Exploring chaotic time series and phase spaces
de Carvalho Pagliosa, Lucas

IMPORTANT NOTE: You are advised to consult the publisher's version (publisher's PDF) if you wish to cite from it. Please check the document version below.

Document Version
Publisher's PDF, also known as Version of record

Publication date:
2020

Link to publication in University of Groningen/UMCG research database

Citation for published version (APA):

Copyright
Other than for strictly personal use, it is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license (like Creative Commons).

Take-down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from the University of Groningen/UMCG research database (Pure): http://www.rug.nl/research/portal. For technical reasons the number of authors shown on this cover page is limited to 10 maximum.

Download date: 14-02-2020
5.1 Initial Considerations

In Chapter 1, we introduced our hypothesis and subsequently refined it into five research questions (RQ1–RQ5). Chapters 2 to 4 have introduced the fundamentals of dynamical systems, benchmark datasets used for evaluating such systems and their optimal embeddings, and techniques for computing such embeddings, respectively.

In this chapter, we start exploring our first research question:

*RQ1. “Does the optimal phase space have indeed low levels of entropy?”*

To tackle RQ1, we initially considered to use (and combine) different types of entropies, measured in function of states in the phase space, to propose an optimization problem that once solved would lead to the optimal embedding (Section 5.7). However, as stated in Chapter 1 and further refined in Chapter 4, computations in the phase space are not trivial in practical scenarios, as they depend on several parameters and conditions that may drastically vary according to the attractor, such as how the space is partitioned and the considered open-ball radius for the computation of nearest neighbors, among others.

In order to bypass those limitations, we relied on the correlation the entropy has with the independence among phase states (Myers et al., 1992) — a high-entropy configuration refers to independent states (in phase space), whereas a low-entropy configuration is associated with less independent states — to tackle the problem from a different perspective. However, measuring independence of states is itself complicated. Therefore, instead of estimating the independence of phase states explicitly (which should be great for optimal embeddings), we based on the Statistical Learning Theory (SLT) (Vapnik, 1998; Luxburg and Schölkopf, 2011), a mathematical framework that supports learning with supervised classifiers, to measure it intrinsically. Within SLT, a highly-independent set of samples should realize better learning than a highly-dependent set of samples. So, at this point, we propose to estimate independence
by looking at the accuracy of a classifier trained and tested on the respective set of samples.

To this end, we test different embedding pairs and choose the one that leads to the best forecasting accuracy under the prediction horizon, computed using the Lyapunov exponents (Section 2.6.3) of its attractor. Next, we provide both theoretical and experimental results based on a cross-validation strategy as evidence of generalization, which allows us to assume that input data is independent. This is an important finding, since SLT can only be used if several assumptions about the input data hold, among which data independence is required (as detailed in Section 5.2).

The structure of this chapter is organized as follows. In Section 5.2, we introduce the Statistical Learning Theory (SLT). Section 5.3 connects SLT with our domain of interest, i.e., dynamical systems. Section 5.4 discusses the issue of data independency. Section 5.5 details on how to forecast time series. Section 5.7 reports experiments that demonstrate how our proposal works. Finally, Section 5.8 concludes this chapter.

5.2 Statistical Learning Theory

The Statistical Learning Theory (SLT) provides the theoretical foundation to enable machine learning in supervised scenarios (Luxburg and Schölkopf, 2011). This framework considers an input space \( \mathcal{X} \) and an output space \( \mathcal{Y} \) in which every \( x_i \in \mathcal{X} \) corresponds to a data sample (or feature vector) and \( y_i \in \mathcal{Y} \) is its expected class or label. For simplicity, we next consider that \( x_i \in \mathbb{R}, y_i \in \mathbb{R} \). SLT assumes a joint probability distribution \( P(\mathcal{X} \times \mathcal{Y}) \) for all possible combinations of input examples in \( \mathcal{X} \) and classes in \( \mathcal{Y} \). In this context, learning is defined as the process of finding a classifier \( f : \mathcal{X} \rightarrow \mathcal{Y} \) that provides the minimum risk (error) as possible. The best classifier corresponds to the function that best represents the joint probability distribution \( P(\mathcal{X} \times \mathcal{Y}) \).

From this perspective, supervised learning requires the definition of a loss function to measure the risk of a classifier \( f \in \mathcal{F} \), in which \( \mathcal{F} \) is the algorithm bias, i.e., the set of all functions used by the supervised learning algorithm to represent every possible classifier. For example, the bias of Naïve Bayes (Raschka, 2014) is composed of linear and orthogonal hyperplanes, while the Multilayer Perceptron (Haykin, 2009) may have a more complex bias consisting of hyperplanes not necessarily orthogonal to the feature-space dimensions.
The two most common functions used in supervised learning are the \(0-1\) loss function and the squared-error function, defined respectively as

\[
\ell(x_i, y_i, f(x_i)) = \begin{cases} 
1 & \text{iff } f(x_i) \neq y_i, \\
0 & \text{otherwise,}
\end{cases}
\]  
(5.1)

and

\[
\ell(x_i, y_i, f(x_i)) = (y_i - f(x_i))^2.
\]  
(5.2)

Then, learning algorithms typically adapt the classifier by changing its parameters to minimize such losses, which in practice often corresponds to shifting the decision hyperplanes following the gradient of the loss function (Bergstra et al., 2011).

However, can one guarantee that an algorithm that performs perfectly on training data, \textit{i.e.}, for which the training error is equal to zero, performs similarly on \textit{unseen} examples? SLT aims to answer this question. To this end, Vapnik (1998) proves several properties to ensure the generalization of supervised learning algorithms. These, however, only hold if the following assumptions are respected:

\begin{itemize}
  \item[A1.] examples must be independent from each other and sampled in an identical manner;
  \item[A2.] no assumption is made about the joint probability distribution \(P(X \times Y)\), otherwise one could simply estimate its parameters;
  \item[A3.] labels \(y \in Y\) can assume nondeterministic values due to noise and class overlapping;
  \item[A4.] the joint probability distribution is fixed over time;
  \item[A5.] the joint probability distribution is unknown at training time; it must be estimated using training examples.
\end{itemize}

Assumptions A1 and A4 are mandatory given Vapnik decided to use the Law of Large Numbers to build up the Empirical Risk Minimization Principle (ERMP), which ensures that

\[
P(|R(f) - R_{\text{emp}}(f)| > \epsilon) \to 0, \ n \to \infty, \tag{5.3}
\]

where \(| \cdot |\) is the absolute-value norm. The remaining assumption A5 gives general purpose to his analysis. As result, Equation 5.3 states that the empirical risk \(R_{\text{emp}}(f) \in [0, 1]\) of a classifier converges, probabilistically, to the real risk \(R(f) \in [0, 1]\) (also known
as expected risk or, simply, risk) as the sample size \( n \) goes to infinity. In this context, the empirical risk corresponds to the training error (computed using the training set), defined as

\[
R_{\text{emp}}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(x_i, y_i, f(x_i)),
\]

while the (expected) risk is defined as

\[
R(f) = E(\ell(X, Y, f(X))),
\]

with \( E(\cdot) \) as the expected value.

The motivation for Equation 5.3 came from the definition of generalization as \( |R(f) - R_{\text{emp}}(f)| \), which means a classifier \( f \) is expected to provide a similar risk (error) over training data and unseen examples, otherwise it overfits and, therefore, it does not model the target problem. Note that a classifier with good generalization is not necessarily the one that provides a low risk but the one in which the empirical risk (training error) is a good estimator for the expected risk (over all possible inputs, including unseen ones). Finally, learning only happens when \( f \) generalizes and its empirical risk \( R_{\text{emp}}(f) \) is small enough according to some problem-specific criterion.

Vapnik observed the need to define an upper limit for such probability in order to ensure learning for any supervised algorithm. He proved the following bound to be valid

\[
P(\sup_{f \in \mathcal{F}} |R(f) - R_{\text{emp}}(f)| > \epsilon) \leq 2N(\mathcal{F}, 2n) e^{-n\epsilon^2/4},
\]

which considers the worst-to-best convergence scenario as the sample size \( n \to \infty \), if and only if the algorithm bias \( \mathcal{F} \) is characterized by a polynomial shattering coefficient \( N(\mathcal{F}, 2n) \). This coefficient, or function, must grow polynomially, otherwise learning is not guaranteed according to ERMP (Equation 5.3). This is one of the most important formal steps for machine learning, as it ensures the learning conditions for supervised algorithms.

Following the above theoretical foundation, Equation 5.6 can be rewritten as

\[
P(\sup_{f \in \mathcal{F}} |R(f) - R_{\text{emp}}(f)| > \epsilon) \leq \delta,
\]

where \( \delta = 2N(\mathcal{F}, 2n) e^{-n\epsilon^2/4} \), thereby defining the probability \( P \) for which both risks do not diverge more than \( \epsilon \) measurement units, \( i.e., \) the empirical risk approximates the expected one. Solving the above for \( \epsilon \) yields

\[
\epsilon = \sqrt{\frac{4}{n} (\log N(\mathcal{F}, 2n) - \log \delta)}.
\]
This gives the divergence between the empirical and the expected risks as

\[ R(f) \leq R_{\text{emp}}(f) + \epsilon, \]  

(5.9)

which is also referred to as the Generalization Bound. The main goal of any supervised learning algorithm is to obtain the smallest combination of empirical risk and \( \epsilon \), so that the expected risk will also be small and one will obtain the optimal classification performance.

5.3 CONNECTING SLT AND DYNAMICAL SYSTEMS

According to Bayes’ theorem (Raschka, 2014), every data sample \( x \in \mathcal{X} \) must have meaningful attribute values to estimate the output class \( y \in \mathcal{Y} \). Hence, input attributes must present some dependency level on the expected outcome (Bishop, 2006). In our context, such dependence is revealed after applying a kernel function (e.g., Takens’ embedding theorem) on the time series to unfold the phase space. In this setting, states intrinsically represent the temporal dependencies of observations, but they do not depend on each other\(^1\). More precisely, if the phase space is reconstructed using the embedding pair \( (m, \tau) \), then for each state \( \phi_i(t), \forall t \in [0, N_i] - 1 \), the pair \( \{x(t), \cdots, x(t + (m - 2)\tau)\} = x_t \in \mathcal{X}, x(t + (m - 1)\tau) = y_t \in \mathcal{Y} \) (Equation 4.18) contains such a dependency. For illustration purposes, Table 1 this dependency for the Logistic map phase space.

<table>
<thead>
<tr>
<th>State</th>
<th>( \mathcal{X} )</th>
<th>( \mathcal{Y} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( (x(t), x(t + 1)) )</td>
<td>( x(t) )</td>
<td>( x(t + 1) )</td>
</tr>
<tr>
<td>( (x(t + 1), x(t + 2)) )</td>
<td>( x(t + 1) )</td>
<td>( x(t + 2) )</td>
</tr>
<tr>
<td>( (x(t + 2), x(t + 3)) )</td>
<td>( x(t + 2) )</td>
<td>( x(t + 3) )</td>
</tr>
<tr>
<td>( (x(t + 3), x(t + 4)) )</td>
<td>( x(t + 3) )</td>
<td>( x(t + 4) )</td>
</tr>
<tr>
<td>( (x(t + 4), x(t + 5)) )</td>
<td>( x(t + 4) )</td>
<td>( x(t + 5) )</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

The important consequence of the above is that one can uniformly sample phase states in any order to infer some classification

---

\(^1\) We remind the reader that deterministic systems define a unique state in the future.
or regression model, a process which can be seen as a selection of points under the same probability distribution (Carlsson and Mémoli, 2013). Moreover, if we have enough states to represent the attractor, we also ensure they are identically distributed into its structure, as illustrated in Figure 5.1).

![Figure 5.1: Uniform sampling of states in the Logistic map phase space. Circles illustrate regions respective states (in the center of each circle) were sampled from. Adapted from (Pagliosa and de Mello, 2017).](image-url)

In the above context, assumptions A1 and A4 of SLT can be satisfied after applying such a kernel function, once phase states are now seen as i.i.d. data inputs. Assumptions A2, A3, and A5 are straightforwardly fulfilled. If we assume we are dealing with samples coming from a single phenomenon, the joint probability distribution is time-independent, which satisfies A4. Chapter 7 addresses separately the case of concept drift scenarios (Gama et al., 2014) in which time-series samples may derive from different phenomena (with distinct joint probability functions).

Since we now have satisfied all prerequisites needed to treat our problem in a machine learning setting, we can perform regressions on the reconstructed phase space. For this, we use a simple Distance-Weighted Nearest Neighbors (DWNN) supervised learning algorithm, which builds a classifier $f : \mathcal{X} \rightarrow \mathcal{Y}$ in form

$$f(\mathbf{x}_{\text{new}}) = \frac{\sum_{t=0}^{N_t-1} w_t y_t}{\sum_{t=0}^{N_t-1} w_t},$$

(5.10)

in which $w_t$ is the Gaussian-based distance between the new state $\mathbf{x}_{\text{new}}$ and the known sample $\mathbf{x}_t$, defined as

$$w_t = e^{-|\mathbf{x}_{\text{new}} - \mathbf{x}_t|^2 / 2\sigma^2}, \quad \forall t,$$

(5.11)

where $\sigma$ is the dispersion of the radial-basis activation function. This function produces values in $[0, 1]$, having 1 when $\mathbf{x}_t$ is at the same space location as $\mathbf{x}_{\text{new}}$ and 0 when they are very far from
each other, \( i.e., \sigma \) is too small to include \( x_t \) in the neighborhood of \( x_{\text{new}} \).

We next analyze the behavior of DWNN with respect to the value of \( \sigma \). By setting \( \sigma \to 0^+ \), we have the smallest open ball \((\text{Mendelson, 1975})\) around the new state that contains no other neighbor but itself, so \text{Equation 5.11} \ yields 0 for all examples. Consequently, we have that

\[
f(x_{\text{new}}) = \frac{\sum_{t=0}^{N_i-1} w_t y_t}{\sum_{t=0}^{N_i-1} w_t} = 0, \tag{5.12}
\]

which characterizes an undefined situation. Hence, there is no learning guarantee in this scenario. This is the situation in which the classifier learns to handle only the training samples \((\text{Luxburg and Schölkopf, 2011})\), or, in other words, it memorizes outputs and only reports correct label when \( x_{\text{new}} = x_t \), for all \( t \). This classifier would never generalize to further unseen data, which represents a typical case of overfitting.

On the other hand, when \( \sigma \to +\infty \), \text{Equation 5.11} tends to 1 for all samples, we have

\[
f(x_{\text{new}}) = \frac{\sum_{t=0}^{N_i-1} w_t y_t}{\sum_{t=0}^{N_i-1} w_t} = \frac{\sum_{t=0}^{N_i-1} 1}{\sum_{t=0}^{N_i-1} 1} \approx \mu_{y_t \in Y}, \tag{5.13}
\]

so our classifier \( f \) always outputs the average class referred to as \( \mu_{y_t \in Y} \). This is the situation where the classifier underfits data, leading to poor regressions. For the sake of illustration, given a binary classifier, this is equivalent to flipping a coin in attempt to hit its output.

We conclude there is a need for a balance between \( \mathcal{F} = \mathcal{F}_{\text{all}} \) and \( |\mathcal{F}| = 1 \) to model a target problem adequately \((\text{both extremes are illustrated in Figure 5.2})\). This trade-off is also known as Bias-Variance Dilemma in Machine Learning literature \((\text{Geman et al., 1992; Luxburg and Schölkopf, 2011})\). For DWNN, the bias is mainly described by the value of \( \sigma \). Finding a good stationary point between under and overfitting should be in agreement with the Statistical Learning Theory \((\text{Luxburg and Schölkopf, 2011})\). Separately, we should set \( \sigma \) large enough to contain sufficient neighbor states for every unseen sample, so that the classifier is capable of extrapolating from these to the unseen points.

5.4 ON THE KERNEL FUNCTION TO DEAL WITH DATA DEPENDENCIES

Takens’ embedding theorem is a dynamical system tool to reconstruct data from the temporal space into the phase space. If this reconstruction is adequate, we assume states in the phase
supervised learning guarantees for time-dependent data

Figure 5.2: DWNN biases: (a) the space contains every possible classifier in $\mathcal{F}_{\text{all}}$ as $\sigma \to 0^+$. (b) the space contains only one classifier when $\sigma \to +\infty$.

space to be independent on each other, as they can be sampled in an independent-and-identically-distributed (i.i.d.) manner (see Figure 5.1). Then, to evaluate the independence of points in the phase space, we relied on the SLT framework as it ensures learning only when training-data samples are independent from each other.

To this end, we proceed as follows. We conducted several experiments to produce an extensive set of phase spaces, using Monte Carlo simulation (Landau and Binder, 2005) for the embedding parameters, which were evaluated in terms of their ability to build regressive functions (those behave in the same manner as discrete-based label classifiers). For each embedding, we have selected random states in the phase space (with replacement to keep data distribution unchanged, as discussed in (Luxburg and Schölkopf, 2011)) to build learning models using the DWNN supervised algorithm. We chose this method as it holds all properties of SLT and, therefore, it is guaranteed to learn when $\sigma$ does not overfit (when training samples are represented) nor underfit (when all training samples are considered simultaneously). In addition, DWNN allows to change its bias by adapting $\sigma$, and measure its influence in terms of training and test errors to confirm the occurrence of both cases.

We define the ideal independent phase space as the embedding for which DWNN learns the most while respecting the Generalization Bound (Equation 5.9). In other words, we have chosen the set of kernel function parameters — namely the embedded dimension $m$, time delay $\tau$, and radius $\sigma$ — that minimized both the empirical risk (training error) and $\epsilon$ to ensure $R_{\text{emp}}(f)$ is a good estimator of $R(f)$. After defining such embedding, we verify if this space is indeed capable of generalizing for unseen data in practical scenarios, which allows us to confirm whether over or underfitting have
happened. To prove that, we use a ghost sample as supported by the Symmetrization Lemma (Vapnik, 1998)

\[
P(\sup_{f \in \mathcal{F}} |R(f) - R_{\text{emp}}(f)| > \epsilon) \leq 2P(\sup_{f \in \mathcal{F}} |R_{\text{emp}}(f) - R'_{\text{emp}}(f)| > \epsilon/2) \leq \delta, \quad n \to \infty,
\]

having \(R'_{\text{emp}}(f)\) as the empirical risk computed for the ghost sample.

This lemma states that if two independent samples have empirical risks that do not diverge more than \(\epsilon\), then it is expected that the empirical risk is also a good estimator for the expected risk. To employ the Symmetrization Lemma, we use the recently-computed optimal parameters to embed two independent, randomly-sampled subseries from \(T_i\). Moreover, in order to be in accordance with ERMP, we simulate the growing of the input space (i.e., \(n \to \infty\)) by equally increasing the length of both sets, starting with \(10 \times H\) (\(H\) is the prediction horizon (de Mello and Yang, 2009) estimated using the Lyapunov exponent as defined in Equation 2.31) up to the total number of \(n_i/2\) observations. Finally, we compute using DWNN, for each round, the respective empirical risks for both series, representing \(R_{\text{emp}}(f)\) and \(R'_{\text{emp}}(f)\), respectively.

Thus, we validate the choice of the optimal embedding by taking the difference between both empirical risks. If this difference is small, then a small \(\epsilon\) satisfies Equation 5.14 and, based on the Generalization Bound (Equation 5.9), the expected risk \(R(f)\) is also small. Besides, if Equation 5.14 is held for most cases, the shattering coefficient \(\mathcal{N}(\mathcal{F}, 2n)\) grows polynomially along \(n\), which improves consistency for the learning algorithm (Vapnik, 1998). This aspect ensures learning, providing empirical evidence that ideal phase spaces contain independent points. This is a very important step towards supervised learning when it comes to time-dependent data.

In our experiments, we show that when the ideal (i.e., the best setting for \(m\) and \(\tau\)) phase space is found after applying Takens’ embedding theorem as kernel function, the supervised learning algorithm models unseen data with significant low errors. Those experiments considered time series produced by synthetic systems. Consequently, we know their generation rule (i.e., the equation \(R(\cdot)\) used to generate data) and the most common setting for the embedding dimension \(m\) and the time delay \(\tau\) to reconstruct phase spaces (the ideal kernel function parametrization). Therefore, we can compare other possible reconstructions against the mostly used ones and measure their generalization capabilities (prediction of unseen observations). In this manner, the analysis of synthetic data allows to take conclusions based on well-known expectations. In
addition, we also considered the Sunspots dataset (Andrews and Herzberg, 1985) to illustrate how our approach addresses a real-world scenario.

5.5 Concrete Example: Predicting Time Series

As discussed earlier in Section 5.2, the risk of a classifier $f$ for any supervised algorithm is defined using a loss function $\ell(x_i, y_i, f(x_i))$, in such a way that the best classifier achieves the smallest expected risk $R(f)$. Moreover, such a loss function should be defined accordingly to the problem goal, space, and restrictions. In the context of time-series modeling, the $0−1$ loss function (Equation 5.1) and the squared-error one (Equation 5.2) may be inadequate to compute the error (the difference between two series) since time-shifted observations can still preserve the nature of the series. To improve comparison, we used instead the Mean Distance from the Diagonal Line (MDDL) (Rios, 2013). MDDL uses Dynamic Time Warping (DTW) to ideally match expected-to-forecasted observations, which produces a path in the two-dimensional space having each series along one axis (Figure 5.3(a)). Next, MDDL calculates the average distance from each path to the diagonal line (Figure 5.3(b)), providing a final distance estimation.

![Figure 5.3: (a) DTW warping path between series. (b) MDDL computes the cumulative distances from the warping path to the diagonal line (shaded). Adapted from (Pagliosa and de Mello, 2017).](image)

An advantage of MDDL is that it summarizes the distance between series considering time displacements (such as time-series trends), improving upon both the traditional Euclidean distance and DTW. For instance, Figure 5.4 illustrates a sinusoidal function with a stochastic process defined by a Normal probability distribution $N(0, 0.72^2)$ (with 0 mean and 0.72 standard deviation$^2$) added to it, and two possible forecasted series: i) the mean-valued series,

---

2 Usually, the Normal distribution is represented by $N(\mu, \sigma^2)$. However, most of our routines in R simulating data distribution received as input the standard deviation. Thus, we decided to report, from now on, Normal distributions highlighting the standard deviation, i.e., explicitly showing it in the form $\sigma^2$. 

76
and ii) the noise-free series. When considering the DTW distance, the mean-valued series would be taken as more similar (normalized distances to (i) and (ii) are 0.31159 and 0.30894, respectively), which implies DTW does not take into account trend components (very important when analyzing time series). In contrast, MDDL would indicate the sinusoidal series as the most adequate solution: distances are 5.84667 and 170.65778 for (i) and (ii), respectively. It is fair to mention that DTW-D (ratio between the DTW and the Euclidean distance) could also have been used in this scenario.

![Figure 5.4: Two forecasted series are compared against the expected sinusoidal series with noise added to it (represented by the dashed line with sinusoidal form): the noise-free series (solid line) and the mean-valued one (straight-dashed line). Adapted from (Pagiosa and de Mello, 2017).](image)

We then consider MDDL as loss function to propose the $k$-Fold Time Cross Validation ($k$FTCV), a novel approach to validate time-series classification and forecast time series. Given a time series $T_i$ and its phase space $\Phi_i$, the algorithm splits $T_i$ into $k$ subseries as $T_i = \{S_1, S_2, \cdots, S_k\}$ (we assume $k = 10$ in our experiments), then it uses one of those subseries $S_j$ as expected data (test samples) and the remaining observations to infer a classifier (training samples), as illustrated in Figure 5.5.

To predict new observations and compare against one of the folds, we perform the following steps: i) we use DWNN to build up a regression function with 90% (because $k = 10$) of the points in phase space; ii) we predict the first instance for the remaining 10% of points ($S_j$); iii) we employ the newly predicted value to recursively forecast next observations (note that the error will accumulate along predictions); iv) we repeat step (iii) $h$ times. Due the chaotic behavior, $h$ assumes the prediction horizon $h = H$ for each series, or any constant $h \leq H$. Finally, the forecasted series is compared to the expected one using MDDL, and the average MDDL value over all folds is used as empirical risk.
Figure 5.5: Execution of 3-Fold Time Cross Validation on our system. For each fold (dashed line in (a)–(c)), the algorithm trains with the remaining series (solid line) to recursively predict $H$ observations (crossed line). Notice that by increasing the number of folds, the learning confidence also increases. Adapted from (Pagliosa and de Mello, 2017).

The best set of embedding parameters is defined as the one whose respective phase space provides the smallest risk. In other words, we select the embedding dimension $m$ and time delay $\tau$ (i.e., the kernel function parametrizations) that best unfold the phase space and transform time-series observations into i.i.d. instances. Lastly, we confirm the true risk to be also small in terms of the Symmetrization Lemma (Equation 5.14).

5.6 EXPERIMENTS

The Distance-Weighted Nearest Neighbors (DWNN) algorithm, a variation of the $k$-nearest neighbors, was used to perform experiments because it holds, for an adequate $\sigma$, the principles established by the Statistical Learning Theory (Luxburg and Schölkopf, 2011). Two different types of experiments were conducted. The first one assessed different phase-space reconstructions (kernel function parametrizations) for synthetic and real-world data, and then we proceeded by selecting the space producing the best classifier, i.e., with the greatest generalization capacity given by $|R(f) - R_{\text{emp}}(f)| \approx 0$. In the second one, we used the recently above-found kernel parametrization to quantify the accumulated
error and evaluate the generalization capacity in terms of recurrent forecasting.

Our assumption is that if:

i) time-series dependencies are intrinsically defined in terms of every data sample (phase state);

ii) data samples can be uniformly selected to compose a training set in order to infer a classification or regression model;

iii) we obtain good enough results, measured by cross-validation for training and test sets,

then we can conclude that the kernel function produced an immersion in a multidimensional space which is capable of translating observations into independent-and identically-distributed (i.i.d.) examples. As a consequence, a supervised learning algorithm, such as DWNN, is capable of modeling uniformly-sampled states from the phase space and, consequently, predict unseen observations (which are associated to class labels).

This section is structured as follows. First, we describe the setup of the entire experiment, including algorithmic choices and parameter settings (Section 5.6.1). Next, we describe the first set of experiments, aimed at selecting the optimal phase-space embedding for a given time series, from both synthetic and real-world data (Section 5.6.2). Finally, we use this optimal embedding to show how the learned model can be used to generalize, i.e., predict, or forecast, time series data (Section 5.6.3).

5.6.1 Experimental Setup

To avoid underfitting and overfitting, we set $\sigma$ as the minimal pairwise distance containing an average percentage of nearby states in state space. In order to define the best percentage, 20 values equally spaced from 0.001 to 0.2 were processed by a Monte Carlo simulation. In addition, to break eventual time dependencies, we randomized the training examples and applied the 10-Fold Time Cross Validation (Section 5.5) to compute the empirical risk. We also made sure every $\sigma$ was sufficiently large to include at least one neighbor, avoiding a memory-based, overfitting, classifier.

We penalized high embedding dimensions and time delays to simplify (in the parsimony sense of Occam’s razor (Rasmussen and Ghahramani, 2001)) as much as possible the phase space. We then selected classifiers (regression functions) by assessing the risk computed by MDDL as described in Section 5.5. We look for the best embedding while varying $m$ and $\tau$ in range $[2, 15]$ and $[1, 15]$, respectively. However, we only changed the embedding parameters
supervised learning guarantees for time-dependent data

and $\tau$ when this risk was smaller than the current (best) risk according to a threshold (here defined as $5 \times 10^{-2}$ when increasing space dimensions and $1 \times 10^{-2}$ when increasing time delays). In order to estimate a fair upper limit for the expected risk, we defined $\epsilon$ as the 20% quantile of all absolute differences of two training sets (as we increased the input length $n$) in an attempt to hold Equation 5.14 and, therefore, obtain a polynomial shattering coefficient $\mathcal{N}(\mathcal{F}, 2n)$.

We also defined a constant for the number of forecasted observations $h$, which is justified by: i) the fact that Lyapunov exponent estimators (Kantz, 1994; Rosenstein et al., 1993) might overestimate the prediction horizon; ii) when $h \leq H$, we have a higher confidence in forecasting (less accumulated errors); iii) to avoid $h > H$, and consequently lead to poor results, we defined $h = 10$ during the training stage (to estimate the best classifier), which is expected not to surpass $H$ for most cases.

The final design decision regards the selection of the Relevant Neighbors ($R$), which we explain next. When learning, DWNN requires the definition of pointwise neighborhoods in order to estimate a class for unseen examples. This process involves the weighted mean for all neighbors under the influence of a Gaussian activation function (Equation 5.11) with standard deviation $\sigma$. Thus, even after narrowing $\sigma$, distant neighbors may still have more influence than they should have. For instance, suppose a state having two nearest neighbors equally far away; DWNN will assign to this state the mean class of both neighbors, no matter how far they are. This is inadequate in our scenario in which labels are real values. In such situations, we have experimentally concluded that such a point with no relevant neighbors should be discarded.

Our experiments suggest that $R$ can be smaller for conservative, deterministic, and dense embeddings, whereas it should be greater for spread-out and noisy phase spaces. Although we have achieved good results, we reinforce that each state (or group of states) should have its (or their) own parameter(s) $R$, considering a phase space can be formed by local structures. We have relied our selection for $R$ on the fact that the Gaussian activation function can be interpreted as a Normal probability distribution, thus, by setting $R = 1.5$ (i.e., setting the distance for relative neighbors to be as far as $1.5 \times \sigma$), the closest states are considered relevant if and only if they are at most $1.5$ standard deviations from the mean (the mean is defined by the position of the unseen sample under analysis). According to the $z$-table (Schefler, 1988), this setting takes an area of 86% of data into account (Figure 5.6). Interestingly, a similar discussion appears when selecting the so-called perplexity parameter when reducing data dimensionality using the t-SNE projection algorithm (van der Maaten and Hinton, 2008). Globally put, the
selection of a “good” perplexity value (which should consider only a given range of nearest neighbors for a high-dimensional point) is conceptually similar to our discussion of selecting a “good” $\sigma$ value for our classification purposes.

Figure 5.6: Even after mitigating the influence of far states by decreasing $\sigma$ in Equation 5.11, distant neighbors (triangles) can still jeopardize the classification of unseen examples (circle) when using DWNN. In this form, only relevant neighbors (squares) should be taken into account. Adapted from (Pagliosa and de Mello, 2017).

5.6.2 Assessing Phase-Space Reconstruction

In the first set of experiments, we assessed different phase-space reconstructions for synthetic and real-world data. Next, we proceeded with the selection of the space $\Phi_i$ that produces the most generalizing classifier. Before showing our results, it is worth to remind that the discrete nature of maps like Logistic, Ikeda and Hénon imposes an ideal phase-space reconstruction. Consequently, the embedding dimension $m$ and time delay $\tau$ are well-known. On the other side, the continuous nature of systems such as Lorenz and Rössler determines different kernel function parametrizations depending on the rate of data sampling. For these systems, we use a default sampling rate of $t_s = 0.01$.

5.6.2.1 Synthetic Time Series

Table 2 summarizes all relevant results from $k$FTCV. Columns describe, from left to right: the time series $T_i$; embedding dimension $m$; time delay $\tau$; $\sigma$ used in DWNN; relevant neighbors $R$; empirical risk $R_{\text{emp}}(f)$; divergence parameter $\epsilon$; and the upper limit for the expected risk $R(f)$ (namely $R(f)^*$), computed as described in Equation 5.9 for the best-learning phase space $\Phi_i$. 

81
Table 2: Using the Statistical Learning Theory to find the best embedding parameters for the kernel function (Takens’ immersion theorem).

<table>
<thead>
<tr>
<th>$T_i$</th>
<th>$m$</th>
<th>$\tau$</th>
<th>$\sigma$</th>
<th>$R$</th>
<th>$R_{\text{emp}}(f)$</th>
<th>$\epsilon$</th>
<th>$R(f)^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistic</td>
<td>2</td>
<td>1</td>
<td>0.00318</td>
<td>1.96</td>
<td>0.13719</td>
<td>0.02734</td>
<td>0.16</td>
</tr>
<tr>
<td>Ikeda</td>
<td>3</td>
<td>1</td>
<td>0.13277</td>
<td>1.3</td>
<td>0.60625</td>
<td>0.03432</td>
<td>0.64</td>
</tr>
<tr>
<td>Hénon</td>
<td>4</td>
<td>1</td>
<td>0.07924</td>
<td>1.5</td>
<td>0.36391</td>
<td>0.04017</td>
<td>0.4</td>
</tr>
<tr>
<td>Lorenz</td>
<td>3</td>
<td>8</td>
<td>0.81707</td>
<td>4</td>
<td>1.30174</td>
<td>0.02440</td>
<td>1.32</td>
</tr>
<tr>
<td>Rössler</td>
<td>3</td>
<td>8</td>
<td>0.26946</td>
<td>1.5</td>
<td>0.44782</td>
<td>0.03017</td>
<td>0.47</td>
</tr>
<tr>
<td>Sinusoidal</td>
<td>2</td>
<td>1</td>
<td>0.00001</td>
<td>0.5</td>
<td>0</td>
<td>0.04143</td>
<td>0.04</td>
</tr>
<tr>
<td>Sunspots</td>
<td>2</td>
<td>4</td>
<td>1.52</td>
<td>1.5</td>
<td>1.15081</td>
<td>0.03577</td>
<td>1.18</td>
</tr>
</tbody>
</table>

According to Table 2, we observe that the parameters found after the phase-space assessments match the commonly used ones (ground-truth parameters) in almost all cases. For clarity, let us mention that researchers typically use $m = 2$ for the Ikeda and Hénon maps only for visualization purposes; while the real embedding dimensions for them are $m = 3$ and $m = 4$, respectively.

Moreover, it is worth mentioning that if we had proceeded without recursive forecasting, different embeddings would have been found. This is due to the fact one-step prediction creates a bias that forces the algorithm to (usually) estimate $\tau = 1$ (which is indeed enough to predict a next but not a recursive sequence of observations).

5.6.2.2 Synthetic Time Series With Noise Added

In an attempt to correlate noise components with the time delay, we have added signals produced by two Normal distributions $\mathcal{N}(0, 0.05^2)$ and $\mathcal{N}(0, 0.1^2)$ to the data produced using the sinusoidal function, whose embeddings with $m = 2$ and $\tau = 1$ are illustrated in Figure 5.7. As expected, noise wrongly correlates phase-space trajectories, leading to poor learning results. This situation is overcome when time delay $\tau$ is increased, stretching the phase space and deviating embedding points. Nonetheless, the embedding dimension $m$ is also affected, as more dimensions are necessary to stabilize false nearest neighbors (Kennel et al., 1992). As a conclusion, we state that supervised algorithms tend to find more complex embeddings when dealing with noisy observations.

Figure 5.8 illustrates the phase space with the smallest empirical risk for each case, having the pair $(m = 2, \tau = 9)$ for both cases in Figure 5.8. This scenario illustrates how even a small amount
of noise significantly changes the ideal embedding \((m = 2, \tau = 1)\) (de Mello and Yang, 2009).

Figure 5.8: In order to learn, DWNN demands a more complex phase space when analyzing noisy data. For the sinusoidal series with \(\mathcal{N}(0, 0.05^2)\) and \(\mathcal{N}(0, 0.1^2)\) noise added, the best found phase space-reconstruction parameters were \((m = 2, \tau = 9)\) in both cases. Adapted from (Pagliosa and de Mello, 2017).

5.6.2.3 Real-World Data

Let us now highlight the importance of the number of relevant neighbors \((R)\) and how this parameter influences our analysis. For real-world scenarios, the area under the Gaussian activation function (Equation 5.11), used to define the relevant neighbors, may vary depending on the phase-space behavior. Different aspects must be considered: i) if phase-space points are affected by noise, decreasing \(R\) may be the best strategy so that only closest points are considered by DWNN; however, ii) if the phase space is formed by
spread points, $R$ should be large enough to involve distant neighbors.

In addition, when predominately deterministic, dense, and continuous phase spaces are analyzed, more points could be taken as neighbors to build the classifier. Take the Logistic map as example. Its phase space is very smooth, forming a well-structured attractor. This structure allows us to set DWNN to consider more neighbors in such a space. In fact, better results were observed when we set $R = 1.96\sigma$ (i.e., covering 95% of neighbors). On the other hand, we defined $R = 4$ for the Rössler system (see Figure 5.9). Although this space has a well-defined structure, some regions must take more neighbors into account to correctly classify points due to the local point density. Therefore, we conclude that $R$ should be defined locally on every point or on attractor regions.

Figure 5.9: Rössler three-dimensional phase space. Despite its deterministic structure, regions in which points are located more apart from each other (darker-thicker lines) force greater values for $R$, so distant neighbors can be correctly involved during classification. Therefore, different values of $R$ should be assigned to different points, depending on their locations and attractor neighborhoods. Adapted from (Pagliosa and de Mello, 2017).

As we know, the Sunspot dataset was produced by a natural phenomenon, so we expect that data-measurement errors and unexpected behavior may occur. In addition, we also know this data behaves similarly to a sinusoidal function with noise added to it. As a consequence, we assumed $R = 1.5$ to form pointwise neighborhoods, which allows us to find embedding dimension $m = 2$ and time delay $\tau = 4$. We believe that this is a good approximation considering that this dataset contains sinusoidal influences (the sinusoidal function is typically embedded using $(m = 2, \tau = 1)$). As previously discussed, real-world scenarios lead to more complex phase spaces (a greater value for $\tau$, in this case), typically due to the presence of noise.
5.6.3 Evaluating The Generalization Capacity When Forecasting

We now measure the generalization capacity of the found embeddings (obtained as described in Section 5.6.2). This allows us to assess whether learning has indeed happened as desired. For this, we computed MDDL for different training samples with 60%, 70%, 80% and 90% of the total number (\(n_i\)) of observations from \(T_i\). For the sake of brevity, we only illustrate the last scenario, i.e., training with 90% of data, as the other cases yielded similar results. Figure 5.10 shows the MDDL average result for all training set sizes.

![Graphs showing MDDL results for different training samples.](image)

Figure 5.10: Forecasted (dotted lines) and expected (solid lines) observations. From (a) to (h): Logistic, Ikeda, Hénon, Lorenz, Rössler, sinusoidal, sinusoidal with \(N(0, 0.05^2)\) noise added, Sunspot. In all plots, observation values were normalized to \([-1, 1]\). Adapted from (Pagliosa and de Mello, 2017).
As expected, all results (estimators of the real risk) were smaller than the upper limit bound given by Table 2. This strongly indicates all phase spaces provided a good generalization capacity for the supervised learning algorithm and, therefore, DWNN was capable of learning. Also, it is worth mentioning that results became worse as time elapses due to the error accumulated during the recursive forecasting (something expected and measurable using the Lyapunov exponent). However, when testing with other values for parameters $m, \tau, \sigma$ and $R$, we observed generalization was jeopardized. This indicates that the method we proposed earlier in Section 5.6.2 to find optimal parameter values is indeed reliable.

5.7 Entropies and Probabilities

For completeness, let us turn back to the initial point mentioned in Section 5.1, i.e., directly estimating the entropy. As it was stated, doing this is complicated. In this section, we detail on this matter and show some of the difficulties in deterministically correlating such measurement with optimal embeddings.

Entropy can be measured in several ways. Among the most common methods, Shannon’s entropy (Hammer et al., 2000) is defined as

$$E_{sh} = - \sum_j P_j \log P_j,$$

(5.15)

where $P_j$ is a probability function. In the context of phase-space reconstruction, Equation 5.15 can be translated to

$$E_{sh}(\phi_i) = - \sum_{t=0}^{N_i-1} P(\phi_i(t)) \log P(\phi_i(t)),$$

(5.16)

where $P(\phi_i(t))$ is a probability function over each phase state $\phi_i(t)$. Such probability can also be measured in different ways. Among alternatives, we initially estimate $P(\phi_i(t))$ throughout the complex network given by the graph $G(V,E)$ (Newman, 2003), where $V$ is the set of vertices representing each state, such that $v_t = \phi(t)_i$, and $e(v_t, v_n) \in E$ is the set of edges where $v_t$ is connected to $v_n$ only if $\phi(t)_i$ is a predecessor of $\phi(k)_i$, i.e., when

$$|x(t + t_w) - x(k)| \leq \epsilon,$$

(5.17)

where $\epsilon$ is a radius factor, and $t_w = (m - 1)\tau$ is the time delay window, respectively. From a different perspective, Equation 5.17 tries to keep track on the intersections of states trajectories with the phase-space hyperdiagonal, as illustrated in Figure 1.2. Moreover, if phase states are represented in a tabular form as in Table 1,
then Equation 5.17 will hold when the last column (the $m$th component) of $\phi(t)_i$ matches the first component of $\phi(n)_i$ according to $\epsilon$. Then, the probability $P(\phi(t))$ is set as a function of the number of edges, i.e., the valency or degree of $v_t$, namely $\nu_t$. Thus, one can naively define $P(\phi(t)) = 1/\nu_t$, but other options may be considered (Buhmann and Buhmann, 2003; Delashmit and Manry, 2005).

Alternatively, one can describe the uncertainty (and, as consequence, the level of determinism) in the phase space, embedded with the pair $(m, \tau)$, in terms of Von Neumann’s entropy (Han et al., 2012), described as

$$E_{vn} (\phi_t) = -\sum_{j} \lambda_j \log \lambda_j,$$

(5.18)

where $\lambda_j$ is one of the eigenvalues of $\phi_t$. Moreover, Equation 5.18 can also be defined in function of the probability of the eigenvaules, as presented in Equation 4.7. In both cases, the entropy will be higher when the attractor fully expands in all directions, and lower when an a subset of dimensions mainly describes the phase space.

Lastly, a third possible way to compute the entropy in the phase space is to partition it into $m$-dimensional cells and count the number of occurrences (states) inside of them. In that form, the partition usually follows some criterion. For instance, the space can be recursively and equally divided into quadtrees or octrees (Sperber, 2017) according to a minimal density of states $n$ (if a cell has more states than $n$, it is split again), or using other geometrical property (Mount, 2010). A relation between the number of empty and non-empty cells can be used to formulate a probability function. Using the same example that the space was split using the density of states, Equation 5.15 could be computed for different values of $n$. This approach goes in the same direction of the correlation dimension $D_2$ (Equation 2.23), but presented from a different perspective.

The last entropy modeling approach above also motivated us to consider an entropy definition based on the conditional probability of Naive Bayes (Raschka, 2014). Given a uniform partition of the space $\Phi = \{X, Y\}$, for instance, where $X$ and $Y$ are the output and input subspaces (Equation 4.18 and Table 1), respectively, then $n^m$ $m$-dimensional cells $c_{i,j\in[1,n]}$ can be created, so that the following relation is expected to be maximal for optimal phase spaces

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} = \frac{P(Y,X)}{P(X)} = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{n_{c_{i,j}}}{n_{c_{i}}},$$

(5.19)
where $n_{c_i}$ represents the total of occurrences in the $i$th subspace of $\mathcal{X}$, and $c_{i,j}$ is number of occurrences in the intersection of such space with the $j$th subspace of $\mathcal{Y}$ (joint probability). Figure 5.11 illustrates an example of such computation for the cell $c_{1,3}$ in two scenarios. Although such approach is expected to work in theory, as the output space should be strongly dependent on the input space for deterministic embeddings, i.e., the conditional probability of $\mathcal{Y}$ to happen given $\mathcal{X}$ should be high, unfortunately our experiments led to unsatisfactory results: no consistent patterns were observed for general systems.

Figure 5.11: Conditional probability computation on the phase space for 2D (a) and 3D (b). The cell $c_{1,3}$ is highlighted in red for both cases, whereas the $i$th subspace of $\mathcal{X}$ is represented in gray. For the 3-dimensional plot, $\mathcal{X} = [X_1, X_2]$.

Summarizing, estimating the entropy is dependent on some pre-requisites, such as the radius $\epsilon$ in Equation 5.17 and how the space is partitioned. This makes computing entropy difficult. Moreover, all entropies referenced in this section struggle from two additional problems: they are not unique descriptors and they suffer from the curse of dimensionality (Chen, 2009). In order to mitigate the last drawback, the Singular Entropy (SE) method (cf Section 4.2.1.4) relies on the first eigenvalue (Equation 4.7).

We have next tried to use those entropies to estimate the optimal embedding. Firstly, we aimed for a phase space where the number of intersections in the cobweb plot was minimal, so that

$$H_{sh}(\phi) = \minimize (E_{sh}(\phi)).$$

However, usually the hyperdiagonal-based attractor is found for this optimization problem, since this structure has the maximum determinism among its states (Rosenstein et al., 1994).

Apart from the above, Von Neumann’s entropy could be used in an alternative way to find the most spread attractor in form

$$H_{vn}(\phi) = \maximize (E_{vn}(\phi) - \beta \|v\|_2),$$
where \( \mathbf{v} = [\alpha_1, \alpha_2, \ldots, \alpha_m] \) is a vector indicating which dimensions should be used, \( \text{i.e.} \)
\[
\alpha_i = \begin{cases} 
1 & \text{dimension } i \text{ composes the phase space,} \\
0 & \text{otherwise.} 
\end{cases} \tag{5.22}
\]
Here, \( \psi \) is a penalty factor to decrease the importance of higher dimensions in Equation 5.21. However, this process is complex to implement and compute.

Thirdly, we considered to combine Equation 5.20 to Equation 5.21 in order to find some balance between under (redundance) and overestimated (irrelevance) phase spaces, in the form
\[
H(\phi) = \maximize (H_{\psi}(\phi) - CH_{sh}(\phi)), \tag{5.23}
\]
where \( C \in [0, 1] \) is a constant weight that defines the trade-off between both quantities.

Although a solution is guaranteed to exist for Equation 5.23 (both entropies have concave characteristics (Boyd and Vandenberghe, 2004)), they are not ensured to converge to benchmark embeddings.

To test whether RQ1 holds, we computed the above entropies for well-known embeddings of the described in Chapter 3. The hypothesis was that, when optimal embeddings (as known by the literature for specific datasets) are used, then the computed entropies would be smaller when compared to other embeddings. The results of this experiment, however, proved inconsistent: we did not observe the expected correlation between minimal entropy and optimal embedding (again, the latter as given by the literature) for the analyzed datasets (Chapter 3). A similar behavior was reported for the SE algorithm (Section 4.2.1.4), where inconsistencies were found for attractors with genus more than one.

However, the fact that RQ1 was disproved in the above form does not mean that the reconstructed embeddings, estimated with these measurements, are not good enough to unfold phase spaces in some cases. Regressions on non-ideal phase spaces can still lead to adequate models and forecasting, if the estimated embedding is not too much far from the optimal. However, we have always aimed for an estimation as closest as possible to the generating rule \( R(\cdot) \), such that overestimated embeddings (mainly in terms of \( \tau \)) are not acceptable as valid. Chapter 9 will present a different way for estimating optimal embedding parameters, using deep learning, which has led to consistent results for all tested datasets.

As a final (and side) note, based on Equation 5.21, we have also considered to apply the Least Absolute Shrinkage and Selection
Operator (LASSO) (Tibshirani, 1996) to estimate the embedding pair, which models the problem as

$$b = \arg \min_{b \in \mathbb{R}^m} \frac{1}{N_i} \| y - Xb \|_2^2 + C \times \| y \|_1,$$

(5.24)

where $X$ and $y$ are matrix representations of $X$ and $Y$, respectively.

The goal is to find a vector $b$ of $m$ coefficients

$$y_j = b_0 + X_j^T b = b_0 + x_{j,1}b_1 + x_{j,2}b_2 + \cdots + x_{j,m}b_m + \epsilon_j, \forall j \in [0, N_i - 1],$$

(5.25)

so that the sum of the squares of the differences between the dependent and exploratory variables is minimized. In order to use Equation 5.24, firstly $X$ is standardized and $y$ centered. Later, states are overembedded so the penalty factor will shrink some coefficients in $b$ to zero. As consequence, we filter the most representative dimensions out by the absolute values of $b$, and estimate $\tau$ according to the lags among non-zero coefficients. However, LASSO performs a linear regression, which leads to the fact that estimations are not consistent for nonlinear systems. As a refinement of this idea, we also considered to apply a kernel function to map nonlinear phase spaces into linear attractors. The idea was to find a map that would lead states into a linear space, preserving as much as possible their topological properties, perform LASSO on this space, and then perform an inverse mapping into the found coefficients of $b$ to have an estimation with respect to the original space. Whether this approach leads to consistent results is, however, still subject to research.

5.8 FINAL CONSIDERATIONS

The goal of this chapter was to show that states are organized in an independent-and-identically manner (i.i.d.) in the optimal phase space, so that low levels of entropy are observed. To find this optimal space, we used the Statistical Learning Theory (SLT). Simply put, we construct a series of regressors, for different values of the embedding parameters $m$ and $\tau$, and choose as optimal embedding the one for which the constructed regressor generalizes best to unseen test data.

As a consequence, instead of relaxing the assumption of data independence in the context of time-series modeling as performed in other studies (Faria et al., 2013; Tan et al., 2011; Al-Khateeb et al., 2012), the usage of a kernel function to reconstruct time series into phase spaces is also important to ensure supervised learning in those scenarios. To the extent of our knowledge, there is no previous work that tackles this goal.
To prove the above statement, we show that forecasted series generated by building regression functions on phase spaces produce small empirical risks and $\epsilon$, which restricts the expected risk to small upper limits and should be enough to theoretically prove learning. However, we also show in practical experiments that good forecasting results were generated, giving us strong evidences that learning occurred. According to SLT, however, learning can only be proven if examples are independent and identically sampled (given other assumptions are already held – as is the case). This allows us to conclude that our kernel function (Takeš's embedding theorem), under the best parametrization, unfolds data observations into i.i.d. examples.

To confirm that points in ideal phase spaces are independent and identically distributed, we conducted experiments by producing an extensive set of phase spaces, using Monte Carlo simulation for the embedding parameters, which were evaluated in terms of their ability to build regressive functions. Training to predict the label of one single unseen observation at a time instant led to wrong phase-space parameters and poor prediction results. To find more reliable spaces, we therefore recursively predicted $h \leq H$ observations (given $H$ is the prediction horizon). This latter approach results in optimal phase spaces, which match the most commonly used embedding parameters from the literature (Ikeša, 1979; Hitzl, 1981; Rössler, 1976). We also noticed that comparing forecasted and expected series using the Euclidean distance could lead to wrong results (a subject previously discussed in (Rios, 2013)), as mean-valued observations could be classified as more similar to the expected series than when comparing noisy observations. To overcome such scenarios and also consider data trends, MDDL was used as loss function.

In addition, to provide a fair training using different time instants of the series, we selected the best phase space that produced the lowest accumulative according to our $k$-Fold Time Cross Validation, whose steps can be summarized as: i) forecast $h$ observations; ii) use MDDL as loss function; iii) proceed with a $k$-cross folding.

As a main contribution, this chapter provides theoretical and empirical evidences that phase spaces transform time-series observations into i.i.d. examples. Moreover, the answer to (RQ1) is, partially, positive: as low levels of entropy correlates to higher levels of independence among phase states, optimal phase spaces can (consequently) be described in terms of such measurements. However, it is still difficult to come up with a single definition (and way of computing) of entropy of an embedding, and we could not conclude that optimal embeddings (for the cases where such information was known) generate minimal entropy values over all possible embeddings.
Another consequence of this work is that supervised learning can be performed and ensured for time-dependent data. With this knowledge one can, for instance, employ supervised-learning algorithms to tackle classification or regression tasks in time series and have learning guarantees according to SLT. Other classifiers can also be used instead of DWNN. As main drawback, our approach requires at least thousands of observations to proceed with the phase-space reconstruction and obtain a meaningful attractor. However, this should not be a problem in practice, when dealing with real-world applications which already have access to large (big data) collections of measurements.

There are several directions of possible future work based on the results presented in this chapter: i) adapting the joint probability distribution so one can rely on learning guarantees to proceed with concept drift detection on data streams; ii) an adaptive strategy to define values for $\sigma$ and/or $R$ according to different attractor points or regions, by exploiting information on how points are locally spread over the phase space (Rios, 2013); iii) designing a more robust approach to penalize higher embedding dimensions as well as time delays to avoid unnecessarily complex phase spaces; and, finally, iv) using the proposed approach to forecast data from other real-world datasets from climate, biology, medicine, and other domains.