UNIVERSITÀ DEGLI STUDI DI MILANO
FACOLTÀ DI SCIENZE E TECNOLOGIE

Corso di Laurea Triennale in Fisica

SHORT-TIME FORECASTING USING CHAOS THEORY: THE CASE OF THE ITALIAN ENERGY MARKET

Relatore:
Marco Gherardi
Correlatore:
Pietro Rotondo

Tesi di Laurea di:
Luca Zilli
Matricola: 911029

Anno Accademico 2019/2020

Abstract

The scope of this work is to design an algorithm to predict the energy's unbalances of the Italian power grid controlled by Terna S.p.A. To achieve this aim we followed three different paths. Firstly we implemented an algorithm based on insights from dynamical systems and chaos theory implemented by George Suigihara and Robert M. May in the nineties. The idea takes inspiration from Takens's embedding theorem and it is similar, in spirit, to the k-nearest neighbors method commonly employed in machine learning. Secondly, since the algorithm was reveled to be very slow in the computational sense, we decided to implement a more manageable k-nearest neighbors. Summing up the latter was reveled to be similar in effectiveness and faster in execution. At last, we decided to implement an algorithm of artificial intelligence using a recurrent neural network. This method is the one with more potential due to its predictive power that increases with the number of data available. We are going to start by reviewing the basics elements of chaos theory and strange attractors, including the illustration of three chaotic maps (the Logistic map, the Lorenz system and the Mackey-Glass equation) on which will be tested the three methods. Subsequently we present a brief introduction about artificial intelligence in which we presented the k-nearest neighbor algorithm and we shew the necessary elements to understand the functioning of the recurrent neural networks. In chapter three and four we shew respectively the analysis of predictions of synthetic data (i.e. the Logistic map, the Lorenz system and the Mackey-Glass equations) and the analysis of the energy's unbalances' predictions. Concerning the chaotic maps, we couldn't understand which was the best methods among the three ones. However, concerning the energy's unbalances, we noticed that the predictions made by the recurrent neural network are the only satisfying ones.

Contents

Introduction

In the modern management business, *forecasting* is an important part of the *decision-making* process. *Online Business Dictionary* defines forecasting as *A planning tool that helps management in its attempts to cope with the uncertainty of the future, relying mainly on data from the past and present and analysis of trends. Forecasting starts with certain assumptions based on the management's experience, knowledge, and judgment. These estimates are projected into the coming months or years using one or more techniques such as Box-Jenkins models, Delphi method, exponential smoothing, moving averages, regression analysis, and trend projection. Since any error in the assumptions will result in a similar or magnified error in forecasting, the technique of sensitivity analysis is used which assigns a range of values to the uncertain factors (variables).* Companies use forecasting methods to predict business outcomes, since this is useful to create estimations that could be used by managers to develop a more suitable strategy. Nowadays every organization has a specialized team who uses statistic and numerical tools to overcome the problem of prediction. All those companies, who are interested in prediction, tend to develop programs that make large use of machine learning. With this thesis we want to test also alternative tools which might solve the same problems in a more manageable way. The method that a company might use varies accordingly to the type of data available. We believe that physics and, the chaos theory in particular, could be involved in this. Speaking of this, we would like to highlight the work of two scientists, Suigihara and May, who, in the nineties, published a method by which it could be possible to distinguish a noisy limit cycle from a chaotic pattern. Their aim was to make the distinction by looking at the trend of the correlation coefficient between both the values of the time series and the values of that

time series predicted with the method. What matters is that the method is effective when it comes to make short-time predictions and, in fact, it will be tested on the Italian energy market.

1

ELEMENTS OF CHAOS THEORY

The chaos theory is commonly defined as a branch of mathematics which focuses on the study of dynamical systems whose apparently random states of disorder and irregularities are often governed by deterministic laws that are highly sensitive to initial conditions. The apparent randomness of complex systems hides underlying patterns as self-similarity and fractals. Those universal features that characterize chaotic systems can be spotted in the fields of biology, electrical engineering, weather systems and many more, giving to chaos theory an interdisciplinary role which comes in support of Physics and of other several scientific fields.

In this chapter we will illustrate the basics features of chaos theory that inspired the creators to build the algorithm, whereas, in chapter 3, we will explain how to implement the method of Suigihara and May.

1.1 History and a little overview about chaos in physics

The first works related to chaos dates back to the 1880s, when Henri Poincaré was studying the three-body problem [4]. He found out that, for some ranges of parameters and initial conditions the dynamic led to nonperiodic orbits that do not approach a fixed point. Later in 1898, Jacques Hadamard published an influential study about the chaotic motion of a free particle gliding frictionlessly on a surface of constant negative curvature, this study is called *Hadamard's billiards* [19]. Hadamard shows that all solutions present unstable trajectories, meaning that all nearby particles' motions diverge exponentially from one other, with a positive Lyapunov exponent (see 1.2). In the first part of the XIX century some scientists were focusing on studies related to nonlinear differential equations (like George David Birkhoff, Andrey Nikolaevich Kolmogorov and Mary Lucy and many more). However, they were more concerned about nonlinear oscillations in dynamic, which found their application in physics and engineering, than on chaos, that remained on the background. The turning point arrived in the 1950s with the invention of the first electronic computer. Much of the mathematics in chaos theory involves the repeated iteration between simple finite-difference equations, which would be counter-productive if done by hand. Electronic computers made these repeated calculations faster and practical, while figures and images made the visualization of their solution possible. In 1963 a simulation experiment made by Lorenz [26], accidentally brought the physics community to the discovery of a strange attractor (1.3.1) that gave light to the chaotic world. He was studying a model of convention rolls in the atmosphere when he discovered that small changes in initial conditions were producing large changes in long-term outcomes and trajectories which were never settling down to equilibrium, but they were continuing their motion in an irregular aperiodic way. Tiny errors in the measurements of initial conditions of some variable led to heavy errors in predictions. Actually, even detailed atmosphere modeling cannot, in general, make precise long-term weather predictions.

The 1970s were the most important years for chaos theory, Lorenz's work became the main point of reference for the studies carried on in the following years. Important were the works of Ruelle and Takens on turbulence in fluid [29] and the work of May which brought chaos theory to study about the changes of populations in biology [2]. The most important discover for the Physics community was made by Mitchell Feigenbaum (1975) [13] and Coullet & Tresser (1978) [10], who discovered

the universality feature of chaos, allowing the application of the theory to many different phenomena and, indeed, building a link between the laws governing the phase transition and chaos. The following discoveries and the numerous insights on applications led, in the 1980s, several scientists (physicists and not) to to the study of dynamical systems and chaos theory.

1.2 When the dynamic leads to chaos

In common words, chaos means disorder. Whereas, in mathematics the definition of chaos is more precise, although there is not an universally accepted one.

Strogatz, in his book defines chaos as *an aperiodic long-term behavior in a deterministic system that exhibits sensitive dependence on initial conditions.* Where

1. With *aperiodic long-term behavior* he implies the existence of an open set of initial conditions, which leads to aperiodic trajectories. This means that, given random initial conditions, aperiodic trajectories must occur with nonzero probability.

2. With *deterministic* it is meant that the system has no noisy elements . The irregular behavior is originated by the nonlinearity of the equations that rule the motion.

3. With *sensitive dependence on initial conditions* it is meant that nearby points in the phase space separate rapidly from each other.

The last property is the most practical and significant one, however, in many cases it can be forgotten, because under appropriate hypothesis, it is the consequence of two other statements. In these cases the description of Strogatz is redundant and it can be expressed with the following definition [12]. *Let $f : I \longrightarrow I$ be a continuous map on I, where I is an interval, is said to be chaotic if:*

1. *f is transitive*

2. *The set of its periodic points P is dense in I.*

The Lyapunov Characteristic Exponent as a measurement of sensitivity to initial conditions. *Sensitivity to initial conditions* means that each point in a chaotic system has significantly different future paths from its neighbor points. So, either a little perturbation or a small error in the measurements of the current trajectory might lead to a relevantly different future behavior.
In mathematics, the Lyapunov exponent measures the sensitivity to initial conditions in the form of the rate of exponential divergence from the perturbed initial conditions. To examine the behavior of an orbit around a point $\vec{X}'(t)$ in the phase space, we need to perturb the system with a quantity $\vec{U}(t)$ like

$$\vec{X}(t) = \vec{X}'(t) + \vec{U}(t) \tag{1.1}$$

where $\vec{U}(t)$ is the average deviation form the unperturbed solution at time t. In the region of the phase space in which we have a chaotic behavior, the Lyapunov characteristic exponent (LCE) σ is independent from $\vec{X}'(0)$ and its value is given by the Oseledec theorem, which states that

$$\sigma_i = \lim_{t\to\infty} \frac{1}{t} \ln |\vec{U}(t)|. \tag{1.2}$$

For an n-dimensional phase space mapping we have n LCEs and one Lyapunov characteristic exponent needs to be 0, since there will never be a divergence of a perturbed trajectory into the direction of the unperturbed motion. Moreover the value of the LCE is a measurement of the rate of exponential divergence: : the higher it is the greater it will be the rate that satisfies a law similar to $|\delta\vec{X}(t)| \simeq e^{\sigma_i t}|\delta\vec{X}(0)|$. Let (x_0, y_0) be the initial condition of a trajectory of a 2-dimensional Cauchy's problem and, let $(x', y') = (x_0 + d_x, y_0 + d_y)$ be the initial conditions of nearby trajectories of the same system. Here $d_0 = |(d_x, d_y)|$ is the distance at time $t_0 = 0$, while the distance between trajectories at the k-th iteration is

$$d_k = |(x'_k - x_k, y'_k - y_k)|. \tag{1.3}$$

Indeed the mean exponential rate of divergence of the trajectories is defined by

$$\sigma_1 = \lim_{k\to\infty} \frac{1}{k} \ln \frac{d_k}{d_0}. \tag{1.4}$$

For n-dimensional phase space map, the n LCEs are ordered $\sigma_1 > \sigma_2 > ... > \sigma_n$. Since σ_1 is bigger then the others, it will dominate the other exponents and, for this, 1.4 is useful to find only the maximal Lyapunov exponent (MLE) that is, actually, the most significant, since it determines overall the predictability of the system [35].

Non periodicity. Chaotic systems are just a class of dynamical systems, in fact, in several cases chaotic motion appears just under determinate initial conditions and in some areas of parameters' space. Indeed the same dynamical that leads to chaos, might lead to other kind of motion or attractor, even to periodic orbit. What characterizes chaotic systems is the repelling property of the periodic sequence. The motion could starts arbitrarily close to a periodic sequence, that the flow will anyway bring the system state away from it. Thus, for almost all initial conditions, the variable evolves chaotically following a non-periodic orbit.

Topological transitivity. There is more than one definition for Topological transitive system, an intuitive one is the following.

May X be a metric space with some metric d and let $f : X \longrightarrow X$ be continuous. The dynamical system (X, f) is called topologically transitive if for every pair of non-empty open sets $U, V \subset X$, there is a positive integer n such that $f^n(U) \cap V \neq \emptyset$ [25].

Intuitively in a topological transitive system, for a point x in U it exists a point y close to x whose orbit passes through V. Topological transitivity is sometimes omitted in common accounts of chaos, which equate it with only to a system sensitive of initial conditions. However the latter property alone does not imply chaos.

An example could be the simple dynamical system produced by repeatedly doubling an initial value: $x_{n+1} = 2x_n$ or $x_n = 2^n x_0$. This system has a sensitive dependence on initial conditions everywhere, since every pair of nearby points eventually becomes more and more separated from each other and the MLE of the equation is positive and easy calculate.

$$\sigma = \lim_{n\to\infty} \frac{1}{n} \ln \frac{x_n}{x_0} = \lim_{n\to\infty} \frac{1}{n} \ln \frac{x_0 2^n}{x_0} = \lim_{n\to\infty} \ln 2 = \ln 2 > 0.$$

However, its behavior is rigid, not chaotic: all positive points tend to positive infinity and all

negative point tend to negative infinity, except 0, the only fixed point.

Density of periodic orbits. Having dense periodic orbits, in a chaotic system, means that every random point of the phase space is, anyway, arbitrarily close to periodic orbits. For example the Logistic map (1.3.2) $x_{n+1} = 4x_n(1-x_n)$ is probably the simplest dynamical system provided with dense periodic orbits. If we start the evolution from the point $x_0 = \frac{5-\sqrt{5}}{8}$, we begin a periodic motion of period 2 consisting of a repetition of the value $\frac{5+\sqrt{5}}{8}$ and $\frac{5-\sqrt{5}}{8}$. A characterization was made by Sharkovskii [30]. He elaborated a theorem which states that every continuous one-dimensional system showing a cycle of period three, must display regular cycles of every other length.

Strange attractors as an example of fractals in Physics. As it previously said, some dynamical systems display chaotic behavior only in some subsets of the phase space. The most fascinating cases happen when the chaotic behavior takes place on an attractor. Two arbitrarily close points on the attractor might be significantly apart during following moments, thus nobody knows exactly where the system will be. What's more, this attractor never closes on itself: the motion never repeats. There is only one restriction: the system will remain on the attractor which, for this behavior, it is commonly called *strange*.
What we have just described is something complicated to define in a meticulous way. There are still different optional definitions. In this thesis, we will agree with the definition formulated by Strogatz in his book [31].
An attractor is a closet set A with the following properties:

1. A is an invariant set: any trajectory $x(t)$ that start in A, stays in A for every t.

2. A attracts an open set of initial condition: there is an open set U containing A such that if $x(0) \in U$, then the distance between $x(t)$ and A tends to zero as $t \to \infty$. This means that A attracts all trajectories which start sufficiently close to it. U is called the *basin of attraction* of A.

3. A is minimal: there is no proper subset of A that satisfies both conditions 1 and 2.

Unlike fixed-points and limit cycles, strange attractors, appearing from a chaotic dynamic, are detailed and complex. They occur in both continuous systems (such as the Lorenz system) and

some discrete ones (such as the Hénon map). Sometimes dynamical systems can display a repelling structure called *julia set*, that can be found at the boundary between the basins of attraction of fixed points. Julia sets behave like a strange "repeller" and, like strange attractors, they have a fractal structure.

A fractal is a geometric object which satisfies the *self-similarity* property. If you zoom on a fractal object, it will always have the same shape as in the original scale.

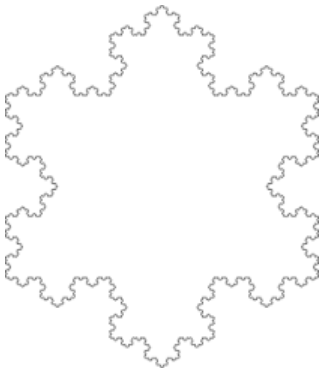


Figure 1.1: A fractal is an object that is equal to itself at every scale. An example is the perimeter of *Koch snowflake*, the *Koch curve*.

The property of self-similarity that defines a fractal is strictly related to the concept concept of dimension. If you divide a segment into N identical copies of itself, each copy will be scaled by a quantity $r = \frac{1}{N}$. In a similar way, if we want to divide a square in four smaller squares, each part will be scaled by a factor $r = \frac{1}{4}^{\frac{1}{2}}$.

The criterion can be generalized. If we want to divide a D-dimensional object in N parts, each part will be scaled by a factor

$$r = \frac{1}{N}^{\frac{1}{D}}. \tag{1.5}$$

Now we just need to invert the formula to get

$$D = \frac{\ln N}{\ln \frac{1}{r}}. \tag{1.6}$$

The 1.6 is the *Fractal dimension* also called *Similarity dimension* [7]. As an example, let's

compute the dimension of the Koch curve built by the following iterating commands:

1. Draw a straight line segment.
2. Divide the line in three equal segments.
3. Build an equal triangle on the central segment.
4. Remove the initial central segment.
5. For every new segment repeat from point 2.

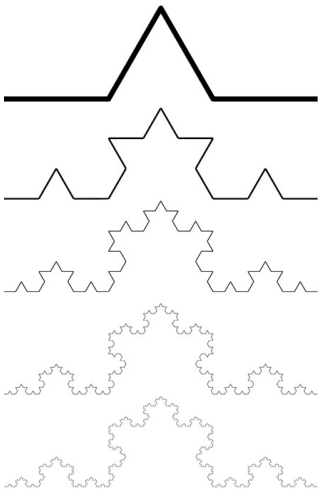


Figure 1.2: In this figure *Koch curve*'s building mechanism.

At each step, from an initial object we obtain $N = 4$ new elements, each scaled by a factor $r = \frac{1}{3}$. The fractal dimension is easy to calculate. Using 1.6 we get:

$$D = \frac{\ln 4}{\ln 3} = 1.26186.$$

Fractals are uncommon geometric objects, which can satisfy special properties. Koch constructed his curve in 1904 as an example of non-differentiable curve, in other words, a continuous curve that miss a tangent line at any of its points. If you start building the curve on each side of an equal triangle, you will obtain a geometric figure called *Koch snowflake* (see figure 1.1): this is the example of an object having both a finite area and an infinite perimeter.

1.3 Chaotic maps: examples

In this section it is given a brief overview about three important chaotic maps on which it will be tested the method that will be used to predict the Italian energy's unbalances.

1.3.1 Lorenz system

As it has already been said, a turning point in the history of chaos occurred thanks to the simulation made by Ed Lorenz in 1963 [26]. Precisely, during his studies on the convention rolls in the atmosphere, he invented an extremely simplified model from which extracted a three dimensional system of equations:

$$\begin{cases} \dot{x} &= \sigma(y-x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= -\beta z + xy \end{cases} \tag{1.7}$$

where $\sigma, \rho, \beta > 0$ are parameters. σ is the *Prandtl number*, ρ is the Rayleigh number, and β, although it does not have a name, in the convection problem studied by Lorenz, it is linked to the height of the fluid layer. The same equations can be find in other models like laser, dynamos and, as reported in [31], this system can perfectly describe the complex motion of a waterwheel.
After deriving his equations, Lorenz discovered that, over a wide range of parameters, those equations display an intricate dynamics: the solution oscillate irregularly, never returning exactly in the same trajectories of the previous loops, but always remaining in the same bounded region. The function founded by Lorenz, informally called *strange attractor*, is not a curve (like limit cycles could be) nor a surface. It is a fractal.

The properties of the Lorenz equations.

1. The system has only two nonlinearities: the quadratic terms xy and xz.

2. There is an important symmetry in the Lorenz system, the motion is invariant for the transformation $(x,y) \to (-x,-y)$: indeed if $(x(t),y(t),z(t))$ is a solution, $(-x(t),-y(t),z(t))$ is a solution.

3. The Lorenz system is dissipative: in the phase space the volume shrinks under the flow. Generally speaking, we could consider a motion like $\dot{x} = f(x)$. Let $V(t)$ and $S(t)$ to be the volume and its surface in the phase space at time t, then we could let them evolve for an infinitesimal dt. If we call $S(t+dt)$ the surface at time $t+dt$, what can we say about the volume $V(t+dt)$? May be n the normal of the surface, since f is the instantaneous velocity, $n \cdot f$ is the normal component of the velocity to the surface. Indeed in an interval dt, from a small area dA emerges a volume $(n \cdot f\text{dt})\text{dA}$. Therefore in mathematical language:

$$V(t+dt) = V(t) + \int_S (n \cdot f dt) dA. \tag{1.8}$$

Using divergence theorem:

$$\dot{V} = \frac{V(t+dt) - V(t)}{dt} = \int_S n \cdot f. \tag{1.9}$$

Finally, we rewrite the integral above by the divergence theorem, and we get

$$\dot{V} = \int_V \nabla f dV. \tag{1.10}$$

And for the Lorenz system,

$$\nabla f = \frac{d[\sigma(y-x)]}{dx} + \frac{d[\rho x - y - xz]}{dy} \frac{d[xy - \beta z]}{dz} = -\sigma - 1 - b < 0. \tag{1.11}$$

Since the divergence is constant the integral is trivial and the equation becomes

$$\dot{V} = -(\sigma + 1 + \beta)V, \tag{1.12}$$

whose solution is

$$V(t) = V(0)e^{-(\sigma+1+\beta)}. \tag{1.13}$$

The volumes in the phase space decades exponentially fast. This short proof explains us that all trajectories could start from everywhere, but they would eventually finish to end up

somewhere in a limiting set; that consists of fixed points, limit cycles and, if we choose the right parameters, a strange attractor.

The Lorenz attractor. Letting aside what's related to the fixed points and cycles, we can focus on the peculiar element of chaos: the strange attractor. Lorenz studied a particular case $\sigma = 10, \beta = 8/3, \rho = 28$. He begun integrating from the position (0,1,0), close to the saddle point at the origin. After an initial transient phase at the beginning, the system establishes itself in an irregular aperiodic oscillation that will go on for $t \to \infty$. Lorenz discovered that a magnificent frame will appear if the solution is plotted in a phase space. This is the strange attractor. Which is the numerical structure of the strange attractor?

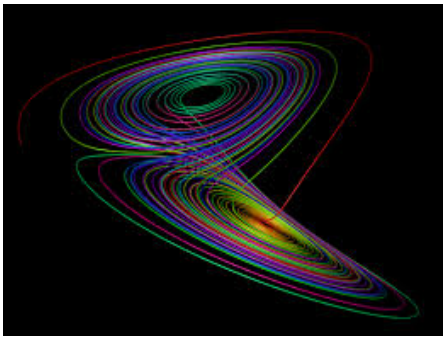


Figure 1.3: The *Lorenz attractor*.

The Figure 1.3 suggests the idea of a pair of figures melding together in the central part of the plot. This could be wrongly confused with a paradox which confutes the uniqueness theorem. However, Lorenz, in 1963, gave us an explanation: the strong volume contraction of the flow plus the insufficient numerical resolution just give us the illusion of intersecting trajectories ...*we see that each surface is really a pair of surfaces, so that, where they appear to merge, there are really four surfaces. Continuing the process for another circuit, we see that there are really eight surfaces... and finally we conclude that there is an infinite complex of surfaces, each extremely close to one or the other of two merging surfaces.* Nowadays this *infinite complex of surfaces* would be called a fractal. It is a set of points with zero volume but infinite surface area.

1.3.2 Logistic map

Unlike the Lorenz system, which uses a set of differential equations, the Logistic map uses a non-linear difference equation to find values following a discrete time-step. The name *Logistic* comes from the propriety of mapping the population at any time step to its value at next time step:

$$x_{t+1} = rx_t(1 - x_t) \tag{1.14}$$

In our frame x represents the population at time t, and r is the growth rate parameter. If r is too low the population will decrease until it extinguishes. By using higher values of r the population might converge to a stable value or fluctuate between a series of populations' ups and downs. Specifically for some values of the growth rate the Logistic map can provide a chaotic behavior. It is in this domain that we will analyze the map will be explored here. In the simulations made and shown in the third chapter it has been used $r = 4$ [5].

1.3.3 Mackey-Glass equation

The Mackey-Glass equation is a nonlinear time delay differential equation

$$x_{k+1} = c \cdot x_k + \frac{a \cdot x_{k-d}}{b + x_{k-d}^e} \tag{1.15}$$

where a,b,c,e are real numbers, and d represents the delay of the variable x at time t. As happens with the other systems, this equation provides a range of periodic and chaotic dynamics.

Originally, Mackey and Glass formulated this equation to display the features of complex dynamics occurring in the physiological control systems that they were studying [28]. They thought that many physiological disorders, called *dynamical diseases*, were characterized by changes in qualitative features of dynamics. The qualitative changes in physiological dynamics corresponded mathematically to the bifurcations of the system that, in the equation's dynamic, might be induced by some change in parameters' values.

On the other other hand, the main feature, distinguishing this system from the other two analyzed, is the presence of a lagged coordinate in the equation. In feedback systems' theory - that is the system class in which physiological control systems take part - there usually is a time lag between

the sensing of the value of a variable under control, and the mounting of an appropriate response and, in physiological systems, this time lag can be substantial. For example, after a loss of blood cells, it can take up to several days before new blood cells are produced through the activation, differentiation and proliferation of the appropriate blood stem cells. One of these equations, which displays a sequence of period-doubling bifurcations and chaotic dynamics, has become known as the Mackey-Glass equation (1977). In this equation, x is the concentration of circulating blood cells, a and e are constants involved in the description of the dependence of the production of these cells as a function of x_τ [15].

1.4 Takens' Embedding Theorem

During experiments, the measurements of times series, due to either technologies limitations or ignorance of the variables in the system state vector, can provide us only with a one-dimensional time series $y(t) = \Phi(\vec{x}(t))$, where $\Phi : \mathbb{R}^N \longrightarrow \mathbb{R}$ is a smooth *observation* or *measurement function*. Takens' embedding theorem tell us that information about the hidden states of a dynamical system can be preserved even in a one-dimensional output.

Suppose we have a dynamical system having states $\vec{x}(t) \in \mathbb{R}^N$, which evolve satisfying the differential equation:

$$\dot{\vec{x}} = \Psi(\vec{x}) \tag{1.16}$$

where $\Psi : \mathbb{R}^N \longrightarrow \mathbb{R}^N$ is the vector field of the dynamical system. Note that when the evolution of the system states are known, the system sates can be recovered over time from the one-dimensional output measurements by the Kalman filter [24]. But without its knowledge, to get the information on the time series, we must use the theorem of Takens. We assume that Ψ is a smooth function and that is confined to a *submanifold* $\mathscr{M} \subset \mathbb{R}^N$. To lighten the notation and talk easily about system states $\vec{x}(t) \in \mathscr{M}$ at a given time t, we define the flow function $G : \mathscr{M} \times \mathbb{R} \longrightarrow \mathscr{M}$ by

$$\vec{x}(t_0 + T) = G(\vec{x}(t_0), T), \tag{1.17}$$

for some $T \in \mathbb{R}$. G is not explicitly calculable, but can be related to the vector field Ψ by

$$\frac{d}{dt}G(\vec{x}(t_0),t) = \Psi(G(\vec{x}(t_0),t)). \tag{1.18}$$

As we are interested in those systems that are sampled uniformly in time using some "sampling time" T, we can define the time-T map $G_T : \mathscr{M} \longrightarrow \mathscr{M}$ by

$$\vec{x}(t_0+T) = G_T(\vec{x}(t_0)), \tag{1.19}$$

which for any $k \in \mathbb{N}$ satisfies

$$\vec{x}(t_0+kT) = G_T^k(\vec{x}(t_0)) = G_T \circ G_T \circ \cdots \circ G_T(\vec{x}(t_0)). \tag{1.20}$$

Takens' Embedding. Firstly Takens defines the *delay coordinate map* with E delays $F_{(\Psi,G_{-T_s})} : \mathscr{M} \longrightarrow \mathbb{R}^E$ by creating a vector with E previous entries of the times series $y(t)$ measured at time interval T_s:

$$\begin{aligned} F(\vec{x}(t)) &= F_{\Psi,G_{-T_s}}(\vec{x}(t)) \\ &= [y(t), y(t-T_s), \ldots, y(t-(E-1)T_s)]^T \\ &= [\Phi(\vec{x}(t)), \Phi \circ G_{-T_s}(\vec{x}(t)), \cdots, \Phi \circ G_{-T_s}^{E-1}(\vec{x}(t))]^T. \end{aligned} \tag{1.21}$$

F is a map that goes from the ambient space $\mathscr{M} \subset \mathbb{R}^N$, where the systems states are defined, to a *reconstruction space* (what Suigihara and May call *state space*) $\mathbb{R}^E$ formed by the times series measurements. When a dynamical system is confined to a manifold $\mathscr{M}$, Takens showed that, given certain conditions on the time map G_{-T_s}, the delay coordinate map F is an embedding of $\mathscr{M}$ into the reconstruction space for almost every choice of the measurement function Φ.

Theorem 1 *Let $\mathscr{M}$ be a compact manifold of dimension K and suppose we have a dynamical system defined by equation* 1.16 *that is confined on the manifold. Let $E > 2K$ and suppose that the periodic points of G_{-T_s} with periods less than or equal to $2K$ are finite in number, and G_{-T_s}, has distinct eigenvalues on any such periodic points.*

Then the observation functions Φ, for which the delay coordinate map F 1.21 is an embedding,

form an open and dense subset of $\mathscr{C}^2(\mathscr{M},\mathbb{R})$.

In the theorem $\mathscr{C}^k(\Omega_1,\Omega_2)$ refers to the space of all functions $\mathscr{F}:\Omega_1\longrightarrow\Omega_2$ whose k-th derivate is continuous. F, however, is a injective immersion operator. This means that, firstly, different points of the system states are not mapped to the same point in the reconstruction space. Secondly being an immersion means that the differential operator at any point x in the state space, D_xF, is itself an injective map. Note that more specific conditions on periodic points are necessary. suppose $\mathscr{M}$ is a limit cycle. The conditions of the theorem dictate that the number of points with periodic orbit on the manifold is finite. Suppose now that for an unlucky choice of T_s we have $G_{-T_s}(x)=x$ for all $x\in\mathscr{M}$, this would mean that that all periodic points are the manifold itself. Indeed, no matter what observation function or how many delays we pick, the delay coordinate map would be $F(x)=[Phi(x),Phi(x),\cdots,Phi(x)]^T$ for any $x\in\mathscr{M}$. This implies that the limit cycle will be mapped into a line in the reconstruction space, and this would violate the injective property [33][34].

Summing up, this theorem says that if the conditions, concerning the periodic points on the dynamical systems are fulfilled, then for any function $\Phi'\in\mathscr{C}^2(\mathscr{M},\mathbb{R})$, there will be a function Φ in an arbitrarily small neighborhood around Φ' such that the delay coordinate map $F_{(\Phi,G_{-T_s})}$ is an embedding. This means that for a large class of observation functions Φ, F preserves the topology of $\mathscr{M}$, therefore information about $\mathscr{M}$ can be retained in the time series measurements. Indeed, by preserving the topology of the manifold in the reconstruction space, many of its properties are conserved, including its dimensionality and its Lyapunov exponents.
In the third chapter we will see how Suigihara and May took inspiration exactly from the way on which Takens have built the reconstruction space using the delay coordinate map. This procedure, together with a simple a simple nearest neighbor algorithm permitted us to implement a forecasting method.

2

INTRODUCTION TO ARTIFICIAL INTELLIGENCE: NEAREST NEIGHBORS METHOD AND RECURRENT NEURAL NETWORKS IN MACHINE LEARNING

On the internet there are several definitions for artificial intelligence (AI). For instance Amazon defines AI as *the field of computer science dedicated to solving cognitive problems commonly associated with human intelligence, such as learning, problem solving, and pattern recognition.* Computers have always had more processing power than the human brain: 10^{10} transistors with a switching time of 10^{-9} seconds against 9×10^{10} neurons in the whole nervous system, with a

switching time of the order of 10^{-3} seconds; but the brain outperform a computer in many ways, for example, it has the ability to learn and make a decision through knowledge and past experiences. In fact, it is this ability the one that the AI craves to steal to adapt it to to the wide world of programming.

Machine learning (ML), data science and statistics are those fields that explain how to learn from (and make prediction about) data. In a period where data analysis has became an aspect on which modern science is concerned about, Ml and data science are playing increasingly important roles in modern technology. Therefore, the understanding of these concepts is an increasingly important skill in and out the world of Physics.

2.1 Machine Learning: algorithms that generate algorithms

Usually, the advanced digital world we see nowadays has always been carried on by algorithms. Through algorithms of every kind, programmers, can instruct computers in new tasks, solve problems and organize data and services. Anyway, a programmer is needed: an human being who is able to write the instructions required by the task. Thanks to the ML the situation changed. In machine learning, learning algorithms - and not computer programmers - create the rules. The idea behind is that, now, the programmer needs only to write the instructions allowing the computer to learn from data, without the necessity to write a code step by step. This approach gives us the possibility to solve those problems, whose resolutions could not be manually programmed. Tasks such as photo recognition or the translation of pictures in speech [3].

The basic process of machine learning is to give training data to a learning algorithm. The generating algorithm, subsequently, puts on a set of rules based on the inference from the data given for the training. Moreover, by using different training data, the learning algorithm could be used to train different models.

Inferring new instructions from data is the core strength of machine learning: higher the quantity of available data is, more the machine will learn from them. In fact, recent improvements in ML have been achieved not through innovations in learning algorithms, but thanks to the higher quantity of data on the internet.

There are various techniques to make a machine learn from data, but the methods for learning are usually divided in just three different classes:

1. *Supervised Learning*: it consists in making the machine learn from labelled data (each one of them is tagged with the correct label). The learning algorithm just needs the desired output and the training labelled set. Supervised learning usually deals with classification and regression problem.

2. *Unsupervised Learning*: has to deal with unlabelled input data and it needs to identify patterns and structures in them. Examples of this type of learning are: clustering, associations and generative models. This model is able not only to separate data into classes, but also to underline the pattern to learn more about it.

3. *Reinforcement Learning*: the machine learns through the interaction with the surrounding environment. The machine will receive either rewards when performing correctly, or penalties when performing incorrectly. The agent learns without the intervention of a human being by maximizing its reward and minimizing its penalty.

In this chapter we will mainly focused on the understanding of a simple algorithm of supervised learning called *K-nearest neighbors* (KNN). We will give a brief overview of this type of learning without mentioning the other two classes. In particular, our aim is to make the reader aware of the universality of the method that will be implemented in the chapter 3 of this thesis, using it to predict the energy's unbalances of the Italian power grid. The KNN is commonly used for regression and classification problems, its implementation is so simple that industries use it to solve several issues.

2.1.1 Supervised Learning

Before getting to the K-nearest neighbors algorithm, we want to concentrate on supervised learning. Several problems can be resolved using supervised ML and the solution requires usually the same structure.

The information about the problem are collected in a dataset that we call $D = (X, Y)$, where X denotes a set of independent variables and Y denotes the set of dependent ones.

More than that, there is the model of the problem that we can associate to a function that takes as input both the dataset and a parameter p: $f(X;p)$. Anyway $f : X \longrightarrow Y$ is considered a function only of the parameter p and it will be used to predict a vector of the set Y through a vector of the set X.

The last component needed is a device allowing both the machine to understand whether it is learning correctly and us to evaluate the performances of the model. This element is called the cost function $c[Y, f(X,p)]$ and it depends on both the set of observed values and the one of predicted values. Indeed, the model is shaped when the p, minimizing the cost function, is founded.

Usually ML researchers and data scientists follow a standard scheme to obtain the most effective model which solves the problem. First, the dataset D is divided in two mutually exclusive groups: D_{train} and D_{test}, respectively the training set and test (or validation) set. Usually, a larger number of data is partitioned in the training set (e.g. 80%) and the remaining ones go into the other set. To solve the problem, the machine needs to find $P = \arg\min_{p}\{C[Y_{train}; f(X_{train}; p)]\}$. The programmer can get the goodness of the performance by evaluating the cost function using the test set $C[Y_{test}; f(X_{test}; p)]$. The value of the cost function for the best fit model on the training set is called the *in-sample* (or *training*) error $E_{in} = C[Y_{train}; f(X_{train}; p)]$, while the value of the cost function on the test set is called the *out-of-sample* (or *validation/generalization*) error $E_{out} = C[Y_{test}; f(X_{test}; p)]$. Since the validation set is usually used only after the learning process, very often $E_{out} > E_{in}$. The distribution of data into two mutually exclusive sets, provides an unbiased estimate to the predictive performance of the model. This, in ML and statistics vocabulary, is called *cross-validation* procedure .

Usually in Statistic we start by using a supposed true mathematical model, and our final goal is to determine the value of some of its unknown parameter of the model. Problems in ML, on the opposite, involve typically inference from data of complex systems, whose mathematical model that describes the system is yet unknown. This is why ML researchers usually have to choose between more than one candidate model by comparing them. The model chosen as the best, is usually, the one minimizing E_{out} the most. We must mention that there is a relationship between E_{in} and E_{out}, even though this concerns more the *statistical learning theory*, about which we will give a brief introduction to understand what else is hidden behind the choice of a model in supervised ML [14].

2.1.2 Statistical Learning Theory: the problem of limited data in machine learning

The main results of statistical learning theory can be summarized in two simple points.

1. Let's assume we have a certain set of data following a complicated distribution that we can not perfectly learn. How do E_{in} and E_{out} will change as the number of points in the set increases?

   On one hand, after an initial drop, E_{in} will increase since our model is not powerful enough to learn the real function we are looking for. On the other hand, E_{out} will decrease as the number of points gets higher. As the number of data increases, E_{in} and E_{out} will approach asymptotically the same value and the training data set will become more representative of the real distribution. The limit value reached by both errors is a function of the complexity of the model and is called the *bias* of the model. Generally, the more complex the model is, lower the bias' value will be. However, since, generally, we tend not to have an infinite amount of data available, we can not often know the value of the bias and we just concentrate on minimizing the value of E_{out}.

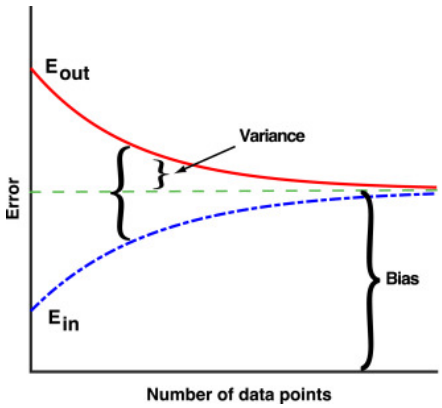


Figure 2.1: In this figure the trend of E_{in} and E_{out} as function of the number of data.

2. Moreover statistical learning theory warns us that it might not be enough minimizing the training error, because the out-of-sample one can still be high. E_{out}, does not depend only on the value of the bias, but it depends also on the number of data used for training, that number, being limited, will generate an increasingly wider *variance* as the model complexity rises up. Summing up, a well-performing model should be characterized by a small E_{out}. Taking into

account both the bias and the variance, we can minimize it by using models with intermediate complexity. As shown in figure 2.2, at the beginning E_{out} decreases since the value of the bias decreases as the model complexity raises up. however, at a certain point, the model becomes too complex for the amount of data in the validation set, so the generalization error becomes larger due to the contribution given by the variance component.

Actually, a programmer, to minimize the out-of-sample error, must evaluate whether it is better to create a more-biased model or a less biased one but with a larger variance. This problem is commonly called the *bias-variance* tradeoff.

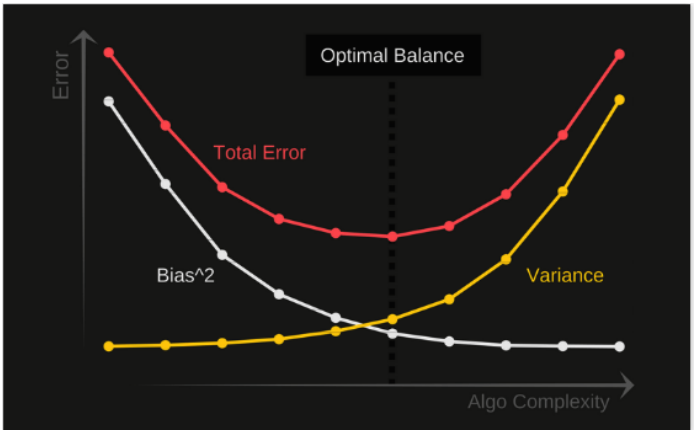


Figure 2.2: In this figure the trend of E_{out} as function of the model complexity. The programmer must find the region that minimize the total error (in red).

2.2 K-Nearest Neighbors Algorithm: the basics of Supervised Machine Learning

Using the k-nearest neighbors algorithm (KNN) is one of the easiest ways to compute supervised machine learning and it can be used to implement both classifications and regression problems.
A classification problem has a discrete value as its output. For example *Likes carbonara with guanciale Likes carbonara with pancetta* are discrete choices: there are no middle levels. In a typical set of classification data we have a predictor (or a set of predictor) and a label. Someone might think that Italian people's opinion may depend on the culture of the region where they live. If this was true, a person's opinion could be predicted by the area where he/she grew up. This is

that kind of problem which can be efficiently solved by using a KNN. For instance we could make a poll among Italians, sort the population by the native region and label each person either with 1 when he/she *Likes carbonara with guanciale* or with 0 when he/she *Likes carbonara with pancetta*. Than, we could try to predict somebody's choice from the one of his/her neighbors.
However, a regression problem is composed by independent variables and a dependent ones. An example of a dataset could be a set of people, whose height and weight are the known parameters. For instance we could choose height as the independent variable and weight as the dependent one. What we are looking for is the proportional constant between the two quantities, but this might be a function of the independent variable. In this case, if someone wants to predict a person's weight given the height, he won't choose the proportional constant calculated using the entire set, instead he will just look at the values of those people with similar height.
The KNN algorithm takes advantage from the labeling of its data, therefore it is considered a type of supervised machine learning. In particular, it assumes that similar things are close to each other: two elements are similar if the distance between them is short. There is more than one way to calculate the distances, and one of them is preferable depending on the type of problem we are solving. In this thesis, as we will see in chapter 3, we will used the Euclidean distance.

The KNN Algorithm.

1. Prepare the data on which you want to apply the algorithm.
2. Choose a number of neighbors K.
3. Calculate the distance between the query example and every other example of the data.
4. Collect all the data with their indices in a vector.
5. Sort the vector of indices by the distances calculated at the previous point.
6. Pick the first k entries from the sorted vector.
7. Get the indices of the selected entries.

8. If you are trying to solve a regression problem, return the average of the K data. If you are trying to solve a classification problem, return the mode (the most frequent of the k elements chosen).

To select the most appropriate K for the data available, the KNN algorithm must be run several times. To be used for the following prediction, it should be chosen the K which reduces the most the number of errors. Obviously, when the number of K decreases, the predictions become less and less stable because there is an higher chance of finding outliers which contaminate the goodness of the data. On the opposite, we should see an increment in stability when the value of K is enlarged: the more K increases, the more the predictions will be accurate and the less influent the weight of outliers will be on the average (if we reasonably suppose a dataset with a small number of them). However, by averaging too many elements, at some point, will bing to take into consideration elements which are "far" from the value involved, therefore the number of errors will increase. The right choice of K depends on the dataset and it is the programmer who should notice which value is the adequate one. We can summarize the advantages of this algorithm in three main points:

1. The algorithm is simple and easy to implement.

2. Is a non-parametric method and indeed there is no need to create a model with assumptions and variables.

3. The algorithm is versatile. It can be used for both classification, regression and search.

The only disadvantage is that it gets significantly slower as the numbers of examples increase. This makes it an impractical choice when facing problems whose data set is huge and when the prediction must be rapid. Furthermore, it already exists much better algorithms that are not only more effective, but also faster than the KNN. Anyway, the KNN remains the simplest algorithm available, therefore, when we have the power of speedily handle the data we are using, the KNN will be useful to solve those problems whose resolution depends on the identification of similar elements [20].

We will see that the algorithm implemented to predict the Italian energy's unbalances of the Italian power grid, can be splitted in two parts. In the first part we build the reconstruction space which

Takens built when invented his theorem - what Suigihara and May call the state space - in the second one we just implement the KNN algorithm.

2.3 Deep Learning

Over the last decade most of the machine learning algorithms designed involved the use of neural networks (NNs). Although they have a long history, they emerged only in the mid 2000s after being labelled as *DeepLearning* and *Deep Neural Networks* (DNNs).
It might be helpful to divide neural networks in four categories:

1. neural networks for supervised learning, which are designed to solve general problems.
2. Neural networks built for image processing, the most widely-used example is convolutional neural networks(CNNs).
3. neural network for unsupervised learning such as deep Boltzmann machines.
4. neural network for sequential data such as recurrent neural networks (RNNs).

In Physics both DNNs and CNNs have found numerous applications. Among the fields of statistical mechanic and quantum statistical physics have been utilized to detect phase transitions and, by contrast, methods from statistical physics have been applied to the field of deep learning to study the efficiency of learning rules and to explore the representation capabilities of DNNs. Whereas deep CNNs were used for reconstruct the cosmic microwave background, DNNs were used to improve the efficiency of Monte-Carlo algorithms. Even in the field of control of quantum system DNNs were discovered useful. They can be applied in such a huge number of areas that it is impossible to mention all of them here. In this brief section, we will outline the basic functioning of DNNs, focusing specifically on what we use in the following chapters [14].

Biology behind the idea. There is a natural concept behind the idea of Neural Network, in fact, this concept is inspired by human nervous system, whose smallest unit element is called neuron. A neuron is just a simple cell able to communicate with other cells of the same type. This brings an enormous processing power, which we want to reproduce by implementing DNNs. In other words

we can say that neurons behave like switches. Each neurons, first, takes an input signal, then, from that, it elaborates an output that, will become itself an input for another neuron.
The neuron consists of three components: soma (the body of the cell), dendrites and axon.

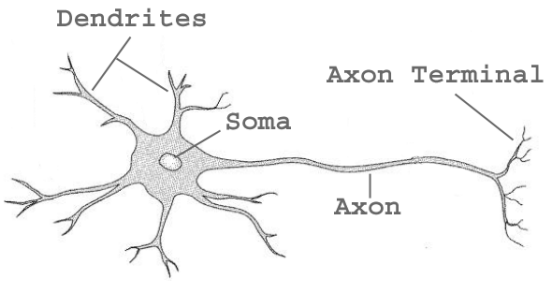


Figure 2.3: In this image is shown the neuron in all its three most important parts.

The soma carries out the basic life processes of the cell. The axon is the long element of the neuron and it acts as a cable to send output signals. Dendrites are, by contrast, responsible of receiving input signals sent to the neuron. The mechanism ruling the shipping of the signal between axon and dendrites comes closer via axon terminals: the synapses. The signal travels from neuron to neuron in the form of neurotransmitter molecules whose amount and type classify the strength of the signal.

Artificial Neurons. Before explaining what a neural network is, we should focus on something more simple: the elements building the DNN, called the artificial neurons. As a first example of artificial neuron, there is a Perceptron. Inspired by the works of Warren McCulloch and Walter Pitts, the scientist Frank Rosenblatt in the 1950s and 1960s developed the perceptron as a binary classifier. In particular it takes several binary inputs $x_1, x_2, \ldots$ to produce a single binary output y. Rosenblatt introduced weights $w_1, w_2, \ldots$ as real numbers expressing the importance of the respective inputs to the output. The output, 0 or 1, is determined by whether the value $\sum_i w_i x_i$ is lower or higher than the threshold value, which is a parameter of the output.

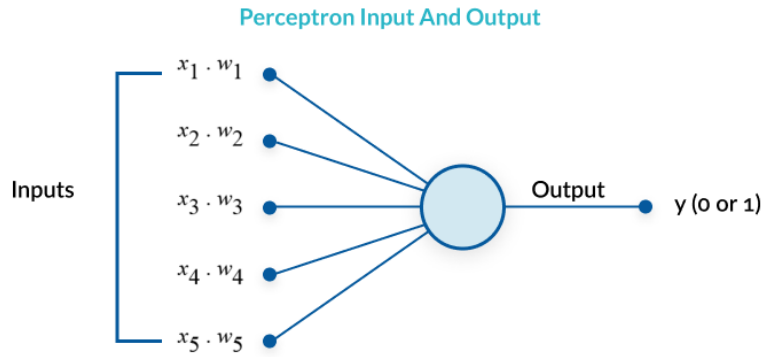


Figure 2.4: The perceptron of Rosenblatt.

A perceptron can be thought as a device able to make a decision by weighing up data. Is quite surprising that something so simple can model the decision-making process of an human brain. However a complex network of perceptrons can be set to make rather correct decisions. Anyway, there will be a problem if our network is made only by perceptrons: small changes in the values of weights of thresholds could completely overturn the output. This limit to the gradual modification of the weights makes it hard to implement the network to perform the correct behavior.
However, it is still possible to overcome this problem by introducing a new type of artificial neuron called sigmoid neuron. Sigmoid neurons, although similar to perceptrons, are designed in a way that allows the modification of the weights and bias values without strongly altering the output. thanks to this feature, a network of sigmoid neurons can learn. As for the perceptron, the sigmoid neuron has inputs $x_1, x_2, \ldots$ although the output differs from the perceptron's one. While perceptrons can return only two output values 0 or 1, sigmoids have a more complicated output function $\sigma(z) = \frac{1}{1+exp(-z)}$, where σ is called the *sigmoid function*, $z = w\dot{x} + w_0$ and w_0 (sometimes called b) is the bias. Anyway the exact form of σ is not that relevant, what really matters is its trend. σ must have a smooth shape, since its smoothness would decrease its sensitivity to weights and bias changes. Indeed, there are various possible activations functions for neurons and it is the duty of the programmer evaluating the most suitable function among the available ones.

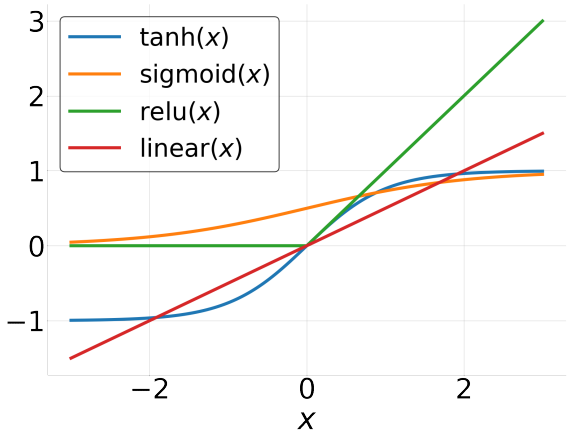


Figure 2.5: More activation functions are available to the programmer. In this photo are shown some possible choices of linear and nonlinear maps.

The architecture of DNNs. When it comes to build a network, the fashion of neurons layers can influences the output. Usually, the network is built following a hierarchical structor. The leftmost layer in the network is called the input layer, and the neurons positioned there are called input neurons. The rightmost one is called output layer and it contains the single output neurons. The middle layers are intuitively called hidden layers.

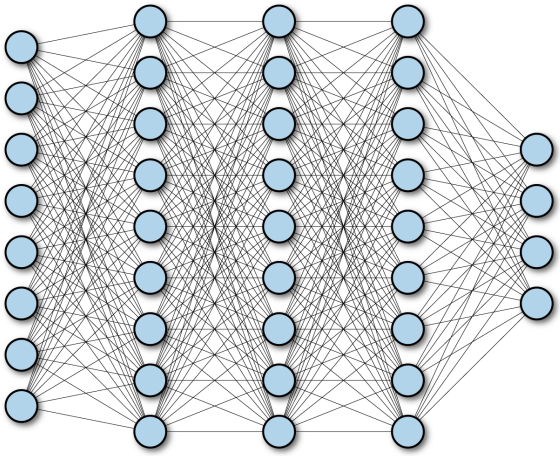


Figure 2.6: A typical deep neural network. In this picture 8 input neurons, 3 hidden layers and 4 output neurons.

Whereas it does not exist a right way to design the hidden layers, the structure of the input and output ones should adapt to the dataset $D = (X,Y)$. For example if we want the machine to learn how to recognize handwritten images, we need to design an output layer of 10 neurons. We label

each neuron with a digit and, to know what the machine has chosen after visualizing the picture, we need to look at the activation function of the output neurons. The answer will be the label of the neuron with the highest value of the activation function.

Universal approximation theorems. Generally speaking, we can think of the neural network as a device able to modify inputs X into output Y through complicated nonlinear transformations which depends on the weights and biases of all the neurons in the input, in the hidden and in the output layers. Specifically, the use of hidden layers enlarge the representational power of a neural network. To explain this concept we should mention the universal approximation theorem: a *a neural network with a single hidden layer can approximate any continuous, multi-input/multi-output function with arbitrary accuracy* [11] [23]. On the other hand, whereas feed-forward networks with a single hidden layer are universal approximators, the width of such networks has to be exponentially large. To explaining this argument in 2017 Lu et al. proved as valid the universal approximation theorem for width-bounded deep neural networks. In particular, they showed that width-n+4 networks with ReLU activation functions can approximate any Lebesgue integrable function on n-dimensional input space if the depth of the network is can grow [27]. These results were essential to underline that, whichever function we are interested in fitting, there will always be a neural network able to carry on the work. The idea behind this the theorem is that hidden layers can generate a piecewise function with arbitrary offsets and heights. Those can be added to approximate the function as shown in figure 2.7.

$$
\begin{aligned}
n_1(x) &= Relu(-5x - 7.7) \\
n_2(x) &= Relu(-1.2x - 1.3) \\
n_3(x) &= Relu(1.2x + 1) \\
n_4(x) &= Relu(1.2x - .2) \\
n_5(x) &= Relu(2x - 1.1) \\
n_6(x) &= Relu(5x - 5)
\end{aligned}
$$

$$
\begin{aligned}
Z(x) = &-n_1(x) - n_2(x) - n_3(x) \\
&+ n_4(x) + n_5(x) + n_6(x)
\end{aligned}
$$

Figure 2.7: A neural network approximating a function using Relu functions.

Moreover, demonstrating the theorem makes it clear that, the more complicated a function is, higher is the number of neurons and free parameters necessary to compute it. Indeed, many studies suggest that it is algorithmically easier to train deep neural networks rather than shallow nets. Anyway, there are many researcher going on about this, therefore any consideration about it can still be contradicted by future discoveries. Generally, the best architecture of a problem depends on the task, on the amount and on the type of data available.

Training deep networks. To fit a model with a DNN we must train the neural net. The training is made by using the cross-validation technique minimizing the loss function for both the training set and test set. The minimization of the loss function together with the search of the minimum, are then executed through some gradient descent technique, which randomly explores the space of parameters to follow the direction that minimize the loss. For example, the loss functions *mean squared error*

$$E(\vec{W}) = \frac{1}{n}\sum_{i=1}^{n} |\vec{Y}_i - f(\vec{Y}_i;\vec{w})|^2$$

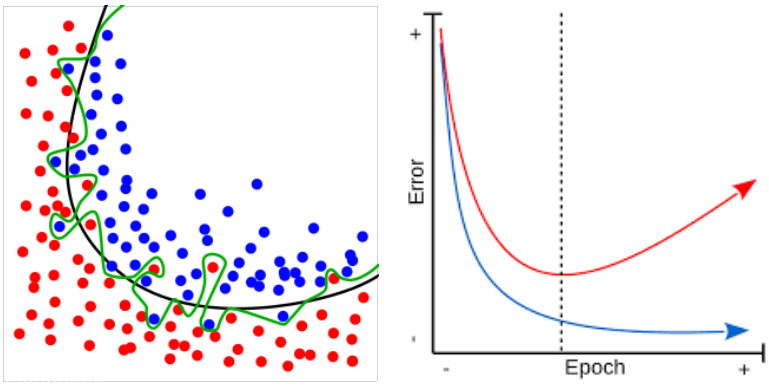
and *mean absolute error*

$$E(\vec{W}) = \frac{1}{n}\sum_{i=1}^{n} |\vec{Y}_i - f(\vec{Y}_i;\vec{w})|$$

are commonly used for continuous data. While, for categorical data, when the output layer is taken to be a classifier for binary data, the *cross entropy function*

$$E(\vec{W}) = -\sum_{i=1}^{n} Y_i \log f(Y_i;\vec{w}) + (1-Y_i)\log[1-f(Y_i;\vec{w})],$$

between the true labels $Y_i \in \{0,1\}$ and the prediction $f(Y_i;\vec{w})$, is the most commonly used.
After having defined the architecture and the cost function, we then select an optimizer to explore the space of parameters. Our choice must depends on the architecture, data and computational resources.



(a) (b)

Figure 2.8: In the figure (a) an example of two models. The black line fits the data perfectly, because was trained for the right number of epochs. The green line, indeed, is an example of a function that 'overfit' the data: the neural network was trained for so much time that the noise of that dataset had badly influenced the shape of the model.
In the figure (b) the graph of the validation (red) and training (blue) error as a function of the number of the epochs used to train the dataset. If the network is not 'too' trained, the error decreases at the increasing of the number of epochs, but when the network becomes to take into account the noise of the training set, becomes more and more unable to fit the validation set and this causes the growth of E_{out}, while E_{in} on the other hand continues to decrease.

The calculation of the gradient requires a specialized algorithm, called *Backpropagation*, able to calculate the derivate of the cost function with respect to all the neural network's parameters (all neurons' weights and biases in the hidden and visible layers).

What is harder in the training phase is understanding when it is time to stop. This can't be understood by looking only at the graph of e_{in}, because the mean-square error decreases as the number of epochs used for training increases. In particular, it could happen to "overfit" the data, if the training session is not stopped at the right time. To overcome this problem, we need to look at a graph that permit us to compare e_{in} and e_{out}. The right number of epochs that must be used during the training is the one which minimizes the e_{out} (see figure 2.8 for a better explanation).

2.4 Recurrent Neural Networks

Recurrent neural networks (RNNs) are a class of artificial neural network that are often used for series (i.e. sequential data) due to their ability to remember what they have learnt from prior inputs while generating outputs. A recurrent neural network is built generalizing the simpler feedforward neural network, adding to it an *internal sate* which acts as an internal memory. The term "recurrent"

refers to the fact that it performs the same function for every input of data, while the output of the current input depends on the computation of the previous one. After producing the output, the RNN copies it and then sends it back into itself. Subsequently, it takes into account both the current input and the output, learned from the previous input, to generate another output. Unlike feedforward neural networks, RNNs can use the internal state (memory) to process sequences of input data. This makes them fitted for tasks such as connected handwriting recognition, speech recognition or time series predictions. In other neural networks, all the inputs are independent from each other, while in RNNs, all inputs are related to each other. The research for the best model for the series can be implemented by minimizing the loss using a cross-validation procedure as explained for deep neural networks, but the architecture of a RNN is actually rather different and we will explain it in the next paragraph [21].

The architecture of a RNN. Architecturally RNNs are different from feedforward neural networks since they aren't formed by common artificial neurons, but by units (or cells) and, according to Tensorflow (just a Python's library) documentation, *An RNN cell, in the most abstract setting, is anything that has a state and performs some operation that takes a matrix of inputs*. RNN cells distinguish themselves from the regular neurons since that they have a state, thus they can remember past information. The three most common types of cell are:

- Vanilla (or Basic) RNN-cell.
- Long short-term memory (LSTM) [22].
- Gated recurrent units (GRU) [8].

In this thesis we have decided to use a simple RNN, built only by a Basic RNN-cell. On a mathematical level, a sequence of inputs are passed on by one through an RNN cell, one at a time. However, to understand clearly the architecture diagrams like figure 2.9, is important to note that the RNNs' inputs needs to have 3 dimensions. Typically it would be:

- The batch size.
- The number of steps.

- The number of features.

What is important to get, in the architecture of the RNN, is that the number of time steps/segments fed into the RNN (shown in figure 2.9) is not decided in advance, but it is depicted by the "the number of steps" dimension of the input dataset. The RNN unit in TensorFlow is called the “RNN cell”. There are many questions on the internet that inquire if “RNN cell” refers to one single cell or to the whole layer. Probably there is no correct answer, however it might be more accurate to refer to “RNN cell” as the whole layer. The reason for this is that the connections in RNNs are recurrent, thus they follow a “feeding to itself” approach. Basically, the RNN layer is comprised of a single rolled RNN cell that unrolls according to the “number of steps” value provided. This procedure is shown in figure 2.9.

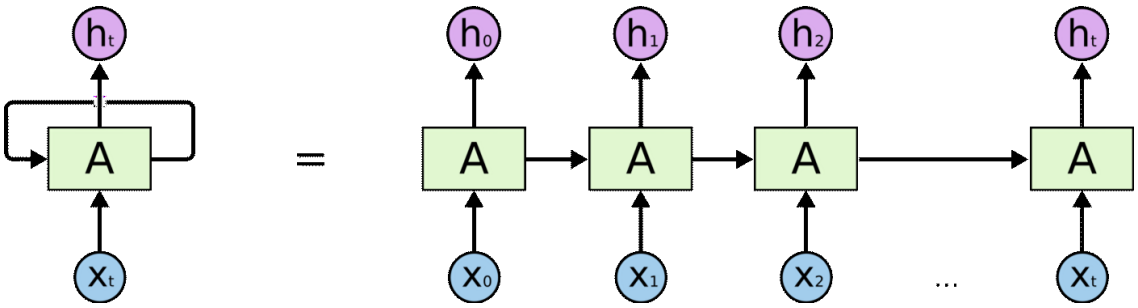


Figure 2.9: A vanilla recurrent neural network. In the image the hidden state retains information from one time step to another flowing through the unrolled RNN units. Referring what is written on the text, in this picture the network has been feeded by a batch with t number of steps.

3

SIMPLE MACHINE LEARNING ALGORITHMS FOR TIME-SERIES PREDICTION INSPIRED BY TAKENS EMBEDDING THEOREM

In this chapter we illustrate the methods used for the predictions. We start introducing the algorithms by reviewing an article written by Suigihara and May (S-M) [32] and the results that they obtained. After we explain step by step the methods used and the difficulties encountered in the implementation of the algorithm of S-M - in this thesis called the *simplex projection* (SP) - and we will motivate why a simpler K-nearest neighbors (KNN) might be a better choice. Moreover we

will illustrate a basic architecture of a recurrent neural network (RNN) that has been tested on the data. It allows to use of a big quantity of data improving the goodness of the prediction, without slowing down significantly the process.

Afterward we will show the results of the predictions' analysis made using the three methods on synthetic data (i.e. the three chaotic maps introduced in 1.3). To measure the goodness of the predictions, we will use the correlation coefficient (ρ) between predicted and actual values of the time series. Since the SP and KNN's methods are based on the past pattern of the time series we want to forecast, we will follow the procedure used in machine learning, dividing our set in two parts: the first (training set) needs to implement the algorithms, while the second (test set) needs to verify if the methods works. For this thesis, we will use a data set of 1000 points divided into two equal part, firstly because both methods become slower when are used big data set and secondly because this procedure is the same one adopted by S-M. Moreover, we tried to increment the size of the training sets, but the goodness of the predictions didn't improve. However, when it comes to forecast on a RNN, it is possible to increment the dataset to bring significant improvements. For this thesis, when dealing with a RNN, we will use a dataset of 13000 points, whose first 10000 are used for the training set and the last 3000 are used for the test set.

In the section 3.6, we will show the analysis on the predictions of a noisy limit cycle to see if the methods SP or KNN are suitable to discern whether a time series is purely stochastic or it displays dynamic motion. Concerning this, we will show that, following our analysis, we cannot draw up the same conclusions S-M drew using the simplex projection. However, a way to solve this problem might be the one shown in figure 3.12 in which we use the KNN.

In the last section is illustrated the trend of the KNN's accuracy as the prediction-time interval (T_p) changes. We showed that the trend of $\rho(T_P)$ can be explained by the trend of the autocorrelation of the time series over the time.

3.1 Review of the article of Suigihara and May of the 19 April 1990

As it has already been said that, the trajectories of chaotic systems are very hard to be predicted due to their strong dependence on the initial conditions. However, it is possible to make short term predictions. An example of approach is presented in the article that Suigihara and May (S-M) published on the 19^{TH} of April 1990 [32]. They invented a method that was supposed to discern whether a time series was showing dynamical components or it was purely stochastic in nature. The method consists in forecasting the time series using an algorithm and observing how qualitatively varies the correlation coefficient between the actual and predicted time series as the parameters of the algorithm change. In this thesis we aim to reproduce that algorithm to predict the time series of energy's unbalances provided by Terna. The algorithm could be explained in a short summary, but it seems more instructive to the author of this work to report the article step by step.

Introduction According to S-M there are two sources of errors in forecasting a trajectory of a natural dynamical system (e.g. animal population or densities of plant): the errors of the measurements and the complexity of the dynamical (when the dynamic leads to chaos). The method is based on the past pattern of the time series and it will be able to make distinction between dynamical chaos and measurement errors.
S-M underline the utility of their algorithm when the number of data points available is low, while other methods (they name the method of Grassberger and Procaccia [17] [16]) require a larger number of data points, as it happens with neural network.

Forecasting for a chaotic time series The first case on which the method is tested is the series generated from the first-difference transformation $(x_{t+1} - x_t)$ on the tent map, which is an equivalent method to generating the logistic map with $r = 4$. They explain that they first-difference the data either to give higher density in phase space to the chaotic attractors and to clarify nonlinearities by reducing the effects of any short-term linear autocorrelations. However, they noticed that they obtained the same results also when working with raw times series. In fact, after having verified that, we have chosen in this thesis, to work with raw times series. After this brief introduction S-M

shown the results of their forecasts on the tent map, displaying plots, similar to those we will show in section 3.5 and finally explain the algorithm that they used. *...Specifically, we first choose an 'embedding dimension' E, and than use lagged coordinates to represent each lagged sequence of data points* $(x_t, x_{t-\tau}, x_{t-2\tau}, ..., x_{t-(E-1)\tau})$ *as a point in this E-dimensional space; for this example we choose* $\tau = 1$*, but the results do not seem to be very sensitive to the value of* τ*, provided that it is not large...* Indeed, now the problem has changed: the element to be predicted is now a sequence of E points, that S-M call *predictee*. *...we now locate all nearby E-dimensional points in the state space, and choose a minimal neighborhood defined to be such that the predictee is contained within the smallest simplex formed from its* $E+1$ *closest neighbors; a simplex containing* $E+1$ *vertices (neighbors) is the smallest simplex that contain an E-dimensional point as an interior point (for points on the boundary, we use a lower-dimensional simplex of nearest neighbors). The prediction is now obtained by projecting the domain of the simplex into its range, that is by keeping track of where the points in the simplex end up after P time steps (that S-M call* T_P*). To obtain the predicted value, we compute where the original predictee has moved within the range of this simplex, giving exponential weight to its original distances from the relevant neighbors...* The model just outlined can be applied to any dynamic system and, indeed, even a chaotic one.

Forecasting with uncorrelated noise In this short paragraph S-M want to underline the differences between a chaotic system and a noisy limit cycle. Basically, a simple limit cycle soiled of white noise is characterized by a constant correlation coefficient (ρ), while a time series whose dynamic is ruled by a chaotic system presents a correlation coefficient which depends strongly on the prediction-time interval. A similar plot, published by them, will be shown at the end of the chapter and we will show that, for our reproduction of the predictions' analysis, it is impossible to drive to the same conclusions.

The embedding dimension In this section was discussed the dependence on the parameter E. For what concerns chaotic maps, S-M explain that the predictions made with high embedding dimensions are less accurate than the predictions made with small values of E. Indeed, for chaotic maps, we should find a decreasing graph $\rho(E)$. S-M thought that this feature ... *is caused by a contamination of nearby points in the higher-dimensional embedding dimension with points whose*

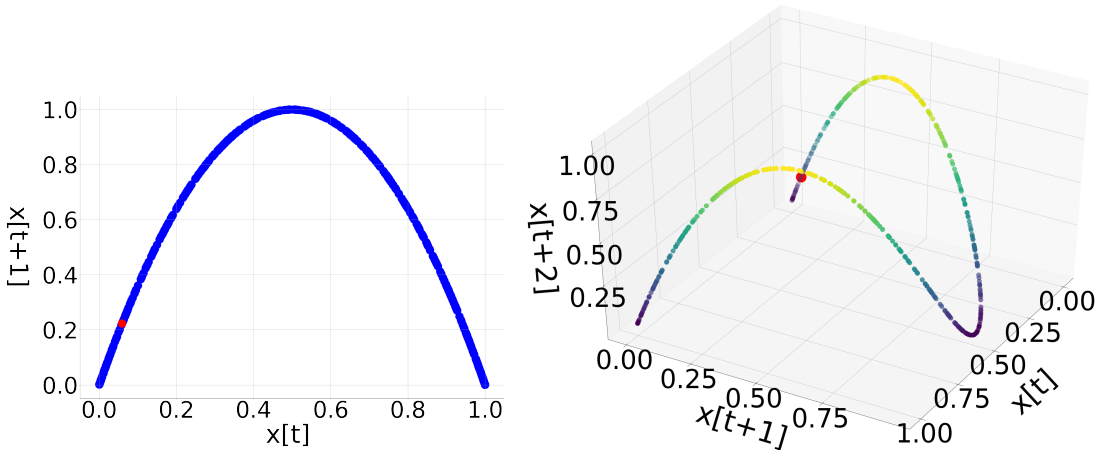
earlier coordinates are close, but whose recent (and more relevant) coordinates are distant... This happens in the analysis made by the correspondent author only on the Logistic map, while in Mackey-glass and Lorenz systems the trend of $\rho(E)$ is not decreasing.

Problems and other approaches This section begins with a list of 'famous' nonlinear maps on which S-M tested the method. They explain that basically all the chaotic systems show the same behavior in the prediction's accuracy, even in more complicated cases. A problem arises when it comes to compare the various $\rho - T_P$ relationships generated by colored noise spectra, which present only short-term, and not long-term autocorrelations. Such colored noise can clearly lead to decreasing graphs $\rho(T_P)$. They conjecture, however, that an autocorrelated pattern like that one, would in general give a flatter $\rho - E$ graph than those generated by chaotic dynamic. Anyway, Suigara and May, concluded suggesting that an observed time series will be regarded as deterministically chaotic if, in addition to a decaying $\rho(T_P)$ signature, the correlation between observed and predicted data obtained by their method is significantly better than those generated by autoregressive linear models. The last part of this section consists in a discussion about other methods to estimate the dimension of an attractor. Since this part is not the focus of this thesis, the reader is invited to look at it directly from the S-M'S article.

Conclusion After having tested the method on natural time series - the cases of Measles and Chickenpox in NYC and the population of diatoms in seawater samples in San Diego - Suigihara and May drew their conclusion....*The forecasting technique outlined here is phenomenological in that it attempts to assess the qualitative character of a system's dynamics - and to make short-range prediction based on that understanding - without attempting to provide an understanding of the biological or physical mechanisms that ultimately govern the system. This often contrasts strongly with the laboratory and field-experiment approaches that are used to elucidate detailed mechanisms by, for example, many population biologists... Our approach works with artificially generated time series (for which we know the actual dynamics, and the underlying mechanisms, by definition), and it seems to give sensible answers with the observed time series for measles, chick-enpox and diatoms... We hope to see the approach applied to other examples of noisy time series in population biology, and in other disciplines in which times seres are typically sparse.*

3.2 The Simplex Projection method

In our reproduction of Suigihara and May's method, the *simplex projection* (SP), we noticed that the algorithm struggles when searching the simplex for the chaotic maps. This is caused by the disposal of the points in the state space, since dispose themselves to form a line that do not allow the predictee's neighbors to enclose it in a simplex. This is a feature of every *Poincaré's plot* [6], the name by which the "state space" is known in chaos theory. Specifically, it is a plot which is supposed to help the user to discern if a times series present deterministic components: if it does the points will display a certain autocorrelation, if it does not, the points, will look purely stochastic. Since the logistic map is a chaotic map, it forms an autocorrelated function in the state space: a curve (Figure 3.1). This basically means that, all points are themselves the border. This takes to a hard search for the vertexes of the simplex which enclose the "predictee". This specially happens in high dimensional state spaces, while in smaller ones the search is overall simpler. Instructive are the cases in which $E = 2, 3$, because the state space can be plotted and be observed in a figure, like the 3.1.



(a) (b)

Figure 3.1: In these figures are displayed two Poincaré plots of the logistic map (1.14 with $r = 4$). In figure (a) in two dimensions and in figure (b) in three dimensions. Both the lines in the pictures are built using 500 points of the map. The red point, on the other hand, is made with the 501-th point and represents the point on which we want to forecast: the predictee. It is rather intuitive that the algorithm struggles to find a simplex made of neighbors that enclose the red point.

The simplex projection

Let $x(t)$ be a set of samples of a time series measured at evenly-spaced times up to some maximum time t_{max}, $\{x(t)\}_{t=t_0}^{t=t_{max}}$ and let Φ_{T_P} be the time evolution operator, which acts on the series like $\Phi_{T_P}x(t) = x(t+T_P)$, the procedure consists of the following commands.

1. Choose an embedding dimension E and a value of lag τ. Then split the series in two parts: the first composed by the earliest N_{space} values, the second composed by the last $N = t_{max} - N_{space}$ ones.

2. Use the first part to build an E-dimensional space like $\{\{x(t = N_{space}), x(t = N_{space} - \tau), ..., x(t = N_{space} - (E-1)\tau)\}, \{x(t = N_{space} - \tau), x(t = N_{space} - 2\tau), ...\}, ...\}$ obtaining $N_{space} - E + 1$ points, which form what Suigihara and May call the *state space*.

3. Given a value $x(t) \in \{x(t = N_{space} + \tau), ..., x(t_{max})\}$, of which we want to make a prediction T_P steps in the future, take the previous $E-1$ values of the series in order to build an E-dimensional point $(x(t), x(t-\tau), ..., x(t-(E-1)\tau))$, which Suigihara and May call the *predictee*.

4. Find the $E+1$ points - $\{x(t_i)\}_{i=1}^{i=E+1}$ - in the state space, that form the smallest E-dimensional simplex, which enclose the predictee:

   (a) Take the $K = E+1$ nearest neighbors of the vector you built at the point 3.

   (b) Check if the predictee is contained in the simplex.

   (c) If it is contained, go to point 5.

   (d) If it is not contained, take the $K' = E+2$ nearest neighbors.

   (e) Check if the combinations $\binom{K'}{E+1}$ contain the predictee.

   - If the predictee is contained in only one of the combinations, go to point 5.
   - If the predictee is contained in more than a combination, sort the simplexes by the volume, take the combination with the smallest one and go to point 5.
   - If it is not contained in any combination, increases K' by one and go to point 4(e).

5. For each point obtained, apply the time evolution operator Φ_{T_P}:
   $\{\Phi_{T_P} x(t_i)\}_{i=1}^{i=E+1} = \{x(t_i + T_P)\}_{i=1}^{i=E+1}$.

6. Predict the E-dimensional point as the exponential average:

$$\langle x \rangle = \frac{\sum_{i=0}^{E+1} x(t_i + T_P) e^{-d_i}}{\sum_{i=0}^{E+1} e^{-d_i}} \tag{3.1}$$

   where d_i are the euclidean distances in $\mathbb{R}^E$.

7. Then the prediction of x(t) is the first coordinate of $\langle x \rangle$.

Observations

- To check if a point $x \in \mathbb{R}^N$ is in a n-simplex it has been used a method that makes use of barycentric coordinates [1]. Let $S = (v_1, ..., v_{n+1})$ be a n-dimensional simplex, where $v_i \in \mathbb{R}^N$, $i = 1, ..., n+1$ are the vertexes of the simplex.
  If the simplex is non-degenerated, the barycentric coordinates are linear independent and are defined by

$$\lambda = T^{-1}(x - v_1) \tag{3.2}$$

  where $T = (v_1 - v_{n+1}, ..., v_n - v_{n+1})^T$. The point x is contained in the simplex S if the barycentric coordinates $\lambda = (\lambda_1, ..., \lambda_n)$ and $\lambda_{n+1} = 1 - \sum_i \lambda_i$, $i = 1, ...n$ satisfy:

  1. $\lambda_i \geq 0 \; \forall i = 1, ...n$
  2. $\sum_{i=1}^{n} \lambda_i \leq 1$

- To calculate the volume of a simplex in n dimensions has been used the *Cayley-Menger* Determinant [9] [18].
  Let S be a n-dimensional simplex with vertexes $v_1, ..., v_{j+1}$ and $B = (b_{ik})$ the $(n+1)x(n+1)$ matrix given by

$$b_{ik} = |v_i - v_k|_2^2, \tag{3.3}$$

then the volume V is given by

$$V(S) = \sqrt{\frac{(-1)^{n+1}}{2^n(n!)^2} DET(\hat{B})}, \tag{3.4}$$

where $\hat{B}$ is the $(n+2)x(n+2)$ matrix obtained by bordering B with a top row $(0,1,...,1)$ and a left column $(0,1,...,1)^T$. The symbol $|\bullet|_2$ indicates the $L2$-norm, the vectors $|v_i - v_k|_2$ indicate the edges' lengths and the determinant in 3.4 is the *Cayley-Menger* determinant.

- It has been said already that the main problem about this algorithm regards the search of the simplex, and it is caused, for chaotic maps, by the form of the points in the state space. Indeed, it is quite rare for the algorithm to find the simplex at the first pick, usually it takes a lot of combinations (point 4(e)) and this slows down the process. This is the reason why we choose to stop the search of the simplex when the number of combinations of the point 4(e) exceeds 500. If the simplex is not found in the first 500 combinations, the prediction becomes a simple KNN, with $K = E$.

- We studied if, for the prediction, it is better to make an arithmetical average or an exponential weighted one. We found out that the results are not really sensitive to the average type, however, to be precise, the exponential one (3.1) it is slightly better.

- The results are not sensitive to the value of τ neither. However, to get a better accuracy, we need to use the smallest value possible.

3.3 K-Nearest Neighbors method

Given the difficulty in searching the simplex, for the subsequent analysis, we have implemented a simpler algorithm. It is almost the same method implemented by Suigihara and May, the only difference consists in avoiding the use of the simplex and simply take the first K-nearest neighbors of the predictee. This method was revealed as effective as the algorithm of S-M, but simpler and faster.

The algorithm

Let $x(t)$ be a set of samples of a time series measured at evenly-spaced times up to some maximum time t_{max}, $\{x(t)\}_{t=t_0}^{t=t_{max}}$ and let Φ_{T_P} be the time evolution operator, which acts on the series like $\Phi_{T_P}x(t) = x(t+T_P)$, the procedure consists of the following commands.

1. Choose an embedding dimension E and a value of lag τ. Then split the series in two parts: the first composed by the earliest N_{space} values, the second composed by the last $N = t_{max} - N_{space}$ ones.

2. Use the first part to build an E-dimensional space like $\{\{x(t=N_{space}), x(t=N_{space}-\tau), ..., x(t=N_{space}-(E-1)\tau)\}, \{x(t=N_{space}-\tau), x(t=N_{space}-2\tau), ...\}, ...\}$ obtaining $N_{space}-E+1$ points, which form what Suigihara and May call the *state space*.

3. Given a value $x(t) \in \{x(t=N_{space}+\tau), ..., x(t_{max})\}$, of which we want to make a prediction T_P steps in the future, take the previous $E-1$ values of the series in order to build an E-dimensional point $(x(t), x(t-\tau), ..., x(t-(E-1)\tau))$, which Suigihara and May call the *predictee*.

4. Take the K nearest neighbors of the predictee from the state space, $\{x(t_i)\}_{i=1}^{i=K}$.

5. For each point obtained, apply the time evolution operator Φ_{T_P}:
$\{\Phi_{T_P}x(t_i)\}_{i=1}^{i=K} = \{x(t_i+T_P)\}_{i=1}^{i=K}$.

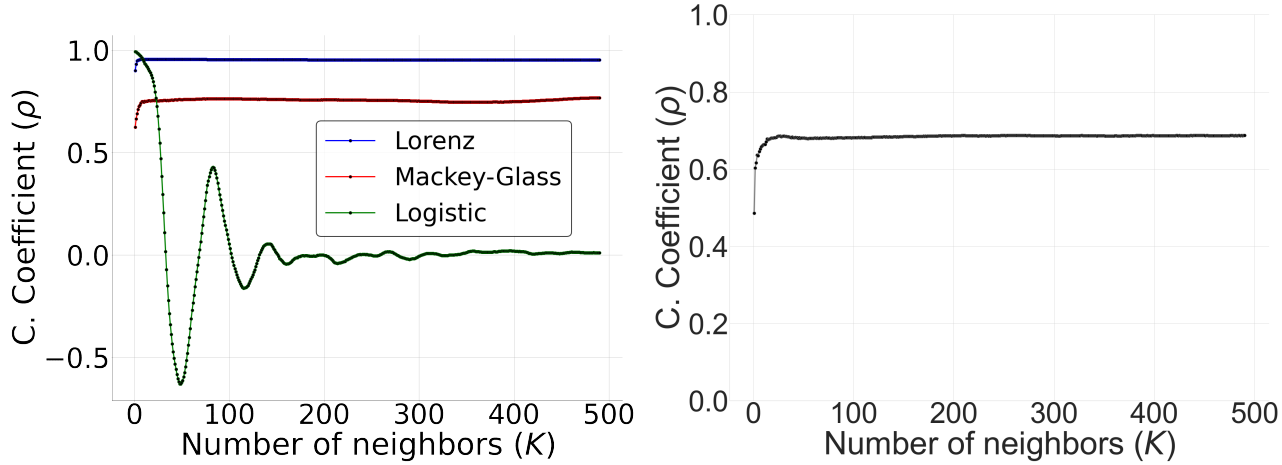
6. Predict the E-dimensional point as the exponential average:

$$\langle x \rangle = \frac{\sum_{i=0}^{K} x(t_i+T_P)e^{-d_i}}{\sum_{i=0}^{K} e^{-d_i}}$$

where d_i are the euclidean distances in $\mathbb{R}^E$.

7. Then the prediction of x(t) is the first coordinate of $\langle x \rangle$.

For this method, as for the SP's, are true the same last two observations made the previous section. Moreover, it is appropriate, for the prediction, to find that value of K which maximizes the correlation coefficient. However, the use of the exponential average, defined as the 3.1, should let us avoid the search for the best K.



(a) (b)

Figure 3.2: (a) in this figure the correlation coefficient of the three chaotic maps. The goodness of the prediction maximizes after few steps for Mackey-Glass and Lorenz, while the trend of ρ for the Logistic map is quite unexpected: seems that the influence of neighbors is such important for this map that even weighing the elements up does not block the decreasing of the average. (b) On the right the correlation coefficient calculated for the sum between the sine wave and white noise described in figure 3.11b. As expected the correlation saturates after few steps.

Automatically, weighing up the elements with an exponential function of the distance should decrease the contribution of both the most far neighbors and the programmer, to find that value of K necessary to not underestimate the average value. In fact, what we expect when looking at a graph $\rho(K)$ would be a correlation coefficient increasing until finding a maximum value, which does not change significantly even when the number of neighbors increases. This is the case of Lorenz and Mackey-Glass systems. However, what happens when using the logistic map is not clear to the author.

3.4 The Recurrent Neural Network

As explained in chapter 2, since to reproduce a time series by using artificial intelligence, it needs a neural network able to recognize the progression over time, we choose a RNN. A great gain is obtained when using a neural network: the possibility to increase the size of the training set, without slowing down significantly the code. The increment of the performances of the model are clearly visible. For the subsequent analysis we used a training set of size 10000 and a data set of size 3000, then we reshaped them in a tensor, whose dimensions are (100,100,1) for the training

set and (30,100,1) for the data set. According to what is written in section 2.4, the RNN units will "unroll" 100 times for each batch provided by the training set and data set. In the program, we used the classes provided by the python library Tensorflow, plus we tested architectures. We choose to use:

- A BasicRNNCell with num_units= 1000.
- Relu activation functions.
- Adam optimizer.
- Mean squared error as loss function.

3.5 Forecasting of chaotic time series

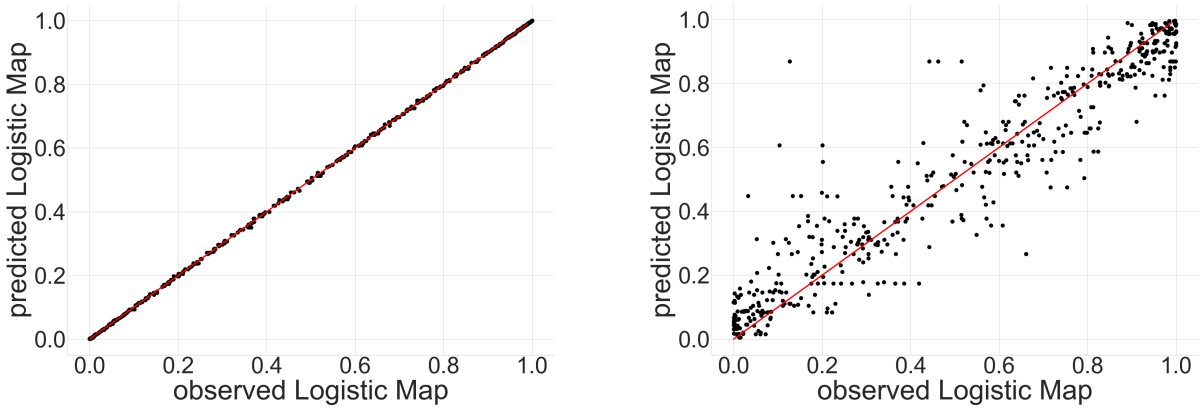
In this section, our aim is to test the three methods on the three chaotic maps shown in section 1.3, mainly, to discover if one of these methods could be more effective than the other two when it comes to the prediction over time. However, the results do not help to drawing a final conclusion, since it shows that the most effective method differs for each map.
As for the predictions made with the methods of KNN and SP, we used as dataset 500 points to build the reconstruction space and other 500 to verify if the prediction works; as for the predictions made with the RNN, we used a training set composed by 10000 points and a test set of 3000 points. The predictions have been made by varying the available parameters:

- For the KNN, the elements involved are: the number of neighbors K, the embedding dimension of the state space E and the prediction-time interval T_P. Since the trend of $\rho(K)$ is already shown in figure 3.2, in this section we will be displayed only $\rho(E)$ and $\rho(T_P)$.
- For the SP the elements involved are: the embedding dimension of the state space E and the prediction-time interval T_P.
- For the RNN, the element involved is just the prediction-time interval T_P.

Subsequently the predictions are analyzed by using the correlation coefficient between the predicted and actual time-series as function of the parameters. Suigihara and May explain how they used the series of the first differences $x_t - x_{t-1}$, to make the state space denser and to reduce short-term linear autocorrelation; however they admitted, through a later analysis, that nothing changes when the series itself is used. Due to this fact, for all the analysis made in the thesis, we will directly use the original time series. As it happened to Suigihara and May, the correlation coefficient tends to decrease as T_P grows because the predictions are heavily influenced by the natural divergence between the trajectories of nearby points, a feature of chaotic time series (see 1.2).
Moreover following the procedure applied by S-M, we analyzed the trend $\rho(E)$ of each map and we were able to find the same qualitative result only regarding the Logistic map. In particular we found out that the trend of $\rho(E)$ for the simplex projection's method is strictly related to the number of neighbors (K) used for the prediction, because is always $E+1$. This in high dimension causes an heavy contamination between nearby points and points, whose coordinate are far from the vector we want to predict. Anyway, in the KNN method, the number of neighbors is independent from E and the graph of $\rho(E)$ is always smooth and quite flat for almost every values of E.

3.5.1 Logistic map

For this section we used the 1.14 with the parameter $r = 4$.



(a) (b)

Figure 3.3: In figure (a) and (b) are shown the plots of the observed time series vs the predicted time series with both $E = 2$ and respectively $T_P = 1$ and $T_P = 6$.

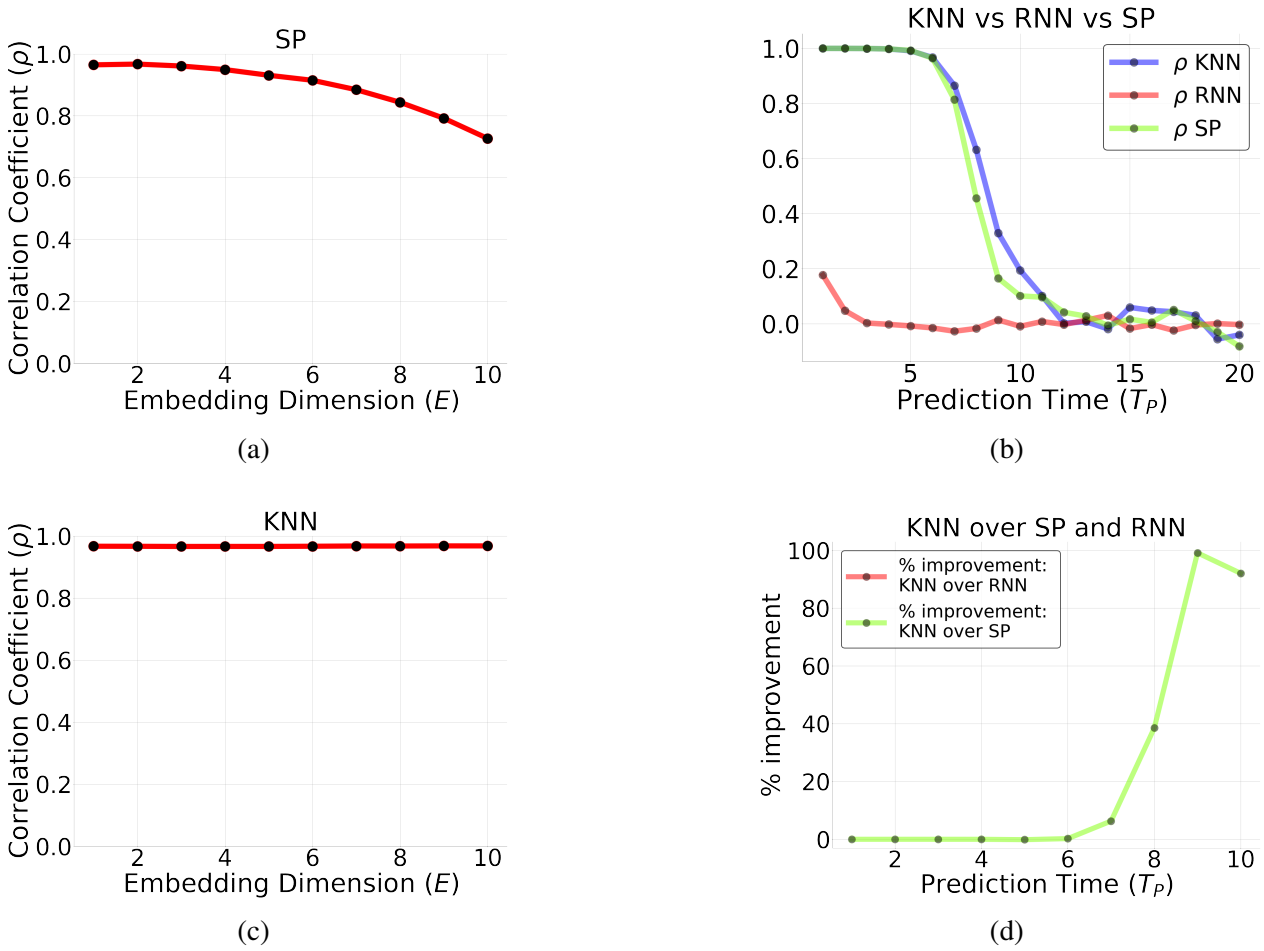


Figure 3.4: In all four figures is shown the correlation coefficient between the data of the map and the data predicted by the algorithms as the coefficients E and T_P vary. For both the methods of KNN and SP, in figure (b) $E = 1$, while in figures (a) and (c) $T_P = 6$ It can be seen that, while the KNN's $\rho(E)$ is rather constant, the SP's $\rho(E)$ is a decreasing function. This happens because the the logistic map is very sensitive to the number of neighbors K in the state space and the predictions made using high values of E are contaminated by the higher number of neighbors K, that for SP is equal to $E+1$. Figure (b) shows the trend of $\rho(T_P)$ of the predictions, while figure (d) shows the percentage increment of the KNN's $\rho(T_P)$ over the other methods' $\rho(T_P)$. The predictions of the SP are similar to the KNN only for $\forall T_P \leq 6$, while the predictions made with the RNN are visibly far worse. For every $T_P \leq 10$ the KNN's $\rho(T_P)$ is more than 100% better than the RNN's.

This is exactly the same map used by Suigihara and May in their work $x_{t+1} = 4x_t(1-x_t)$. In the figure 3.4 are shown the analysis of the Logistic map. The correlation coefficient of the prediction has been calculated for all the three methods. It is immediately clear that the results obtained by using the RNN displayed the worst correlation. However, the other 2 methods could be almost comparable for small values of T_P, even though the predictions by KNN are slightly better. To make the predictions using the Simplex Projection and varying T_P, $E = 1$ was kept the same, because the method finds the simplex nonzero times only under this condition. Precisely it finds the simplex for the 100% of cases. However, To make the predictions using the KNN, it has been kept $K = 1$

because is the case which maximizes the predictions' accuracy, as the figure 3.2 shows.

3.5.2 Lorenz system

In this section we explore the Lorenz system of differential equations 1.7 and we analyze the predictions as in the previous section. This is one of the classic systems of non-linear differential equations. It exhibits a range of different behaviors as the parameters (σ, β, ρ) change, for more details the reader is invited to read the first chapter. In the analysis displayed below we use the x coordinate of the system simulated using ($\sigma = 10.0$, $\beta = \frac{8}{3}$, $\rho = 28.0$).

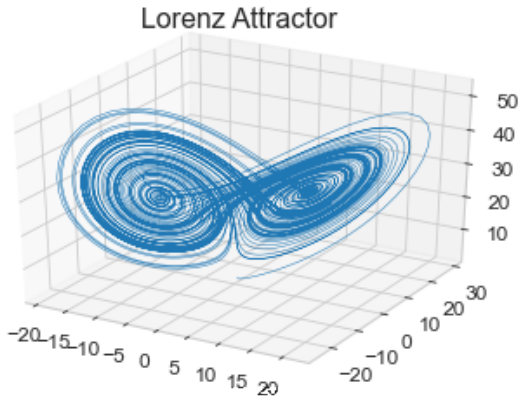
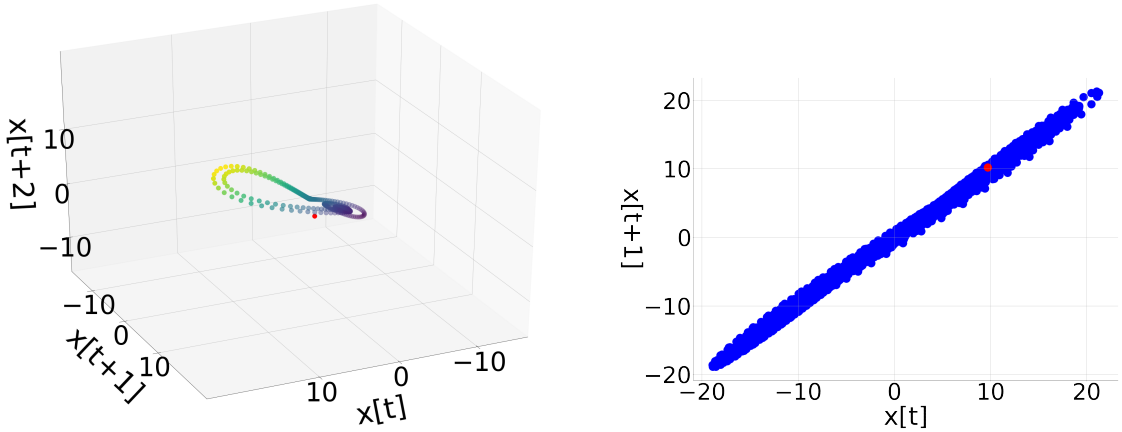


Figure 3.5: In this figure is shown the Lorenz attractor of the simulation made using Python. In the following pages, the analysis of the motion of x-coordinate in this picture will be shown.



(a) (b)

Figure 3.6: As the 3.1 in this images it is shown the Poincaré plots of the Lorenz system. In other words the state space for $E = 2, 3$.

Following what we did for the Logistic Map, for the predictions made with KNN and SP we use the first 500 values to fill the state space and it has been be used the last 500 values to make the predictions. Figure 3.7 shows how the correlation coefficient changes as function of E and T_P.

Precisely in 3.7a is plotted the graph which shows $\rho(E)$ calculated between the data of the times series and the data predicted with the method of SP. We found the same problem of the logistic map concerning the search of the simplex. For $E = 1$ the simplex is found 98% of the times cases, for $E = 2$ 75%, for $E \geq 3$ less than the 3% and for $E \geq 5$ 0%.

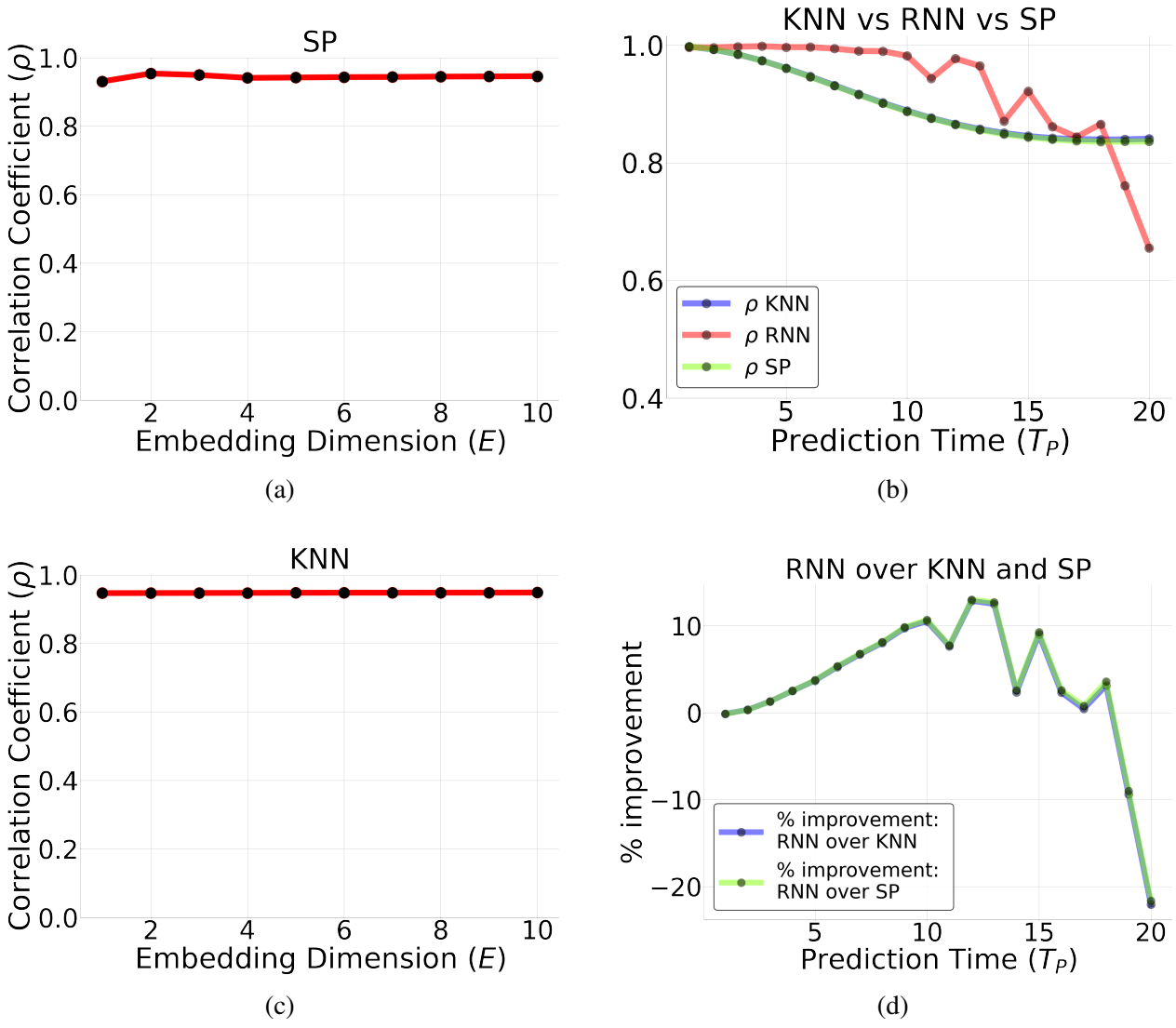


Figure 3.7: In all four figures is shown the correlation coefficient between the data of the map and the data predicted by the algorithms as the coefficients E and T_P vary. For both the methods KNN and SP, in figures (a) and (c) $T_P = 6$; however, in figure (b) $E = 1$ for the KNN and $E = 2$ for the SP. It can be seen that, while the KNN's ρ is rather constant, the SP's ρ displays a maximum when $T_P = 2$. Suigihara and May thought that this feature could estimate an upper bound for the dimensionality of the attractor in the chaotic maps [32]. Figure (b) shows the trend of $\rho(T_P)$ for the predictions by all methods, while figure (d) shows the percentage increment of RNN's $\rho(T_P)$ over the other methods's $\rho(T_P)$. It can be noticed that, for the Lorenz system, the RNN makes the best predictions, while SP and KNN show a very similar correlation coefficient for every T_P.

Whereas a bigger quantity of simplexes is found for $E = 1$, the method seems more effective when keeping the parameter $E = 2$. However, the differences are so tiny to be almost invisible. As

for the Lorenz system, the best method for the predictions seems to be the artificial intelligence's algorithm, at least when $T_P \leq 18$. However, SP and KNN methods display a similar value of correlation coefficient for every T_P. Unlike the Logistic map, when is used the KNN, the predictions' accuracy saturate quickly when the number of neighbors K increases, indeed we used $K = 20$ for the forecasts analyzed.

3.5.3 Mackey Glass system

In this section it will be analyzed the nonlinear time delay differential equation of Mackey and Glass.

$$x_{k+1} = c \cdot x_k + \frac{a \cdot x_{k-d}}{b + x_{k-d}^e} \tag{3.5}$$

where a,b,c,e are real numbers, and d represents the delay of the variable x at time t. To see more details the reader is invited to look at the chapter 1.3.3.

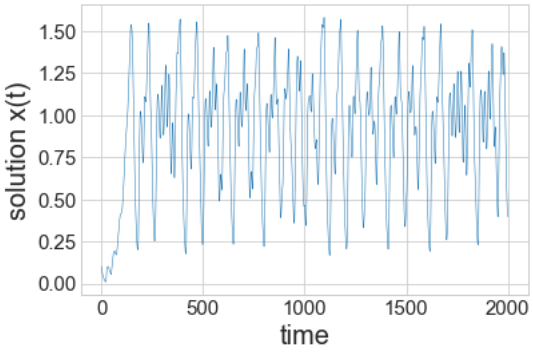
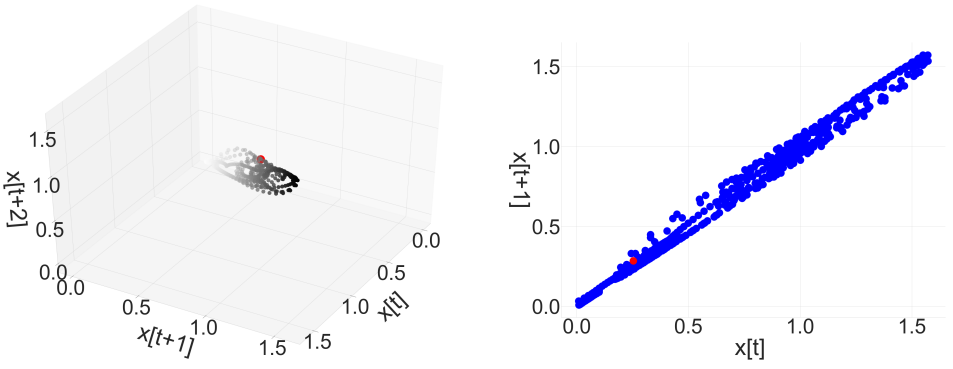


Figure 3.8: In this figure the solution of the Mackey Glass equation as function of t.



(a) (b)

Figure 3.9: As the 3.1 in this images it has been shown the Poincaré plots of the Mackey-Glass equation, or rather the state space for $E = 2$ and $E = 3$.

As always in this work, we used the first 500 values to fill the state space in which the prediction will be made and the subsequent 500 values as predictees to forecast. As it happens in the Lorenz system, when the predictions are calculated by SP the trend of $\rho(E)$ displays a maximum in proximity of $E = 2$ (3.10a). Despite the fact that the method finds the simplexes only for the 83% of the 500 predictees versus the 100% obtained when the predictions are calculated by using $E = 1$.

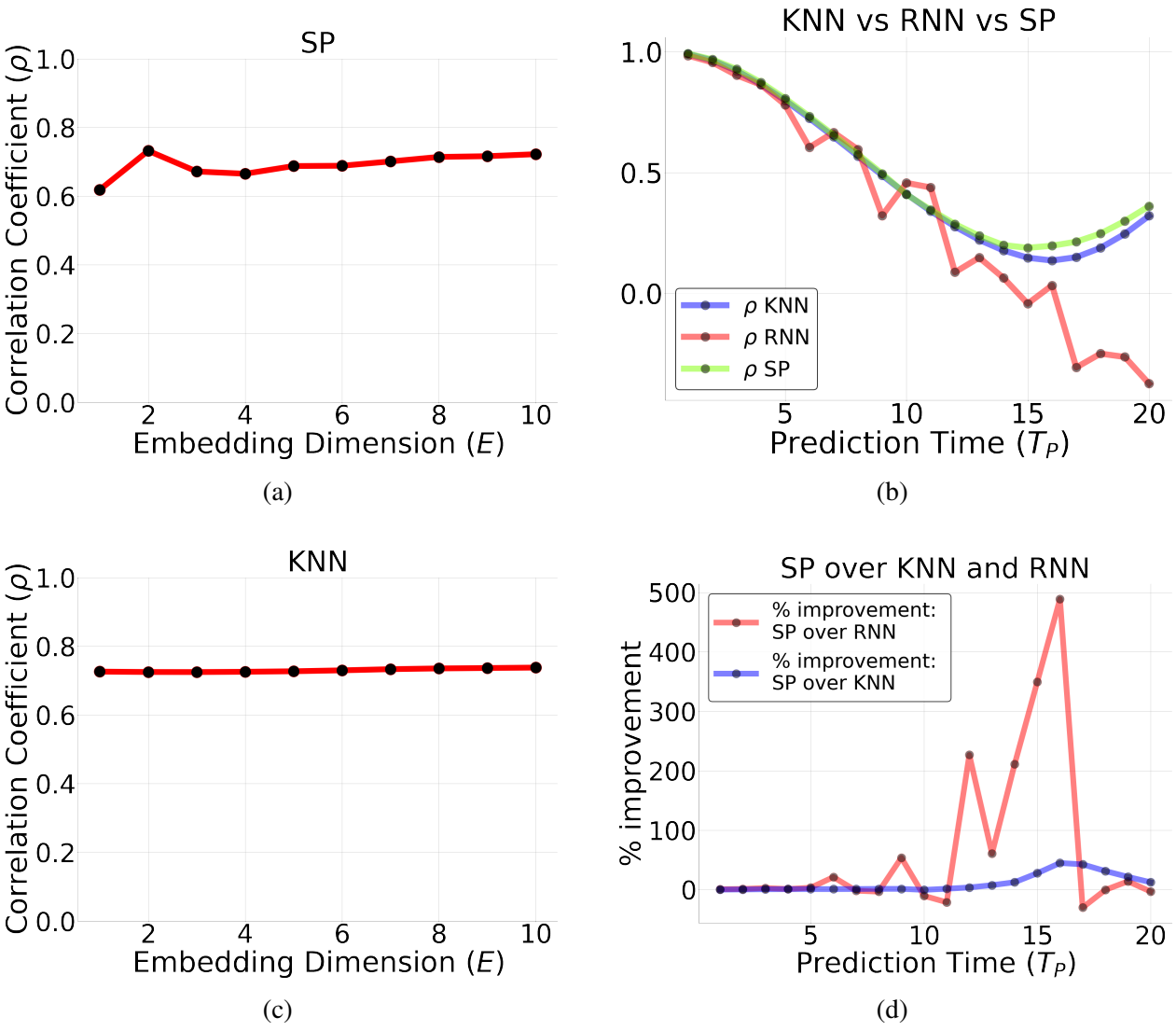


Figure 3.10: All the four figures is show the correlation coefficient between the data of the map and the data predicted by the algorithms as the coefficients E and T_P vary. For both the methods KNN and SP, in figures (a) and (c) $T_P = 6$, however in figure (b) $E = 2$. It can be seen that, while the KNN's ρ is rather constant, the SP's ρ displays a maximum for $T_P = 2$, as it happens in the Lorenz system. It is important to underline that, for the solution of the Mackey-Glass system, the graph $\rho(E)$ visibly increases when the predictions are made using SP. This feature is a consequence of the ρ's tendency to increase when the number of neighbors K increases too (remember that for SP $K = E + 1$). Figure (b) shows the trend of $\rho(T_P)$ for all methods' predictions, while figure (d) shows the percentage increment of the SP's $\rho(T_P)$ over the other methods' $\rho(T_P)$. It can be noticed that for the M-G system the SP makes the more trustworthy predictions, even if the ρ displayed is very similar to the KNN's.

Even using the solution of the Mackey-Glass equation, the ability of finding simplex decreases rapidly with the increasing of E: for $E = 3$ is less then 36% and for $E \geq 6$ is already 0%. The predictions calculated for low values of T_P ($T_P \leq 11$) are almost similar for all the methods. However, for higher values of T_P, the RNN's predictions fails completely, whereas the SP's and KNN's display a similar and higher accuracy. As it happens in the Lorenz system, when is used the KNN, the predictions' accuracy saturate quickly when the number of neighbors K increases, indeed we used $K = 30$ for the forecasts analyzed.

3.6 Forecasting of a nosy limit cycle

Although at the first sight, a plot of a noisy limit cycle and a plot of a chaotic series seems to be very similar to each other as we can observe in the figures 3.11a and 3.11b, the method implemented should distinguish the two different class of time series. In their article, Suigihara and May, explained that one of the main features that makes an analysis of predictions of a chaotic time series using the SP differ so much from the one of a noisy limit cycle, is the sensitivity to the parameters E and T_P. In particular, while a chaotic time series display a decreasing trend as these two parameters increase, the noisy limit cycle seems to remain constant as E and T_P change, or at least it seems to vary smoothly.

To outline the different behavior of the two types of time series in figure 3.11 shows the same analysis made by Suigihara and May, where as for limit cycle, it has been taken an uncorrelated noise superimposed on a sine wave defined as the following time series:

$$x(t_i) = \sin(0.5t_i) + R_i \tag{3.6}$$

where R_i is a random number sampled uniformly in the interval $[-0.5, 0.5]$. On the other hand, after having analyzed the chaotic times series trying to reproduce the same graphs Suigihara and May did, it has been noted that the behavior showed by ρ for higher values T_P and E, can't bing to the same conclusions.

Concerning the predictions of the noisy limit cycle, even if $\rho(T_P)$ is quite smooth for small T_P, it has a total different behavior when the graph shows a larger interval.

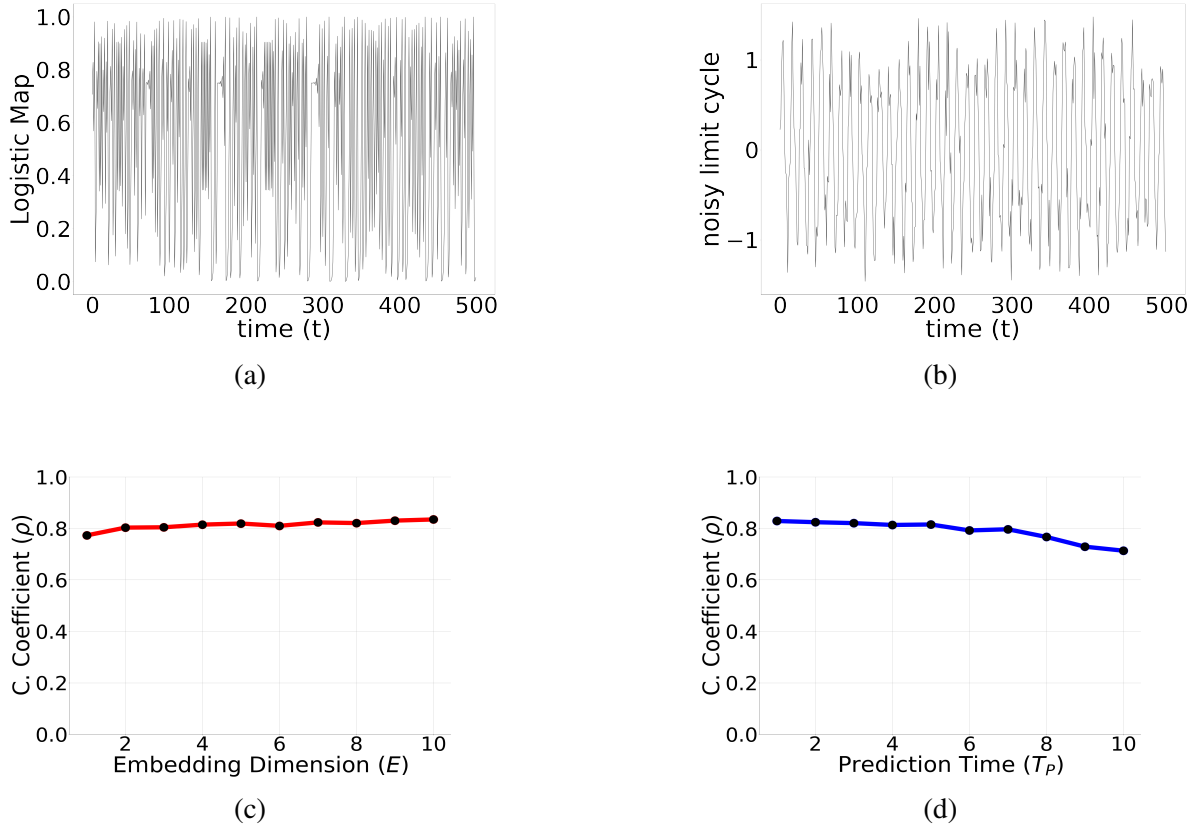


Figure 3.11: Figures (a) and (b) show respectively a plot of 500 points of the Logistic map and a plot of 500 points of a noisy limit cycle. The limit cycle is built adding to sin(0.5t) a random number chosen uniformly between [-0.5,0.5]. Figures (c) and (d) show how ρ varies as function, respectively, of E and T_P for the limit cycle. We can see the same results in the analysis made by Suigihara and May as the graphs are almost constant and ρ seems to be insensitive to the variation of the two parameters in the range $\rho, T_P \in [1, 10]$.

The trend is periodic as the limit cycle is and, as figure 3.13 shows, it overlaps almost perfectly on the graph of the autocorrelation function. Moreover it has been observed, analyzing all the maps, that the graph $\rho(E)$ after having analyzed all the maps, it is noticeable that in each case the graph $\rho(E)$ is quite flat, decreasing, but very slowly. These behaviors differ from those that S-M outlined in their article. Because of this, we can state that our method of SP can not understand if a time series has chaotic components.

However, we hypothesize that the graph $\rho(E)$ calculated using the values predicted with the KNN might help us to discern whether a time-series is purely stochastic or it presents dynamical elements. While analyzing the three chaotic times-series, we noticed that by using the method of KNN, fixing the number of neighbors K and letting the embedding dimension E vary, we could observe different behaviors between a chaotic time series and a noisy limit cycle. Whereas chaotic maps display a smooth graph of ρ, whose variations are visible for great values of E, noisy limit cycles display a

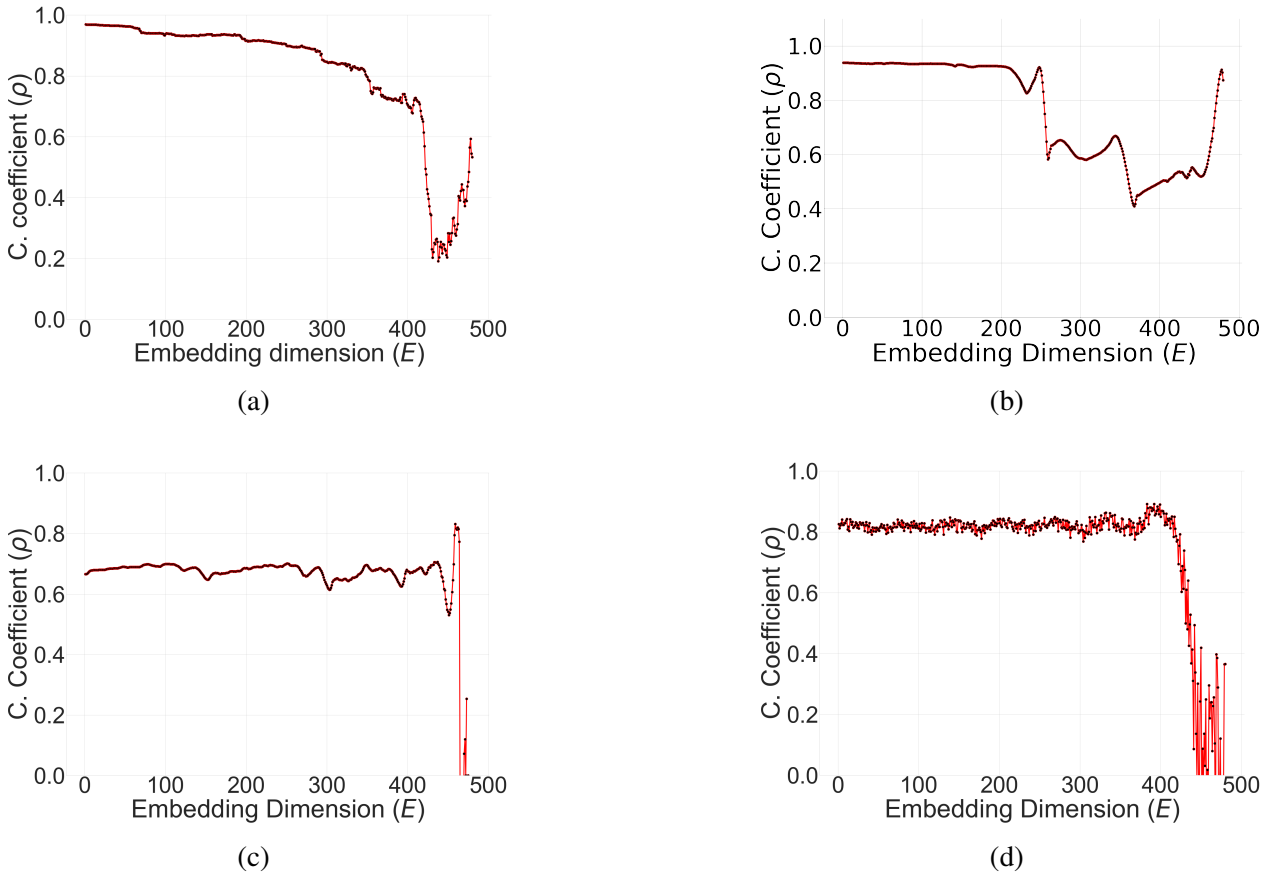


Figure 3.12: All the four figures is shown the correlation coefficient between the data of the map and the data predicted by the algorithms as the coefficient E increases, while $T_P = 6$. For all these analysis, 500 points were used to make the state space and 500 points to verify if the predictions work. The values of K are different in every graph, although previous simulations made by the author confirmed that the trend do not qualitatively depend from it. In the figures (a), (b) and (c) are shown the correlation coefficients of the chaotic maps, respectively Logistic map, Lorenz system and Mackey-Glass equation. In figure (d) is shown the trend of noisy limit cycle's ρ, as it can be easily seen in the images shown, this is the roughest among of all four.

rough graph of ρ and the values arrange themselves around the same mean value $\forall E$. In the figure 3.12 are shown the graph $\rho(E)$ for big E for a limit cycle and the three chaotic maps. When E asymptotically reaches 500, the value of ρ drops. However this feature has no dynamical reasons: simply, increasing E, the number of points in the state space decreases to be approximately close to K. This contaminates the average, when we make the prediction.

3.7 The predictability of a time series depends on the autocorrelation

Suigihara and May in their article stated that the method implemented displays either a flat $\rho(T_P)$ when implemented on a noisy limit cycle or a decreasing one when implemented on a chaotic time series. However, for large values of time step (T_P), the correlation between predicted and actual time series is not flat for noisy limit cycle and it generally can increase if the time series is chaotic. To gain insight of this problem, we implemented the auto-correlation function defined as

$$\chi(T_P) = \frac{\langle Z(t)Z(t+T_P)\rangle - \langle Z\rangle^2}{\sigma_Z^2} \tag{3.7}$$

where $Z(t)$ is a time series, $\langle Z\rangle$ is its mean value, σ_Z^2 is its variance and T_P is the lag-time. In the series used in this thesis time is discrete, therefore $\chi(t)$ can be calculated directly only for a finite set of time steps. When we have a set of samples of the time series Z(t) measured at evenly-spaced times up to some maximum time t_{max}, the correct formula for the autocorrelation function is

$$\chi(T_P) = \frac{\frac{1}{t_{max}-T_P}\sum_{t=0}^{t_{max}-T_P} Z(t)Z(t+T_P) - \frac{1}{t_{max}-T_P}\sum_{t=0}^{t_{max}-T_P} Z(t)\frac{1}{t_{max}-T_P}\sum_{t=0}^{t_{max}-T_P} Z(t+T_P)}{\frac{1}{t_{max}}\sum_{t=0}^{t_{max}} Z(t)^2 - (\frac{1}{t_{max}}\sum_{t=0}^{t_{max}} Z(t))^2}. \tag{3.8}$$

When the formula 3.8 has been used on the chaotic time series it has been noted that the drops of $\rho(T_P)$ qualitatively correspond to the drop of $\chi(T_P)$. In fact, the conclusion of the author is that the graph of $\rho(T_P)$ cannot say us more than what the autocorrelation 3.7 might do. Moreover, the predictions' accuracy can be qualitatively predicted just by looking at the graph of 3.8.

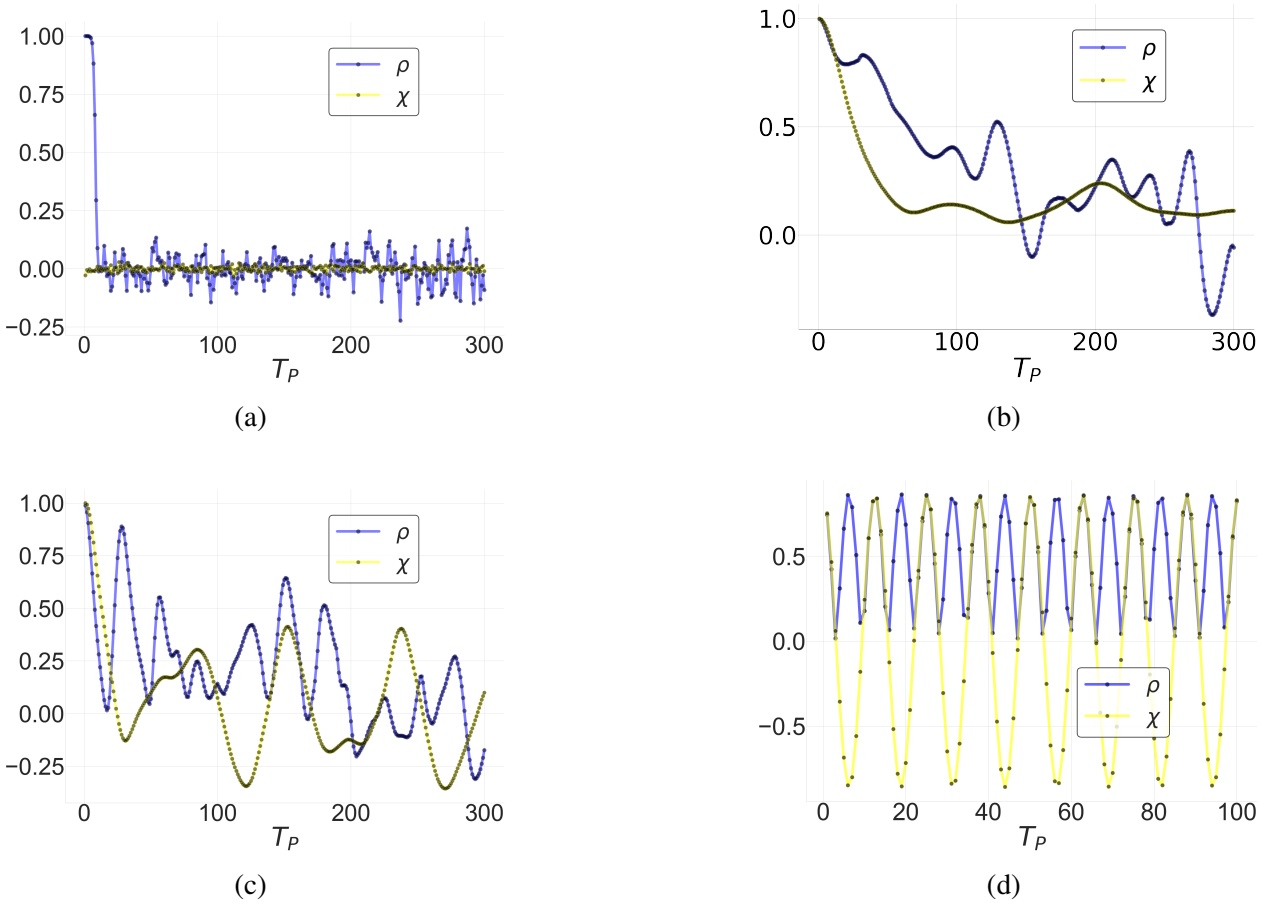


Figure 3.13: All the four figures is show the correlation coefficient (in blue) between the data of the map and the data predicted by the algorithms and the autocorrelation (in yellow) as the coefficient T_P varies. Figures (a), (b) and (c) show the graphs of the chaotic maps, respectively Logistic map, Lorenz system and Mackey-Glass equation. Figure (d) shows the trend of the noisy limit cycle's ρ and χ. The Lorenz system seems the most predictable in the future because its solution is the most autocorrelated among the three. On the other hand, the Logistic map is the most uncorrelated and thus the most unpredictable.

3.8 Achievements of the third chapter

1. We noted that the simplex projection is only a version of the simpler K-nearest neighbors. Moreover it displays a similar accuracy, but is slower. See section 3.5.

2. From the analysis of the predictions of the simplex projection we can not distinguish a chaotic map from a noisy limit cycle in the same way Suigihara and May did. See section 3.6.

3. We suppose that a method to discern whether a map is a noisy limit cycle or not, is looking at the trend of $\rho(E)$ when the predictions are made using the KNN. We do not have a demonstration. We just tested it on synthetic data and observed the results. See section 3.6.

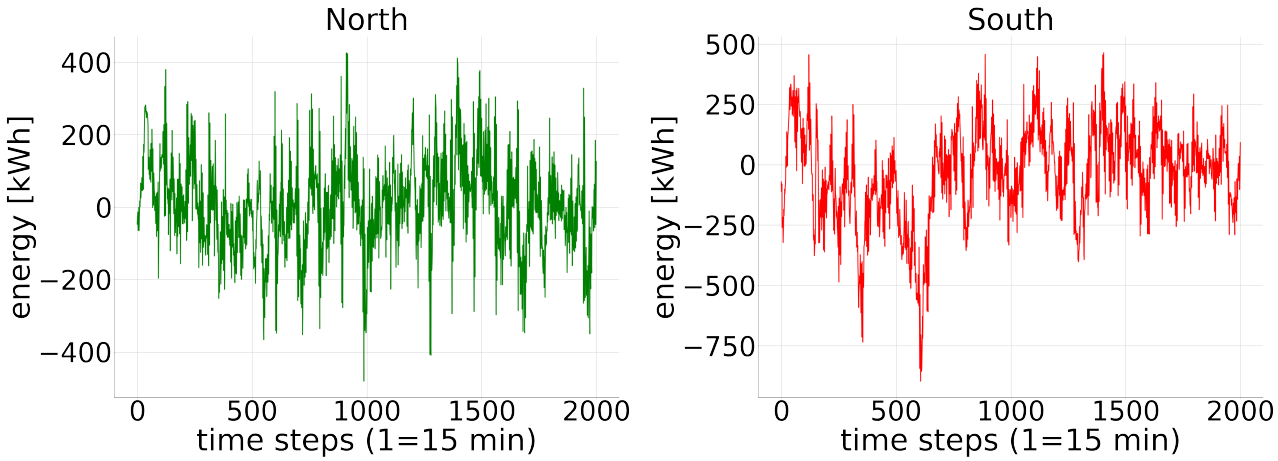
4. We noted that, when the predictions' accuracy is measured with the correlation coefficient, its trend is qualitatively similar to the trend of the autocorrelation function. See section 3.7.

4

APPLICATION OF THE ALGORITHMS TO THE PREDICTION OF THE ENERGY'S UNBALANCES OF THE ITALIAN POWER GRID

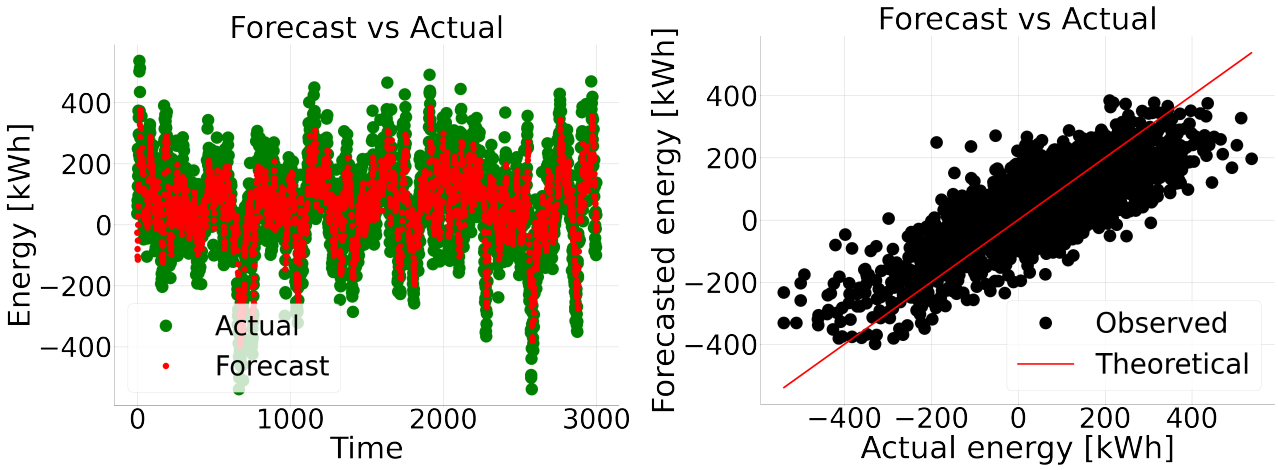
In this section we will apply the methods explained in chapter 3 to forecast on the energy's unbalances time series offered by Terna S.p.A. We start by introducing in section 4.1 the functioning of the Italian energy market, where Terna operates, in order to provide an overview of the problem to the reader. Then, we introduce the results of the forecasting techniques explained in the last chapter on the data provided by Terna. In section 4.2 we use the K-nearest neighbors (KNN) and simplex

projection (SP), while in section 4.3 we try to reproduce the time-series using a recurrent neural network (RNN). Terna has collected old time series of energy's unbalances every 15 minutes, and the values are labelled differently whether the energy consumed is in the northern or in the southern part of Italy. Figure 4.1 shows the time series that we are going to forecast.



(a) (b)

Figure 4.1: In this figures the energy's unbalances times series provided by Terna S.p.A. On the left (a) in the north Italy, on the right (b) in the south Italy. The unbalances are distributed every 15 minutes; indeed 2000 time-steps correspond to 20 days and 20 hours.



(a) (b)

Figure 4.2: (a) In this figure is displayed the predicted energy's unbalances vs the observed ones of the test set as function of time. (b) In this figure is shown on the y-axis the predictions of energy's unbalances and on the x-axis the actual energy unbalances of the test set. In both figure the predictions are calculated 4 time steps in the future ($T_P = 4$) using a recurrent neural network.

4.1 The ancillary services market and the task of Terna S.p.A.

The day-ahead market. In 1999, in Italy the liberalization of the wholesale electrical market took place giving it a new structure, essentially similar to the other European countries: there is a "commercial" market, organized in auctions, on which operate consumers and manufacturers (i.e. operators of power stations). In the auction day, operators make an offer to sale/purchase the energy for each hour of the following day and every proposal consists of a pair of a quantity in MWh and a price in €/MWh. The market is "commercial". This means that at each trade corresponds an economical exchange. The sellers collect the moneys established by the auction and the buyers pay, however it does not exist an obligation imposing the buyer to consume the energy contractualized for that hour. This is commonly called the *day-ahead market* (DAM).

The problem of the instantaneous energy transmission: the work of Terna S.p.A. Electrical laws require that at each instant must be guaranteed the right balance between injections and withdrawals of the power grid, since there are no accumulation systems allowing the collection of temporary energy excesses that would permit the rebalancing of the grid an instant later. Clearly, it is not possible to rely on the results of the DAM to guarantee the energy balance, as operators may not comply with commercial injection/withdrawal commitments. Moreover, the high standardization of the products traded on the DAM is such that completely different $P(t)$ power profiles, with the same integral value over the course of an hour, are equivalent to each other from a commercial point of view, but not from a physical one.
It is therefore necessary an entity who:

- Constantly monitors the transmission power grid.
- Compensates the imbalances between injections and withdrawals in real time to guarantee the correct functioning of the grid.

In Italy this entity is Terna S.p.A., which is centralized at national level (ie: it is a monopolistic subject, given the nature of its functions) and, for its role in the energy transmission system, is called Transmission System Operator (TSO).

The ancillary services market. The second phase of the market, subsequent to the DAM, is the *ancillary services market* (ASM). This market is structured to guarantee that Terna S.p.A. manages the problem of transmission, trough a competitive process between the operators. Terna has no available electrical resources and, to compensate the unbalances between injections and withdrawals, it buys the services of the operators. Terna, indeed, in the ASM is the only buyer for the ancillary services offered by the different companies, which offer their services and are rewarded following the *pay-as-bid* rule: Terna pays the price requested by the emanation of the service.
For instance, suppose that at the 15:30 of the day D Terna needs a quantity of energy (let's say 1 MWh), which must be supplied in a certain time period (let's say between 16:00 and 16:15 of the same day D), to restore the balance in the power grid. Through the analysis between the price of the offers presented by the different operators, Terna finds a company that produces 1 MWh at the price of 100 €/MWh. By selecting this offer, Terna makes the chosen company's installations produce 1 MWh more than the usual level production, paying 100 € for the supply of the service. Clearly Terna, for the same service, selects the cheaper offer from the various operators, but this choice is not always so simple and clear. Terna's algorithm of selection takes into account several factors, most of them not even clear to the market operators, for example:

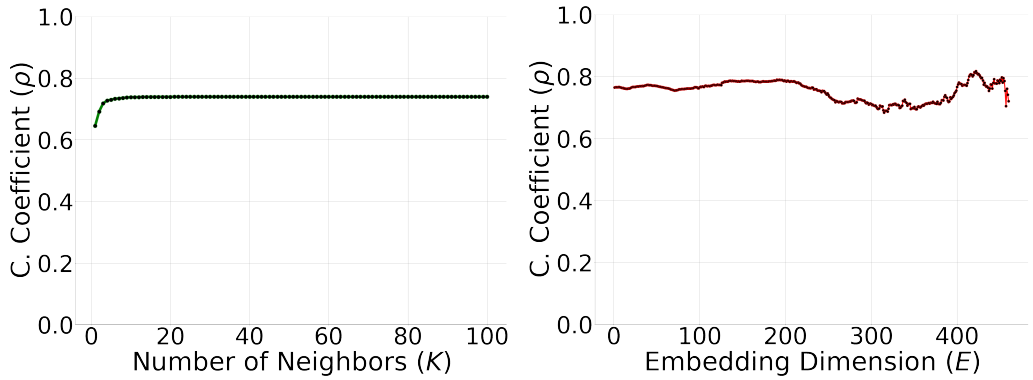
- Bonds for the maintenance of the tension. The supply of reactive power to the grid requires that a certain number of installations could be turn on even in low demand condition.
- Bonds related to the location. For instance, an unbalance in Sicily might not be fixed by a modulation on the production in Lombardy, because the grid could have interruptions.
- Bonds related to time. Turning on/off an installation might requires a time t in which it needs to be maintained on/out of service.

Basically, starting from an analysis of the available data, the problem is so complex that is often impossible to understand the choices operated by Terna on the offers' selection. Moreover the techniques operated by Terna have a relative long time evolution and, due to some factors, the operators might sometimes start from scratch in the comprehension of the ASM. It is not unusual to see significant fluctuation on the profits coming from the ASM of the operating companies in the electrical sector. This is a relevant risk for both Terna and the market's operators.

In the view of the foregoing, the aim of this research project is to recreate the time series of energy's unbalances in order to put Terna and the energy market's operators under a minor risk when operating on the ancillary services market.

4.2 Predictions of the energy's unbalances using the K-nearest neighbors and the simplex projection

As we did in the last chapter, we will use 500 points of the series to build the state space and 500 to forecast. Then, we will analyze the correlation coefficient between the predicted and actual time-series as function of K, E and T_P. The analysis of $\rho(K)$ shown in figure 4.3a displays what expected: the value predicted using the exponential average saturates after few steps. Indeed, for the subsequent predictions we will use $K = 30$. Figure 4.3b shows a smooth trend of $\rho(E)$ that seems more similar to the one of the chaotic maps than to the trend of the noisy limit cycle of figure 3.12. If the conclusions drew up in 3.6 are true, the energy's unbalances should display a chaotic trajectory.

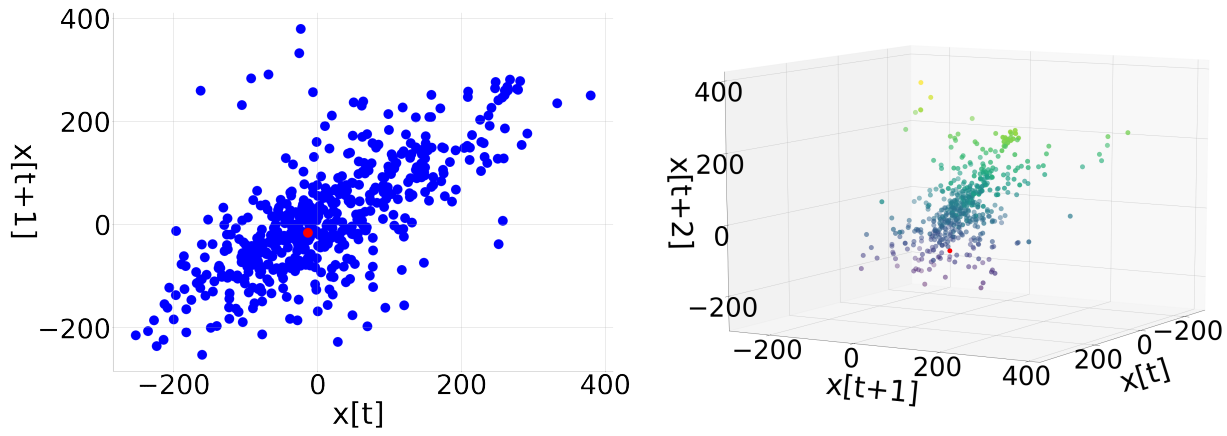


(a) (b)

Figure 4.3: In both figures it is used time-series of energy's unbalances in January 2019 in north Italy. (a) In this figure $E = 2$ and $T_P = 4$. Using more than 10 neighbors the goodness of the prediction is maximized. (b) On the right the correlation coefficient between the actual and predicted time series. $T_P = 4$ and $K = 30$. The correlation coefficient displays a smooth graph $\rho(E)$, as for the chaotic maps of figure 3.12 do. Are the energy's unbalances of the Italian power grid displaying a chaotic motion?

On the other hand, when Looking at the Poincaré plots in figures 4.4a and 4.4b it seems that the points arrange themselves in a casual disposition. Probably stochastic elements contribute to the

evolution of the time series and, even if it has dynamical components, the casual ones contaminate the Poincaré plots.



(a) (b)

Figure 4.4: This figures show the Poincaré plots of the time series. When confronted to the plots of the chaotic maps of section 3, it seems that the time series of the energy consumption are more stochastic than chaotic.

Given these preliminary remarks, in the next pages, we show the trend of the predictions' accuracy versus the increasing prediction-time interval (T_P). For the analysis we have used both the KNN and the simplex projection. In particular the simplex projection was implemented with a small value of E because, as it happens with chaotic maps, for higher values the algorithm does not find those simplexes that enclose the predictee, even when the Poincaré plots (shown in figure 4.4) suggest a suitable state space.

Figures 4.5c and 4.5d display the predictions' accuracy of the KNN and the SP as function of T_P. Two maximums are visible for $T_P = 4, 8$. This means that the time series are very suitable to be predicted one hour and two hours in the future. However it seems harder to increase the goodness of prediction for higher T_P, at least when using both the methods of KNN and SP.

The prediction obtained of the KNN/SP are not satisfying

How can we evaluate whether the predictions obtained by the KNN and simplex projections are a bad result?

We can implement a naive method of prediction and compare its predictions' accuracy with the one of our methods. If these first are the most trustworthy, we conclude that our methods are not

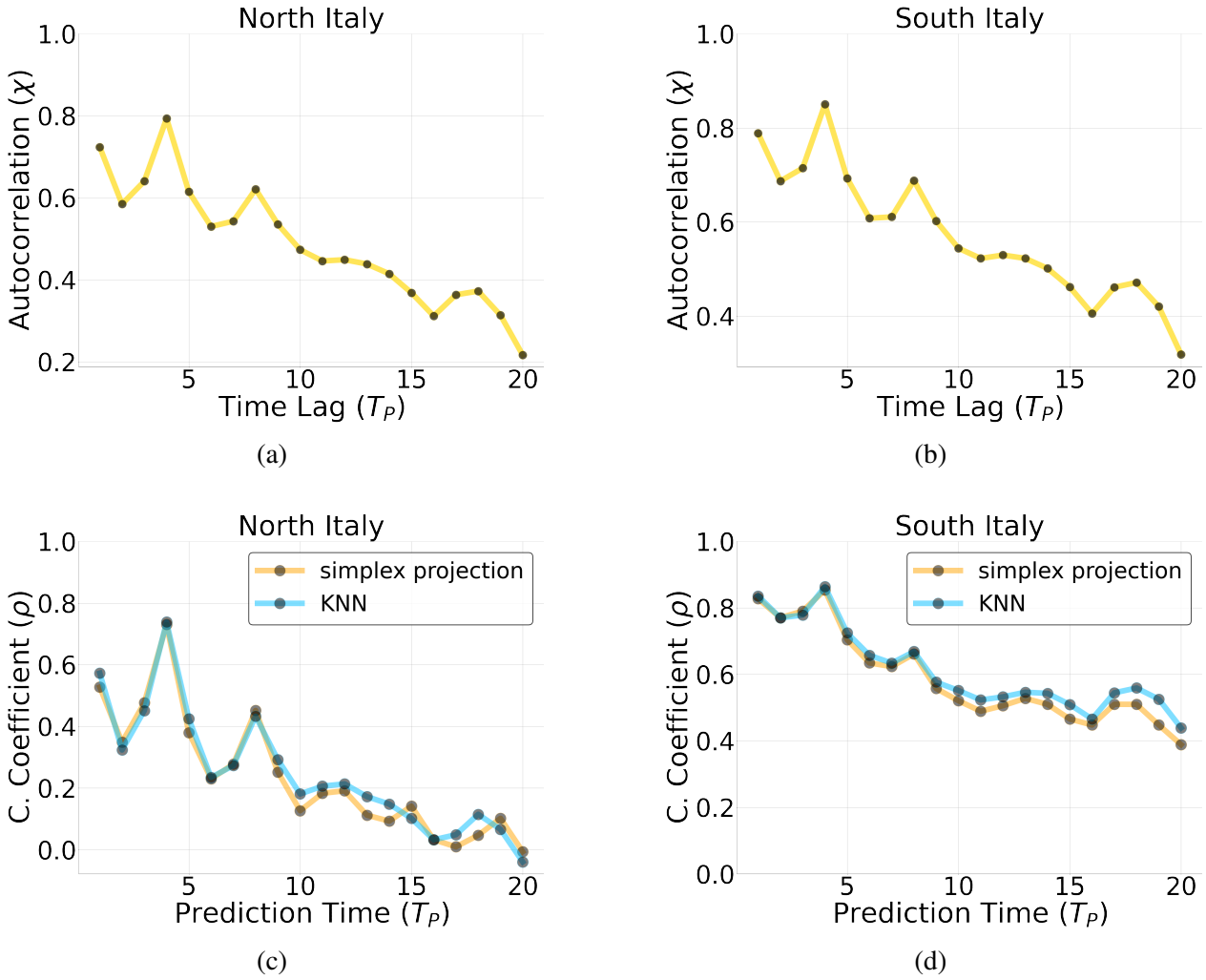
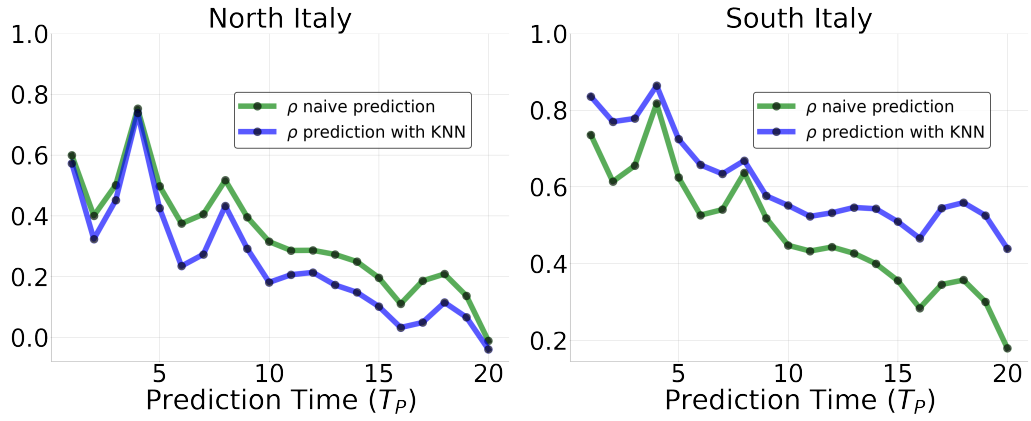


Figure 4.5: Figures (a) and (b) show the autocorrelation coefficient of the time series as the time-lag T_P varies. In the figures (c) and (d) it is compared the correlation coefficient as T_P varies for the two methods: KNN (with $K = 20$) and simplex projection (with $E = 2$). From the graphs we can see that $\rho(T_P)$ follows the same qualitative trend of $\chi(T_P)$, as expected. Moreover, as noted predicting the chaotic maps, we did not see a good difference between the simplex projection and KNN.

effective. Given a time series $x(t)$, the naive method consists in predicting the time series at time $x(t+T_P)$ with the value $x(t)$.

$$x(t+T_P) \longrightarrow x(t)$$

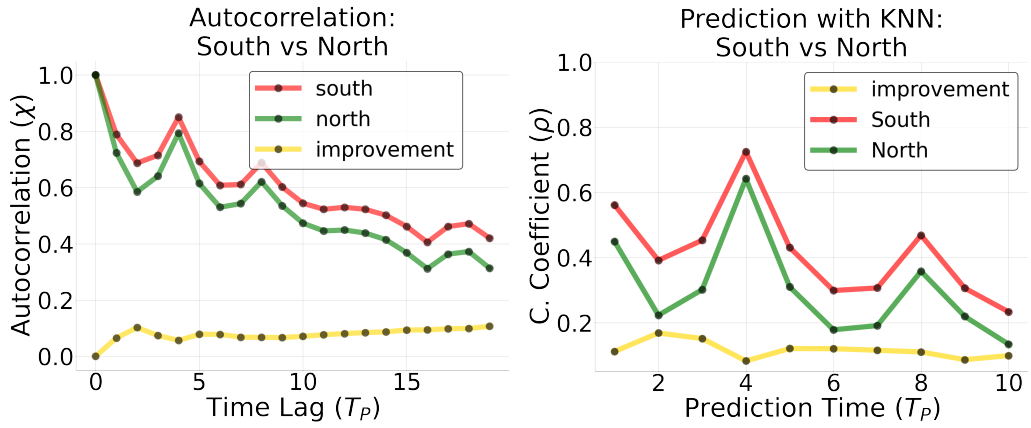
This is the most reasonable method we can implement when our time series is a random walk and we do not know anything abut the future steps, which are completely random by definition. This is exactly what we did on the energy's unbalances time series. We saw that the accuracy displayed by the KNN and SP changes according to the dataset used. While, for some dataset the KNN and SP's accuracy is dominated by the naive method's one, for others, it is the naive method's accuracy dominated by the one of the other methods.



(a) (b)

Figure 4.6: The figure shows the correlation coefficient between the predicted and the actual time series of the south and north unbalances. We can see the difference between the correlation coefficient whether the prediction of the KNN is calculated on the north or on the south. Since the SP's and KNN's accuracy is often similar for every dataset, the figure 4.6 shows only the KNN's one.

Moreover, we noted that, on average, southern predictions performs better than the northern ones. We suppose that this difference can be explained by looking at the values of autocorrelations of the time series time as explained in figure 4.7 .



(a) (b)

Figure 4.7: (a) This figure shows the autocorrelations of the time series of energy's unbalances in north and south Italy. The yellow line represents the difference between the autocorrelations of the two parts of Italy. (b) On the right the correlation coefficient between the actual and predicted time series of the energy's unbalances as T_P varies (using the method of KNN). In yellow is shown the difference between the correlations of the two parts of Italy. We suppose that the improvement of the KNN in the south is caused by the better autocorrelation of the dataset.

4.3 Predictions of the energy's unbalances using a recurrent neural network

In conclusion, we show the predictions' accuracy of the recurrent neural network. The "predictive power" of the neural network increases as the size of the training set grows and, computationally speaking, big training sets do not significantly slow down the code. For the subsequent analysis we used a training set of size 10000 and a data set of size 3000 and the increment of the performances of the method are clearly visible.

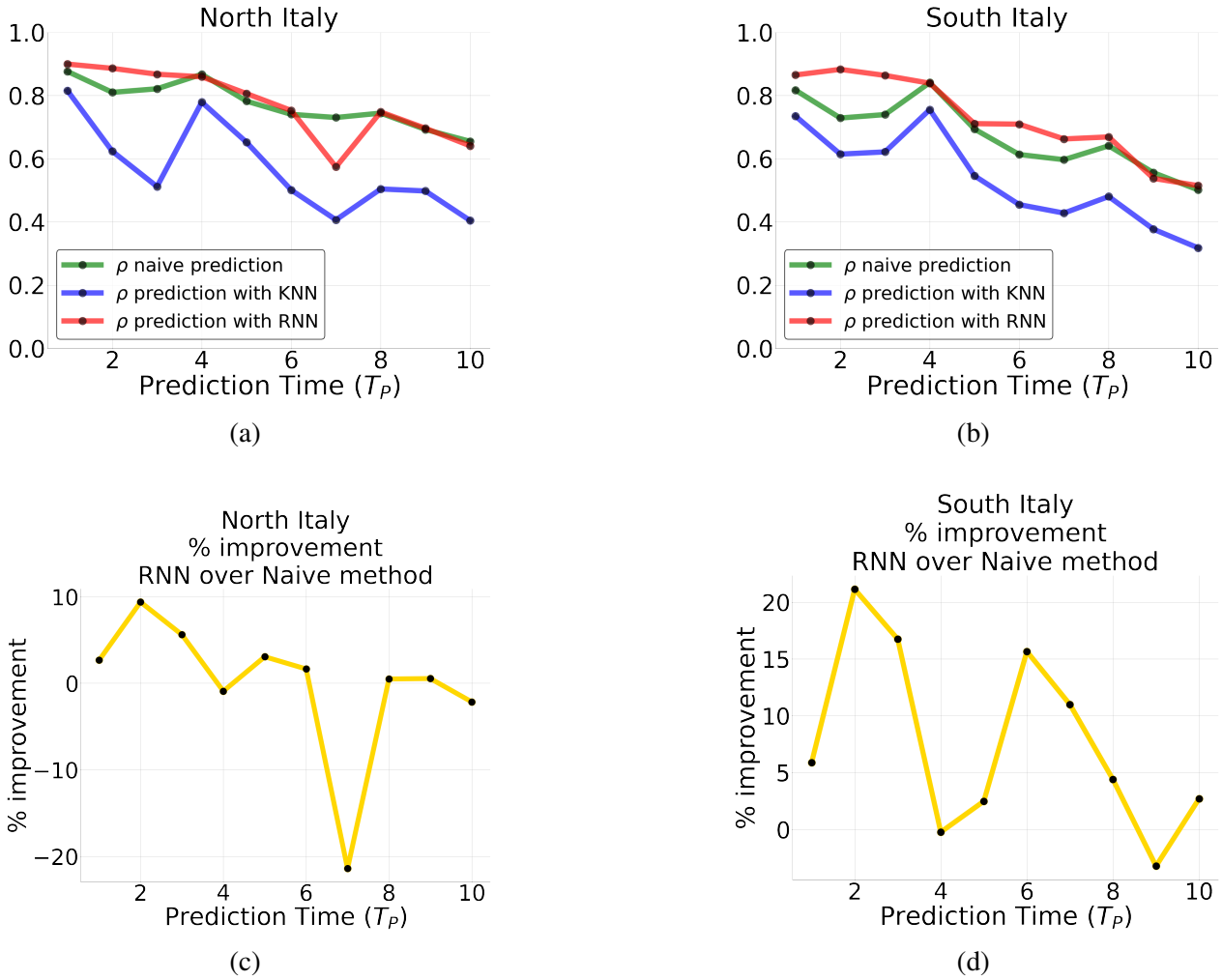


Figure 4.8: In these figures are compared the correlation coefficients between the actual and predicted time series calculated with the algorithm of KNN, with a RNN and with the naive method. The analysis shows that the KNN's accuracy is always dominated by the one of the other two methods. For almost every T_P the RNN displays the best accuracy among all the methods tested. Moreover, unlike the K-nearest neighbors and simplex projection, the RNN's accuracy does not follow the same trend of the autocorrelation of the series: it is overall decreasing for every T_P.

In the program has been used the classes provided by the Python's library Tensorflow, and we chose to use the same architecture explained in section 3.4.
In figure 4.8 we can see the difference between the correlation coefficients for each method used. We omitted the SP's accuracy because, as said in section 3.8, is similar to the KNN's one. To implement the correlation coefficient, we used the same dataset of 3000 values, overall the RNN performs better than the other methods and, moreover, its accuracy is higher than the naive method's one.

4.4 Conclusions

1. The Southern time series of energy's unbalances is more autocorrelated than the Northern one.

2. The predictions in the South Italy made using the KNN/SP are more accurate than the ones made in the North Italy. We hypothesize that this is caused by the point 1.

3. The predictions of the energy's unbalances made using the KNN/SP are not satisfying.

4. The predictions of the energy's unbalances made using the RNN displayed the best accuracy among all the methods used in this thesis.

Bibliography

[1] niekas (https://math.stackexchange.com/users/178268/niekas). *How to check if point $x \in \mathbb{R}^n$ is in a n-simplex?* Mathematics Stack Exchange. URL:https://math.stackexchange.com/q/1226825 (version: 2015-04-09). eprint: https://math.stackexchange.com/q/1226825. URL: https://math.stackexchange.com/q/1226825.

[2] Roy M Anderson and Robert M May. "Population biology of infectious diseases: Part I". In: *Nature* 280.5721 (1979), pp. 361–367.

[3] *Artificial Intelligence and Machine Learning: Policy Paper*. URL: https://www.internetsociety.org/resources/doc/2017/artificial-intelligence-and-machine-learning-policy-paper/?gclid=Cj0KCQjwnqH7BRDdARIsACTSAdvUEPsOTwl98dBUOPZypmqPMAysh_yKf3uZf-GlFJ7c02zMthJZ_6MaAho_EALw_wcB.

[4] June Barrow-Green. *Poincaré and the three body problem*. 11. American Mathematical Soc., 1997.

[5] Geoff Boeing. *Chaos Theory and the Logistic Map*. URL: https://geoffboeing.com/2015/03/chaos-theory-logistic-map/.

[6] Geoff Boeing. *Visualizing Chaos and Randomness*. URL: https://geoffboeing.com/2015/04/visualizing-chaos-and-randomness/.

[7] Larry D. Bradley. *Chaos & Fractals*. URL: https://www.stsci.edu/~lbradley/seminar/fractals.html.

[8] Junyoung Chung et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint arXiv:1412.3555* (2014).

[9] Karen D. Colins. *Cayley-Menger Determinant.* MathWorld–A Wolfram Web Resource. URL: https://mathworld.wolfram.com/Cayley-MengerDeterminant.html.

[10] Pierre Coullet and Charles Tresser. "Iterations d'endomorphismes et groupe de renormalisation". In: *Le Journal de Physique Colloques* 39.C5 (1978), pp. C5–25.

[11] George Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[12] Saber N Elaydi. *Discrete chaos: with applications in science and engineering*. CRC Press, 2007.

[13] MJ Feigenbaum. "Universality in complex discrete dynamics". In: *Los Alamos Theoretical Division Annual Report* 1976 (1975).

[14] Davide Emilio Galli. *Numerical Simulation Laboratory. Lecture 11, Machine Learning*. Nov. 2019.

[15] L. Glass and M. Mackey. "Mackey-Glass equation". In: *Scholarpedia* 5.3 (2010). revision #186443, p. 6908. DOI: 10.4249/scholarpedia.6908.

[16] P. Grassberger. "Grassberger-Procaccia algorithm". In: *Scholarpedia* 2.5 (2007). revision #91330, p. 3043. DOI: 10.4249/scholarpedia.3043.

[17] Peter Grassberger and Itamar Procaccia. "Characterization of strange attractors". In: *Physical review letters* 50.5 (1983), p. 346.

[18] Peter Gritzmann, Victor Klee and David Larman. "Largestj-simplices inn-polytopes". In: *Discrete & Computational Geometry* 13.3-4 (1995), pp. 477–515.

[19] Jacques Hadamard. "Les surfaces à courbures opposées et leurs lignes géodésique". In: *J. Math. pures appl.* 4 (1898), pp. 27–73.

[20] Onel Harrison. *Machine Learning Basics with the K-Nearest Neighbors Algorithm*. URL: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761.

[21] Suleka Helmini. *All you need to know about RNNs*. URL: https://towardsdatascience.com/all-you-need-to-know-about-rnns-e514f0b00c7c.

[22] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[23] Kurt Hornik. "Approximation capabilities of multilayer feedforward networks". In: *Neural networks* 4.2 (1991), pp. 251–257.

[24] Rudolf E. Kálmán. "A New Approach to Linear Filtering and Prediction Problems". In: *Journal of Basic Engineering* 82 (1960), pp. 35–45.

[25] S. Kolyada and L'. Snoha. "Topological transitivity". In: *Scholarpedia* 4.2 (2009). revision #149449, p. 5802. DOI: 10.4249/scholarpedia.5802.

[26] Edward N. Lorenz. "Deterministic Nonperiodic Flow". In: *Journal of the Atmospheric Planet* 20 (1963), pp. 130–141.

[27] Zhou Lu et al. "The expressive power of neural networks: A view from the width". In: *Advances in neural information processing systems* 30 (2017), pp. 6231–6239.

[28] Michael Mackey and Leon Glass. "Oscillation and chaos in physiological control systems". In: *Science* 197 (1977), pp. 287–289.

[29] David Ruelle and Floris Takens. “On the nature of turbulence”. In: *Les rencontres physiciens-mathématiciens de Strasbourg-RCP25* 12 (1971), pp. 1–44.

[30] A. Nikolayevich Sharkovsky. “Sharkovsky ordering”. In: *Scholarpedia* 3.5 (2008). revision #91762, p. 1680. DOI: 10.4249/scholarpedia.1680.

[31] Steven H. Strogatz. *Nonlinear Dynamics and Chaos*. 1994.

[32] George Sugihara and Robert M. May. “Nonlinear forecasting as a way of distinguishing chaos from measurement error in time series.” In: *Nature* 344 (1990), pp. 734–741.

[33] Floris Takens. “Detecting strange attractors in turbulence”. In: *Dynamical systems and turbulence, Warwick 1980*. Springer, 1981, pp. 366–381.

[34] *Takens' Embedding Theorem*. URL: https://cnx.org/contents/k57_M8Tw@2/Takens-Embedding-Theorem.

[35] Eric W. Weisstein. *Lyapunov Characteristic Exponent*. URL: https://mathworld.wolfram.com/LyapunovCharacteristicExponent.html.