Climate tipping points: Detection and analysis of patterns using an ordinal regression approach

Final Report

Authors: P.A. Gutiérrez¹, C. Hervás-Martínez¹, F. Fernández-Navarro², I. Dicaire², A. Nikolaou², M. Pérez-Ortiz¹, J. Sánchez-Monedero¹
Affiliation: ¹University of Córdoba, ²ESA ACT

Date: 21/04/2014

Contacts:

Pedro Antonio Gutiérrez
Tel: (+34) 957218153
Fax: (+34) 957218630
e-mail: otri@uco.es pagutierrez@uco.es

Leopold Summerer (Technical Officer)
Tel: +31(0)715654192
Fax: +31(0)715658018
e-mail: act@esa.int

Ariadna ID: 13-9202
Ariadna study type: Standard
Contract Number: 4000108222/13/NL/MV

Available on the ACT website
http://www.esa.int/act
Ariadna AO/1-7415/13/NL/KML: Climate tipping points: Detection and analysis of patterns using an ordinal regression approach [13-9202]

Gutiérrez, Pedro Antonio¹, Hervás-Martínez, César¹, Fernández-Navarro, Francisco², Dicaire, Isabelle², Nikolaou, Athanasia², Pérez-Ortiz, María¹ and Sánchez-Monedero, Javier¹

¹: Department of Computer Science and Numerical Analysis, University of Córdoba, Spain, pagutierrez@uco.es, chervas@uco.es, i22perom@uco.es, jsanchezm@uco.es
²: Advanced Concepts Team, European Space Research and Technology Centre (ESTEC), European Space Agency (ESA), Noordwijk, Netherlands, isabelle.dicaire@esa.int, athanasia.nikolaou@esa.int, francisco.fernandez.navarro@esa.int, i22fenaf@uco.es
Contents

1 Introduction 3

2 Detection of early warning signals in paleoclimate data using a genetic time-series segmentation algorithm 8
  2.1 Segmentation Algorithm 10
  2.1.1 Mathematical description of the segmentation problem 10
  2.1.2 General overview of the segmentation algorithm 10
  2.1.3 Chromosome representation 11
  2.1.4 Initial population 11
  2.1.5 Segment characteristics 11
  2.1.6 Clustering: k-means Algorithm 14
  2.1.7 Fitness 14
  2.1.8 Selection and replacement processes 15
  2.1.9 Mutation Operator 15
  2.1.10 Crossover Operator 15
  2.2 Experiments 17
  2.2.1 Climate datasets 17
  2.2.2 Algorithm parameters 17
  2.3 Results 17
  2.4 Discussion 17
  2.5 Additional details about the Segmentation Algorithm 22
  2.5.1 Generation of each individual for the initial population 22
  2.5.2 Application of the k-means algorithm 22
  2.6 Additional Examples of Segmentation for the GISP2 and NGRIP datasets 23

3 Alternative fitness functions and a new evaluation method 24
  3.1 Measuring the quality of the clustering process 24
  3.2 Automatic evaluation method and experimental setting 26
  3.2.1 Experimental setting 26
  3.2.2 Automatic evaluation method 26
  3.3 Results 28

4 Time Series Forecasting by Evolutionary Recurrent Product Unit Neural Networks 30
  4.1 Models 31
  4.1.1 Short memory model: Autoregressive Product Unit Neural Network (ARPUNN) 31
4.1.2 Long memory model: Recurrent Product Unit Neural Network (RPUNN) ........................................ 32
4.2 Parameter Estimation .................................................. 34
4.3 Experiments .............................................................. 35
   4.3.1 Dataset Selected ............................................... 35
   4.3.2 Metrics Considered for Evaluation ......................... 36
   4.3.3 Algorithms Selected for Comparison Purpose ............. 36
4.4 Results ........................................................................ 37
5 Conclusions ................................................................. 39
Chapter 1

Introduction

Climate variability as it was regionally manifested during the past millenia is documented in paleoclimatic proxy data extracted from various spots around the globe. This information is, however, nothing but straightforward in the derived time series which are obscured by noise signals, error margins in time resolution that is reflected in uncertain timing of events and an unknown number of internal and external modes that interact cumulatively to form the observed behavior (Kanner et al., 2012; Crucifix, 2012). This elusive internal variability is consecutively under-represented in current climate models. Predicting future climate transitions based on their output has still a long way to go, that also passes through the challenging task of thorough analysis of the past climate transitions. In response to this challenge the development of suitable tools for statistical analysis of paleoclimate time series has seen a boost during the last years.

Certain events stand out in a paleoclimate proxy time series because of their abrupt character and discontinuity. Some of them are encountered in more than one proxy. Combined study of proxies from different spots on earth help infer the global/regional character of events encountered in the time series, or phase differences of the same event across the globe, which makes their timing precision important. The comparison between the Greenland ice cores (GISP: Dansgaard et al., 1969; Johnsen et al., 1972) and Antarctic ice cores (Tabacco et al., 1998; Schwander et al., 2001) leads to the hypothesis of the bipolar seesaw effect, which improved our understanding of the Atlantic thermohaline circulation inter-hemispheric effects and identified it as a global propagator of freshwater flux anomalies (Broecker, 1998; Stocker and Johnsen, 2003). The abrupt transitions that were detected throughout the data sets are related to the Dansgaard-Oeschger (DO) events described in Dansgaard et al. (1993) and Alley et al. (2003) as abrupt warming in Greenland followed by gradual cooling. These events are regarded as critical transition points and indicate that the climate can demonstrate threshold behaviour (Alley et al., 2003). Therefore analysing the evolution of the climatic variables in the time period preceding the transition helps in identifying possible warning signals.

The statistical tools used to extract knowledge from time series analysis have undergone considerable development during the past decade (see Livina and Lenton, 2007; Livina et al., 2011; Lenton et al., 2012; Scheffer et al., 2009; Dakos et al., 2008; Held and Kleinen, 2004; Cimatoribus et al., 2013). Driven by the
ultimate aim of predicting future transitions, the effort is concentrated in overcoming the limitations of the finite length of the time series and simultaneously reveal statistical parameters that unfold the system’s global properties. Focusing on a single transition event, identifying the slowing down as an early warning signal (EWS) before the Younger Dryas period draw a lot of attention in 2008 in the work of Dakos et al. (2008). Discussion was raised on what the precursors of a bifurcation point are, naming increase in autocorrelation as a prominent indicator. Ditlevsen and Johnsen (2010) brought forward that respecting the fluctuation-dissipation theorem (Kubo, 1966) imposes both increasing autocorrelation and variance before crossing a bifurcation. The fluctuation-dissipation theorem is in the centre of the theoretical analysis of the statistical findings, as it relates the response of the system to external perturbations to its fluctuations when in thermal equilibrium (Palmer and Weisheimer, 2011). Based on the same theorem, Cimatoribus et al. (2013) suggested a shortcoming of the previous statement based on the fact that the climate system is not in thermal equilibrium. Preprocessing or filtering the raw proxy data in order to achieve stationarity and infer more straightforward diagnostics is suggested, such as the Detrended Fluctuation Analysis (DFA) coefficient and the modified DFA coefficient. Livina and Lenton (2007) measured the proximity of a system to a tipping point not in units of critical parameter difference, but by monitoring the evolution of the DFA propagator. Held and Kleinen (2004) proposed the method of degenerate fingerprinting and hypothesised the system as an auto-regressive process with lag-1 autocorrelation, including only exponentially decaying modes and white noise in the dynamics of the system. Livina et al. (2011) applied tools of potential analysis to describe the location and stability of distinct global states of a system from climate time series data. The above studies involve an implicit association of the performed analysis to a selected transition point. It makes sense to focus on certain abrupt events which are seen in the data series with bare eye, in terms of comprehensive analysis, but certain algorithms can also be used for a quantitative description of the time series. Rahmstorf (2003) tried to automatise the characterisation of the DO events by introducing an event detection algorithm based on the data slope. The author also proposed estimating systematic and dating errors and suggested that the DO events are very likely synchronised to an external forcing of Bond cycle periodicity (Bond et al., 1992). Cimatoribus et al. (2013) went a step forward in the use of algorithms employing bootstraping, an advanced bootstrapped ensemble method to estimate the probability error in detecting DO events within the Pleistocene ensemble of DO events.

While debate on the statistical parameters suitable for detecting a transition is still vivid, attributing DO events to one among three types of transitions as defined in Ashwin et al. (2012) is a requirement of direct relevance to the seek of EWSs. The hypotheses on the causality of the DO events include noise-induced transitions, relaxation oscillations and excitability of the system of which the stochastic resonance is a subcase (Crucifix, 2012). Each case stems from a different simple mathematical model which demonstrates complex behaviour. The patterns preceding a transition can include EWSs which can be diagnosed and studied in the controlled environment of simulated data. Simulated time series are used to complement the findings from the finite time series analysis, since in lack of long and regular observational time series of climate variables, the study of the statistical properties is hindered. Concepts drawn from various dy-
namical systems are merged into EMICs (Models of Intermediate Complexity) (Ganopolski and Rahmstorf, 2001; Stocker and Johnsen, 2003; Claussen et al., 2003; Arzel et al., 2012) in order to produce evolution of variables in thousands of model years and test hypotheses for the forcings and processes that possibly shaped the proxy time series. Since the simplified forcings are in direct correspondence with an idealised mathematical model, hypotheses of underlying mechanisms can be verified or rejected by additional comparison of simulated data behaviour to the proxy data observable. In this reverse engineering approach more than a single model can reproduce the variability encountered in the ice core or any other paleoclimatic record, so interpretation of dynamical systems should be used with caution and in combination with scientific insight (Crucifix, 2012). Returning to the DO and Younger Dryas abrupt changes encountered in the ice core records, grouping them according to the behaviour preceding them is only advantageous for reinforcing or weakening the existing hypotheses on their incidence.

This Ariadna study proposes a different approach to address the problem of EWS detection in time series. We supply no prior knowledge of tipping point to the algorithm and employ a segmentation method for classifying the different kinds of segments presented in the time series. The goal of time-series segmentation is to provide a more compact representation of time series data through dividing time series data into segments and using a high level representation to approximate each segment. The time-series segmentation problem has been widely studied within various disciplines. For example, time-series segmentation algorithms have been successfully applied in phoneme recognition (Xiong et al., 1994; Prandom et al., 1997), paleoecological problems (Bennett, 1996), telecommunication applications (Himberg et al., 2001) or financial problems (Tseng et al., 2009). For an excellent review of time-series segmentation see Keogh et al. (2001).

On the other hand, climate time series can also be modelled by using predictive models. The term Time Series (TS) refers to a succession of data values chronologically sorted that belongs to a magnitude or phenomenon that has been sampled at a certain rate. An example of time series could be the evolution of the maximum daily temperature, the unemployment rate of a country or the amplitude of the seismic waves of an earthquake. Time series are present in most of the science fields like econometrics (Gonzalo and Ng, 2001), weather forecasting (Arroyo and Mate, 2009) or control engineering (Lee and Davier, 2013). Nowadays, TS research concerns about TS Analysis (TSA) and TS Forecasting (TSF). The goal of TSA is to extract the main features and characteristics that describe the underlying phenomena, while the objective of TSF is to find a function to predict the next value of the time series using its p lagged values.

Artificial Neural Networks (ANNs) are a very popular Machine Learning (ML) tool used for TSF (Hansen and Nelson, 1997). In the field of TSF, there are different ANN architectures. Feedforward Neural Networks (FFNNs) are the most common and simplest type of ANNs, where the information moves in a forward direction (Johansson et al., 1991; Ahmed and Raul, 1991). For example, the Time Delay Neural Network (TDNN) consists on a FFNN whose inputs are the delayed values of the TS (Sitte and Sitte, 2000). Instead, Recurrent Neural Networks (RNN) are based on a different architecture where the information moves forming a direct cycle (Connor et al., 1994). This
cycle can storage information from previous data in the internal memory of the network, which be useful for certain kind of applications. One example of RNN is the Long Short Term Memory Neural Network (LSTMNN) \cite{Hochreiter1997}, whose main characteristic is the capability of its nodes to remember a time series value for an arbitrary length of time. Robot control and real time recognition \cite{Baccouche2011} are example of real applications of LSTMNN. Echo State Networks (ESNs) are RNNs whose architecture includes a random number of neurons whose interconnections are also randomly decided. This provides the network with a long term memory and a competitive generalisation performance \cite{Jaeger2002, Gallicchio2011, Rodan2011}. From the previous introduction to ANNs in TSF, it can be derived that one of the main differences between FFNNs and RNNs lies on their storage capacity. RNNs have a long term memory because of the architecture of the model, whereas the memory of FFNNs is provided by the lagged terms at the input of the network.

The parameter estimation algorithm is also very important when analyzing the different algorithms. The more complex the structure of a neural network is, the more challenging its weight matrix estimation turns. Traditional Backpropagation (BP) algorithms can result in a very high computational cost, specially when dealing with complex nonlinear error surfaces. The Extreme Learning Machine (ELM) is an example of an algorithm that can estimate the parameters of a FFNN model efficiently \cite{Pan2009}. It is a highly implemented algorithm that determines the hidden layer parameters randomly and the output layer ones by using the Moore-Penrose (MP) generalised inverse \cite{Huang2012}, providing a better generalisation performance than traditional gradient-based learning algorithms for some problems. As will be analysed later in this document, this Ariadna study also contributes a new algorithm for TSF.

The results of this Ariadna study have been included in two journal publications and one international conference contribution:


This study report is organised in five chapters: after this Introduction, a new algorithm for time series segmentation is presented in Chapter 2, analysing its performance for EWS detection. Chapter 3 is devoted to the study of alternative
fitness functions and evaluation methods for the considered algorithm. Then, Chapter 4 presents a new modelling technique for time series. Finally, Chapter 5 summarises the contributions of the study.
Chapter 2

Detection of early warning signals in paleoclimate data using a genetic time-series segmentation algorithm

This chapter presents the main characteristics of the segmentation algorithm presented in this study. The algorithm is presented and its results are analysed for the different datasets considered.

As previously discussed, prior to EWS detection, this study introduces a segmentation method as a first step to better understand the time series. This segmentation provides a more compact representation of the time series through splitting it into segments with similar behaviour Keogh et al. (2001). A segmentation analysis avoids the necessity of specifying predefined sliding windows for the different TPs, which is one of the main difficulties of previous TP detection methods Dakos et al. (2012). Moreover, the segmentation algorithm is able to detect differences between the TPs. We address the segmentation problem as a heuristic search problem with the proposal of a Genetic Algorithm (GA) to overcome the limitations of traditional statistical methods. The GA segments the data trying to obtain diverse clusters of segments based on six statistical properties.

The segmentation problem is usually converted into an optimisation problem that could be addressed using local or global algorithms (like GAs). For example, several GA-based approaches to segment time-series were proposed in Chung et al. (2004). In a similar way, Tseng et al. (2009) also proposed a GA to address the segmentation of the time series. The main novelty of this last approach is the inclusion of a clustering technique within the optimisation procedure to assign a class label to each segment.

In this study, a time-series segmentation algorithm is proposed by combining a clustering technique and a GA to automatically find the proper segmentation points and segment classes of a climate time series with abrupt changes. Interest in GAs applied to climate tipping points is rising, e.g. Lenton et al. (2009) used a GA to tune 12 physical parameters of an Earth System Model to study
the tipping of the Atlantic thermohaline circulation following a multi-objective optimisation method. The time-series segmentation algorithm presented in this chapter is a significant extension of the one in Tseng et al. (2009). The final goal of the proposed GA is to minimise the distance of each segment to its centroid in a six-dimensional space where the six dimensions are statistical properties of each segment. The proposed approach first groups the segments into $k$ clusters according to their statistical characteristics by using the $k$-means clustering technique (MacQueen et al., 1967). The Euclidean distance is used to calculate the distance of each segment with respect to its centroid. Because two segments may have different lengths and characteristics, six statistical metrics are measured for each segment such that the distance can be calculated and the clustering technique can be applied using this six-dimensional space. The algorithm is specifically adapted to time-series with abrupt changes.

The proposed approach features the following characteristics:

- Assigns a class label to the different segments via the combination of the GA with the clustering technique; traditional approaches would only provide the segmentation points (Sclove, 1983; Himberg et al., 2001; Keogh et al., 2001). This is specially useful for finding common pattern analysis from climate data.

- As mentioned in the previous point, the focus in the present study is not just on the determination of the cut points ($t_i, i = 1, \ldots, m - 1$). Rather, the idea underlying the development here is that of transitions between classes. As stated previously, the labels $C_j, j = 1, \ldots, k$ are categorised using six statistical measures. The analysis of this transitions is crucial to the detection of EWSs.

- The algorithm considers all segments of the paleoclimatic record and attempts to find common characteristics within the different segments. This approach is not unlike that of Cimatoribus et al. (2013), who considered the average behaviour of the ensemble of DO events with the added bonus that our approach can also analyze each transition and their underlying statistical properties by applying a label to each climate segment.

- Each segment is represented by a six-dimensional vector where the dimensions are several statistical metrics some of which have been previously considered in detecting EWSs of critical transitions by various authors (Cimatoribus et al., 2013; Dakos et al., 2008, 2012).

- Instead of representing the time series evolution by plotting one of its metrics, the approach proposed in this chapter allows to visualise several metrics simultaneously and to compare several sections of the time series to find common patterns.

This chapter is organised as follows. Section 2.1 introduces the segmentation algorithm with a detailed description of the embedded genetic algorithm, the six statistical metrics and the clustering process. Section 2.2 presents the paleoclimate datasets used in this study and the algorithm parameters. Section 2.3 presents the main results of the segmentation algorithm including a detailed analysis of the statistical metrics preceding DO events. Finally, Section 2.4 discusses the results from the point of view of the stochastic resonance model and possible limitations of the algorithm.
2.1 Segmentation Algorithm

2.1.1 Mathematical description of the segmentation problem

The problem of time-series segmentation considered here is the following: Given a time-series \( Y = \{ y_n \}_{n=1}^N \), partition the set of values of \( y_n \) into \( m \) segments within which the behavior of \( y_n \) is homogeneous. The segmentation algorithm should provide a partition of the time index set \( (n = 1, \ldots, N) \) into subsets:

\[ S_1 = \{ y_1, \ldots, y_{t_1} \}, S_2 = \{ y_{t_1+1}, \ldots, y_{t_2} \}, \ldots, S_m = \{ y_{t_{m-1}+1}, \ldots, y_N \} \],

where \( t \)'s are the cut points and are subscripted in ascending order \( (t_1 < t_2 < \cdots < t_{m-1}) \). Each subset \( S_l, l = 1, \ldots, m \) is a segment. The integer \( m \) and the cut points \( t_i, i = 1, \ldots, m-1 \), have to be determined automatically by the algorithm. Formally, the segmentation problem is a special case of the general clustering problem.

Furthermore, the segments considered in this study are grouped into \( k \) different classes \( (k < m) \), where \( k \) is a parameter defined by the user. Therefore, each \( S_l \) segment has associated a class label: \( (S_1, C_1), (S_2, C_2), \ldots, (S_m, C_m) \), where \( C_l, l = 1, \ldots, m \), is the class label of the \( l \)-th segment. The class label of each segment, \( C_l \), has \( k \) possible values.

2.1.2 General overview of the segmentation algorithm

This chapter proposes a novel Genetic Algorithm (GA) from the field of time series segmentation (see Sclove, 1983; Himberg et al., 2001; Keogh et al., 2001; Chung et al., 2004). The general objective of the GA is to identify segments with common characteristics by applying a label to these segments. In practice this means finding the cut points of the time-series defining the different segments to be discovered in the time-series together with the class labelling of these segments. As in traditional GAs, the proposed approach considers a population of candidate solutions (representing different possible segmentations) which are evolved towards better segmentation solutions. Each possible segmentation is represented in an array of integer values (chromosome representation) which can be mutated and recombined. The evolution starts from a population of randomly generated segmentations. After that, every segment in every chromosome is categorised using six statistical metrics. It is important to point out that most of these six statistical metrics were previously considered in the climate community (variance, autocorrelation, skewness, etc.). The clustering technique is applied over this six-dimensional space for every chromosome and a fitness value is assigned to every chromosome according to the degree of homogeneity of the segments with respect to their centroids. The class label is assigned during the clustering process. After that, different mutation and crossover operators are applied to explore and exploit the search space. This procedure is repeated during \( n \) generations. The main steps of the proposed algorithm are summarised in Figure 2.1.

The different characteristics of the GA are defined in the following subsections. Further information of the algorithmic flow of the GA proposed is included in Appendix 2.5.
**Time series segmentation:**
**Input:** Time series.
**Output:** Best segmentation of the time series.

1. Generate a random population of $t$ time series segmentations.
2. Evaluate all segmentations of the initial population by using the fitness function.
3. **while not** Stop Condition **do**
4. Store a copy of the best segmentation.
5. Select parent segmentations from current population.
6. Generate offspring: apply crossover and mutation to construct new candidate segmentations.
7. Evaluate the fitness of the offspring segmentations.
8. Replace current population with offspring.
9. **end while**
10. **return** Best segmentation from final population

**Figure 2.1:** Main steps of the algorithm

### 2.1.3 Chromosome representation

A direct encoding of the final segmentation solution is adopted where each individual chromosome consists of an array of integer values \( \{ \text{Michalewicz} \} \) [1996]. Each position stores a cutting point of the time-series. A chromosome of \( m \) segments in the time-series is represented by \( \{ t_1, \ldots, t_{m-1} \} \), where the value \( t_i \) is the index of the \( i \)-th cutting point of the time series. In this way, the first segment is delimited by the cutting points 1 and \( t_1 \), the second by the cutting points \( t_1 \) and \( t_2 \) and so on. An example of this chromosome representation is given in Figure 2.2.

### 2.1.4 Initial population

A GA requires a population of feasible solutions to be initialised and updated during the evolutionary process. As mentioned above, each individual within a population is a possible segmentation result for the time-series considered. An initial set of chromosomes is thus generated with some constraints to form feasible segments. This initial population of \( t \) individuals is randomly generated. The number of individuals will be kept constant during the evolution. Further information of the creation of each initial individual can be found in 2.5.1.

### 2.1.5 Segment characteristics

As a result of the genetic operators, the segments in a chromosome may have different length. Thus, an approach has to be designed to transform all the segments to the same dimensional space. In our case, six statistical metrics are measured for all the segments included in a chromosome allowing the GA to calculate similarities between segments using the same dimensional space. For the sake of simplicity, all the following characteristics are going to be referred to the segment \( S_s \) which is the part of the time series limited in the following way \( S_s = \{ y_{t_s-1}, \ldots, y_{t_s} \} \):
a) Example chromosome. Each position represents an index of a time series value. 

b) Segments of the time series resulting from the chromosome. 

<table>
<thead>
<tr>
<th>Segment 1</th>
<th>Segment 2</th>
<th>Segment 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3 4</td>
<td>4 5 6 7 8</td>
<td>8 9 10 11 12 13</td>
</tr>
<tr>
<td>13 14 15 16 17 18</td>
<td>18 19 20 21 22</td>
<td></td>
</tr>
</tbody>
</table>

Segment 4 Segment 5 

C) Corresponding segmentation and time series. The characteristics of each segment will be obtained for the corresponding part of the time series.

Figure 2.2: Chromosome representation (Online version in colour)
1. Variance ($S^2_s$): It is a measure of variability that indicates the degree of homogeneity of a group of observations. The mathematical expression of this metric is:

$$S^2_s = \frac{1}{t_s - t_{s-1}} \sum_{i=t_{s-1}}^{t_s} (y_i - \bar{y}_s)^2$$

(2.1)

where $(t_s - t_{s-1})$ is the number of points of the segment, $t_s$ is the index of the first point in the $s$-th segment, $t_{s-1}$ is the index of the last point in the segment, $y_i$ are the time series values of the segment, and $\bar{y}_s$ is the average value of the segment.

2. Skewness ($\gamma_{1s}$): The skewness represents the asymmetry of the distribution of the series values in the segment. Segments can be skewed either up or down with respect to the arithmetic mean. The skewness is defined as:

$$\gamma_{1s} = \frac{1}{t_s - t_{s-1}} \sum_{i=t_{s-1}}^{t_s} (y_i - \bar{y}_s)^3 \frac{1}{S^3_s \cdot (S^2_s)^{\frac{1}{2}}},$$

(2.2)

where $S_s$ is the standard deviation of the $s$-th segment.

3. Kurtosis ($\gamma_{2s}$): It measures the degree of concentration that the values present around the mean of the distribution. Positive kurtosis (i.e. long tails) indicate large excursions away from the arithmetic mean. Kurtosis is defined as:

$$\gamma_{2s} = \frac{1}{t_s - t_{s-1}} \sum_{i=t_{s-1}}^{t_s} (y_i - \bar{y}_s)^4 \frac{1}{S^4_s \cdot (S^2_s)^{\frac{1}{2}}} - 3.$$  

(2.3)

4. Slope of a linear regression over the points of the segment ($a_s$): A linear model is constructed for every segment trying to achieve the best linear approximation of the points of the time-series in the evaluated segment. The slope of the linear model is a measure of the general tendency of the segment. The slope parameter is obtained as:

$$a_s = \frac{S_{yt}^s}{(S^2_s)^{\frac{1}{2}}},$$

(2.4)

where $S_{yt}^s$ is the covariance of the time indexes, $t$, and the time series values, $y$, for the $s$-th segment; and where $S^2_s$ is the standard deviation of the time values. The mathematical expression for the covariance is:

$$S_{yt}^s = \frac{1}{t_s - t_{s-1}} \sum_{i=t_{s-1}}^{t_s} (i - \bar{t}_s) \cdot (y_i - \bar{y}_s).$$

(2.5)

5. Mean Squared Error ($MSE_s$): This statistic measures the degree of non-linearity of the segment. As done for the slope, we also fit a linear model for the segment and then we measure the $MSE_s$ of this linear fitting. This is defined by:

$$MSE_s = S^2_s \cdot (1 - r^2_s),$$

(2.6)

where:

$$r^2_s = \frac{S_{yt}^s}{S^2_s \cdot (S^2_s)^{\frac{1}{2}}}.$$  

(2.7)
6. Autocorrelation coefficient ($AC_s$): It measures the degree of correlation between the current values of the time-series and the values of the time-series in the previous time stamp. The $AC_s$ is defined as:

$$AC_s = \frac{\sum_{i=t_s-1}^{t_s} (y_i - \bar{y}_s) \cdot (y_{i+1} - \bar{y}_s)}{S_s^2}.$$  

(2.8)

Once the six statistical metrics have been calculated for each segment in a chromosome, a clustering technique is applied over this six-dimensional space.

2.1.6 Clustering: $k$-means Algorithm

A clustering process has to be applied in order to obtain the value of the fitness function for each segment. The algorithm chosen, $k$-means, is applied to the time-series segments. Further information on the application of $k$-means and the initialisation procedure can be found in Appendix 2.5.2.

Before applying the clustering algorithm one should normalise the values of the segment metrics, as the distance of each segment to its centroid strongly depends on the range of values of each metric (e.g. variance can have a much broader range of variation than skewness). Thus, distances from each metric with larger ranges would disrupt others from smaller ranges. Scaling is used to avoid this problem. For a given segmentation, the segment metrics are normalised to the range $[0,1]$ using the min-max normalisation:

$$v^* = \frac{v - v_{\min}}{v_{\max} - v_{\min}},$$  

(2.9)

where $v$ is the value of the metric for a given segment, $v^*$ is the normalised value, $v_{\min}$ is the minimum value for this metric when considering all the segments and $v_{\max}$ is the maximum one. A constant value of $v^* = 0.5$ will be assigned whenever the metric is constant for all segments.

2.1.7 Fitness

All GAs need a measure which allows assigning a quality index for each individual of the population. If we are dealing with a clustering process, a way to evaluate the obtained groups is to consider the Sum of Squared Errors ($SSE$), which consists of the sum of squared distances between each segment and its cluster centroid:

$$SSE = \sum_{i=1}^{m} d_i^2$$  

(2.10)

where $i$ is the segment being evaluated, $m$ is the total number of segments, and $d_i$ is the Euclidean distance between segment $i$ and its closest centroid.

Our goal is minimise this $SSE$ in order to obtain more compact clusters (where each point is as closer as possible to its centroid, but the centroids are as far as possible from each other). However, when the GA tries to minimise the $SSE$, it tends to minimise the number of segments as much as possible, in the extreme case producing a partition where each cluster is a single segment. For instance, assuming that the number of clusters considered is five and that a chromosome include only five segments, the $SSE$ would be minimum in this
case, $SSE = 0$, because each segment would constitute a cluster. Taking into account that this is not an acceptable solution, the fitness function is redefined considering also the number of segments:

$$fitness = \frac{m}{SSE}.$$  \hspace{1cm} (2.11)

In this way, the algorithm tries to find partitions of the time series where the number of segments is sufficiently high to assure the acquisition of valuable information from the clustering process.

### 2.1.8 Selection and replacement processes

In this study, a direct selection with the criteria *all individuals are selected* is adopted. That is, in each generation, all individuals within the population are selected for reproduction and generation of offspring. Thus, a greater diversity is promoted, because the parents are not selected based on their fitness.

The replacement process has been performed by roulette wheel selection, i.e. a selection probability for each individual chromosome is calculated from its fitness value. The number of individuals selected is the population size minus one, and the vacant place is occupied by the best segmentation (that with the highest fitness) of the previous generation, thus being an elitist algorithm.

As can be seen, the selection process promotes diversity, while the replacement process promotes elitism.

### 2.1.9 Mutation Operator

The algorithm has been endowed with four mutation operators whose principal function is to perform a better random exploration of the search space, with the aim of reducing the dependency with respect to the initial population and escaping from local optima. The probability $p_m$ of performing any mutation is decided by the user. Once a mutation is decided to be performed, the kind of perturbation applied to the chromosome is randomly selected from the following list: 1) add a cut point, 2) remove a cut point, 3) move half of the cut points to the left, and 4) move half of the cut points to the right.

When adding or removing cut points, the number of cut points to be added or removed is also determined randomly. When moving cut points to the right or the left, the number of points to move is approximately $m/2$ (half of the available points), they are randomly selected, and each cut point is randomly pushed to the previous or the following cut point (with the constraint that it never reaches the previous or the next point). An example of the four mutation operations is included in Figure 2.3 where two cut points are removed, one cut point is added and half of the cut points are moved to the left and to the right.

### 2.1.10 Crossover Operator

The algorithm includes a crossover operator, whose main function is to perform an exploitation of the existing solutions. For each parent individual, the crossover operator is applied with a given probability $p_c$. The operator randomly selects the other parent, a random index of the time series, and it interchanges the left and right parts of the chromosomes with respect to this point. An illustration of the crossover operator can be seen in Figure 2.4.
a) Mutation operator: remove two cut points (18 and 52).

b) Mutation operator: add a cut point: 39.

c) Mutation operator: randomly move cut-points to the left.

d) Mutation operator: randomly move cut-points to the right.

Figure 2.3: Mutation operator (Online version in colour)

a) Before applying crossover operator.

b) After applying crossover operator. The crossover point was randomly decided to be 60.

Figure 2.4: Crossover operator (Online version in colour)
2.2 Experiments

2.2.1 Climate datasets

The datasets chosen for this study are the GISP2 Greenland Ice Sheet Project Two and the NGRIP North Greenland Ice Core Project δ18O ice core data (Grootes and Stuiver, 1997; Stuiver and Grootes, 2000; Andersen et al., 2004; Svensson et al., 2008). The δ18O water isotope record is a proxy for past atmospheric temperature but also reflects changes in water temperature and seasonal snow accumulation. In this study we focus on the 20-yr resolution δ18O isotope records from both drilling sites.

Pre-processing the datasets in the form of a 5-point average was found to help reduce short-term fluctuations within the datasets and improve the analysis of time series segmentations. If \( \{y_n\}_{n=1}^N \) is the original time series, then the time series we have considered is \( \{y^*_n\}_{n=1}^N \) with \( y^*_n = \frac{1}{5} \sum_{j=5i+4}^{5i+9} y_i \).

2.2.2 Algorithm parameters

GAs usually involve adjusting a notable set of parameters. However, their search dynamics, which adapts to the problem evaluated, results in a performance which is negligibly affected by minor changes in these parameters. In our case, all the parameters were initially set by trial and error and then we used the same values for all the problems analysed.

The number of individuals (segmentations) of the population is \( t = 80 \). The crossover probability is \( p_c = 0.9 \) and the mutation probability \( p_m = 0.075 \). The number of clusters to be discovered from each candidate segmentation is \( k = 5 \). This number is possibly the most important parameter, but we experimentally found that \( k = 5 \) clusters is high enough to discover new information among derived clusters but not so high that the interpretation and reproducibility of the results could be threatened. The maximum number of generations is set to 2000, and the \( k \)-means clustering process is allowed a maximum of 20 iterations.

It is important to point out that the algorithm estimates the type of segments and the cutpoints without any additional climate knowledge or supervision of the climate experts. The data time-series obtained from the system undergoing a transition is the only information available to the algorithm.

Finally a GA is a stochastic optimisation algorithm with an embedded random number generator. Given that the results can be different depending on the seed value the algorithm should be run several times with different seeds. For each dataset, the GA was run 10 times, with seeds in the set \( \{10, 20, \ldots, 100\} \) to evaluate and remove the dependence of the results on the seed value. It is also a means to evaluate the accuracy of the algorithm.

2.3 Results

This section presents the main results of the segmentation algorithm for the two paleoclimate datasets under study. The segmentation returned by the GA in the last generation was analyzed, and the following approach has been considered for its analysis. First it was verified whether the DO events were belonging to
Table 2.1: Detection accuracy for early warning signals of Dansgaard–Oeschger events when considering the results of the GA for the 10 seeds.

<table>
<thead>
<tr>
<th>DO event</th>
<th>Detectability success (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GISP2</td>
</tr>
<tr>
<td>End of Younger Dryas</td>
<td>80</td>
</tr>
<tr>
<td>1</td>
<td>90</td>
</tr>
<tr>
<td>2</td>
<td>70</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
</tr>
<tr>
<td>4</td>
<td>90</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
</tr>
<tr>
<td>6</td>
<td>60</td>
</tr>
<tr>
<td>7</td>
<td>70</td>
</tr>
<tr>
<td>8</td>
<td>90</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>50</td>
</tr>
<tr>
<td>11</td>
<td>70</td>
</tr>
<tr>
<td>12</td>
<td>80</td>
</tr>
</tbody>
</table>

different classes or if they were grouped according to some common characteristics. Second the behaviour of each metric in the six-dimensional parameter space was observed on the onset of DO events to find common patterns that would be indicative of EWSs, e.g. increasing variance and autocorrelation coefficient. This was done for two independent datasets and for the ten seed values. The detection accuracy when considering the results of the 10 seeds is included in Table 2.1. Following such approach five main results have been obtained. They are listed below:

1. The DO events are grouped into two main classes, sometimes three because the values of autocorrelation, variance, and MSE may differ significantly from one DO event to another. The high number of classes considered here (5 classes in total) allows for flexibility within the algorithm as warning signals may have different strengths in agreement with the stochastic resonance model (Ganopolski and Rahmstorf, 2001, 2002).

2. EWSs of DO events can be found by the segmentation algorithm in the form of an increase in autocorrelation, variance, and mean square error (MSE). These EWSs are robustly (70%+) found in the GISP2 $\delta^{18}$O dataset for DO 1, 2, 4, 7, 8, 11, 12, and end of Younger Dryas and for DO 1, 4, 8, 10, 11, 12, and end of Younger Dryas for the NGRIP $\delta^{18}$O dataset (see Table 2.1 for more details).

3. The increase in mean square error (MSE) is suggested here as another indicator of abrupt climate change. The increase in MSE, which suggests nonlinear behaviour, has been found to correspond with an increase in variance prior to DO events for ~90% of the seed runs for the GISP2 $\delta^{18}$O dataset (e.g. see Figure 2.5) and for ~100% of the seed runs for the NGRIP $\delta^{18}$O dataset.

4. The increase in the autocorrelation coefficient cannot be solely used as indicator of climate change. The algorithm sometimes found an increase
in MSE and variance but a decrease in autocorrelation coefficient on the onset of DO events. This signature was minor in the GIPS2 $\delta^{18}O$ dataset (e.g. DO 2, 10) but much more present in the NGRIP $\delta^{18}O$ dataset (e.g. DO 0, 1, 5, 7, 8, 10). Hints of this behaviour could already be found for DO 1 by Lenton et al. (2012). We stress that the increase in variance and MSE is a much more robust EWS for NGRIP especially.

5. Analysis of paleoclimate records GIPS2 and NGRIP did not find any consistent change in skewness nor kurtosis on the onset of DO events.

Figure 2.5: Time Series metrics after the clustering process (i.e. the segments found by the algorithm are replaced with their clusters centroids). The increase in MSE is associated with an increase in variance and autocorrelation on the onset of DO events. Several DO events are represented for reference (GISP2 $\delta^{18}O$ ice core, seed = 10) (Online version in colour).

Figure 2.6 presents the detailed segmentation results for GISP2 and NGRIP $\delta^{18}O$ ice core data for a given seed value, including the segmentation, the DO events and the centroids for each cluster. The Dansgaard-Oeschger events are found grouped into two or three main classes with high autocorrelation, MSE, and variance corresponding to classes $C_1$ and $C_2$ for GISP2 and classes $C_1$ and $C_5$ for NGRIP for that run. Class $C_5$ (cyan curve in Fig. 2.6b) is considered the main DO class in NGRIP data for that particular run with a highly linear relationship (ratio of 1:1) between variance and MSE within that class and a constant high autocorrelation coefficient. This is illustrated in Figure 2.7b.

Class $C_3$ for GISP2 dataset was the third main class grouping segments with the lowest MSE, variance, and autocorrelation for that seed run and was found at the onset of several DO events (e.g. 1, 4, 8, 12) collocated with the Heinrich events H1, H3, H4, H5 as well as during the Holocene period (for an introduction to Heinrich events see Hemming, 2004). Classes $C_4$ and $C_5$ have been found outside the plotted area (in the -50ka, -60ka range) and therefore do not appear in the graph. As for the NGRIP dataset classes $C_2$ and $C_4$ with the lowest MSE, variance, and autocorrelation have been found at the onset of several DO events as well (e.g. 4, 7, 8, 10 and 12) with a strange behaviour.
in the autocorrelation coefficient for DO 1. A detailed analysis of their six-
dimensional vector revealed that classes $C_2$ and $C_4$ differ only from the point
of view of kurtosis in that run. This is further discussed in the discussion
section about the limitations of the algorithm. Considering algorithm runs with
different seed values revealed minor differences such as DO events belonging
to other classes but the main characteristics described here and in the five
main points remained robust throughout the results. The reader is invited to
Appendix 2.6 for the detailed segmentation results of GISP2 and NGRIP $\delta^{18}O$
ic core data for another seed value.

![GISP2 oxygen isotope data](image1)

GISP2 oxygen isotope data
Time before present (kyrs)

![NGRIP oxygen isotope data](image2)

NGRIP oxygen isotope data
Time before present (kyrs)

Figure 2.6: Results of segmentation algorithm on $\delta^{18}O$ ice core data (seed = 10). The Dansgaard-Oeschger events are found grouped into two or three main
classes with high autocorrelation, MSE, and variance corresponding to classes
$C_1$, $C_2$ and $C_3$ for GISP2 and $C_1$ and $C_3$ for NGRIP. Several Dansgaard-Oeschger
events are numbered for reference. (Online version in colour)

2.4 Discussion

As expected the two ice core studied showed a few differences with respect
to the detectability of DO events (see Table 2.1 but overall the main
classifiers could be captured within the two datasets. For instance changes
in statistical parameters were detected in the GIP52 ice core for DO 3, 5, 6 and
10 with medium success (30%-60%) and with medium to low success (20%-60%)
for DO 2, 3, 5, 6, 7 and 10 in the NGRIP ice core, suggesting that these particular
DO events possess weak EWSs. Furthermore the segmentation algorithm could
not find any EWS for DO event 9. We suggest here that this particular event
does not possess any EWS, i.e. that the transition to a warm ocean circulation
mode close to a bifurcation point is taking place because of internal noise. The
detection of EWSs at the onset of some DO events and absence in other events
is a strong argument in favour of the stochastic resonance model as proposed in
Ganopolski and Rahmstorf (2001, 2002). It is worth mentioning that DO 9 is
also different from the point of view of its amplitude within a given time period.
Using a simple event detection algorithm based on the data slope [Rahmstorf (2003)] could not detect DO 9 as the associated warming was slower than in the other events documented in Dansgaard et al. (1993).

When analyzing the results of segmentation algorithms one must also consider the segment lengths to obtain meaningful information. It can happen that the algorithm is not able to assign a proper class to a segment and prefers to divide the segments into smaller sections to reduce e.g. MSE and kurtosis values. The new smaller sections are likely to be grouped together in this parameter space, allowing the algorithm to perform the clustering process. Moreover, analyzing Eq. (2.11), fitness is directly proportional to the number of segments so segmentations with a high number of segments will be preferred. One signature of this effect is seen in the fact that all small segments are found in a single class with very low kurtosis ($\gamma_2 = [-1.6, -1.9]$), constant skewness (equal to 0), and a large range of slope coefficients. They are represented by a straight line in Fig. 2.8. Special care was taken to discard those small segments (e.g. containing 2 or 3 points) in the analysis of EWSs. The difficulty of evaluating the clustering quality is the origin of these problems. Sum of squared errors (SSE) directly depends on the number of segments of the segmentation in such a way that, if the number of segments is not taken into account for the fitness function, the segmentations tend to be too simple (few segments per cluster) resulting in too coarse-grained information. As discussed, when introducing the number of segments, the algorithm tends to discover small segments. However, as consecutive small segments are usually labelled with the same class, the final result is still useful. More advanced quality metrics for clustering could avoid this kind of problems.

On the other hand, when considering the EWSs discovered by the algorithm in future events, the length of the segments should be defined. This could be done by analyzing the historical data obtained for the time series (for one or even several different seeds).
2.5 Additional details about the Segmentation Algorithm

2.5.1 Generation of each individual for the initial population

We apply the following steps to generate the initial population of segments:

1. We determinate randomly the numbers of cutting points, \( m \), as a uniform value in the interval \([10k, 15k]\), where \( k \) is the number of clusters to be discovered among the different segments (and it is a user parameter). In this way, we guarantee the existence of at least ten segments on average for each cluster, so we can assure a minimum number of patterns to discover proper clusters.

2. The procedure generates randomly the indexes of these \( m \) cutting points. These random indexes are generated in such a way that \( t_1 < t_2 < \ldots < t_{m-1} \), not allowing the repetition of indexes in the chromosomes, which will result in a zero-length segment.

2.5.2 Application of the \( k \)-means algorithm

The algorithm is applied in the following way:

1. Initialisation of the centroids: we have modified the classic algorithm in the sense that, instead of randomly choosing the initial centroids from the list of segments, we consider a deterministic selection. This deterministic process ensures that a chromosome will have always the same fitness value. First, we choose the characteristic of the segment where the difference between the maximum and minimum value is the highest, i.e. the characteristic with the most extreme values. The first centroid is the segment with the highest value in that characteristic. The second centroid
is the segment with the highest Euclidean distance from the first centroid previously selected. The third centroid will be that which is farthest from both, and so on. This assures a deterministic initialisation process, as the same time that the initial centroids are as far as possible from each other.

2. Then the usual $k$-means algorithm is applied, i.e.:

(a) We calculate the distance between each segment and all the centroids.
(b) Each segment is assigned to the cluster of the closest centroid.
(c) The centroids are recalculated, as the average value of all the segments belonging to the corresponding cluster.
(d) If the stop condition is not fulfilled, return to (a). The algorithm stops when the centroids are not modified for two consecutive iterations.

### 2.6 Additional Examples of Segmentation for the GISP2 and NGRIP datasets

Figure 2.9 presents the detailed segmentation results for GISP2 and NGRIP $\delta^{18}$O ice core data for a seed value of 100. The Dansgaard-Oeschger events are found grouped into two main classes with high autocorrelation, MSE, and variance corresponding to classes $C_1$ and $C_4$ for GISP2 and classes $C_1$ and $C_3$ for NGRIP for that run.

![Figure 2.9: Results of segmentation algorithm on $\delta^{18}$O ice core data (seed = 100). The Dansgaard-Oeschger events are found grouped into two main classes with high autocorrelation, MSE, and variance corresponding to classes $C_1$ and $C_4$ for GISP2 and $C_1$ and $C_3$ for NGRIP. Several Dansgaard-Oeschger events are numbered for reference (Online version in colour).](image)

<table>
<thead>
<tr>
<th>Variable</th>
<th>$C_1$ (group)</th>
<th>$C_1$ (means)</th>
<th>$C_4$ (group)</th>
<th>$C_4$ (means)</th>
<th>MSE</th>
<th>Autocorrelation</th>
</tr>
</thead>
<tbody>
<tr>
<td>GISP2</td>
<td>2.142</td>
<td>-1.194</td>
<td>-0.91</td>
<td>0.406</td>
<td>3.94</td>
<td></td>
</tr>
<tr>
<td>NGRIP</td>
<td>3.088</td>
<td>-0.228</td>
<td>-0.972</td>
<td>0.507</td>
<td>3.23</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2.9: Results of segmentation algorithm on $\delta^{18}$O ice core data (seed = 100). The Dansgaard-Oeschger events are found grouped into two main classes with high autocorrelation, MSE, and variance corresponding to classes $C_1$ and $C_4$ for GISP2 and $C_1$ and $C_3$ for NGRIP. Several Dansgaard-Oeschger events are numbered for reference (Online version in colour).
Chapter 3

Alternative fitness functions and a new evaluation method

The experiments in Chapter 2 showed some important limitations for the fitness function considered. In this chapter, we consider alternative fitness functions for the segmentation algorithm. Moreover, we approach a method for automatic clustering result evaluation. Measuring the quality of a segmentation can be only achieved by expert evaluation of the solutions given by the algorithm. We present a quantitative method to perform comparisons with respect to an expected ideal segmentation of the series to assess the robustness and stability of the method. This method allows evaluating a segmentation algorithm with a minimal effort by the expert, who has only to provide the ideal segmentation.

The rest of the chapter is organised as follows. Section 3.1 presents some of the new alternative fitness functions, while Section 3.2 presents a proposal for segmentation comparison and discusses the experimental setting. The results are included in Section 3.3.

3.1 Measuring the quality of the clustering process

As described in Section 2.1.7, the last step of the evaluation of the chromosome is to measure how well the segments are grouped (compactness of the clustering). It is clear that different clustering algorithms usually lead to different clusters or reveal different clustering structures. In this sense, the problem of objectively and quantitatively evaluating the clustering results is particularly important and this is known in the literature as cluster validation. There are two different testing criteria for this purpose (Xu and Wunsch, 2008): external criteria and internal criteria. When a clustering result is evaluated based on the data that was clustered itself, this is called internal evaluation. In external evaluation, clustering results are evaluated using for example known class labels. Based on these concepts, the internal criteria evaluation metrics will be a suitable option for the evolution, because the GA is not given any a priori information of the
segments to be found. Note that the segments metrics are normalised at this step as well. We have considered four different metrics:

1. **Sum of squared errors (SSE):** The simplest error measure is the sum of squared errors (considering errors as the distance from each point to their centroid), i.e.:

   \[
   SSE = \frac{1}{N} \sum_{i=1}^{k} \sum_{x \in C_i} d(x, c_i)^2, \tag{3.1}
   \]

   where \( k \) is the number of clusters, \( c_i \) is the centroid of cluster \( C_i \) and \( d(x, c_i) \) is the Euclidean distance between pattern \( x \) and centroid \( c_i \). This function does not prevent clusters to fall very close in the clustering space.

   As this index has to be minimised, the fitness will be defined as \( f = \frac{1}{1 + SSE} \).

2. **Caliński and Harabasz index (CH):** This index has been found to be one of the best performing ones for adjusting the value of \( k \). It is defined as:

   \[
   CH = \frac{\text{Tr}(S_B)}{\text{Tr}(S_W) (k-1)}, \tag{3.2}
   \]

   where \( N \) is the number of patterns, and \( \text{Tr}(S_B) \) and \( \text{Tr}(S_W) \) are the trace of the between and within-class scatter matrix, respectively. Note that the value of \( k \) will be fixed in our algorithm. As this index has to be maximised, the fitness will be defined as \( f = CH \).

3. **Davies-Bouldin index (DB):** This index also attempts to maximise the between-cluster distance while minimising the distance between the cluster centroids to the rest of points. It is calculated as follows:

   \[
   DB = \frac{1}{k} \sum_{i=1}^{k} \max_{i \neq j} \left( \frac{\alpha_i + \alpha_j}{d(c_i, c_j)} \right), \tag{3.3}
   \]

   where \( \alpha_i \) is the average distance of all elements in cluster \( C_i \) to centroid \( c_i \), and \( d(c_i, c_j) \) is the distance between centroids \( c_i \) and \( c_j \). As this index has to be minimised, the fitness will be defined as \( f = \frac{1}{1 + DB} \).

4. **Dunn index (DU):** The Dunn index attempts to identify clusters that are compact and and well-separated. In this case, the distance between two clusters is defined as

   \[
   d(C_i, C_j) = \min_{x \in C_i, y \in C_j} d(x, y), \tag{3.4}
   \]

   that is, the minimum distance between a pair of points \( x \) and \( y \) belonging to \( C_i \) and \( C_j \). Furthermore, we could define the diameter \( \text{diam}(C_i) \) of cluster \( C_i \) as the maximum distance between two of its members, such as: \( \text{diam}(C_i) = \max_{x, y \in C_i} d(x, y) \). Then, the Dunn index is constructed as:

   \[
   DU = \min_{i=1, \ldots, k} \left( \min_{j=i+1, \ldots, k} \left( \frac{d(C_i, C_j)}{\max_{l=1, \ldots, k} \text{diam}(C_l)} \right) \right). \tag{3.5}
   \]

   The Dunn index has been found to be very sensitive to noise, but this disadvantage can be avoided by considering different definitions of cluster distance or cluster diameter. For example, as suggested in [Xu and Wunsch, 2008], the cluster diameter can be computed as:

   \[
   \text{diam}(C_i) = \frac{1}{N_{C_i}(N_{C_i} - 1)} \sum_{x, y \in C_i} d(x, y), \tag{3.5}
   \]

   where \( N_{C_i} \) is the number of patterns belonging to cluster \( C_i \). This cluster diameter estimation has been found to be more robust in the presence of noise. As this index has to be maximised, the fitness will be \( f = DU \).
3.2 Automatic evaluation method and experimental setting

As previously, the dataset chosen for this chapter is the North Greenland Ice Core Project (NGRIP) $\delta^{18}$O ice core data (Andersen et al., 2004; Svensson et al., 2008). The $\delta^{18}$O water isotope record is used as a proxy for past atmospheric temperature. We focus on the 20-yr resolution $\delta^{18}$O isotope records. The dataset is pre-processed by obtaining a 5-point average in order to reduce short-term fluctuations within the data. In this way, the time series we have considered is $\{y^*_n\}_{n=1}^{N/5}$ with $y^*_i = \frac{1}{5} \sum_{j=i}^{i+4} y_j$.

3.2.1 Experimental setting

The experimental design is presented in this subsection. The GA was configured with the following parameters: the number of individuals of the population is $t = 100$. The crossover probability is $p_c = 0.8$ and the mutation probability $p_m = 0.2$. The percentage of cut points to be mutated is the integer part of the 20% of the number of cut points. For the initialisation, the number of segments is decided by defining the average segment length, which is set to $\overline{l} = 4$. The maximum number of generations is set to $g = 100$, and the $k$-means clustering process is allowed a maximum of 20 iterations. These parameters were optimised by a trial and error procedure, although the algorithm showed a very robust performance to their values. The most important parameters for the final performance of the algorithm were $\overline{l}$ and $k$.

We performed different experiments considering the 4 different fitness functions presented in Section 3.1 and different values of $k$ for the $k$-means algorithm ($k = 2, \ldots, 6$). It is important to recall that the algorithm estimates the optimal segments and clusters them without any prior information of the DO events. The only information given to the algorithm is the time series and the statistic characteristics to use for the clustering in order to validate whether the statistics proposed in the literature are useful for characterising paleoclimate TPs in general. Given the stochastic nature of GAs, the algorithm was run 30 times with different seeds to evaluate its stability and robustness.

3.2.2 Automatic evaluation method

In order to evaluate the results of the algorithm, two evaluation metrics were used. These measures analyse both the homogeneity of cluster assignment with respect to the DO events and the robustness of the results obtained from different seeds. They are not included in the fitness function, serving only as an automatic way of evaluating the quality of the segmentation, avoiding the intervention of the expert. Both are indexes comparing two different clustering partitions:

1. Rand index (RI) This metric is particularly useful for data clustering evaluation (Rand, 1971). It is related to the accuracy, but is applicable even when class labels are not available for the data, as in our case. A set $Y = \{y_n\}_{n=1}^{N}$ is given (in our case, the time series), and two clustering partitions of $Y$ are to be compared: $X = \{X_1, \ldots, X_r\}$ and $Z = \{Z_1, \ldots, Z_s\}$. For a given segmentation, the partitions are defined in the following way:
Figure 3.1: Representation of the ideal segmentation and the different DO events.

$X_l$ is a set containing every $y_i \in s, s \in \mathcal{C}_l$, i.e. the partitions are based on the label assigned to each time series value $y_i$ from the current segmentation. The following two numbers are defined: $a$ (number of pairs in $Y$ that are in the same set in $X$ and $Z$) and $b$ (number of pairs in $Y$ that are in different sets in $X$ and $Z$). Then, the Rand index is defined as: $RI = (a + b) / \binom{n}{2}$. This metric has a value between 0 and 1, with 0 indicating that the two partitions do not agree on any pair of points and 1 indicating that they are exactly the same.

2. Adjusted rand index (ARI): It is a corrected version of the RI trying to fix some known problems with the RI, e.g. the expected value of the RI of two random partitions does not take a constant value and it approaches its upper limit of unity as the number of clusters increases. ARI values range from $-1$ to $+1$, yielding negative values if the index is less than the expected index. The detailed formulation can be found in Hubert and Arabie (1985).

In order to evaluate the segmentation returned by the algorithm, we compare it with an ideal segmentation. The ideal segmentation (Figure 3.1) has been designed by examining the literature about Dansgaard-Oeschger (DO) events, which are associated to TPs. In the Figure, the onsets of the DO events (in a first approximation, we do not consider the error margin) reported in Svensson et al. (2008) are represented by vertical lines and the segments covering the period precursor to the DO events (which we hypothesise as TP) are delimited by the slope close to the corresponding onset. The closer the segmentation returned by the GA is to this ideal segmentation, the better the segmentation. To perform this comparison, RI and ARI indexes will be used (ARI_Ideal and RI_Ideal).

Given that the wishful ideal segmentation would be binary (non DO event or DO event) and the segmentation returned by the GA can have a value of $k > 2$, we need to binarise the segmentation of the GA (i.e. decide which clusters...
Table 3.1: NGRIP average segmentation results for different algorithm settings.

<table>
<thead>
<tr>
<th>Fitness</th>
<th>k</th>
<th>ARI_Ideal</th>
<th>RI_Ideal</th>
<th>ARI_Seeds</th>
<th>RI_Seeds</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB</td>
<td>5</td>
<td>0.315 ± 0.060</td>
<td>0.777 ± 0.015</td>
<td>0.346 ± 0.078</td>
<td>0.727 ± 0.040</td>
</tr>
<tr>
<td>DU</td>
<td>5</td>
<td>0.308 ± 0.067</td>
<td>0.788 ± 0.018</td>
<td>0.341 ± 0.099</td>
<td>0.727 ± 0.046</td>
</tr>
<tr>
<td>CH</td>
<td>5</td>
<td>0.260 ± 0.073</td>
<td>0.772 ± 0.008</td>
<td>0.223 ± 0.105</td>
<td>0.644 ± 0.074</td>
</tr>
<tr>
<td>SSE</td>
<td>5</td>
<td>0.279 ± 0.048</td>
<td>0.770 ± 0.018</td>
<td>0.057 ± 0.018</td>
<td>0.638 ± 0.017</td>
</tr>
</tbody>
</table>

3.3 Results

All these results are included in Table 3.1. The first part of the table compares the different fitness functions for a predefined value of $k = 5$ (as we initially observed that this was obtaining suitable results). As can be seen, both DB and DU fitness functions obtain very good segmentation quality and stability, although DB performs slightly better. In contrast, CH and SSE are performing poorly in both scenarios (it is noteworthy the very low stability obtained by the SSE fitness function, which may be due to the fact that it only minimises the intra-cluster distances and obviates the inter-cluster distances). The result that the algorithm is robust and stable to different initialisations is crucial for the following parts of the study (i.e. develop an early warning system for TPs of climatic component). Concerning the experiment that studies different values of $k$, it can be seen that $k = 5$ is indeed the optimal value for the segmentation. This result indicates that the concept and nature of DO events is too complex to represent the DO events and which not. Preliminary experiments revealed that DO events were usually grouped under one or two clusters, so we evaluated ARI_Ideal and RI_Ideal for all possible combinations of one or two clusters. The final value was the maximum ARI_Ideal and RI_Ideal values of all these combinations. Moreover, the stability of the GA was estimated by comparing the 30 segmentations from the different runs. This was done by averaging RI and ARI comparing all possible pairs of segmentations (ARI_Seeds and RI_Seeds).
Figure 3.2: Best time series cluster assignment after the evolutionary process.

Figure 3.3: Clustering space for the six metrics (each point represents a segment).

the series. However, five events are not detected: 2, 9, 11, 13 and 16 (some of which have been found in the literature to be caused by random fluctuations of the dynamics of the time series and for which there is no evidence of increase in the selected statistics). The clustering space of this segmentation can be analysed in Figure 3.3. This Figure confirms that there are some differences between the two clusters associated to the DO events ($C_1$ and $C_5$), mainly from the values of the $S_2^*$ metric.
Chapter 4

Time Series Forecasting by Evolutionary Recurrent Product Unit Neural Networks

This chapter is focused on Product Unit Neural Networks (PUNNs) and its application on TSF. The basis function of the hidden neurons of PUNNs is the Product Unit (PU) function, where the output of the neuron is the product of their inputs raised to real valued weights. PUNNs are an alternative to sigmoidal neural networks and are based on multiplicative nodes instead of additive ones (Durbin and Rumelhart, 1989). This model has the ability to express strong interactions between input variables, providing big variations at the output from small variations at the inputs. Consequently, it has increased storage information capability and promising potential for TSF. However, they result in a highly convoluted error function, plenty of local minima. This handicap makes convenient the use of global search algorithms, such as genetic algorithms (Li et al., 2011), evolutionary algorithms (Luque et al., 2007) or swarm optimisation algorithms (Cai et al., 2004), in order to find the parameters minimising the error function. PUNNs has been widely used in classification (Hervás-Martínez and Martínez-Estudillo, 2007) and regression problems (Martínez-Estudillo et al., 2006), but scarcely applied to TSF, with the exception of some attempts on hydrological TSA (Dulakshi S. K. Karunasingha and Li, 2011; Piotrowski and Napiorkowski, 2012). It is important to point out that, in TSF, there is an autocorrelation between the lagged values of the series. In this way, theoretically, PUNNs should constitute an appropriate model for TSF because they can easily model the interactions (correlations) between the lagged values of the time series.

The first goal of this chapter is to evaluate the performance of Autoregressive Product Unit Neural Networks (ARPUNN) on TSF. The ARPUNN model should yield high performance for TSF, as it fulfils the requirements that allow the modelling of TS: ability to express the interactions between inputs and increased storage capability. However, as mentioned above, long term memory
ANNs usually obtain better results than FFNNs. For this reason, a second goal of this work is to propose a hybrid ANN combining an ARPUNN with a reservoir network with the objective of increasing the storage capability of the resulting model. The short term memory is provided by the different lags of the TS included in the input layer and the long term memory is supplied by a reservoir network included as one of the inputs of the system. The final model is called Recurrent Product Unit Neural Network (RPUNN).

From the points of view of the learning algorithm, the complex error surface associated to PUs implies serious difficulties for searching the best parameters minimising the error function. A novel hybrid algorithm is proposed in this work to alleviate these difficulties. It combines the exploration abilities of global search algorithms with the exploitation ones of local search methods. The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) algorithm [Hansen 2006, Jastrebski and Arnold 2006] is used to calculate the parameter values of the hidden layer, whereas the weights of the output layer are determined by means of the MP generalised inverse. These combinations provide us with a hybrid model and algorithm capable to afford the difficulties of TSF, obtaining a competitive performance.

This chapter is organised as follows: Section 4.1 describes the ANN hybrid model proposed to be applied in TSF. Section 4.2 explains the hybrid search algorithm designed to get the parameters which optimise the error function. Sections 4.3 and 4.4 explains the experiments that were carried out and the results obtained.

4.1 Models

In this section, we first introduce a ARPUNN model, which is then extended by considering a reservoir to result in the RPUNN model. The models proposed addressed the TSF problem. This problem is mathematically formulated as follows. Let \( \{ y_n \}_{n=0}^{N+p} \) be a TS to be predicted, where \( N + p \) TS values are given for training. In this way, the function \( f : \mathbb{R}^p \rightarrow \mathbb{R} \) is estimated from a training set of \( N \) patterns, \( D = (X, Y) = \{ (x_n, y_{n+p}) \}_{n=1}^{N} \) where \( x_n = \{ y_{n+p-1}, y_{n+p-2}, \ldots, y_n \} \) is the vector of input characteristics (\( p \) past values of the TS) taking values in the space \( \Omega \subset \mathbb{R}^p \), and the label, \( y_{n+p} \), is the value of the TS for the \( n+p \) instant. Both models are now explained in the following subsections.

4.1.1 Short memory model: Autoregressive Product Unit Neural Network (ARPUNN)

This section presents the first model proposed to address the TSF problem, the so-called ARPUNN. The suggested architecture is based on considering PUs as the basis functions for the hidden layer of the network. PUs neural networks models have the ability to express strong interactions between the input variables. The model is composed by an input, hidden and output layer. The input layer has \( p \) input units that correspond to the lagged values of the TS providing the network with a short memory. The hidden layer of the network is composed by \( S \) PUs and the output layer contains only one neuron. A representation of
model proposed has been included in the supplementary material of the paper associated to this chapter \[1\].

The final model is linear in the basis function space together with the initial variables. A similar architecture (which is usually referred to as skip-layer connections) was also considered for classification in previous works for PUs (Hervás-Martínez and Martínez-Estudillo 2007; Gutiérrez et al., 2010) and Radial Basis Functions (RBFs) (Gutiérrez et al., 2011). The TS value is estimated by

\[ \hat{y}_{n+p} = f(x_n, \theta) : \mathbb{R}^p \rightarrow \mathbb{R}, \]

where the final output of the model is defined as

\[ f(x_n, \theta) = \beta_0 + \sum_{s=1}^{S} \beta_s B_s(x_n, w_s) + \sum_{k=1}^{p} \alpha_k y_{n+p-k}, \quad (4.1) \]

where \( \beta_s \in \mathbb{R} \) denotes the weight of the connection between the hidden neuron \( s \) and the output neuron \( s = 1, 2, \ldots, S \), leading the structure that provides the non linear contribution of the inputs. The \( \beta \) vector includes all the parameters connecting the hidden with the output layer and the bias \( \beta = (\beta_0, \beta_1, \beta_2, \ldots, \beta_S) \in \mathbb{R}^S \). The linear contribution of the inputs is controlled by \( \alpha_k \), which is the weight of the connection between the input \( k \) and the output layer \( k = 1, 2, \ldots, p \). The vector \( \alpha \) contains all the parameters connecting the input and the output layer, \( \alpha = (\alpha_1, \alpha_2, \ldots, \alpha_p) \in \mathbb{R}^p \). Another kind of weights, \( w_s \in \mathbb{R}^p \), represent the connections of the hidden neuron \( s \) and the input layer. The \( \theta \) vector contains the full parameter vector \( \theta = \{w_1, w_2, \ldots, w_S, \beta, \alpha\} \). Finally, \( B_s(x_n, w_s) : \mathbb{R}^p \rightarrow \mathbb{R} \) represents the output of the \( s \)-th PU basis function and it is defined as:

\[ B_s(x_n, w_s) = \prod_{i=1}^{p} (y_{n+p-i})^{w_{is}}, \quad 1 \leq s \leq S \quad (4.2) \]

where \( w_{is} \in \mathbb{R} \) is the weight of the connection between the \( i \)-th input node and the \( s \)-th basis function and \( y_{n+p-i} \) denotes the \( i \)-th lagged past value of the TS.

### 4.1.2 Long memory model: Recurrent Product Unit Neural Network (RPUNN)

In this section, the long memory model is presented (called Recurrent Product Unit Neural Network, RPUNN). The RPUNN model is also established on the ARPUNN advanced previously and reuses its network architecture. One aspect that should be considered on TSF is the memory or the amount of information that can be storage in the network. Traditionally, ANNs with longer memory have an enhanced performance for TSF. The main difference between ARPUNN and RPUNN lies in the inclusion on a new structure as an input, a reservoir network. The reservoir provides the whole structure with long term and dynamic memory. The structure of the RPUNN is depicted in Figure 4.1.

As can be seen, the network inherits the architecture of ARPUNN with the linear and non-linear combination of the inputs described in the previous

1[http://www.esa.int/gsp/ACT/cms/projects/rapunn.html]
2[For the sake of clarity, reservoir representation is simplified: there is a link between each reservoir node and each PU and all reservoir nodes receive \( y_{n-1} \) time series value as input. The interconnections between reservoir nodes are random. Internal connections of the reservoir are given by \( \kappa \).]
The output layer contains only one neuron while the hidden layer of the network is composed by $S$ neurons, whose basis function is the PU. The input layer considered has $p+m$ neurons that correspond to the $p$ lagged values of the TS plus the $m$ outputs of the reservoir network. The $p$ lagged values provides to the network the short memory. The reservoir part is formed by a set of $m$ nodes and the output of each of these nodes is considered as an input to the RPUNN, providing the whole structure with a dynamic memory. The only input considered for the reservoir is the first lagged value of the input of the network. The estimated TS value is defined by the final output of the model, $\hat{y}_{n+p} = f(x_n, \theta) : \mathbb{R}^{m+p} \rightarrow \mathbb{R}$, as follows:

$$f(x_n, \theta) = \beta_0 + \sum_{s=1}^{S} \beta_s B_s(x_n, \psi^{(n)}, w_s) + \sum_{k=1}^{p} \alpha_k y_{n+p-k} \quad (4.3)$$

where $\psi^{(n)} \in \mathbb{R}^m$ is the reservoir state vector for time $n$, and $\theta = \{w_1, w_2, \ldots, w_s, \beta, \alpha, \kappa\}$ represents the set of the network weights, composed by the vectors $\beta \in \mathbb{R}^S$ and $\alpha \in \mathbb{R}^p$ (previously defined), $w_s \in \mathbb{R}^{m+p}$, which represents the connections of the hidden neurons and the input layer, $s = 1, \ldots, S$, and, finally, the matrix of the connections for the reservoir network, $\kappa \in \mathbb{R}^{m \times (m+2)}$. At last, $B_s(x_n, \psi^{(n)}, w_s) : \mathbb{R}^{m+p} \rightarrow \mathbb{R}$ represents the basis function considered in the hidden layer yielding the following nonlinear output for the model:

$$B_s(x_n, \psi^{(n)}, w_s) = \prod_{i=1}^{p} (y_{n+p-i})^{w_{is}} \prod_{j=p+1}^{p+m} (\psi_j^{(n)})^{w_{js}} \quad (4.4)$$

where $1 \leq s \leq S$, $w_s = (w_{1s}, w_{2s}, w_{(p+1)s}, \ldots, w_{(p+m)s}) \in \mathbb{R}^{m+p}$ is the hidden layer weight vector, $w_{is} \in \mathbb{R}$ is the weight of the connection between the input neuron $i$ and the hidden neuron $s$, $1 \leq i \leq p$, and $w_{(p+j)s}$ is the weight of the connection between the $j$-th reservoir node and the hidden neuron $s$, $1 \leq j \leq m$. Finally, $\psi_j^{(n)}$ represents the output of the $j$-th reservoir node at time $n$, $1 \leq j \leq m$ and the corresponding vector is $\psi^{(n)} = \{\psi_1^{(n)}, \ldots, \psi_m^{(n)}\}$. 
The reservoir consists of a sparsely connected group of nodes, where each neuron output is randomly assigned to the input of another neuron. This allows the reservoir reproducing specific temporal patterns. All the reservoir nodes are sigmoidal nodes, as this model is more adequate in order to keep a long term memory of the TS:

\[
\psi_j^{(n)} = R_j(\psi_j^{(n-1)}, \kappa_j) = \sigma \left( \kappa_{0j} + \sum_{i=1}^{m} \kappa_{ij} \psi_i^{(n-1)} + \kappa_{(m+1)j} y_{n-1} \right),
\]

where \( \sigma(x) = 1/(1 + \exp(-x)) \) is the sigmoidal activation function, and \( \kappa_j \) is the vector of parameters corresponding to the \( j \)-th reservoir neuron, \( \kappa_j = \{\kappa_{0j}, \kappa_{1j}, \ldots, \kappa_{mj}, \kappa_{(m+1)j}\} \), with \( m + 2 \) elements. As can be observed, self-connections are allowed. The internal structure of the reservoir is randomly fixed and kept constant during the learning process, in the same vein than it is done with ESNs (Gallicchio and Micheli, 2011).

### 4.2 Parameter Estimation

This section discuss the training algorithm proposed to fit ARPUNN and RPUNN parameters. As stated above, PUNNs exhibit a highly convoluted error surface, which can easily make the training algorithm get stuck in local minima and in consequence, avoiding optimum parameters to be obtained. In general, this can be tackled by using global search algorithms, but instead they can be slow to reach the global optimum. The method considered in this work focuses in obtaining a trade-off between both extremes, which is achieved by a hybrid algorithm. The parameter set to be optimised in the ARPUNN model is \( \Theta = \{\beta, \alpha, w_1, w_2, \ldots, w_S\} \), which is composed by the set of weights of the hidden layer nodes (\( w_1, w_2, \ldots, w_S \)), and the set of weights of the output layer, \( \beta \) and \( \alpha \). In the case of the RPUNN, it is also required to estimate the values of the parameters included in the vector \( \kappa \), i.e. the weights of the reservoir interconnections.

The beginning of the algorithm involves the CMA-ES method as a global optimisation procedure (Hansen, 2006). CMA-ES is an evolutionary algorithm for difficult nonlinear non-convex optimisation problems in continuous domain. The evolution strategy defined in this algorithm is based on the use of a covariance matrix that represents the pairwise dependencies between the candidate values of the variables to be optimised. The distribution of the covariance matrix is updated by means of the covariance matrix adaptation method, that attempts to learn a second order model of the cost function similar to the optimisation made in the Quasi-Newton methods (Saini and Soni, 2002). The CMA-ES has several invariance properties and does not require a complex parameter tuning. In this study, the uncertainty is undertaken as proposed in Hansen et al. (2009) and a subtractive update of the covariance matrix is done as in Jastrebski and Arnold (2006). Another consideration is to adapt only the diagonal of the covariance matrix for a number of initial iterations, as stated in Ros and Hansen (2008), leading to a faster learning. The cost function according to which the weights are optimised is the Root Mean Squared Error (RMSE).
For both ARPUNN and RPUNN models, the target parameters under optimisation by the CMA-ES algorithm are the weights from the input layer to the hidden layer \( \{w_1, w_2, \ldots, w_S\} \). The hybrid algorithm starts by randomly generating the values for these weights. Although the rest of the weights are needed to obtain the cost function, they can be analytically calculated by using the MP generalised inverse, as done in the ELM (Huang et al., 2012). This process has to be performed on each iteration of the CMA-ES algorithm and for each individual of the population. Let \( \Phi = (\beta_1, \ldots, \beta_S, \alpha_1, \ldots, \alpha_p)^T \) denote the weights of the links connecting hidden and output layers. The calculation of \( \Phi \) can be done by taking into account that the system is linear is the basis function space is considered. In this way, the nonlinear system can be converted into a linear system where:

\[
Y = H\Phi
\]

where \( H = \{h_{ij}\} \) \((i = 1, \ldots, N \text{ and } j = 1, \ldots, S + p)\) represents the hidden and input layers output matrix: if \( 1 \leq j \leq S \), \( h_{ij} = B_j(x_i, w_j) \) (for the ARPUNN model) or \( h_{ij} = B_j(x_i, \psi^{(i)}, w_j) \) (for the RPUNN model); if \( S < j \leq S + p \), \( h_{ij} = y_{i+p-j} \). Finally, the determination of \( \Phi \) can be obtained by finding the least-square solution of the equation:

\[
\hat{\Phi} = H^\dagger Y
\]

where \( H^\dagger \) is the MP generalised inverse of the matrix \( H \). The solution provided by this method is unique and it has the smallest norm within all least-square solutions. In addition, it obtains a high generalisation performance that decreases the time required to learn the sequence as states (Huang et al., 2004).

The parameters of the reservoir for RPUNN model are randomly fixed before starting the CMA-ES optimisation and then kept constant for the whole evolution. Sparsity is achieved by randomly setting to 0 a percentage (in our case, \( \sim 90\% \)) of the weights for the connections between reservoir nodes (i.e. \( \kappa_{ij} = 0 \), for some randomly selected \( i \) values, \( 1 \leq i \leq m \)).

### 4.3 Experiments

In order to analyze the performance of the proposed methods, an experimental study was carried out. The TS data selected, the metrics considered to evaluate the performance of the models and the algorithms used for comparison purposes are described in the following subsections.

#### 4.3.1 Dataset Selected

The TS datasets used for the experimental setup belongs to the NNGC1, Acont, B1dat, D1dat and Edat forecasting competition\(^3\). These datasets were also considered in Bergmeir et al. (2012). A total amount of 29 time series available in the KEEL-dataset repository\(^4\) (Alcalá-Fdez et al., 2011) have been used. A detailed description of the datasets considered and a table with their characteristics have been included in the supplementary material of the paper.

\(^3\)Available at http://www.neural-forecasting-competition.com

\(^4\)Available at http://sci2s.ugr.es/keel/timeseries.php
The datasets have been preprocessed to adapt the inputs to the mathematical characteristics of the PU-based models: input variables have been scaled in the rank $[0.1, 0.9]$. The experimental design was conducted using a 5-fold cross validation, with 10 repetitions per each fold.

4.3.2 Metrics Considered for Evaluation

The metrics considered in this brief are the Root Mean Square Error (RMSE) and the Number of Hidden Nodes (NHN). Given that all the models consider fully connected neurons, NHN is a measure of the size of the neural network. Neural Networks are very sensitive to this value (generally, large networks require a higher value of NHN and a longer processing time (Crone and Dhawan, 2007)).

4.3.3 Algorithms Selected for Comparison Purpose

In order to evaluate the performance of the RPUNN and ARPUNN models, they have been compared to some of the most promising neural networks models for TSF. Aiming to outline different characteristics of the methods, the compared methods have been grouped in two sets. The main objective behind the first set of models is comparing ARPUNN and RPUNN methods to baseline algorithms. This set is composed by the following algorithms:

- A Nonlinear Autoregressive Neural Network (NARNN) (Chow and Leung, 1996), which parameters have been determined by the Broyden-Fletcher-Goldfarb-Shannon gradient-based algorithm (Nawi et al., 2006).
- The Echo State Network (ESN) (Rodan and Tiňo, 2011).
- The Extreme Learning Machine (ELM) method (Huang et al., 2012).

The second set of models is selected with purpose of analyzing the performance of the PU basis functions for TSF. Due to this, the two models proposed are compared to ANNs models trained with the same algorithm, but considering other basis functions. The models employed in this set are:

- The Nonlinear Autoregressive Radial Basis Function Neural Network (NARRBFNN).
- The Nonlinear Autoregressive Sigmoid Neural Network (NARSIGNN).

All the hyperparameters considered in this chapter were estimated by a nested five-fold cross-validation procedure. The metric considered to determine the best configuration of parameters was the RMSE. The most important hyperparameter was NHN, and the range of possible values considered for model selection depends on the model:

- In the case of the NARNN, NARRBFNN, NARSIGNN, ARPUNN and RPUNN algorithms, the experiment was carried out considering the set $S \in \{5, 10, 15, 20\}$.

5Scaling the input data to positive values is required to avoid having complex numbers as output of the basis function. Additionally, the scaling considered also avoids having inputs equal to zero or one.
The ESN and ELM algorithms require a higher number of hidden neurons that can supply the network with sufficiently informative random projections (Huang et al., 2012). In this case, the set of hidden nodes considered is formed by \( S \in \{10, 20, 50, 100, 150, 200, 300\} \).

Further considerations on the parameters values for the models can be found in the supplementary material of the paper.

4.4 Results

For all of the 29 data series, models were trained, predictions were made on the test set, and the RMSE and NHN were computed for these predictions. The detailed tables of results (for the two metrics considered) can be found in the supplementary material of this brief. Table 4.1 reports the averaged results over all the series for the methods compared (including the averaged value for the metric and the averaged ranking). As can be seen in Table 4.1, the RPUNN model yielded the best mean in RMSE \((RMSE = 0.0583\) and \(R_{RMSE} = 1.793\)) followed by the ARPUNN model \((RMSE = 0.0624\) and \(R_{RMSE} = 2.7586\)). Despite this improvement, there are some datasets where the results were not as accurate as expected, which might be caused by the degrees of freedom of the models proposed. On the other hand, the minimum \(\overline{NHN}\) is obtained by the ESN model with a mean of 12.06 followed by the ARPUNN model with a mean of 12.65. In terms of ranking the best results are obtained by the ARPUNN model with a 2.46 mean position followed by the ESN model with a 2.75 mean position. The RPUNN does not obtains as good results as in the case of the RMSE getting a \(\overline{NHN}\) of 15.10 and a \(R_{NHN}\) of 4.10. These outcomes show that the ARPUNN model is highly competitive regarding the simplicity of the network. However, in the case of the RPUNN model, in order to get the best performance it requires a larger architecture leading to a higher NHN. A boxplot of the results obtained for the ranking of the \(RMSE_G\) and NHN can be seen in the Figure 4.2 where it can be appreciated the performance above mentioned.

Table 4.1: Summary of results for RMSE and NHN as the test variables, including results of the Holm test for rankings.

<table>
<thead>
<tr>
<th>Model</th>
<th>(RMSE_G)</th>
<th>(R_{RMSE_G})</th>
<th>(\overline{NHN})</th>
<th>(R_{NHN})</th>
</tr>
</thead>
<tbody>
<tr>
<td>NARNN</td>
<td>0.0652</td>
<td>3.7586</td>
<td>14.37</td>
<td>3.41</td>
</tr>
<tr>
<td>NARRBFNN</td>
<td>0.0881</td>
<td>5.3448</td>
<td>14.82</td>
<td>3.68</td>
</tr>
<tr>
<td>ESN</td>
<td>0.0718</td>
<td>4.3793</td>
<td>12.06</td>
<td>2.75</td>
</tr>
<tr>
<td>ELM</td>
<td>1.1155</td>
<td>5.1379</td>
<td>163.69</td>
<td>6.70</td>
</tr>
<tr>
<td>NARSIGNN</td>
<td>0.0689</td>
<td>4.8275</td>
<td>16.58</td>
<td>4.86</td>
</tr>
<tr>
<td>ARPUNN</td>
<td>0.0624</td>
<td>2.7586</td>
<td>12.65</td>
<td>2.46</td>
</tr>
<tr>
<td>RPUNN</td>
<td>0.0583</td>
<td>1.7931</td>
<td>15.10</td>
<td>4.10</td>
</tr>
</tbody>
</table>

The best result is in bold face and the second one in italics.

C: control method (Holm test)

•: significant differences wrt. the control method (Holm test)

The results provided in this brief were validated using non-parametric tests. The Friedman test detected significant differences on a significance level of
Figure 4.2: Boxplot for the average ranking of $RMSE_G$ (RMSE over the generalisation set) and NHN over the 29 datasets $\alpha = 0.10$. Based on this fact, the Holm test was applied (with the same level of confidence), considering as the control method for the $RMSE_G$ variable the RPUNN algorithm, and the ARPUNN method for the NHN variable, because they obtain the best mean ranking for these metrics. The Holm test shows that the RPUNN model performs significantly better than the rest of the models considered. Regarding the NHN metric, the ARPUNN method shows a significant lower size than the NARRBFNN, RPUNN, NARSIGNN and ELM methods. A fully description of the non-parametric tests applied and the results of all of them are included in the supplementary material of this brief.
Chapter 5

Conclusions

This Ariadna study has introduced two different tools for analysing paleoclimate data: 1) a novel genetic algorithm (GA) (covered in Chapters 2 and 3), and 2) two novel time series forecasting (TSF) models.

The GA, taken from the field of time series segmentation, has been applied to paleoclimate data to identify common patterns that would act as early warning signals for abrupt climate change. The segments are represented in a six-dimensional space with dimensions corresponding to statistical metrics that contain information about the system undergoing a critical transition, and they are automatically grouped by a clustering algorithm to uncover common prototypes of segments throughout the time series. These common patterns can be visualised in a straightforward manner by looking at their segment class label. The GA presents differentiating characteristics with respect to previous time series segmentation algorithms, specially in the generation of the initial population, in the mutation operators based on moving cut points and in the fitness function. The clustering process and the GA complement each other with the final aim of achieving a higher level representation of the time series information. Despite being a stochastic algorithm, the GA shows a robust behaviour in different datasets, independently of the algorithm seeds, with very low standard deviations for the fitness values.

Experimental results show that early warning signals of Dansgaard-Oeschger events could be robustly found for several of these events in the form of an increase in autocorrelation, variance, and mean square error in both GISP2 and NGRIP δ¹⁸O ice core data. The GA applied to NGRIP δ¹⁸O ice core record showed that increasing autocorrelation coefficient cannot be solely used as an indicator of climate change. The quantitative results presented in this chapter strongly support the hypothesis of stochastic resonance model brought forward to explain abrupt Dansgaard-Oeschger climate events. Finally the proposed approach provides a novel visualisation tool in the field of climate time series analysis and detection of critical transitions.

On the other hand, the GA has been complemented by the evaluation of different fitness functions and a new method for automatically assessing the performance of the algorithm. From the different experiments, the Davies-Bouldin index presented in Section 3.1 is the best performing fitness function.

Future work about this first contribution includes extending the method to find early warning signals and considering other time series datasets, mutation
and crossover operators and fitness functions.

This study has also proposed two new models of artificial neural networks (ANNs) based on the use of product units (PUs) as a basis function for TSF. The interest on the use of PUs arise from its ability to express strong interactions between input variables, a feature truly important in TSF where there is autocorrelation between the lagged values of the time series. Two models of PU neural networks (PUNNs) have been implemented, the autoregressive PUNN (ARPUNN), and the recurrent PUNN (RPUNN) which consists on an enhanced version of the ARPUNN. The architecture of ARPUNN considers a short-term memory provided by the lagged values of the time series, whereas the RPUNN model includes an additional set of inputs supplied by a reservoir network which provides a long-term memory to the model. The parameters of the models were determined by a hybrid learning algorithm that combines a global and a local search methods (the CMA-ES algorithm and the use of the MP generalised inverse, respectively). The proposed models have been implemented, tested and compared to the state-of-the-art ANNs for TSF. The results show that the introduced models present a very good performance in terms of root mean squared error (RMSE).
Bibliography


