](#), [87](#), [88](#), [97–100](#), [120–123](#), [125](#), *Glossary*: [electrocardiogram](#)

**ENSO** El Niño Southern Oscillation [53](#), [54](#)

**ER** Error rate [79–81](#)

**ESN** Echo State Network [8](#)

## F

**FN** False Negative [19](#), [23](#)

**FP** False Positive [19](#), [23](#)

**FPR** False Postive Rate [21](#), [24](#)

## G

**GLM** Generalized linear model [77](#), *Glossary*: [generalized linear model](#)

## I

**IRLS** Iterative Reweighted Least Squares [78](#)

## L

**L-BFGS** Limited-memory Broyden-Fletcher-Goldfard-Shanno [78](#)

**LD** Linear Discriminant [64](#), [83](#), [85](#)

**LDL-C** Low Density Lipoprotein Cholesterol [63](#)

**LOO** Leave-One-Out [14](#)

**LR** Logistic Regression [77](#), [78](#)

**LSM** Liquid State Machine [8](#)

## M

**MIT** Massachusetts Institute of Technology [73](#)

**MLII** Modified Limb lead II [73](#), [97–100](#)

## N

**NLN** NonLinear Node [32–34](#), [36](#), [37](#), [41–43](#), [59](#), [79](#), [92](#), [93](#), [98](#), [103](#), [104](#)

**NMSE** Normalized Mean-Square Error 91, 92, 94–96, 100, 110–113, 115, 121

**NRMSE** Normalized Root-Mean-Square Error viii, x, xii, 54–57, 121

### O

**OvA** One-versus-all 51, *Glossary: one-vs-all*

### P

**PIK** Potsdam Institute for Climate Impact 54, 121

**PMOF** Polarization-Maintained Optical Feedback 105, 106, 108–113, 115

**PROF** Polarization-Rotated Optical Feedback 105, 106, 108–113, 115

### R

**RC** Reservoir Computing 7, 31–33, 40, 63, 77, 87, 90, *Glossary: reservoir computing*

**RNN** Recurrent Neural Network 7, 8, 31, 35, *Glossary: recurrent neural network*

**ROC** Receiver Operating Characteristics 23–25, 52

### S

**s.e.m.** Standard error of the mean 48, 50, 51, 59, 60, *Glossary: standard error of the mean*

**SD** Standard Deviation 55, 76, 91, 92, 94–96

**SDR** Spoken Digit Recognition viii, 47, 52, 58, 60, 79, 107, 120, 122, 123

**SL** Semiconductor Laser 103, 104

**SM** Signal Modelling 64

**SOI** Southern Oscillation Index 53, 54, 121

**SOMLVQ** Self-Organizing Maps with Learning Vector Quantization 64

**SON** Self-Organizing Networks 64

**SVEB** Supra-Ventricular Ectopic Beat 83, 88

**SVM** Support Vector Machine 64

### T

## Acronyms

**TN** True Negative [19](#), [23](#)

**TNR** True Negative Rate [21](#)

**TP** True Positive [19](#), [22](#)

**TPR** True Postive Rate [20](#)

## V

**VEB** Ventricular Ectopic Beat [83](#), [88](#)

## W

**WER** Word Error Rate [48](#), [50](#), [51](#), [59](#), [61](#), [108–110](#), [114](#), [120](#)

**WTA** Winner-take-all [48](#), [51](#), *Glossary*: [winner-take-all](#)



---

## Bibliography

- [1] J. P. Crutchfield, W. L. Ditto, and S. Sinha, "Introduction to focus issue: intrinsic and designed computation: information processing in dynamical systems—beyond the digital hegemony," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 20, no. 3, p. 037101, 2010.
- [2] D. Woods and T. J. Naughton, "Optical computing: Photonic neural networks," *Nature Physics*, vol. 8, no. 4, pp. 257–259, 2012.
- [3] T. Chouard, "Legacy of a universal mind," *Nature*, vol. 482, no. 7386, pp. 455–455, 2012.
- [4] Glosser.ca, "Colored neural network," Licensed under CC BY-SA 3.0 via Wikimedia Commons.
- [5] D. Buonomano and M. Merzenich, "Temporal information transformed into a spatial code by a neural network with realistic properties," *Science*, vol. 267, no. 5200, pp. 1028–1030, 1995.
- [6] H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," *Science*, vol. 304, no. 5667, pp. 78–80, 2004.
- [7] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural computation*, vol. 14, no. 11, pp. 2531–2560, 2002.
- [8] L. Appeltant, M. C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. R. Mirasso, and I. Fischer, "Information processing using a single dynamical node as complex system," *Nature communications*, vol. 2, p. 468, 2011.
- [9] M. Fiers, K. Vandoorne, T. Van Vaerenbergh, J. Dambre, B. Schrauwen, and P. Bienstman, "Optical information processing: Advances in nanopho-

## BIBLIOGRAPHY

- tonic reservoir computing," in *Transparent Optical Networks (ICTON), 2012 14th International Conference on*. IEEE, 2012, pp. 1–4.
- [10] M. C. Mackey and L. Glass, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, no. 4300, pp. 287–289, 1977.
- [11] A. Uchida, R. McAllister, and R. Roy, "Consistency of nonlinear system response to complex drive signals," *Physical review letters*, vol. 93, no. 24, p. 244102, 2004.
- [12] B. Schrauwen, D. Verstraeten, and J. Van Campenhout, "An overview of reservoir computing: theory, applications and implementations," in *Proceedings of the 15th European Symposium on Artificial Neural Networks*. p. 471-482 2007, 2007, pp. 471–482.
- [13] D. Verstraeten, B. Schrauwen, M. d'Haene, and D. Stroobandt, "An experimental unification of reservoir computing methods," *Neural Networks*, vol. 20, no. 3, pp. 391–403, 2007.
- [14] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied logistic regression*. Wiley.com, 2013.
- [15] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.
- [16] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005.
- [17] T. Hastie, R. Tibshirani, and F. Jerome, *The elements of statistical learning*, 2nd ed. Springer, 2009, vol. 2, no. 1.
- [18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *The Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [19] G. Hall, E. Titchener, K. Dallenbach, M. Bentley, E. Boring, and M. Washburn, *The American Journal of Psychology*. University of Illinois Press, 1903.
- [20] H. Ebbinghaus, *Memory: A Contribution to Experimental Psychology*, ser. Columbia University. Teachers College. Educational reprints. no. 3. Teachers College, Columbia University, 1913.
- [21] D. M. W. Powers, "Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.

## BIBLIOGRAPHY

- [22] P. A. Flach and S. Wu, "Repairing concavities in roc curves." in *IJCAI*. Citeseer, 2005, pp. 702–707.
- [23] T. Fawcett, "An introduction to roc analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.
- [24] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (roc) curve." *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.
- [25] K. Hicke, M. Escalona-Morán, D. Brunner, M. C. Soriano, I. Fischer, and C. R. Mirasso, "Information processing using transient dynamics of semiconductor lasers subject to delayed feedback," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 19, no. 4, p. 1501610, July-Aug 2013.
- [26] H. Jaeger, "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*, vol. 148, 2001.
- [27] W. Maass, P. Joshi, and E. D. Sontag, "Computational aspects of feedback in neural circuits," *PLOS Computational Biology*, vol. 3, no. 1, p. e165, 2007.
- [28] D. Buonomano and W. Maass, "State-dependent computations: spatiotemporal processing in cortical network," *Nature Review Neurosciences*, vol. 10, pp. 113–125, 2009.
- [29] M. Rabinovich, R. Huerta, and G. Laurent, "Transient dynamics for neural processing," *Science*, vol. 321, no. 5885, pp. 48–50, 2008.
- [30] L. Larger, M. C. Soriano, D. Brunner, L. Appeltant, J. M. Gutiérrez, L. Pesquera, C. R. Mirasso, and I. Fischer, "Photonic information processing beyond Turing: an optoelectronic implementation of reservoir computing," *Optics express*, vol. 20, no. 3, pp. 3241–3249, 2012.
- [31] Y. Paquot, F. Duport, A. Smerieri, J. Dambre, B. Schrauwen, M. Haelterman, and S. Massar, "Optoelectronic reservoir computing," *Scientific reports*, vol. 2, 2012.
- [32] J. Schumacher, H. Toutounji, and G. Pipa, "An introduction to delay-coupled reservoir computing," in *Artificial Neural Networks*. Springer, 2015, pp. 63–90.
- [33] T. Erneux, *Applied Delayed Differential Equations*. Springer Science and Business Media, 2009.

## BIBLIOGRAPHY

- [34] M. Le Berre, E. Ressayre, A. Tallet, H. Gibbs, D. Kaplan, and M. Rose, "Conjecture on the dimensions of chaotic attractors of delayed-feedback dynamical systems," *Physical Review A*, vol. 35, no. 9, p. 4020, 1987.
- [35] F. Duport, B. Schneider, A. Smerieri, M. Haelterman, and S. Massar, "All-optical reservoir computing," *Optics express*, vol. 20, no. 20, pp. 22783–22795, 2012.
- [36] D. Brunner, M. C. Soriano, C. R. Mirasso, and I. Fischer, "Parallel photonic information processing at gigabyte per second data rates using transient states," *Nature communications*, vol. 4, p. 1364, January 2013.
- [37] L. Appeltant, "Reservoir computing based on delayed-dynamical systems," Ph.D. dissertation, Vrije Universiteit Brussel and Universitat de les Illes Balears, 2012.
- [38] M. C. Soriano, S. Ortín, D. Brunner, L. Larger, C. R. Mirasso, I. Fischer, and L. Pesquera, "Optoelectronic reservoir computing: tackling noise-induced performance degradation," *Optics express*, vol. 21, no. 1, pp. 12–20, 2013.
- [39] L. Appeltant, G. Van der Sande, J. Danckaert, and I. Fischer, "Constructing optimized binary masks for reservoir computing with delay systems," *Scientific reports*, vol. 4, p. 3629, 2014.
- [40] A. Rodan and P. Tino, "Minimum complexity echo state network," *Neural Networks, IEEE Transactions on*, vol. 22, no. 1, pp. 131–144, 2011.
- [41] M. C. Soriano, D. Brunner, M. Escalona-Morán, C. R. Mirasso, and I. Fischer, "Minimal approach to neuro-inspired information processing," *Frontiers in Computational Neuroscience*, vol. 9, p. 68, 2015.
- [42] H. Jaeger, "Short term memory in echo state networks," GMD Reports 152, Tech. Rep., 2002.
- [43] R. Martinenghi, S. Rybalko, M. Jacquot, Y. K. Chembo, and L. Larger, "Photonic nonlinear transient computing with multiple-delay wavelength dynamics," *Physical review letters*, vol. 108, p. 244101, 2012.
- [44] A. Namajūnas, K. Pyragas, and A. Tamaševičius, "An electronic analog of the mackey-glass system," *Physics Letters A*, vol. 201, no. 1, pp. 42–46, 1995.
- [45] M. C. Soriano, G. Van der Sande, I. Fischer, and C. R. Mirasso, "Synchronization in simple network motifs with negligible correlation and mutual information measures," *Physical review letters*, vol. 108, no. 13, p. 134101, 2012.

## BIBLIOGRAPHY

- [46] D. Nikolić, S. Häusler, W. Singer, and W. Maass, "Distributed fading memory for stimulus properties in the primary visual cortex," *PLoS biology*, vol. 7, no. 12, p. e1000260, 2009.
- [47] A. Goudarzi, M. R. Lakin, and D. Stefanovic, "Dna reservoir computing: A novel molecular computing approach," in *DNA Computing and Molecular Programming*. Springer, 2013, pp. 76–89.
- [48] K. Vandoorne, W. Dierckx, B. Schrauwen, D. Verstraeten, R. Baets, P. Bienstman, and J. Van Campenhout, "Toward optical signal processing using photonic reservoir computing," *Optics Express*, vol. 16, no. 15, pp. 11 182–11 192, 2008.
- [49] M. Hermans and B. Schrauwen, "Memory in linear recurrent neural networks in continuous time," *Neural Networks*, vol. 23, no. 3, pp. 341–355, 2010.
- [50] J. Dambre, D. Verstraeten, B. Schrauwen, and S. Massar, "Information processing capacity of dynamical systems," *Scientific reports*, vol. 2, 2012.
- [51] V. N. Vapnik and V. Vapnik, *Statistical learning theory*. Wiley New York, 1998, vol. 1.
- [52] H. Jaeger, *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "echo state network" approach*. GMD-Forschungszentrum Informationstechnik, 2002.
- [53] T. Instruments-Developed, "46-word speaker-dependent isolated word corpus (ti46)," NIST Speech Disc, 1991.
- [54] G. R. Doddington and T. B. Schalk, "Computers: Speech recognition: Turning theory to practice," *Spectrum, IEEE*, vol. 18, no. 9, pp. 26–32, 1981.
- [55] R. F. Lyon, "A computational model of filtering, detection, and compression in the cochlea," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'82.*, vol. 7. IEEE, 1982, pp. 1282–1285.
- [56] B. of Meteorology. Australian Government, "Climate glossary: Southern oscillation index," <http://www.bom.gov.au/climate/glossary/soi.shtml>.
- [57] I. Boutle, R. H. Taylor, and R. A. Römer, "El niño and the delayed action oscillator," *American Journal of Physics*, vol. 75, no. 1, pp. 15–24, 2007.
- [58] P. Whittle, "The analysis of multiple stationary time series," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 125–139, 1953.

## BIBLIOGRAPHY

- [59] A. F. Atiya and A. G. Parlos, "New results on recurrent network training: unifying the algorithms and accelerating convergence," *Neural Networks, IEEE Transactions on*, vol. 11, no. 3, pp. 697–709, 2000.
- [60] H. Jaeger, "Adaptive nonlinear system identification with echo state networks," in *Advances in neural information processing systems*, 2002, pp. 593–600.
- [61] WHO, "The top 10 causes of death." World Health Organization, Tech. Rep. factsheet 310, August 2013.
- [62] M. Nichols, N. Townsend, R. Luengo-Fernandez, J. Leal, A. Gray, P. Scarborough, and M. Rayner, "European cardiovascular disease statistics 2012," *European Heart Network, Brussels, European Society of Cardiology, Sophia Antipolis*, p. 116, 2012.
- [63] S. Mendis, P. Puska, B. Norrving *et al.*, *Reducing cardiovascular risk to prevent heart attacks and strokes. Global atlas on cardiovascular disease prevention and control*. World Health Organization (in collaboration with the World Heart Federation and World Stroke Organization), 2011.
- [64] C. T. Trialists *et al.*, "Efficacy and safety of more intensive lowering of ldl cholesterol: a meta-analysis of data from 170 000 participants in 26 randomised trials," *The Lancet*, vol. 376, no. 9753, pp. 1670–1681, 2010.
- [65] H. P. Selker, R. J. Zalenski, E. M. Antman, T. P. Aufderheide, S. A. Bernard, R. O. Bonow, W. B. Gibler, M. D. Hagen, P. Johnson, J. Lau *et al.*, "An evaluation of technologies for identifying acute cardiac ischemia in the emergency department: executive summary of a National Heart Attack Alert Program Working Group Report," *Annals of Emergency Medicine*, vol. 29, no. 1, pp. 1–12, 1997.
- [66] J. H. Pope and H. P. Selker, "Diagnosis of acute cardiac ischemia," *Emergency medicine clinics of North America*, vol. 21, no. 1, pp. 27–59, 2003.
- [67] P. Brugada and J. Brugada, "Right bundle branch block, persistent ST segment elevation and sudden cardiac death: a distinct clinical and electrocardiographic syndrome: a multicenter report," *Journal of the American College of Cardiology*, vol. 20, no. 6, pp. 1391–1396, 1992.
- [68] J. Pan and W. J. Tompkins, "A Real-Time QRS Detection Algorithm," *Biomedical Engineering, IEEE Transactions on*, vol. BME-32, no. 3, pp. 230–236, 1985.
- [69] Y. H. Hu, W. J. Tompkins, J. L. Urrusti, V. X. Afonso *et al.*, "Applications of artificial neural networks for ECG signal detection and classification," *Journal of electrocardiology*, vol. 26, pp. 66–73, 1993.

## BIBLIOGRAPHY

- [70] S. Osowski and T. H. Linh, "ECG beat recognition using fuzzy hybrid neural network," *Biomedical Engineering, IEEE Transactions on*, vol. 48, no. 11, pp. 1265–1271, 2001.
- [71] M. Lagerholm, C. Peterson, G. Braccini, L. Edenbrandt, and L. Sornmo, "Clustering ECG complexes using Hermite functions and self-organizing maps," *Biomedical Engineering, IEEE Transactions on*, vol. 47, no. 7, pp. 838–848, 2000.
- [72] Y. H. Hu, S. Palreddy, and W. J. Tompkins, "A patient-adaptable ECG beat classifier using a mixture of experts approach," *Biomedical Engineering, IEEE Transactions on*, vol. 44, no. 9, pp. 891–900, 1997.
- [73] P. De Chazal, M. O'Dwyer, and R. B. Reilly, "Automatic classification of heartbeats using ECG morphology and heartbeat interval features," *Biomedical Engineering, IEEE Transactions on*, vol. 51, no. 7, pp. 1196–1206, 2004.
- [74] M. Llamedo and J. Martinez, "Heartbeat Classification Using Feature Selection Driven by Database Generalization Criteria," *Biomedical Engineering, IEEE Transactions on*, vol. 58, no. 3, pp. 616–625, 2011.
- [75] C. Ye, M. T. Coimbra, and B. V. Kumar, "Investigation of human identification using two-lead electrocardiogram (ECG) signals," in *Biometrics: Theory Applications and Systems (BTAS), 2010 Fourth IEEE International Conference on*. IEEE, 2010, pp. 1–8.
- [76] Z. Zidelmal, A. Amirou, D. Ould-Abdeslam, and J. Merckle, "ECG beat classification using a cost sensitive classifier," *Computer methods and programs in biomedicine*, 2013.
- [77] M. Engin, "ECG beat classification using neuro-fuzzy network," *Pattern Recognition Letters*, vol. 25, no. 15, pp. 1715–1722, 2004.
- [78] G. Karraz and G. Magenes, "Automatic Classification of Heartbeats using Neural Network Classifier based on a Bayesian Framework," in *Engineering in Medicine and Biology Society, 2006. EMBS '06. 28th Annual International Conference of the IEEE*, 2006, pp. 4016–4019.
- [79] A. K. Mishra and S. Raghav, "Local fractal dimension based ECG arrhythmia classification," *Biomedical Signal Processing and Control*, vol. 5, pp. 114–123, 2010.
- [80] C. Lainscsek and T. J. Sejnowski, "Electrocardiogram classification using delay differential equations," *Chaos*, vol. 23, no. 2, pp. 023 132–023 132–9, 2013.

## BIBLIOGRAPHY

- [81] R. Mark and R. Wallen, "AAMI-recommended practice: Testing and reporting performance results of ventricular arrhythmia detection algorithms," *Association for the Advancement of Medical Instrumentation, Arrhythmia Monitoring Subcommittee, AAMI ECAR*, 1987.
- [82] A. for the Advancement of Medical Instrumentation and A. N. S. Institute, *Testing and Reporting Performance Results of Cardiac Rhythm and ST-segment Measurement Algorithms*, ser. ANSI/AAMI. The Association, 1999.
- [83] A. Natale and F. Marchlinski, *Handbook of Cardiac Electrophysiology*. Taylor & Francis, 2007. [Online]. Available: <http://books.google.es/books?id=lhZBND65SucC>
- [84] W. Einthoven, "The different forms of the human electrocardiogram and their signification." *The Lancet*, vol. 179, no. 4622, pp. 853–861, 1912.
- [85] F. N. Wilson, F. D. Johnston, A. G. Macleod, and P. S. Barker, "Electrocardiograms that represent the potential variations of a single electrode," *American Heart Journal*, vol. 9, no. 4, pp. 447–458, 1934.
- [86] F. N. Wilson, F. D. Johnston, F. F. Rosenbaum, H. Erlanger, C. E. Kossmann, H. Hecht, N. Cotrim, R. M. de Oliveira, R. Scarsi, and P. S. Barker, "The precordial electrocardiogram," *American Heart Journal*, vol. 27, no. 1, pp. 19–85, 1944.
- [87] R. E. Mason and I. Likar, "A new system of multiple-lead exercise electrocardiography," *American heart journal*, vol. 71, no. 2, pp. 196–205, 1966.
- [88] M. S. Thaler, *The only EKG book you'll ever need*. Lippincott Williams & Wilkins, 2010, vol. 365.
- [89] NIH, *Types of arrhythmia*. National Heart, Lung, and Blood Institute, 2011, vol. [www.nhlbi.nih.gov/health/health-topics/topics/arr/types](http://www.nhlbi.nih.gov/health/health-topics/topics/arr/types).
- [90] R. Mark, P. Schluter, G. Moody, P. Devlin, and D. Chernoff, "An annotated ECG database for evaluating arrhythmia detectors," *Front. Eng. Health Care*, pp. 205–210, 1982.
- [91] Y. Pawitan, *In all likelihood: statistical modelling and inference using likelihood*. Oxford University Press, 2001.
- [92] J. A. Nelder and R. Baker, "Generalized linear models," *Encyclopedia of Statistical Sciences*, 1972.
- [93] S. Menard, *Applied logistic regression analysis*. Sage, 2002, vol. 106.
- [94] J. E. Gentle, *Matrix algebra: theory, computations, and applications in statistics*. Springer Science & Business Media, 2007.



## BIBLIOGRAPHY

- [95] R. Malouf, "A comparison of algorithms for maximum entropy parameter estimation," in *proceedings of the 6th conference on Natural language learning-Volume 20*. Association for Computational Linguistics, 2002, pp. 1–7.
- [96] G. Andrew and J. Gao, "Scalable training of l1-regularized log-linear models," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 33–40.
- [97] F. Takens, "Detecting strange attractors in turbulence," in *Dynamical systems and turbulence, Warwick 1980*. Springer, 1981, pp. 366–381.
- [98] L. Cao, A. Mees, and K. Judd, "Dynamics from multivariate time series," *Physica D: Nonlinear Phenomena*, vol. 121, no. 1, pp. 75–88, 1998.
- [99] E. N. Lorenz, "Deterministic nonperiodic flow," *Journal of the Atmospheric Sciences*, vol. 20, no. 2, pp. 130–141, 1963.
- [100] M. Escalona-Morán, M. C. Soriano, J. García-Prieto, I. Fischer, and C. R. Mirasso, "Multivariate nonlinear time-series estimation using delay-based reservoir computing," *The European Physical Journal Special Topics*, vol. 223, no. 13, pp. 2903–2912, 2014.
- [101] R. Lang and K. Kobayashi, "External optical feedback effects on semiconductor injection laser properties," *Quantum Electronics, IEEE Journal of*, vol. 16, no. 3, pp. 347–355, 1980.
- [102] T. Heil, A. Uchida, P. Davis, and T. Aida, "Te-tm dynamics in a semiconductor laser subject to polarization-rotated optical feedback," *Physical Review A*, vol. 68, no. 3, p. 033811, 2003.
- [103] A. S. Weigend, "Time series prediction: forecasting the future and understanding the past," *Santa Fe Institute Studies in the Sciences of Complexity*, 1994.
- [104] U. Huebner, N. Abraham, and C. Weiss, "Dimensions and entropies of chaotic intensity pulsations in a single-mode far-infrared  $\text{nh } 3$  laser," *Physical Review A*, vol. 40, no. 11, p. 6354, 1989.
- [105] W. Walker, P. Lamere, P. Kwok, B. Raj, R. Singh, E. Gouvea, P. Wolf, and J. Woelfel, "Sphinx-4: A flexible open source framework for speech recognition," 2004.
- [106] D. Verstraeten, B. Schrauwen, D. Stroobandt, and J. Van Campenhout, "Isolated word recognition with the liquid state machine: a case study," *Information Processing Letters*, vol. 95, no. 6, pp. 521–528, 2005.
- [107] A. Rodan and P. Tino, "Minimum Complexity Echo State Network," *Neural Networks, IEEE Transactions on*, vol. 22, no. 1, pp. 131–144, 2011.



