WORDT
NIET UITGELEEND

ningen Faculteit der Wiskunde en Natuurwetenschappen

Wiskunde en
Informatica

Novelty Detection in Time Series

Comparison between a Clustering and Predicting Neural Network

J.S. Spiekstra

supervisors:
Dr. ir. J.A.G. Nijhuis
Drs. M. van Veelen

August 2000

- 6 OKT. 2000

Rijksuniversiteit Groningen
Bibliotheek
Wiskunde / Informatica / Rekencentrum
Landleven 5
Postbus 800
9700 AV Groningen

Technische Informatica

RuG

Abstract

This report discusses novelty detection in time series. Particularly novelty detection with clustering and time series prediction using a neural network. The question discussed in this report is weather a clustering of static relations or the modelling of dynamics in time series provides a better approach for novelty detection.

This question originates from a problem of novelty detection in radiotelescope data gathered by the NFRA/Astron and is posed as a hypothesis after introducing novelty detection in chapter 1.

To answer the question we discuss how to handle time series, specially how to extract information from time series and what kind of novelties can be present in time series. Also important are measures for the quality of a novelty detection system. These topics are discussed in chapter 2.

Another necessary ingredient for answering the question are implementations of the two approaches. The Kohonen self–organizing feature map is chosen as the most appropriate implementation for novelty detection based on clustering static information of a time series system, while a Tapped Delay Line Multilayer Perceptron is used for novelty detection based on time series prediction using dynamic information.

The background of the used neural network architectures and experimental results of these two neural network methods are discussed in chapters 3 and 4.

The results of the experiments and some of the discussed quality measures are used to give the final answer to the hypothesis stated in chapter 1. This answer is presented in chapter 5 together with recommendations for further research.

Contents

Abstract **1**

Contents **2**

Introduction **4**
1.1 Novelty detection 4
1.2 Novelty detection background 5
1.3 Novelty detection problem description 5
1.4 Novelty detection with neural networks 8
1.5 Novelty detection in time series 9
1.6 Thesis problem description 10

Concepts **13**
2.1 Introduction 13
2.2 Handling time series 13
2.3 Quality measures 16
2.4 Used time series and novelties 17
2.5 Conclusion 19

Novelty detection with clustering **20**
3.1 Introduction 20
3.2 ART 20
3.3 The Receptive Novelty Detector (RND) 22
3.4 Self organizing Feature Maps 24
3.5 The Kohonen map 25
3.6 Novelty detection with Kohonen 26
3.7 Conclusion 29

Novelty detection with time series prediction **30**
4.1 Introduction 30
4.2 Time series prediction 30
4.3 The Multilayer Perceptron (MLP) 31
4.4 Time series prediction with the TDL−MLP 33
4.5 Novelty detection with the TDL−MLP 34
4.6 Conclusion 36

Conclusion and Discussion **37**
5.1 Overview 37
5.2 Experiments 37
5.3 Discussion and further research 37

References **39**

Appendixes
Data sets **41**
A.1 Markov model data set 41
A.2 Circle data set 43
A.3 Sinus with two frequencies 44

List of figures

Figure 1.1 The world of novelty detection 5
Figure 1.2 a) non−linear separable sets b) linear separable sets 6
Figure 1.3 Traditional novelty (=fault) detection, Isermann 1984 7
Figure 1.4 Examples of abrupt and incipient novelties in time series 8
Figure 1.5 Clustering, classification, identification 9
Figure 1.6 Two realisations of the univariate time series 1.1 10
Figure 1.7 Radiotelescope signals 11
Figure 2.1 Crude prices off oil per day 14
Figure 2.2 Daily returns of oil prices from figure 2.1 14
Figure 2.3 Fourier transformation 15
Figure 2.4 A Markov model 18
Figure 3.1 ART 21
Figure 3.2 Schematic algorithm for RND 23
Figure 3.3 Sinus with changing sample frequency 23
Figure 3.4 Index of winning receptor 24
Figure 3.5 cummultative sum of anti−receptors created without initial training 24
Figure 3.6 A kohonen map 25
Figure 3.7 Neighborhoods in the outputlayer 25
Figure 3.8 Neighborhoods in circular lattice of 20 output neurons 27
Figure 3.9 Trade off 27
Figure 3.10 Sensitivity graph 28
Figure 3.11 Trade off 29
Figure 3.12 sensitivity graph 29
Figure 4.1 Example of a feedforward neural network 31
Figure 4.2 a TDL−MLP with one input and a delay line of size 5. 33
Figure 4.3 Trade off 35
Figure 4.4 Sensitivity graph 35
Figure 4.5 Trade off 36
Figure 4.6 Sensitivity graph 36
Figure 5.1 Trade off Kohonen map and TDL−MLP 38
Figure A.1 First 500 values of x1 and x2 42
Figure A.2 scatterplot of normal data set 42
Figure A.3 scatterplot x1 and x2 in the novelty state and its transitions 43
Figure A.4 scatterplot of normal data set 44
Figure A.5 scatterplot of data generated in the novelty state 44
Figure A.6 Sinus with frequency 10 Hz and samplefrequency 190 Hz 45

Chapter 1

Introduction

1.1 Novelty detection

Before we discuss the central thesis problem we want to make clear what is meant with novelty detection and novelties in this thesis. This is presented in the first three paragraphs. In paragraph 1.4 and 1.5 novelty detection with neural networks and novelty detection in time series is discussed. In paragraph 1.6 the actual thesis problem and a hypothesis is presented.

Novelty detection is a research area with a wide variety of applications and with even more methodologies and a diverse nomenclature. In literature novelty detection is often called fault detection or abnormality detection. The term fault detection is often used in the mathematical systems theory, statistics and electrical engineering branch. Abnormality detection appears to be the more general term of novelty detection. The term novelty detection itself is typically used in relation with neural networks. In this thesis mostly the last term is used, as this thesis discusses novelty detection with neural networks.

It's important to get a clear understanding of what is mend with a novelty and novelty detection, in order to understand the issues discussed in this thesis.

According to *Webster's ninth new collegiate dictionary* a novelty is:

> "Something new or unusual"

This definition is to general for our research. Lamberts et al. define a novelty in [16] as:

> "Something which doesn't resemble anything we have seen before"

This definition already gives a better and tighter description of the term, but a very specific definition (definition 1.1) of a novelty in data (e.g. a time series) is proposed by Van Veelen, Nijhuis and Spaanenburg in [30]. They state:

Definition 1.1 (Novelties)
Novelties[1] are discrepancies between observed and expected behavior of a data generation process

Definition 1.1 is used in this thesis because it implies some process generating data, which can be seen as a time series. Moreover the novelties are described as difference between normal, expected, behavior and the actual, observed, data. This resembles the way in which we detect novelties as will be discussed later in this thesis.

Remains us to give a definition of the main topic of this thesis, novelty detection. A good definition for novelty detection is the following definition of Lionel Tarrasenko.

Definition 1.2 (Novelty detection)
Novelty detection is the process of building a statistical model of normality in feature space, determining the boundaries of normal data, and then using a means to identify when data has moved outside these boundaries. [Tarrasenko (1997)] □

Definition 1.2 is a very useable definition because it corresponds with the basic recipe for novelty detection presented in paragraph 1.3. But before we discuss the basic concepts of novelty detection in general, first some background is presented.

1. In the original definition the word abnormalities is used. In this context abnormality can be seen as synonym to novelty.

1.2 Novelty detection background

Novelty detection has a wide variety of applications. Some examples are gearboxfailure detection [6][13], character recognition [11][16][26], anomaly detection in MRI images [4][17], power system faults analysis [14][25], motor failure detection [21][24][33] and other fields where huge amounts of data is gathered, e.g. radio astronomy (see paragraph 1.6).

Most paradigms and methodologies of novelty detection are stated in the field of mathematical systems and electrical engineering. The research to novelty detection took in the 1990s also a flight in the area of computational intelligence, in specific the neural networks branch. Many people saw the benefits of neural networks over classical statistics. In this thesis the emphasis is at neural networks.

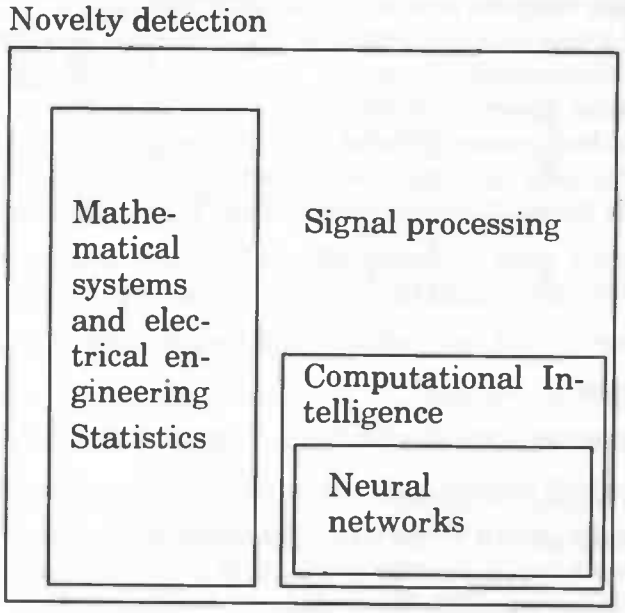

Figure 1.1 *The world of novelty detection*

Despite all the different points of view towards novelty detection the main problems for novelty detection are very generic. The basic problems and concepts are discussed in the next section.

1.3 Novelty detection problem description

Looking at the definition for novelty detection by Tarrasenko (def. 1.2) it's quiet clear it implies the following pre−condition for novelty detection:

"There is a model $\mathfrak{F}$ describing the normal dataset A"

Thus the novelty set is given by the complement $\overline{A}$. One of the keypoints of novelty detection is to find this model $\mathfrak{F}$. Thereby the aim of novelty detection is to create the model in such a way that data samples are transformed into a space where expected (set A) and abnormal data (set $\overline{A}$) can be easily separated. Ideally novelties will be orthogonal to expected signals in this space so they can be linearly separated [30]. In figure 1.2 an ideal linear separable situation and a non−linear situation are drawn.

The next problem, determining the boundaries of the normal dataset is also stated in the definition. Determining the boundaries is often done by determining the boundaries of the

residuals generated by the modelling of the normal data, because usually there isn't a perfect model for the data witch results in that the boundaries of the real data can't be used for the distinction between normal and novelty data.

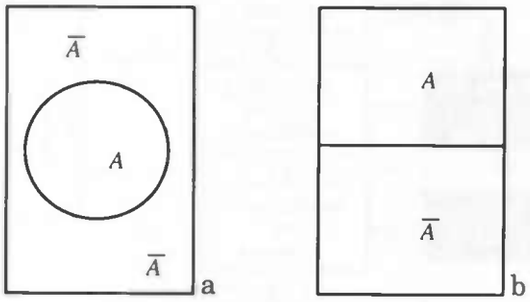

Figure 1.2 *a) non−linear separable sets b) linear separable sets*

The last step of novelty detection is to find a way to identify when data has moved outside the boundaries of the normal data model. In most novelty detectors this is done by some kind of an error−threshold, but sometimes other methods are used (e.g. learning behavior of a model).

Traditional approaches sometimes avoid the determination of the boundaries of the normal data and identifying when data has moved outside these boundaries by assuming the novelty data is known and can be modelled. This leads to these three stages of traditional novelty detection [30][8] :

- Model selection/construction
- Computation of signature (residuals)
- Comparison of signature of observed process with that of known normal/faulty process

In figure 1.3 these steps are plotted in a diagram. For this thesis only the middle section marked as fault detection is of interest.

In contrary with the last stage of traditional novelty detection nowadays it's assumed that the faulty process is not known. This is much more logical because the novelty set $\overline{A}$ of a normal set A can be infinite. So the comparison of signatures of the observed process with that of known faulty process is left out by modern novelty detection.

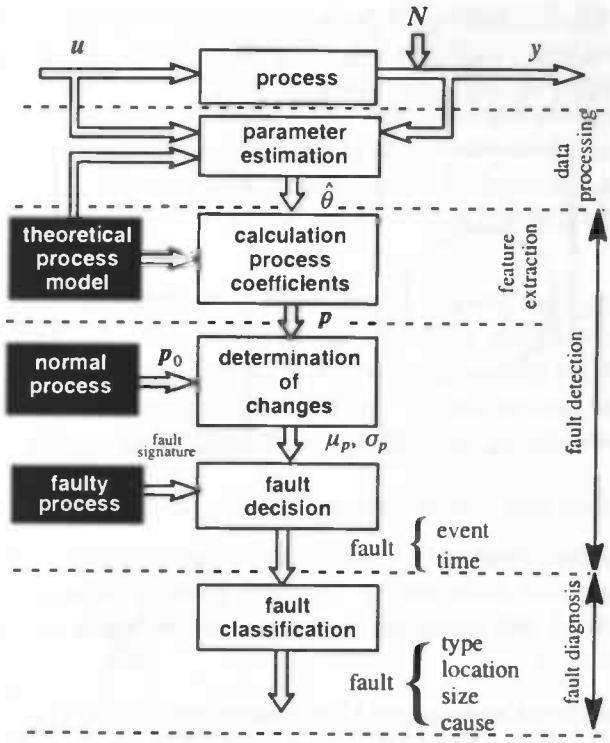

Figure 1.3 *Traditional novelty (=fault) detection, Isermann 1984 This thesis discusses the middle section, fault detection, of the diagram.*

In the third stage of novelty detection a novelty–identification method is used for the comparison and the decision weather a datapoint (or segment of datapoints) is a novelty or not. The novelty–identification method is of great influence to the success of the novelty detection. The novelty–identification method depends on what kind of novelty is to be detected. In literature novelty kinds are characterized by there time–evolution. According to Baskiotes and Rault [30], these types are:

- cataleptic: absent time–evolution, random
- abrupt: sudden change/jump or switching
- incipient: slow change, evolution and drifting

The last two novelty kinds are used in this thesis. Examples of these kinds are plotted in figure 1.4.

The last two noveltykinds can be time–related, however most research is aimed at non time–related novelties. This is because most methods in literature are using a classification or clustering method to detect novelties, these methods often don't use time as a variable which makes it almost impossible only to detect *time*–related novelties. In literature a time–related novelty is called a disturbance.

This thesis discusses the detection of non time–related abrupt and incipient novelties in time series with the use of neural networks. The next section gives an introduction in novelty detection with neural networks.

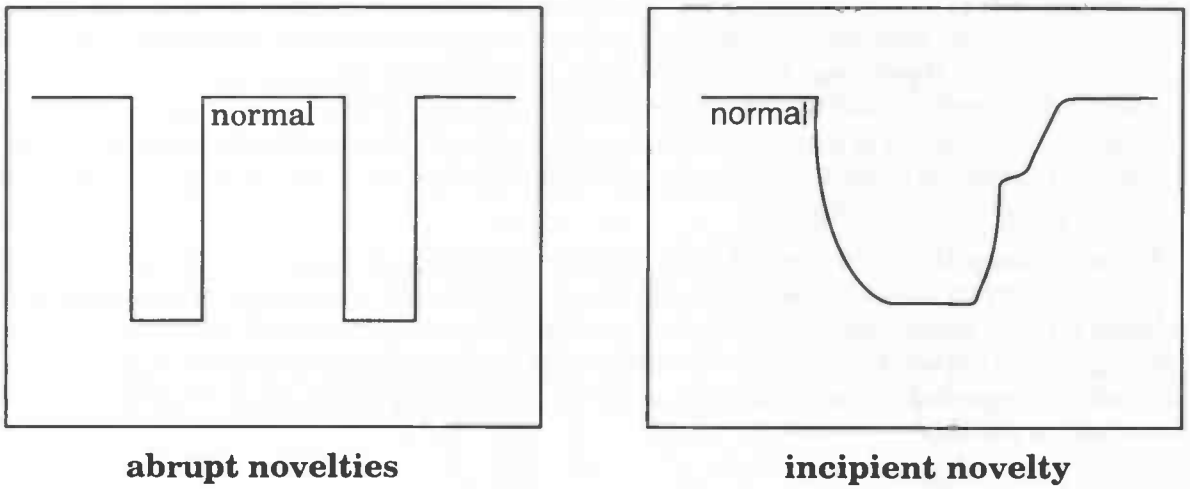

Figure 1.4 *Examples of abrupt and incipient novelties in time series*

1.4 Novelty detection with neural networks

The idea behind artificial neural networks is to simulate the way the brain performs a certain task or function of interest. The architecture and terminology of the neural network thus finds its roots in neuroscience. In Haykin [9] an artificial neural network is defined as an adaptive machine:

Definition 1.3 (neural network)
A *neural network* is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects: □

1. Knowledge is acquired by the network through a learning process.
2. Interneuron connection strengths known as synaptic weights are used to store the knowledge

The most important advantages of neural networks are:

- Generalization
- Non–linearity
- Context adaptivity
- Fault tolerance / robustness

See for more details on neural networks the book of Haykin [9] or Bishop [1].

Novelty detection with neural networks is, as other non–neural methods, based on making a model for the normal data. This can be done in three ways:

- classification
- clustering
- identification

The first two methods look similar. Both methods make an mapping between inputvariables and a certain class or cluster. There is a big difference in these two methods though.

This difference is in learning methodology. A neural network for classification is trained supervised, while a clustering neural network is trained unsupervised (e.g. self–organizing featuremap). This immediately shows the disadvantage of a classifying neural network in comparison with clustering. For classification the different classes in the data have to be known, while with clustering this isn't necessary. The way of detecting novelties with clustering or classification is in general the same though. In both cases an distance to a certain (winning) cluster or class is calculated and based on this result a sample is marked as a novelty sample or normal sample.

The basic idea of the third method, identification, is finding an input–output model, where input and output variables belong to the observed data. An identifying neural network is always trained supervised, an example of a identifying neural network is a time series predicting neural network. Novelties can be detected by measuring the error between network output and expected output. Then this error can be used for the decision whether a sample is or isn't a novelty.

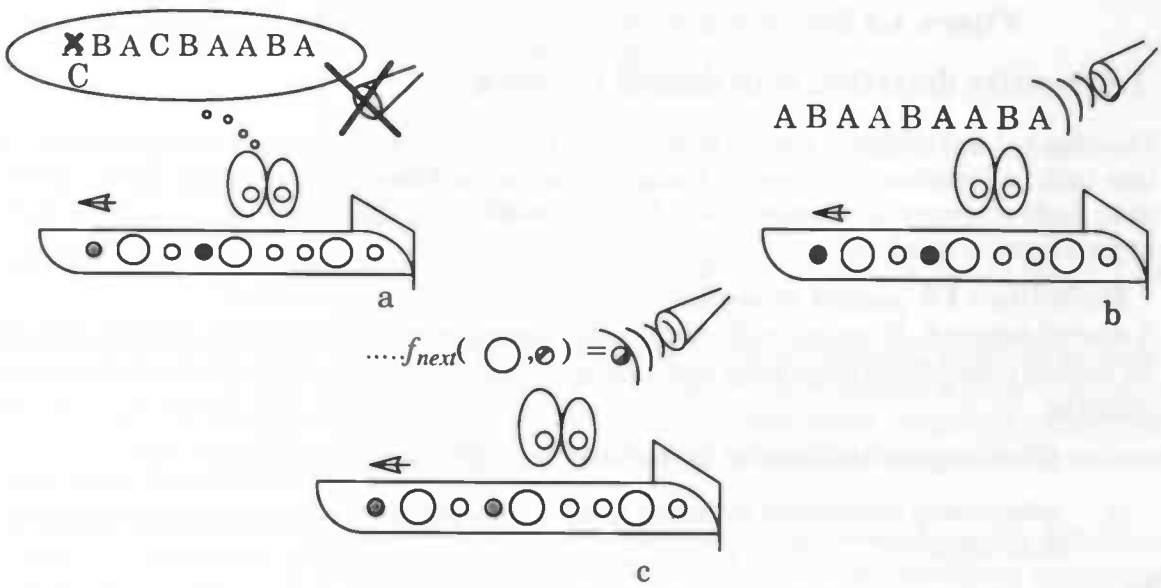

Figure 1.5 *Clustering, classification, identification a) Unsupervised clustering: features: color and size, thus three clusters, b) Supervised Classification: feature: size, color is ignored by the supervisor, c) Identification: prediction of the next object in the series*

1.5 Novelty detection in time series

Time series can be characterized as a string of values with as index time. Time series are studied by people in many different fields. In meteorology weather information is studied as a function of time and in the financial business stock exchanges and interest rates are examples of time series.

In mathematics time series are modelled as a stochastic process where a stochastic process is a family of real–valued random variables $(X_t)_{t\in T}$ defined on a probability space $[\Omega, \mathfrak{F}, P]$. The functions $(X.(\omega))_{\omega\in\Omega}$ are called realizations, trajectories or sample paths of the process. [Mikosch, Time series analysis]

Systems producing time series data can be univariate or multivariate. Univariate time series are time–series with at each t one value. Multivariate time series meanwhile are time series with at each t two or more values. For example:

$$X_t = 0.8 * X_{t-1} + \mathcal{N}(0, 0.01) \tag{1.1}$$

is a univariate time series, but the system

$$X_{1t} = -0.6 * X_{2t} + 0.8 * X_{1(t-1)} + \mathcal{N}(0,1) \wedge X_{2t} = 0.9 * X_{2(t-2)} + \mathcal{N}(0,1)$$

produces a multivariate (bivariate) time series.

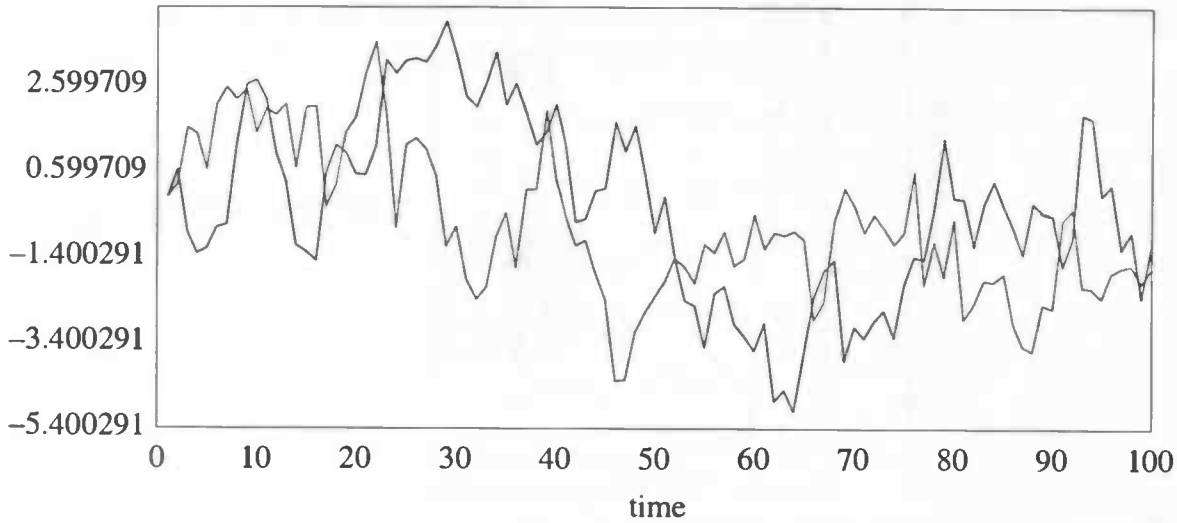

Figure 1.6 *Two realisations of the univariate time series 1.1*

To detect novelties in a multivariate or univariate time series system information from the history of the time series can be used, this kind of information is called *dynamic information*. In other words, dynamic information is the relation between past values of a time series. When dealing with a multivariate time series there is also *static information* in the time series. Static information of a multivariate time series is the relation between the variables at an certain time t. By clustering methods for novelty detection sometimes this static information is used, as showed in chapter 3. In chapter 4 only dynamic information is used. In the next chapter more is told about handling time series.

1.6 Thesis problem description

In the preceding paragraphs we introduced novelties, novelty detection, neural networks and time series. These subjects form the ingredients for my thesis subject: Novelty detection in time series using a clustering and predicting neural network.

The problem was initiated by the Netherlands Foundation for Research in Astronomy (NFRA) and 'Stichting Astronomisch Onderzoek in Nederland' (ASTRON). They record radiosignals from the universe with huge radiotelescopes. The recorded signals contain discrepancies from normal background noise caused by an astronomical object or by human interference such as mobile telephones. An example of measured data with the radiotelescopes is printed in figure 1.7. The descrepancies ,or novelties as we call it, caused by astronomical objects are of interest for the researchers. Because there are huge amounts of data, automatic detection of the novelties is of great interest. Nowadays the detection is done by a human operator, which costs a lot of valuable time.

We chose two approaches to the automatic detection of the novelties. The first is modelling the 'normal' signal by clustering based on the static information of the set of signals (time

series). The second is based on predicting the next values of all signals using dynamic information of the signals. The concepts of static and dynamic information are already discussed in paragraph 1.5, but dynamic information is thoroughly discussed in paragraph 2.2.2.

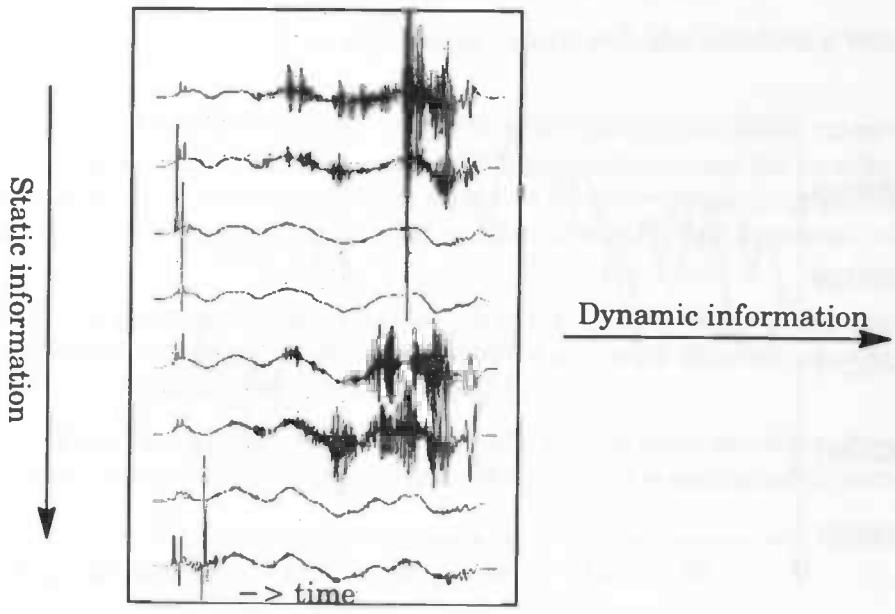

Figure 1.7 *Radiotelescope signals Each signal represents a certain frequency range. The peaks are novelties from normal background noise.*

The detection is based on the difference of expected behavior and real behavior as mentioned in definitions 1.1 and 1.2. If this difference is larger then normal a detection takes place. For prediction this is the difference between predicted value (expected value) and measured value (real value) and for a clustering method this is the distance to the nearest cluster (best fitting cluster).

The goal of this thesis is to give a comparison between the two presented novelty detection methods. In literature such a comparison isn't made. Moreover in literature clustering methods are mostly used while time series prediction is very rare. Only in classical statistics time series models and the realizations of these models are used to detect novelties. Though we think time series prediction performs better when used for novelty detection because it models the real time series due to the use of dynamic information, while the clustering method models only some configurations of static information. This leads to the following hypothesis.

Modelling the time series with dynamic information and predicting the next value of the series is expected to perform better then clustering the static information of the system.

To give this hypothesis a scientific basis we present in chapter 3 and 4 two experiments. The data used for these experiments are artificial time series and have resembling characteristics compared to the NFRA/ASTRON time series.

Concluding, this chapter presented the basic ideas and ingredients for this thesis. The next chapter discusses some general concepts, mentioned in literature, necessary for novelty detection in time series. The next chapter will also present some quality measures for novelty detectors which can be used for comparing the two methodologies for novelty detection in time series systems with each other and will discuss the principles of the time series used for our experiments.

After chapters 3 and 4, in which the used neural architectures and the results of the experiments are presented, the conclusions are stated in chapter NO TAG. Thus in this last chapter the question will be answered weather the mentioned hypothesis holds or not.

Chapter 2

Concepts

2.1 Introduction

As mentioned at the end of the first chapter we discuss in this chapter some other important concepts for novelty detection in time series. An important concept for novelty detection in time series, certainly in real–world time series, is the pre–processing and analysis of the time series. Some techniques, mentioned in the literature, are discussed in paragraph 2.2.

Furthermore the goal of this thesis is the comparison of two novelty detection techniques. To compare methods with each other quality measures for a novelty detector are necessary. Some quality measures are presented in paragraph 2.3.

For the comparison of the two novelty detection methods we also need test data. How these artificial time series are generated and how the are characterized is explained in paragraph 2.4.

2.2 Handling time series

While novelty detection in time series stands by finding a model for the 'normal' time series this section describes some tools to get such a model from the information in time series. Information in some kind of data is often called features of the data. The model describing the data is based on these features. Some features are more important than others, e.g. when dealing with a classification or clustering problem a feature which is the same in every class isn't interesting while a feature which differs per class is. To visualize the important or interesting features from data or time series, preprocessing can be used.

2.2.1 Stationarity and Preprocessing

The first step of preprocessing time series is often making the time series more or less stationary. To extract features from a time series there should be some regularity in the time series, more precise there should be similar patterns or behavior in different segments of the time series. In mathematical language the time series $(X_t)_{t \in \{\mathbb{N} \bigcup -\mathbb{N}\}}$ is said to be *stationary* if the following relations hold [Mikosch, time series analysis]:

3. $E|X_t|^2 < \infty, t \in \{\mathbb{N} \bigcup -\mathbb{N}\}$
4. $EX_t = m, t \in \{\mathbb{N} \bigcup -\mathbb{N}\}$ *for a constant m*
5. $\gamma_X(s,t) = \gamma_X(s+h, t+h)$ *for all s, t, h* $\in \{\mathbb{N} \bigcup -\mathbb{N}\}$

Where $\gamma_X(s,t)$ is the autocovariance function.

In real–life time series, e.g. stock exchanges, often a trend is present. One way to get rid of this trend is looking to the difference of a past value and current value, this is called a return. In financial engineering the following 'returns' of the original time series X_t are proposed [Mikosch, time series analysis]:

- $Y_t = \dfrac{X_t - X_{t-1}}{X_{t-1}}$

- $Y_t = \ln(X_t) - \ln(X_{t-1}) = \ln\left(\frac{X_t}{X_{t-1}}\right)$

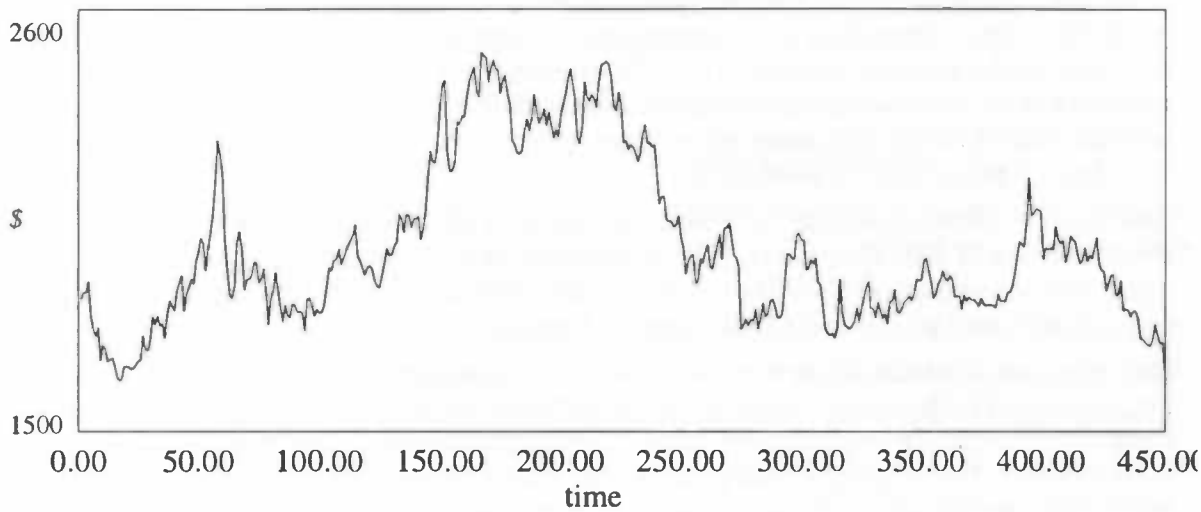

Figure 2.1 *Crude prices off oil per dayThis time series isn't stationary*

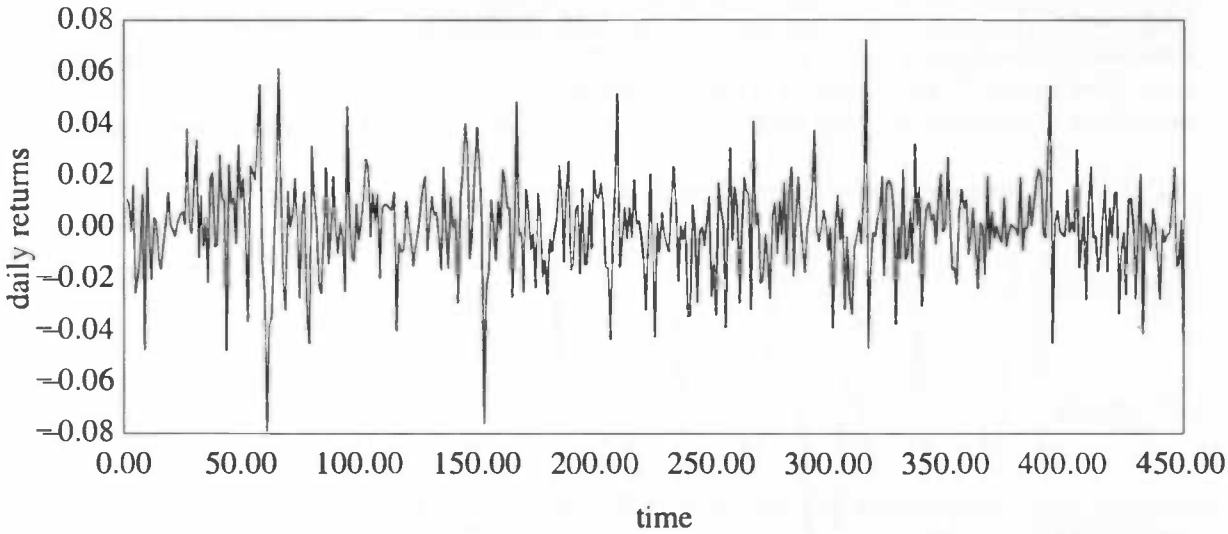

Figure 2.2 *Daily returns of oil prices from figure 2.1 This time series is stationary*

The first return is the normal return, the second is called the logreturn.

When time series are generated by a system which can be in two or more states, the time series generated from this system are at first sight not stationary because the third condition for stationary doesn't hold for the total signal. When this is the case the time series can still be multi−stationary, which in this thesis means that the time series consists of a finite number of stationary parts. Time series with this kind of stationarity are used in the experiments described in in chapter 3 an 4 of this thesis.

Once the time series is made stationary further preprocessing can be conducted. Following there are two ways discussed for further preprocessing a (stationary) time series:

- Transformation to frequency domain

- Quantification

Transformation to frequency domain is often used by applications using speech or vibration signals. The basic idea of a transformation from time to frequency is making a histogram of how often a certain frequency is present in a signal. There are two commonly used methods of time–frequency transformation. The first is the Fourier transformation (figure 2.3), this method is used in article [24]. The second method is wavelet transformation. This method has its roots in image–processing but lately it is also used in time series analysis because this method has some advantages over Fourier transforms [DeVore, Bradley: Wavelets]. This method is used in [6].

Another not so common way to determine peaks in the frequency domain is proposed by Brotherton et al. [2]. They use an autoregressive (AR) process to model a time series segment, the parameters of the AR process are then used to determine peaks in the frequency domain and used as input for their novelty detector.

Quantization of a time series is mostly used as preprocessing for a clustering or classification method. The basic idea of quantization is the division of a continuous range into a discrete range. In other words, every value X_t of the time series is put in a bin corresponding with its value. Every bin has a range of fitting values. Normally the means of the binranges are distributed equidic on the original range. For example, to quantize the continues range $[-1 \ldots 1]$ into 10 bins the bin ranges are $\{[-1 \ldots -0.8)[-0.8 \ldots -0.6), \ldots, [0.8 \ldots 1]\}$, so the series $[-1.000, -0.754, -0.465, -0.518, -0.152, 0.423, 0.923]$ can be transformed into the series $[1, 2, 3, 3, 5, 8, 10]$. This method is used in the articles [3] and [5].

Besides preprocessing there is another important factor for finding an appropriate model for a time series. In paragraph 1.5 we mentioned the use of dynamic information by novelty detection in time series. The next subsection will discuss this topic more thoroughly.

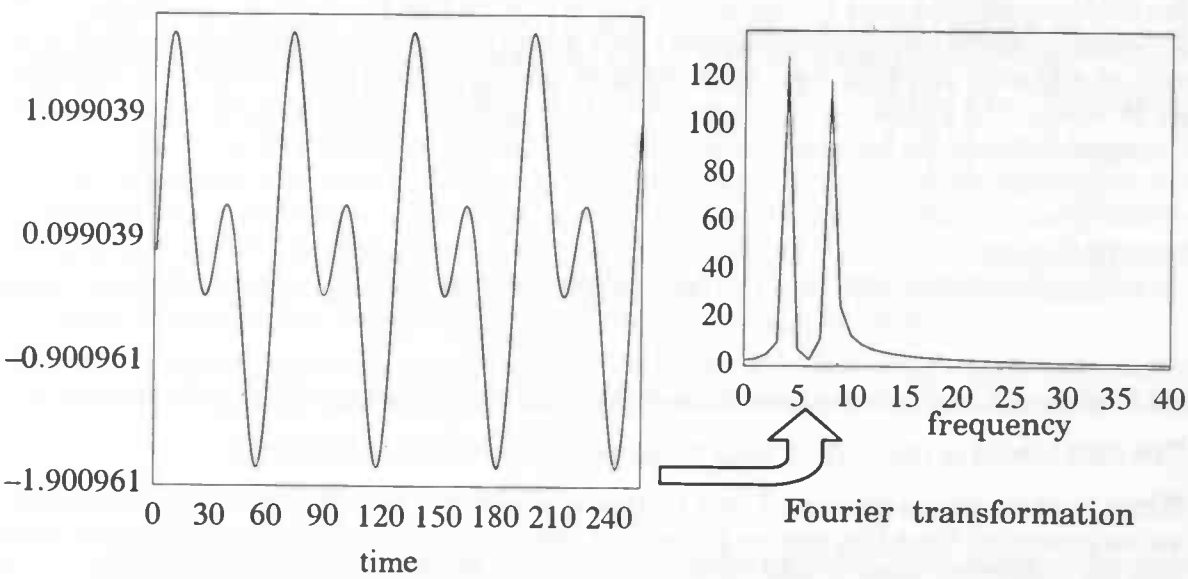

Figure 2.3 *Fourier transformation The original time series is the summation of two sinoídals with frequencies of 4 Hz and 8 Hz. This is clearly visible in the plot of the Fourier transformation*

2.2.2 Dynamic information in time series

The dynamic information of a time series is its history, thus the information given by the values of the time series in time. Dynamic information is also called *temporal information.*

Dynamic information specially is important for predicting future behavior or values of a time series but can also be used by clustering methods. To catch the dynamic information of a time series in a model, the model has to have some notion of time. According to Van Veelen, Nijhuis and Spaanenburg this can be achieved in three ways [29]:

- explicit notion of time : *tell model what time it is*
- finite time window : *look at a limited number of past values*
- feedback : *look at previous outputs or internal states of the model*

Originally only the first method was used to give a static model (model without notion of time) notion of time. This was done just by adding an extra input with as value the absolute time. In this thesis only the second method is used.

The basic idea of the second method is using as input of the dynamic model[2] a time−window of n values, e.g. $(X_t, X_{t-1}, X_{t-2}, \ldots, X_{t-(n-1)})$. The size of this window is of some importance. The determination of the right size of the time window is often a case of personal experience with time series. Though the autocorrelation function $\rho_X(h) = \frac{\gamma_X(h)}{\gamma_X(0)}$ can give a good estimation of the time−window size.

At the end of this section we know what to notice when modelling a certain time series, but this doesn't say anything about the quality of the model or the novelty detection ability of the model. Therefore in the next section some general quality measures for a novelty detection model are discussed.

2.3 Quality measures

According to Baskiotes and Rault [23] a novelty diagnosis system should have the following properties:

- robustness
- flexibility
- adaptivity
- transparency

For a novelty detection system these quality measures are the same except that there are two extra quality measures:

- sensitivity
- promptness

The first four quality measures are discussed shortly because they aren't used to compare the two novelty detection approaches discussed in this thesis.

The *robustness* of a novelty detector is the sensitivity for small parameter variations such as white noise at the inputsignal. Though a novelty detector shouldn't be to robust because small novelties then aren't detected. In this way the robustness relates with the *flexibility*. Flexibility means that every possible novelty and not one particular kind is detected, thus

2. A dynamic model is a model that has some way of capturing dynamic/temporal information in its parameters. This is also referred to as Short Term Memory (STM), not to be confused with Long Term Memory (LTM) which is the capacity of models to store information of a series [29].

the flexibility is perfect when the model for the normal signal exactly fits the real normal data with as consequence that every slight discrepancy from this model can be detected as novelty.

Adaptivity and Transparency are quality measures for the degree of influence the novelty detector and the data generating system have at each other. A novelty detection model is *adaptive* when structural changes in the data−producing process easily can be included without having to reinitiate the learning phase of the model. *Transparency* means that the actual novelty detection can be done outside the normal process thus without interrupting this normal process.

The last two quality measures are of more importance for this thesis. Both are used for the actual comparison of the two novelty detection approaches.

Promptness is the measure for the speed of the novelty detection expressed in time or the number of samples that have to be processed before the verdict can be given weather one sample is a novelty or not. For clear reasons promptness is a valuable quality measure for real life novelty detection applications.

Sensitivity and quality measures related to sensitivity are the most important in this thesis. The results of the conducted experiments, and thus the two methods for novelty detection discussed in this thesis, are compared mainly using this measures.

With sensitivity is meant the false acceptance rate (FAR) versus the false detection rate (FDR). The FDR is the percentage of the 'normal' datapoints which is detected as a novelty weather the FAR is the percentage of 'novel' datapoints accepted as a 'normal datapoint'. As mentioned before a novelty is detected when the modelerror exceeds a threshold, when this threshold drops the FDR rises and vice versa for the FAR. This relation is shown in a sensitivity graph or as a trade off function..

When detecting novelties the FAR should be as low as possible, thus very close to zero percent. Suppose a situation as in the case of the NFRA/ASTRON mentioned in chapter 1. Here the novelty detector is aimed at reducing the work done by hand. This can be done with a novelty detector with an as high as possible threshold but still a FAR of 0%. In the ideal situation the FDR is then also equal to 0%. In this ideal case the novelty detector is perfect, the signals marked as novelty are real novelties and not just normal signal detected as novelty. The reduction of work done by hand is then optimal. Though in real cases the FDR, by a FAR of 0%, is often greater then 0%, say $x\%$. Thus $x\%$ of normal signal is detected false as novelty, but for $(100-x)\%$ of the normal signal it is sure it doesn't contain any interesting novelties and can be put aside save without missing a novelty.

This last percentage gives a concrete valued, thus a suitable and comparable, measure for the quality of a novelty detector and will be referred to as the reduction rate (RR) in this thesis.

2.4 Used time series and novelties

As mentioned in paragraph 1.6 we use two artificial generated time series systems to compare the clustering and prediction method. These time series are thoroughly discussed in appendix sections NO TAG and NO TAG. In this paragraph we will shortly discuss the general features of, the generating model of and the novelties in the used time series.

The used time series systems have to satisfy four conditions without preprocessing the time series:

- The time series have to contain dynamic information

- The time series system have to be multivariate to contain static information
- The time series have to be at least multi–stationary
- The time series system has to have some resemblances with real life time series

The first condition isn't difficult to satisfy . Time series, by definition, always contain dynamic information. Meanwhile the last three conditions are harder to satisfy.

In real life situations the time series are often (sensor) outputs in time of a device, e.g. a radiotelescope or motor. These devices can be in different states of operation. Thus the behavior of the outputs of the system are state dependant and thus can be described with a state model. Then novelties in these systems are new states aberrant from the known normal operation states.

To simulate this state behavior we use for the generation of our artificial time series a Markov state machine.

A Markov state machine or Markov model is a normal finite state machine but the transition from one state to another occurs with a certain probability, figure 2.4 is a graphical representation of a Markov model.

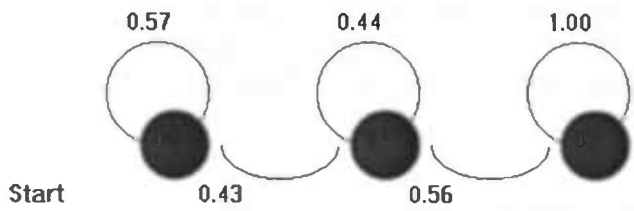

Figure 2.4 *A Markov model The starting state is state 00, the transition probalities are the numbers under or above the transition lines.*

The time series used in this thesis that are generated by a Markov model have two basic stationary functions for all states, thus the time series are bivariate. The static relation between the two functions x_{1t} and x_{2t} in a state is a function f so $x_{1t} = f(x_{2t})$. The basic functions have some adjustable parameters such as mean or factor which values differ per state.

Summarizing this it can be concluded all four conditions are satisfied. Though still there is a problem. The time series contains huge discontinuities at the transition from one state to another. For modelling the time series with dynamic information this is very unpleasant. Beside this disadvantage it also isn't resembling real world processes. In real live processes a change of state takes time, in this time the process is more or less interpolating to the other state. Parallel to this interpretation we interpolate in a number of steps the values of the basic functions parameters of the originating state to the values in the other state when a transition takes place. In this way the time series become smoother and the transitions paths are described by a function, which makes modelling the time series with dynamic information easier.

Due to the 'slow' transition from state to state the kind of novelty in the artificial time series is incipient (see paragraph 1.3).

2.5 Conclusion

In this chapter we discussed how to preprocess a time series, this is especially necessary when dealing with real life time series. The time series used for experiments in this thesis are artificial and don't need any preprocessing.

The results of these experiments can be compared with quality measures. For this thesis the most important one is the sensitivity and the related concept rejection rate.

The time series used for the experiments are generated by a Markov model to simulate common real time series often used in the field of novelty detection.

In the following two chapters the actual neural networks used for the novelty detection are discussed. To test and compare the neural networks the quality measures and the artificial time series presented in this chapter are used.

Chapter 3

Novelty detection with clustering

3.1 Introduction

In this chapter we will discuss three methods for novelty detection with clustering. Two of them are neural network based, the Adaptive Resonance Theory (ART) neural network and the Kohonen neural network, respectively discussed in paragraph 3.2 and 3.5. The method discussed in paragraph 3.3 is a simple linear method but has some similarities with ART.

Eventually we use the Kohonen map to conduct our main experiments. The results are presented paragraph 3.6.

The basic idea of clustering data is modelling the inputspace with a number of prototypes. Every cluster has his own prototype. A vector from the inputspace belongs to the cluster when the vector distance between prototype and input vector is small enough.

An important remark is that clustering should not be confused with classification. Although both methods look like each other the differences in design and learning strategies are huge. The main difference is that a clustering methods creates its prototypes unsupervised, or self−organized, while a classification method uses a supervised strategy.

3.2 ART

The Adaptive Resonance Theory (ART) neural networks belong to the self−organizing clustering neural networks. It was introduced by Stephen Grossberg in 1976. It is claimed that ART solves the stability−plasticity dilemma. An algorithm is called stable if it 'remembers' previously learned knowledge, especially when the algorithm has to learn new knowledge. An algorithm is plastic when previous knowledge is forgotten when new knowledge is offered. The dilemma is posed by Heins and Tauritz as follows [10].

- How can a learning system be designed to remain plastic, or adaptive, in response to significant events and yet remain stable in response to irrelevant events?
- How does the system know how to switch between its stable and its plastic modes to achieve stability without rigidity and plasticity without chaos?
- In particular, how can it preserve its previously learned knowledge while continuing to learn new things?
- What prevents the new learning from washing away the memories of prior learning?

The ART−1 architecture overcomes this dilemma for binary inputvectors and the ART−2 for real−valued inputvectors. The ART−1 architecture is used in the articles [3] and [26], in this section only ART−1 is discussed. The ART−1 architecture consists of two subsystems, the attentional subsystem and the orienting subsystem.

The attentional subsystem consists of two layers of neurons, called F1 and F2. The neurons of this two layers are connected pairwise by synapses with adaptive weights. The connection strengths (adaptive weights) that are emerging from a node in the F2 layer and converging to the nodes in the F1 layer is called a binary template or top−down trace. The con-

nection strengths emanating from a node in the F1 layer and converging to the nodes in the F2 layer are called bottom–up traces [26]. Both belong to the long term memory (LTM) of the system, whereas the F1 and F2 layer belong to the short term memory (STM).

Globally the system functions as follows in steps:

1. An binary input vector is put at the F1 layer.
2. The inproduct of the input vector with all top–down traces is calculated
3. The F2 layer neuron associated with the largest inproduct is signed as winner
4. The binary template associated with the winning F2 layer neuron is compared with the binary input vector by means of the vigilance test performed by the orienting subsystem (novelty detector) :

$$\frac{|I \cap T|}{|I|} \geq \rho \quad \text{, where I is the input vector, T the template and} \quad (3.1)$$

$0 \leq \rho \leq 1$ the vigilance parameter

5. If (3.1) succeeds the old template is updated by (3.2) else if (3.1) doesn't succeed for any other template, a novel input is detected and a new template for this novel input is created.

$$T^{(new)} = T^{(old)} \cap I \quad (3.2)$$

The orienting subsystem is used to control step 5 of the algorithm. A graphical representation of ART is given in figure 3.1.

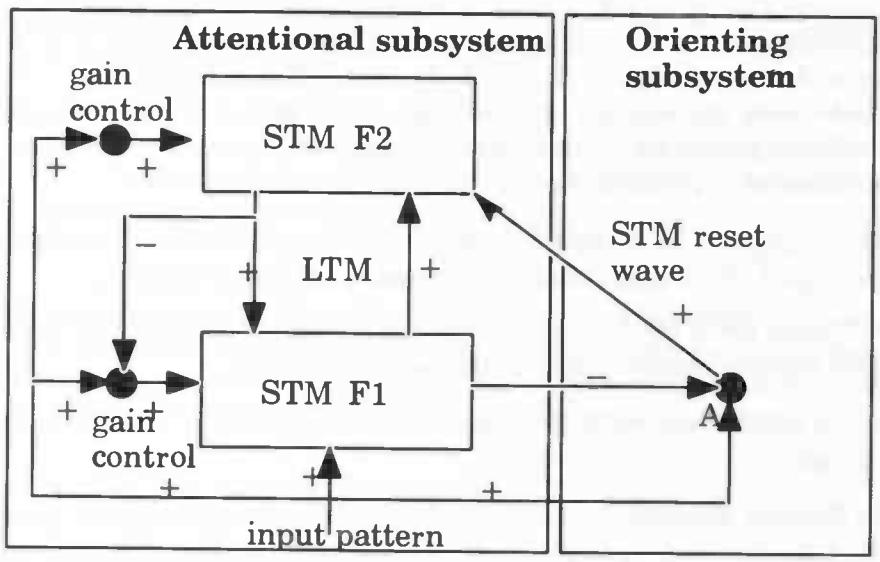

Figure 3.1 *ART Two layers, F1 and F2, of the attentional subsystem encode patterns of activation in the STM traces. Bottom–up and top–down pathways between F1 and F2 contain LTM traces which multiply the signals in these pathways. The remainder of the circuit modulates these STM and LTM processes [10]*

In [3] this system is used for novelty detection in stationary time series. As preprocessing they use a thermometer code for the time series values. The input was a vector of encoded (preprocessed) values of the time series in a time window plus the complements of the en-

coded values. Thus when the time window had size 5 and the number of bits to encode is 20 then the number of input bits would be 2*5*20=200.

In their first experiment they use as normal data a sinus with frequency *f*, the large (abrupt) novelty is an increase of the frequency with 20%, the small novelty is an increase of the frequency with 5%. In both cases the index of which template is activated and the sum of adaptations to templates give a good detection of the novelty. Also the other experiments in this article show a good performance.

Though the restriction to binary inputs is a disadvantage to this method.

In the next section a similar approach is discussed, with the difference this system isn't using binary patterns.

3.3 The Receptive Novelty Detector (RND)

The RND is a mixture of a system proposed by Dasgupta and Forrest [5] and ART. The system proposed by Dasgupta and Forrest uses ideas from immunology. This system is a probalistic method that notices changes in normal behavior without requiring prior knowledge of the changes for which it is looking, the input for this system are binary vectors and it uses templates to store knowledge. In these ways it resembles ART. If differs from ART in the way the templates are calculated and used. There is no such thing as a second layer of competitive neurons. The templates are simply the first n inputs. For the actual detection a complement set is used, called the detector set (compare anti−bodies in immunesystem). The detector set has to satisfy the following relation.

$$\left\{\forall d \in D | d \underset{x}{\notin} T\right\} \text{ where D is the detector set and T is the template set} \tag{3.3}$$

A novelty is detected when an input pattern matches one or more detectors at x consecutive places. The disadvantage of this method is the possibility of a large receptor set, because the novelty set (complement of normal set) is usually (much) larger then the normal set, because a good estimation of the boundaries of the novelty set without knowing the novelties in advance is merely impossible.

This problem is solved in the RND. In the RND the normal templates, called receptors, are also used for the novelty detection, anti−receptors are created along the process and not preliminary. Another advantage is that the RND uses non−binary patterns as input. The input vector for this system is a quantified signal−window.

A novelty is detected when there isn't a match with a receptor in the receptor database. This novelty is then added to the anti−receptor set. So the second way a novelty is detected is when there is a match with a pattern in the anti−receptor set (figure 3.2).

The goal of the RND is to show that novelty detection in time series with a simple system (in comparison to ART) using simple quantification and template creation without hardly any generalization (there is some generalization due to the match factor) is possible in simple cases of time series.

Receptive Novelty Detector

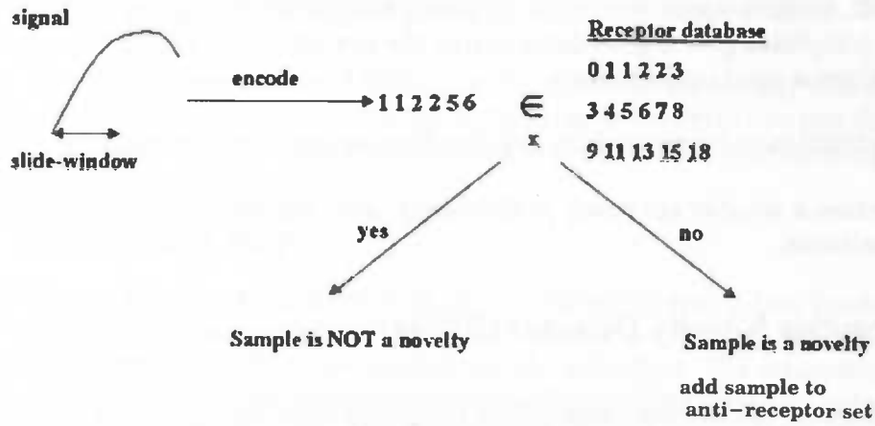

Figure 3.2 *Schematic algorithm for RND Encoding is a quantification of the input signal.* $a \underset{x}{\in} B$ *means that pattern a matches at x consecutive positions with at least one pattern from B. x Is comparable with the vigilance parameter of ART.*

We conducted an experiment with RND with a similar dataset as used in [3]. The time series used in this experiment is a sinus with sample steps of 0.1, after 500 samples the sample step increases with 5% to 0.105 and after 1000 samples it returns to 0.1. The time series is quantized into 30 intervals. As input a quantized window of 10 values is used, this window is shifted every step 1 sample. The vigilance x is 9, thus a receptor matches with an input if they match at 9 consecutive positions.

The results are slightly worse then ART because the stability is worse. This stability problem is caused by matches of normal inputs with anti−receptors which are created by a mismatch with all receptors. In my implementation the anti−receptors have higher priority then receptors, practically they overwrite the normal receptors. Though when using no receptor set, thus without training, and only watch the creation of anti−receptors, the change in the signal at position 500 can be seen clearly by the cummultative sum of creations of anti−receptors. When the signal starts this cummultative sum is obviously raising quickly because there isn't any knowledge yet, but after a while the creation of anti−receptors drops to zero. When the signal changes the creation of anti−receptors raises again. So the change can be detected (and thus the novelty). (see figures)

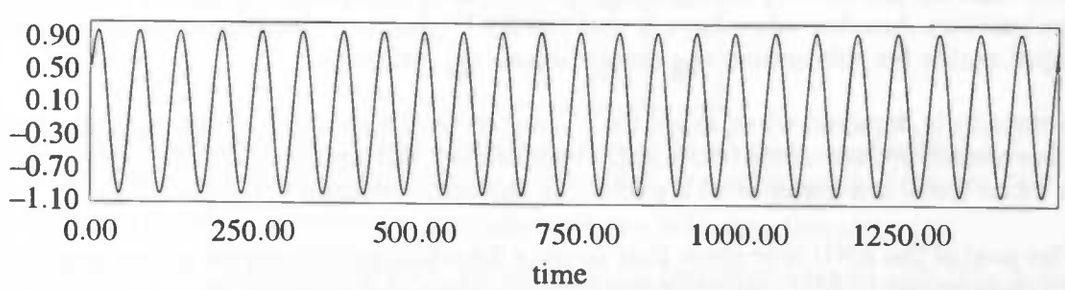

Figure 3.3 *Sinus with changing sample frequency At t=500 the novelty starts, end at t=1000*

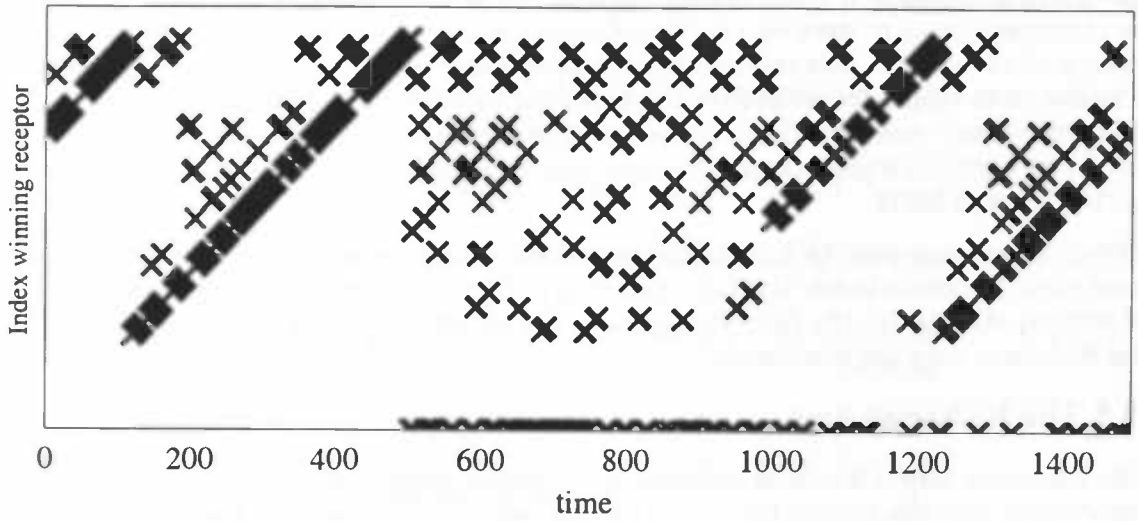

Figure 3.4 *Index of winning receptor The trainset is the first 500 values of the quantized sinus. This plot shows the abnormal behavior of normal receptor activation by the novelties in the signal between t=500 and t=1000.*

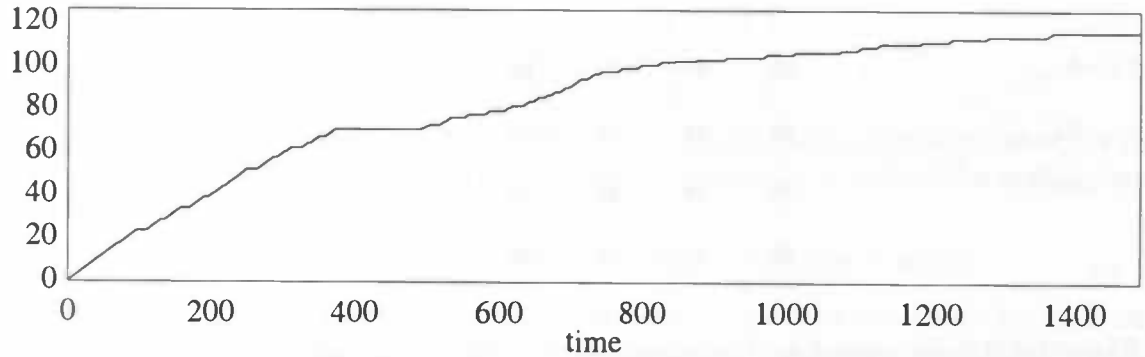

Figure 3.5 *cummultative sum of anti−receptors created without initial training The sum remains stable until the novelty starts and stabilizes again when the novelty ends.*

The conclusion is that it is showed that simple template (prototype) matching, with ART or RND, is sufficient for novelty detection in simple time series. Though a clear verdict weather a sample is a novelty or not isn't available yet. The biggest disadvantages of the above two clustering methods are the discrete valued templates and inputs. This obligates an explicit preprocess with the risk of losing important information.

In the next paragraph another self organizing neural network clustering method is discussed. This method can be used without any preprocessing and gives a clear verdict weather a sample is a novelty or not.

3.4 Self organizing Feature Maps

A Self−organizing Feature Map (SoFM) is a method used for clustering of data based on features, hence 'feature map'. The method is trained unsupervised, hence 'self organizing'. Similar to the ART and the RND the features used for clustering are characterized by templates or prototypes as they are called in literature about SoFM. The advantage of SoFM over ART−1 and RND is the ability to use a continuous input domain and real−valued prototype vectors.

The principal goal of the SoFM algorithm developed by Kohonen in 1982 is to transform an incoming signal pattern of arbitrary dimension into a one– or two dimensional discrete map, and to perform this transformation adaptively in a topological ordered fashion [9]. The Kohonen map is the most used implementation of a SoFM. The Kohonen map provides an easy measurement for the decision weather a sample belongs to a cluster or not and it isn't dependent on a preprocess when input is from a continuous range, the last in contrary to ART–1 and RND.

These advantages over ART and RND plus the expertise we already had with the Kohonen map made us decide to use the Kohonen map as the architecture to test the clustering part of our hypothesis. In the following section the architecture and learning methodology of the Kohonen map are discussed.

3.5 The Kohonen map

The Kohonen map is a neural network architecture containing two layers of neurons, the input layer, and the output layer. The output layer is a competitive layer, the output neurons in this layer are fully interconnected with each other. The connection weights of the connections between outputneurons are called the neighborhood. An output neuron is also connected to all input neurons. The vector of weights emerging from a output neuron and converging to the input neurons is called a prototype.

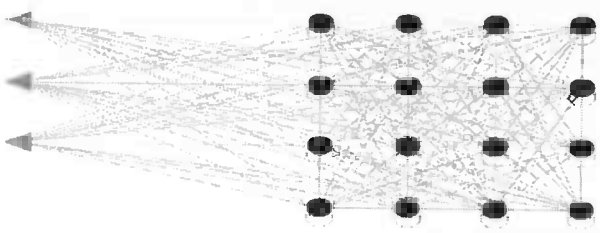

Figure 3.6 *A kohonen map The input neurons are the triangles, the output neurons are the circles.*

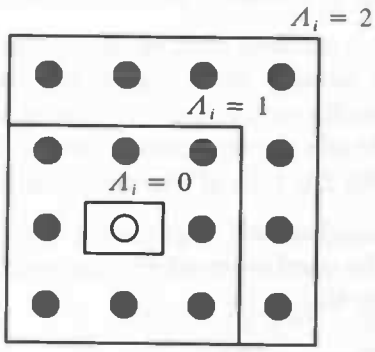

Figure 3.7 *Neighborhoods in the outputlayer Square topological neighborhood Λ_i, of varying size, around "winning" neuron i, identified as a white circle. [Haykin [9]]*

The algorithm for the kohonen map is divided into the following five steps [9]:

1. *Initialization.* Choose random values for all initial prototype vectors $w_j(0)$. The only restriction here is that the $w_j(0)$ be different for $j = 1, 2, \ldots, N$, where N is the number of output neurons (thus the number of prototypes)

2. *Sampling.* Draw a sample **x** from the input distribution with a certain probability; the vector **x** represents the sensory signal.
3. *Similarity Matching.* Find the best–matching (winning) neuron $i(x)$ at time n, using the minimum–distance Euclidean criterion:

$$i(x) = \arg\min_j \|x(n) - w_j\|, \qquad j = 1, 2, \ldots, N \tag{3.4}$$

4. *Updating.* Adjust the prototype vectors of all neurons, using the update formula

$$w_j(n+1) = \begin{cases} w_j(n) + \eta(n)[x(n) - w_j(n)], & j \in \Lambda_{i(x)}(n) \\ w_j(n), & otherwise \end{cases} \tag{3.5}$$

where $\eta(n)$ is the learning–rate parameter, and $\Lambda_{i(x)}(n)$ is the neighborhood function centered around the winning neuron $i(x)$; both $\eta(n)$ and $\Lambda_{i(x)}(n)$ are varied dynamically during learning for best results.

5. *Continuation.* Continue with step 2 until no noticeable changes in the feature map (set of prototypes) are observed.

In the kohonen neural networks used for experiments we used the Gaussian neighborhood function

$$\pi_{j,i} = \exp\left[-\frac{d_{j,i}^2}{2\sigma^2}\right] \tag{3.6}$$

where the parameter σ is the 'effective width' of the topological neighborhood. $d_{j,i}$ Is the similar to the Λ_i shown in figure 3.7. This leads to the following rewrite of the update function (3.5):

$$w_j(n+1) = w_j(n) + \eta(n)\pi_{j,i(x)}(n)[x(n) - w_j(n)] \tag{3.7}$$

Both the learning rate $\eta(n)$ and the neighborhood function are to be decreased during training to get a stable feature map. Therefore we use exponentional decay of the effective width and the learning rate according to Haykin [9].

3.6 Novelty detection with Kohonen

In this section the results are discussed of the experiments of novelty detection in time series with the Kohonen neural network. As discussed in section 1.3 novelty detection consists of three steps.

The first step is to select and construct the model. The model in this case is the Kohonen neural network. In the constructing phase the decision has to be made, based on the time series characteristics, what kind of information is used as input. For the Kohonen approach in this thesis this is static information. The construction of the model is the actual training of the Kohonen network with the 'normal' time series data, thus without novelties.

The second step is the computation of the residuals of the network. The residual of an input vector is the root mean square error (RMSE) between the input vector x and the prototype vector p of the winning output neuron after evaluating the network with the input vector.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(x_i - p_{\hat{i}})^2}{N}}, \qquad where\ N = |x| = |p| \tag{3.8}$$

The third step is the comparison of the residuals, this is done with a threshold. If the RMSE between an input vector and the selected prototype is greater or equal to this threshold this input is marked as a novelty, otherwise as normal.

The first experiment we present uses the data described in appendix section NO TAG. This time series system is bivariate and multi−stationary. Remember we used the static information to cluster the 'normal' input data. Thus the number of inputs of the Kohonen map is two. Every time step the values of x_{1t} and x_{2t} are set at the input neurons. As output we used a one dimensional circular lattice of 20 neurons (see fig. 3.8).

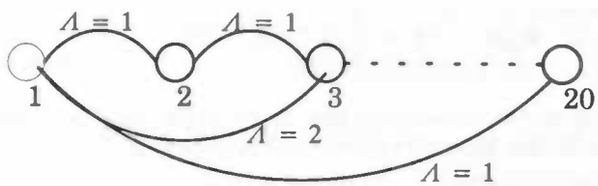

Figure 3.8 *Neighborhoods in circular lattice of 20 output neurons*

We chose this number of outputs after having tried several configurations with more and less output neurons. This showed the accuracy of modelling the normal data becomes significant better until the number of 20 outputneurons is reached.

We conducted the experiment several times, each time the result was similar. In figure 3.9 the trade off function is presented and in figure 3.10 the sensitivity graph is printed.

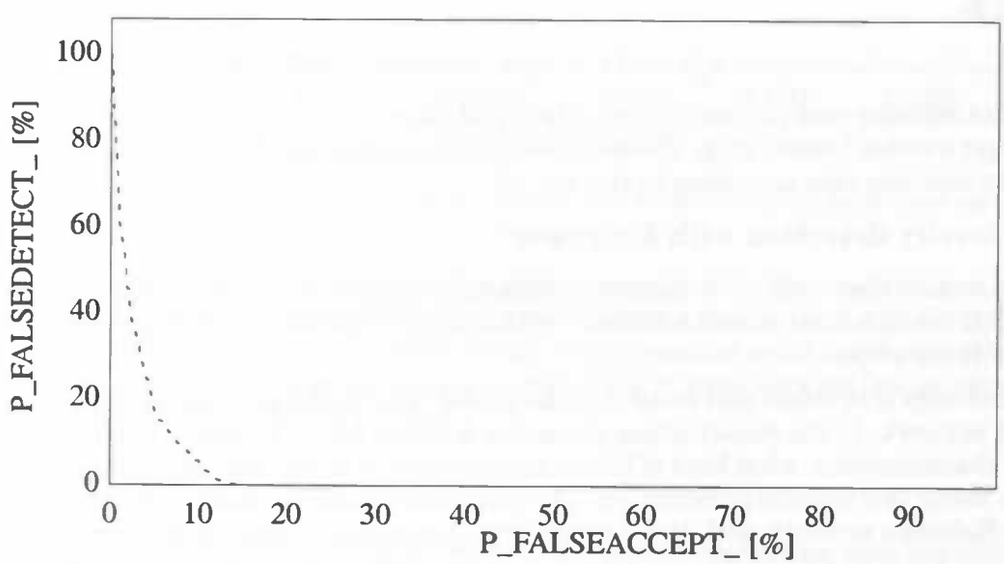

Figure 3.9 *Trade off of a kohonen neural network tested with time series system 1.*

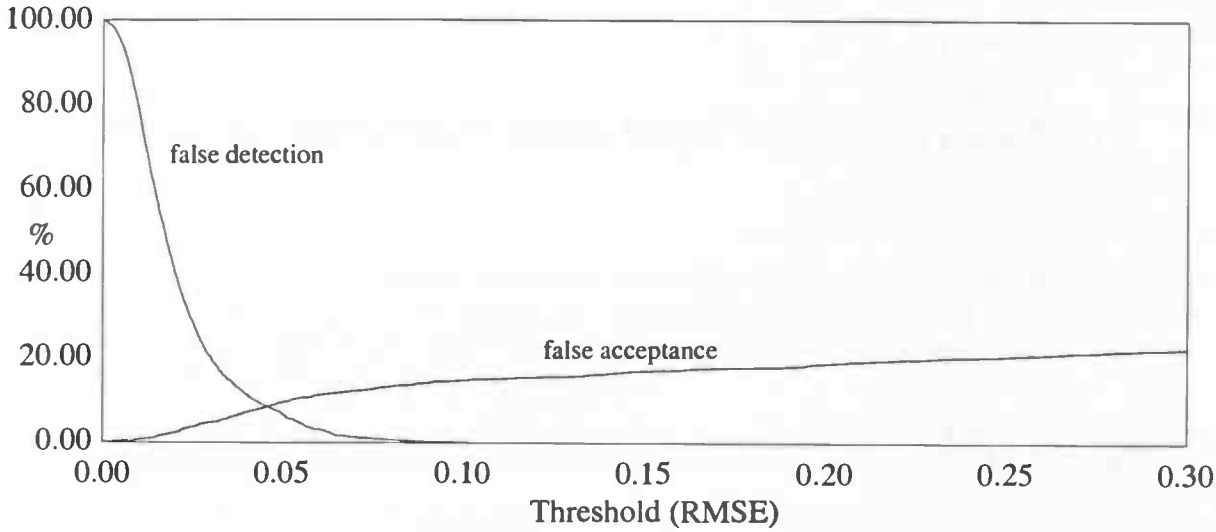

Figure 3.10 *Sensitivity graph of a Kohonen neural network tested with time series system 1*

From these graphs it can be concluded the RR is approximately 20% (100% − 80%). This isn't a very good result. The main reason for this disappointing result is the transition from state 3 to the novelty state, state 4. When looking at the scatterplots of the data which visualizes the static information (see appendix section NO TAG), one can see this transition overlays for a small part the transitions from state 3 to 1 and 3 to 2. This implies that these novelties can't all be distinguished from normal data.

To avoid this problem the second dataset used for an experiment doesn't contain such overlays of novelties with normal data. This dataset is described in appendix section NO TAG. This dataset is also bivariate and multi−stationary. Though this time series system has other differences compared to the series used in the first experiment. The most important for clustering at static information is that the states are non−linear separable when looking at the visualization of the static information in the scatterplot.

Due to the uniform distribution of the transitions, the number of transitionsteps, and the non−linear separability we need more output neurons then in the first experiment. The number of inputs stays the same obviously. After we tried several configurations for the outputs we chose for a two dimensional lattice of 10x10 neurons with the neighborhoods arranged as in figure 3.7.

The performance for this dataset is presented in the figures 3.11 and

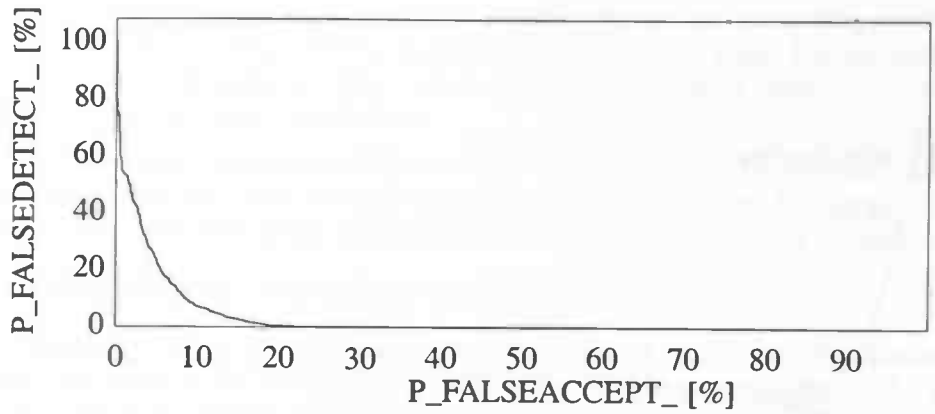

Figure 3.11 *Trade off of a Kohonen neural network tested with time series system 2.*

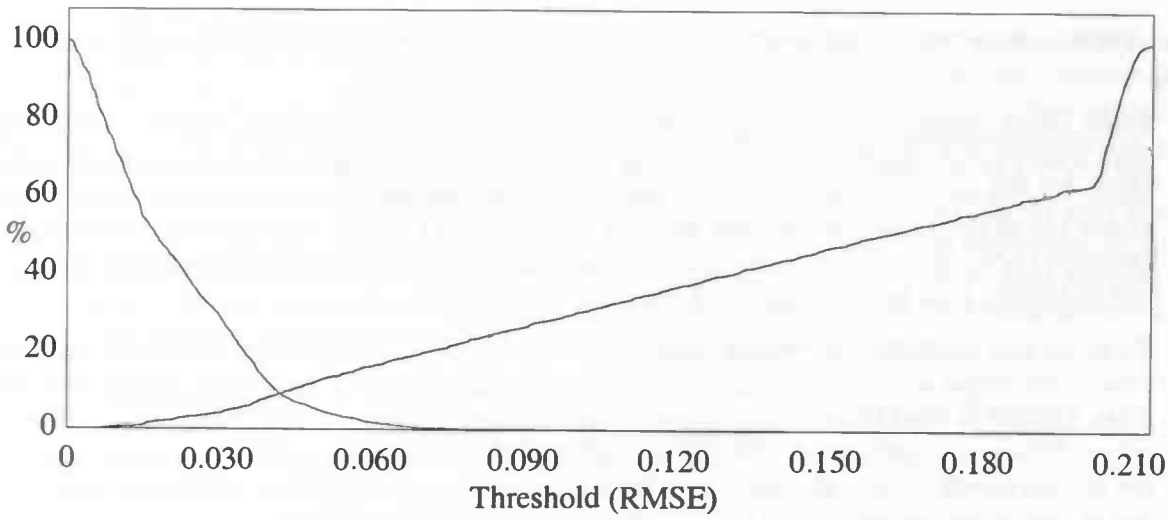

Figure 3.12 *sensitivity graph of a Kohonen neural network tested with time series system 2.*

From these graphs it follows that the RR is approximately 30%. This is a better result then with the first dataset. The problem with this kind of time series lies in the uniform distribution of the static information. In other words, every possible point inside the boundaries of the signal is a potentially sample, thus to model the normal data perfect an almost infinite number of clusters should be used.

3.7 Conclusion

In this chapter we showed that the Kohonen self−organizing feature map has advantages over other clustering methods as ART and RND. We conducted our main experiments with the Kohonen map. The Kohonen map is used to cluster the static information of the dataset. The results for the two datasets are respectively an average RR of approximately 20 % and 30 %.

According to our hypothesis the performance of the time series predicting neural network is better. In the next chapter this method is discussed, followed by the testresults of this method.

Chapter 4

Novelty detection with time series prediction

4.1 Introduction

This chapter discusses the second approach to novelty detection in time series. Referring to our hypothesis of chapter 1 we expect that the novelty detection based on creating a dynamic model of the series and predicting with this model the time series, performs better then the method used in the previous chapter.

Before we present the testresults of this method in paragraph 4.5, we discuss the concepts of time series prediction in paragraph 4.2.

In paragraph 4.3 and 4.4 we focus to the neural network approach of time series prediction.

4.2 Time series prediction

The concept of time series prediction is described by Van Veelen, Nijhuis and Spaanenburg in [29]. They state the problem of time series prediction is to forecast or predict the value or some statistic at any time t of one variable say $Y \in \mathfrak{F}_Y$ from a set of variables $\mathfrak{X} = \{X_1 \ldots X_n\} \in \mathfrak{F}_{\mathfrak{X}} = \mathfrak{F}_{X_1} \times \mathfrak{F}_{X_2} \ldots \times \mathfrak{F}_{X_n}$ where one of the variables in this set may be Y by looking at past values of this set of variables. Thus finding a mapping $f_{W(t)}$ parameterized by the parameter set $W(t) \in \mathbb{R}^{|W|}$ so that $Y(t_p) = f_{w(t)}(\mathfrak{X}(t_1), \mathfrak{X}(t_2), \ldots, \mathfrak{X}(t_m))$ where $f_{W(t)} \in \mathfrak{F}_{\mathfrak{X}}^m \times \mathfrak{F}_Y$ and $t_p > t \wedge t_1 < t_2 < \ldots < t_m < t$. Though many different fields may be encountered in practice, we will assume that all fields $\mathfrak{F}_Y$, $\mathfrak{F}_{X_1}$, $\mathfrak{F}_{X_2} \ldots \mathfrak{F}_{X_n}$ are subsets of $\mathbb{R}$ so that the problem of time series prediction can be summarized by eq. (4.1).

$$Y(t_p) = f_{W(t)}(X(t_1), X(t_2), \ldots, X(t_m))$$
$$X \in \mathbb{R}^n \wedge Y \in \mathbb{R} \wedge t_p > t \wedge t_1 < \ldots t_m < t \tag{4.1}$$

To find the optimal $f_{w(t)}$ two kinds of modelling are used

- linear
- non–linear

Examples of the first approach are the statistical AR(p), MA(q) and ARMA(p,q,r) models. To the second approach belong neural network models.

The statistical approach has been a research area for many years. This has led to good design trajectories and results. Parallel to this development the last decade also the research to time series modelling with neural networks led to convenient solutions.

The discussion which approach is the best is at this moment a hot topic. In literature there are already made some comparisons between neural networks and statistical models, e.g. by Sarle in [27]. The most heard conclusions of such comparisons are that statistical methods and neural networks are equal methodologies with different languages and architectures used to estimate modelparameters, but with the same results due to old statistical restrictions emerging in both fields. Thereby often is concluded that it is highly unlikely neural networks will ever supersede statistical methodology. Though we doubt this last

conclusion, we rather see statistics and neural networks not as competing methodologies, but as tools for different subproblems in the data–modelling area. For example with the help of linear statistical tools for data–analysis, the non–linear neural networks can be optimized for the actual data–modelling.

In this thesis time series data is modelled with neural networks. In the next paragraph we discuss the most common used neural network architecture used for data–analysis and which can be easily adapted to an architecture for dynamic modelling of time series.

4.3 The Multilayer Perceptron (MLP)

The MLP belongs to the so called multi layer feedforward networks. The history of these neural networks dates as far back as the early 1960s. The main breakthroughs were made in the second half of the 1980s after a long silencing period after a pessimistic publication of Minsky and Papert in 1969 about the abilities of neural networks. In the late 1980s the architecture and the learning methodology of the feedforward network as we now know it was developed.

In general MLP neural networks consist out of three layers. An input layer is the interface with the outside world and each input neuron is connected to all neurons in the combining hidden layer. At their turn all hidden neurons are connected to each neuron in the output layer. Given a certain pattern presented to the input neurons, the output neurons of a trained MLP neural network will provide the belonging output pattern. During the learning procedure of such a network, input patterns are presented to the input layer and the weights of the connection are updated using the back–propagation learn algorithm until the difference between the output patterns and the desired target patterns is small enough. Characteristic for this method is that the final behavior of the network is considered to be stored in the hidden layer and the connections of the MLP neural network.

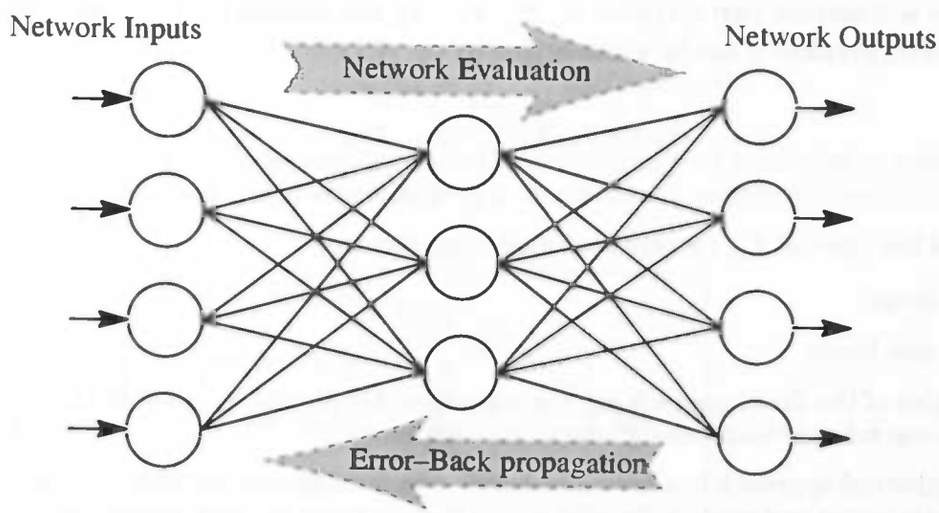

Figure 4.1 *Example of a feedforward neural network Input patterns are provided to the network inputs and are evaluated in forward direction through the network. This results in a estimated output. This output is compared with the desired output which results in an error. This error is propagated in backward direction through the network which results in the adaption of the weights of the connections between the layers.*

Before discussing the back–propagation algorithm we need to know how the output of a single neuron is calculated. The output of a single neuron is calculated by the following pair of equations.

$$v_k = \sum_{j=0}^{p} w_{kj} x_j \tag{4.2}$$

$$y_k(n) = \varphi_k(v_k(n)) \tag{4.3}$$

where

- v_k is called the linear combiner output
- x_j are the input signals
- w_{kj} are the weights of the connections converging to neuron k
- $y_k(n)$ is the actual output produced by input signal n
- φ_k is the activation function

These equations yield for neurons in the hidden layer as well as neurons in the output neuron. The activation function accounts for the non−linearity of the MLP. Several activation functions are proposed. The most commonly used function is the sigmoid, defined as

$$\varphi(v) = \frac{1}{1 + e^{-av}} \tag{4.4}$$

where a is the slope parameter.

The MLP is an identifying neural network, thus trains supervised (see paragraph 1.4).

Training of the MLP is an iterative process which stops when all network output errors are smaller than some ϵ. This is done by minimizing a cost−function, a commonly used cost−function is the summed squared error over all outputs defined as follows

$$\mathcal{E}(n) = \frac{1}{2} \sum_{j \in C} e_j(n)^2 \tag{4.5}$$

where $e_j(n)$ is the desired output minus the actual output at iteration n (eq. (4.6))

$$e_j(n) = d_j(n) - y_j(n) \tag{4.6}$$

The error−back propagation algorithm adapts the connection weights between layers in such a way the cost function is minimized. The weight adaptation is dependent of the errors at the output neurons the connections are converting to. The weight adaptation starts with the weights between the output layer and the hidden layer. These weights adaptions are calculated with the following equation:

$$\Delta w_{ji} = \eta e_j(n) \varphi'(v_j(n)) y_i(n) = \eta \delta_j(n) y_i(n) \tag{4.7}$$

Where$\eta \in [0 \ldots 1]$ is the learningrate and $\varphi_j'(v_j(n)) = \frac{\delta y_{j(n)}}{\delta v_{j(n)}}$. We may define the term $e_j(n)\varphi'(v_j(n))$ for neuron j as the local gradient, denoted by $\delta_j(n)$. The local gradient points to required change in the connection weights. Eq. (4.7) is for the correction at the output neurons, thus the error−term $e_j(n)$ is available, for the hidden neurons this isn't the case, thus the error is propagated back. This results in a local gradient for the hidden neurons as follows:

$$\delta_j(n) = \varphi_j'(v_j(n)) \sum_k \delta_k(n) w_{kj}(n) \tag{4.8}$$

where $\varphi_j'(v_k(n)) = -\frac{\delta e_k(n)}{\delta v_k(n)}$ and $\delta_k(n)$ is the local gradient of the outputs. For a complete derivation of this formulas see [9].

Summarizing, the correction $\Delta w_{ji}(n)$ applied to the connectionweights between any neuron i to neuron j is defined by the delta−rule

$$\begin{bmatrix} Weight \\ correction \\ \Delta w_{ji}(n) \end{bmatrix} = \begin{bmatrix} learningrate \\ parameter \\ \eta \end{bmatrix} \begin{bmatrix} local \\ gradient \\ \delta_j(n) \end{bmatrix} \begin{bmatrix} input\ signal \\ of\ neuron\ j \\ y_i(n) \end{bmatrix} \tag{4.9}$$

As we can see, this rule applies to both hidden and output neurons, where only the calculation of the local gradient changes.

In the next paragraph how the 'normal' MLP can be adapted to a MLP specialized in capturing dynamic information from time series.

4.4 Time series prediction with the TDL–MLP

In literature many architectures are proposed for capturing dynamic information with neural networks. An overview is given in [29]. The basic idea is to add functionality to the neural network to use past values of a data set to calculate the present output(s).

To transform a MLP into a Tapped Delay Line MLP (TDL−MLP) a linear filter is added to the neural network to incorporate dynamics into the model. By Tapped Delay Line MLPs this linear filter is fully connected to an input neuron. The input neuron is then called a Finite Impulse Response (FIR) perceptron when all filters at the connections are causal and have a finite time window. The linear combiner output of this perceptron is calculated as in eq. (4.10)

$$v_j(n) = \sum_{i=0}^{p} \sum_{l=0}^{M} w_{ji}(l) x_i(n-l) \tag{4.10}$$

where M is the size of the time window, thus the amount of past (expected or estimated) outputs used to calculate the current output(s). A schematic representation of a TDL−MLP is showed in figure 4.2.

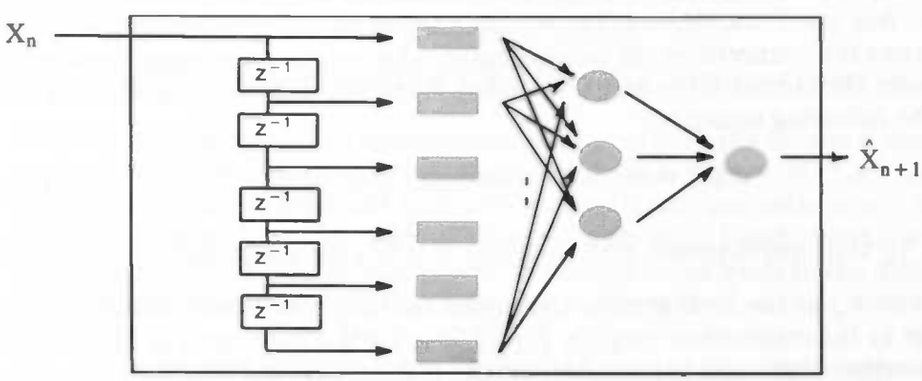

Figure 4.2 *a TDL−MLP with one input and a delay line of size 5. The TDL−MLP is often used for time series prediction, thus for modelling the f in* $X_n = f(X_{n-1}, X_{n-2}, \ldots)$

In this thesis the TDL−MLP is used for the prediction of the next value of a time series. As said earlier training of a MLP continues until the error of the total model is beneath a

certain ϵ. Normally a good measure for this error is the RMSE over the set of trainpatterns (eq. (4.11)).

$$RMSE = \sqrt{\frac{\sum_{n=1}^{N}(d_j(n) - y_j(n))^2}{N}} \tag{4.11}$$

where N is the number of trainpatterns. Though when predicting time series sometimes TDL−MLP neural networks learn simply the id function resulting indeed in a low RMSE because the next value of a time series is often close to the present value. To solve this problem another error measure is proposed, the RSE, defined as

$$RSE = \frac{\sum_{n=1}^{N}(observation(n) - prediction(n))^2}{\sum_{n=1}^{N}(observation(n) - observation(n-1)^2} \tag{4.12}$$

When the neural network model simply predicts the last observed value (*observation(n−1)*) this results in a RSE greater than 1.0, a ratio below 1.0 means a better prediction. The prediction is the best when the RSE is 0. We use this error measure to decide weather the neural network is trained satisfactory.

After the TDL−MLP trained the 'normal' time series novelties can be detected. This is discussed in the next paragraph together with experimental results.

4.5 Novelty detection with the TDL–MLP

The novelty detection with the TDL−MLP is pretty similar to the novelty detection with the Kohonen map discussed in paragraph 3.6. A novelty is detected when the residual belonging to an input pattern is greater than a certain threshold. This residual is, equivalent to the clustering case, calculated as the RMSE with the difference the prototype is the expected value, the input vector is the predicted value and N is the number of outputs of the TDL−MLP. Summarizing the error belonging to an input sample of the TDL−MLP is

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(d_i - y_i)^2}{N}} \tag{4.13}$$

For the experiments with the TDL−MLP we used the same data as with the Kohonen map. Thus the TDL−MLPs had two inputs x_{1t} and x_{2t} with delay lines appended to these inputs and two output neurons with as desired output $x_{1(t+1)}$ and $x_{2(t+1)}$.

For the first experiment, with the data described in appendix section NO TAG, we used a TDL−MLP with 10 hidden neurons and for each input a delay line of length 5, thus an input pattern consisted of *x(t), x(t−1), x(t−2), x(t−3)* and *x(t−4)*. The delay line length and the number of hidden neurons was chosen after trying other delay line lengths and hidden neurons. The delay line length is also chosen based on an autocorrelation analysis (see paragraph 2.2.2).

After a very long training period we reached an RSE of 0.707 for input x1 and 0.587 for input x2. The result of the novelty detection is showed in figures 4.3 and 4.4.

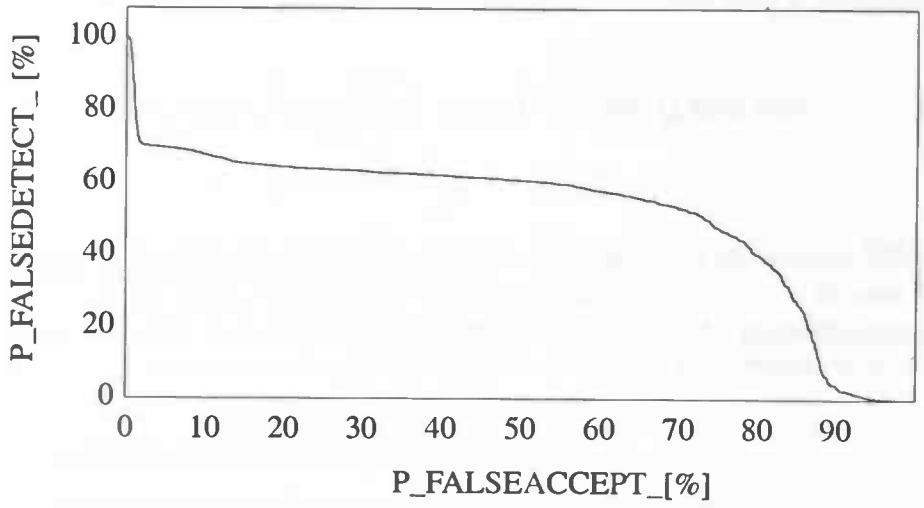

Figure 4.3 *Trade off TDL−MLP tested with time series system 1*

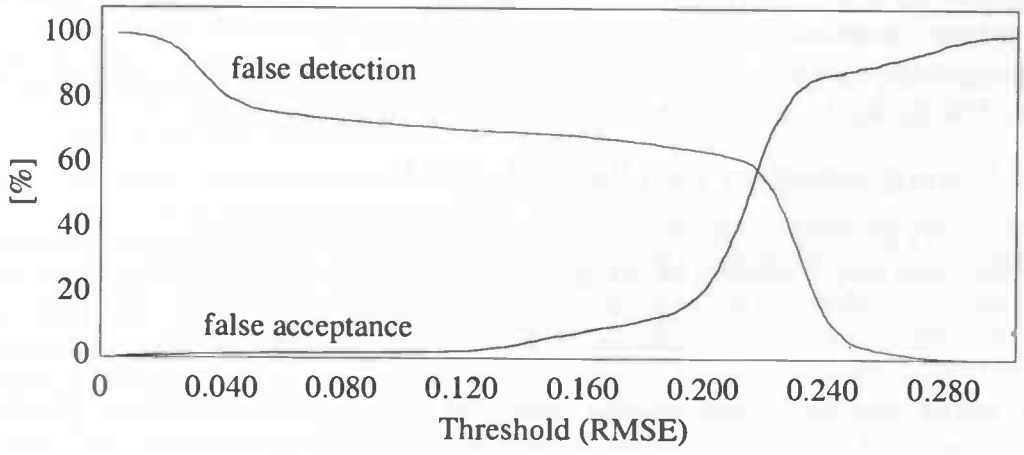

Figure 4.4 *Sensitivity graph of TDL−MLP tested with time series system 1*

These graphs show a RR of approximately 80%, as with the kohonen map this also is a disappointing result. Though the reasons are different. Figure NO TAG shows that the time series used in this experiment have pretty steep transition paths which makes the time series at even small time intervals highly non−stationary, besides that the time series generated in a state have a high noise level, all together this leads to the difficulties the TDL−MLP encounters when modelling this time series system.

The second data set is used to solve these problems, the transition paths are made smoother and the time series generated in a state are sinoïdals which are known to be modelled very well by a TDL−MLP. The TDL−MLP used by the presented results consisted of 10 hidden neurons and 2 inputs with both a delay line of length 5. The RSEs were after training 0.05 for x1 and 0.07 for x2. This is a clear improvement when compared with the first data set. The novelty detection results are presented in figures 4.5 and 4.6.

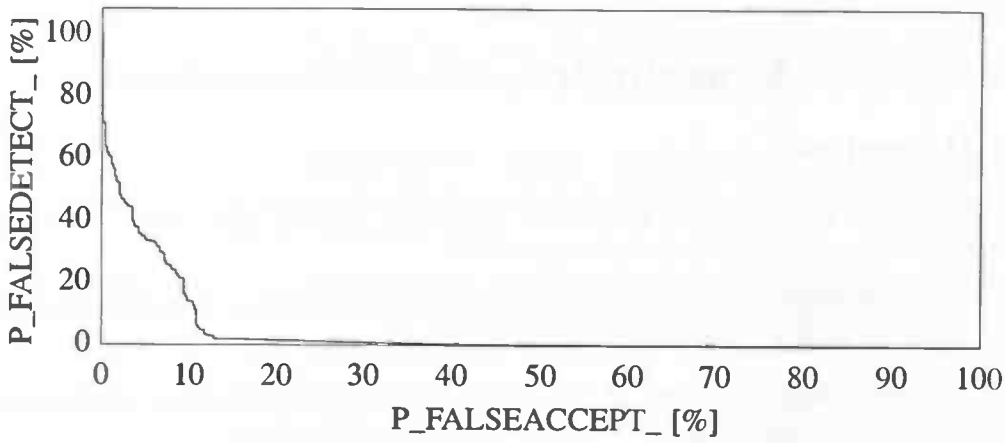

Figure 4.5 *Trade off of a TDL−MLP tested with time series system 2.*

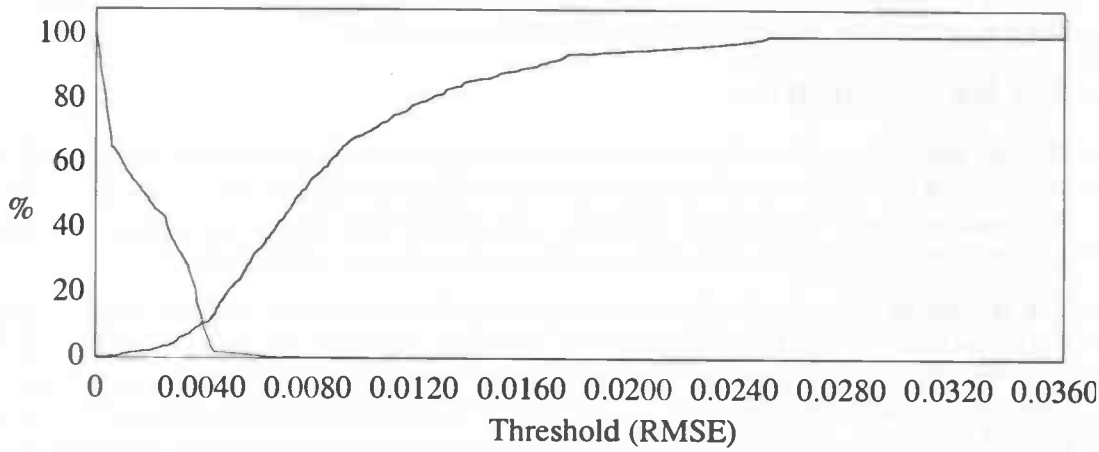

Figure 4.6 *Sensitivity graph of a TDL−MLP tested with time series system 2.*

The results of this experiment showed a mean performance in RR of approximately 30%, again the same performance as the Kohonen map. The reasons for the not expected low RR are the great generalization capabilities of the TDL−MLP. This resulted in a TDL−MLP which even predicted the time series in the novelty state pretty well, while in normal data space the predictions weren't very well at the state transitions.

4.6 Conclusion

This chapter discussed how standard MLPs can capture dynamic information in time series by adding a TDL(s) to the MLP. We expected that these TDL−MLPs performed better then the Kohonen map discussed in the previous chapter. Experimental results showed the contrary. In both test cases the performances were equal. In the next chapter we will discuss the conclusions.

Chapter 5

Conclusion and Discussion

5.1 Overview

In this thesis, after having given the basics of novelty detection in time series, we introduced two methods for novelty detection.

- Novelty detection based on clustering static information in multivariate time series systems with a clustering neural network
- Novelty detection based on modelling dynamic information in multivariate time series systems with a time series predicting neural network

In a hypothesis we stated that we expected that the second approach performed better than the first approach. To make the comparison between the two methods we introduced some quality measurements. The most important one, for reasons from real life applications, is the reduction rate which is deduced from the sensitivy.

5.2 Experiments

For the experiments we chose the Kohonen map as implementation for the clustering approach and the TDL−MLP as implementation for the time series prediction approach. As data sets we used time series systems containing both important static information and dynamic information to give each method a fair chance of succes.

The results of the experiments were suprising. Both methods had the same performances. Both methods scored RRs between 20% and 30%. Besides this promptness is another measurment especially important for real life applications. The promptness of both methods is also equal, with the remark the TDL−MLP has an initializing period of the lenght of the tapped delay line. Disregarding this remark both methods can detect novelties in real time. Summarizing this we can conclude that

- **In the discussed and simillar cases the novelty detection performance of the predicting TDL–MLP using dynamic information, equals that of the Kohonen self–organizing feature map using static information**
- **The hypothesis stated in chapter 1 doesn't hold**

5.3 Discussion and further research

The reasons for the dissapointing performance of the TDL−MLP lays for a big part in the multi−stationarity of the time series. An option for further research is to investigate the opportunity of using seperate TDL−MLPs for each state. The main problem is then the decision in which state the model is in and thus which TDL−MLP is to be used.

Another problem encountered by the TDL−MLP is over generalization which happened in the second experiment. This resulted in good predictions, thus no novelty, while in the novelty state. This could maybe be solved by using separate TDL−MLPs for each time series in the model.

Summarizing this it is pointed out that using a divide and conquer principle is an option for further research.

The main problem of using static information for clustering is that novelties that are close to or at (static) positions of normal data can't be detected. A solution for this is maybe also using dynamic information with the Kohonen map. We expect that is does, also because the Kohonen map showed to be a good novelty detector even only with the use of static information.

So we did another experiment appropriate for further research with the univariate time series described in appendix section A.3. The mentioned problems with the TDL−MLP and Kohonen map are all avoided in this experiment.

The TDL−MLP used for this experiments had 19 hidden neurons and a delay line of length of 6. The Kohonen had as inputvector also the current input and the 5 last inputs, the output lattice was circular and contained 19 output neurons.

The networks were trained with a time series without noise. When tested with a time series also containing no noise the performance was for both methods perfect, thus all novelties were detected and no normal pattern was falsely detected as novelty. Though when 1% noise was added to the time series kohonen was clearly better then the TDL−MLP. A trade-off function for this experiment is presented in figure 5.1.

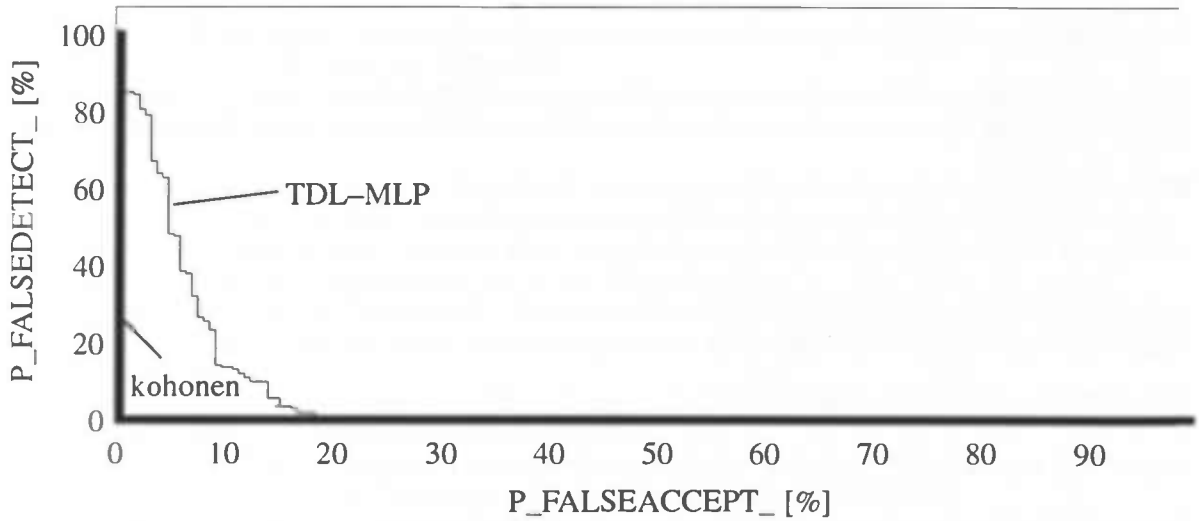

Figure 5.1 *Trade off Kohonen map and TDL−MLP by trainset without noise and testset with 1% noise.*

We performed the same experiment also with as trainset the noisy time series. In this case the Kohonen map became better then the TDL−MLP by 7% noise.

These firts results are very promissing and the comparison between a kohonen novelty detector using dynamic information and a TDL−MLP novelty detector using dynamic information is thus also recommended as subject for further research.

References

[1] C. M. Bishop, *Neural Networks for Pattern Recognition*, Clarendon Press.

[2] T. Brotherton, T. Johnson and G. Chadderdon, *Classification and Novelty Detection using linear models and a Class Denpendent Elliptical Basis Function Neural Network*, Proc. of the International Joint Conference on Neural networks, 1998, vol. 2, pp. 876–879.

[3] T. P. Caudell and D. S. Newman, *An apdative resonance architecture to define normality and detect novelties in time series and databases*, Proc. of the Internation Joint Conference on Neural Networks, 1993, vol. 4, pp. 166–175.

[4] M. Chen, T. Kanade, H. A. Rowley and D. Pomerleau, *Anomaly detection through registration*, Proc. of the IEEE Computer society conference on computer vision and pattern recognition, June 1998.

[5] D. Dasgupta and S. Forrest, *Novelty detection in time series data using ideas from immunology*, Proc. of the 5th International conference on intelligent systems, June 19–21 1996.

[6] M. J. Desforges, P. J. Jacob, J. E. Cooper, H. A. Rowley and D. Pomerleau, *Applications of probability density estimation to the detection of abnormal conditions in engineering*, Proc. of the Instn Mech Engrs, April 1998, vol. 212C.

[7] J. L. Elman, *Finding structure in time*, Cognitive Science, no. 14, pp. 179–211, 1990.

[8] P. M. Frank, *Fault Diagnosis in Dynamic Systems Using Analytical and Knowledge–based Redandancy – A Survey and Some New Results.*, Automatica, vol. 26, no. 3, pp. 459–474, 1990.

[9] S. Haykin, *Neural Networks, A Comprehensive Foundation*, first edition, Prentice–Hall Inc.

[10] L. G. Heins and D. R. Tauritz, *Adaptive Resonance Theory (ART) : An Introduction*, Technical Report SNNS Report, May/June 1995.

[11] T. V. Ho and J. Rouat, *Novelty detection based on relaxation time of a network of integrate–and–fire neurons*, Proc. of the IJCNN ,1998, vol. 2, pp. 1524–1529.

[12] D. M. Hummel, W. Ahmed and M. T. Musavi, *Adaptive detection of small sinusoidal signals in non–gaussian noise using an RBF neural network*, IEEE transactions on Neural networks, vol. 6, no. 1, pp. 214–219, January 1995.

[13] N. Japkowicz, C. Myers and M. Gluck, *A novelty detection approach to classification*, Proc. of the Fourteenth International Conference on Artificial Intelligence, august 1995, vol. 6, pp. 518–523.

[14] M. Kezunovic, I. Rikalo and D. J. Sobajic, *High speed fault detection and classification with neural nets*, Electric power systems research journal, vol. 34, no. 2, pp. 109–116.

[15] R. Kozma, M. Kitamura, M. Sakuma and Y. Yokoyama, *Anomalty detection by neural network models and statistical time series analysis*, Proc. of the International conference on Neural Networks,1994, pp. 3207–3210.

[16] W. J. D. Lamberts, J. A. G. Nijhuis and M. H. ter Brugge, *Detecting Novelties*, Technical Report, University of Groningen, Dept. of Computing Science, August 1998.

[17] K. H. Lee, *A Non–probalistic approach to the histogram method in detecting abnormalities in MRI images*, Proc. of the IEEE International Conference on Neural networks, 1994, pp. 3490–3494.

[18] G. Martinelli and R. Perfeti, *Generalized cellular ceural network for novelty detection*, IEEE transactions on Circuits and Systems I: Fundamental theory and applications, vol. 41, no. 2, pp. 187–189, February 1994.

[19] D. Martinez, *Neural Tree Density Estimation for Novelty Detection*, IEEE transactions on Neural networks, vol. 9, no. 2, pp. 330–338, March 1998.

[20] C. Moustier and R. Atkinson, *An approach to automated fault diagnosis of hydraulic circuits using neural networks*, Technical Report, University of Exeter School of Engineering and E.N.S.E.R Bordeaux , 9 July 1995.

[21] P. P. Meiler and H. L. M. M. Maas, *Condition monitoring of a diesel engine by analysing its torsional vibration, using modern information processing technology*, Technical Report, TNO, Physics and Electronics Laboratory, 1997.

[22] T. Nogami, Y. Yokoi, H. Ichiba and Y. Atsumi, *Gas discrimination method for detecting transformer faults by neural network*, Proc. of the IEEE International Conference on Neural networks, 1994, pp. 3800–3805.

[23] R. Patton, P. Frank and R. Clark, *Fault Diagnosis in Dynamical Systems – Theory and Application*, Prentice–Hall Inc., 1989.

[24] T. Petsche, A. Marcantonio, C. Darken, S. J. Hanson, G. M. Kuhn and I. Santoso, *A neural network autoassociator for induction motor failure prediction*, Advances in Neural Information Processing Systems, vol. 2, pp. 924–930.

[25] C. Rodriguez, S. Rementeria, J. I. Martin, A. Lafuente, J. Muguerza and J. Perez, *A modular neural network approach to fault diagnosis*, IEEE transactions on Neural networks, vol. 7, no. 2, pp. 326–339, March 1996.

[26] R. Sadanada and S. G. R. M. Rao, *ART1: Model algorithm characterization and alternative similarity metric for the novelty detector*, Proc. of the IEEE international conference on Neural Networks, 1995, vol. 3, pp. 2421–2425.

[27] W. S. Sarle, *Neural networks and statistical models*, Proc. of the Nineteenth Annual SAS Users Group International Conference, April 1994.

[28] A. B. Trunov and M. M. Polycarpou, *Automated fault diagnosis in nonlinear multivariable systems using a learning methodology*, IEEE Transactions on Neural Networks, vol. 11, no. 1, pp. 91–101, 2000.

[29] M. van Veelen, J. Nijhuis and B. Spaanenburg, *Neural network approaches to capture temporal information*, Proc. of the Computing Anticipatory Systems, Liege, August 1999.

[30] M. van Veelen, J. A. G. Nijhuis and L. Spaanenburg, *Emergence of learning methodology for abnormality detection*, Proc. of the Twelfth Belgian–Dutch Artificial Intelligence Conference (BNAIC), Kaatsheuvel, November 2000.

[31] M. van Veelen, *Fault diagnosis and isolation*, Technical Report, University of Groningen, Dept. of Computing Science, May 2000.

[32] D. J. H. Wilson and G. W. Irwin, *RBF Principal manifolds for process monitoring*, IEEE transactions on Neural networks, vol. 10, no. 6, pp. 1424–1434, November 1999.

[33] A. Ypma and R. P. W. Duin, *Novelty detection using self–organzing maps*, Proc. of the 3rd Annual ASCI Conference, June 1997, pp. 236–240.

[34] *Neural technology aids early diagnosis of breast cancer*, Neural Edge: Neural computing in health, no. 9, 1995.

Appendix A

Data sets

A.1 Markov model data set

This data set is a bivariate time series generated by a Markov model. The time series element is created by interpolation of the parameters of the basic functions when switching from one state to the other.

The basic functions are :

$$x1 = a^2 - b^2 + \mathcal{N}(0,c) \wedge x2 = a^2 + b^2 + \mathcal{N}(0,c) \qquad \text{(A.1)}$$

The parameters a, b and c are different for each state. In the normal time series there are three states with the following parameter values:

state \ parameter	a	b	c
1	0	0	1
2	8	4	1
3	−4	−4	1

The transition matrix in the normal time series is printed below, the real values are the chances of a transition from one state to the other.

from state \ to state	1	2	3
1	0.7	0.15	0.15
2	0.15	0.7	0.15
3	0.1	0.2	0.7

In my experiments the number of interpolation steps was 16, thus when switching from e.g. state one to two, parameter a is interpolated from 0 to 8 in 16 steps. This results in two reasonable smooth time series as shown in figure A.1. These series are scaled to the range −1 to 1 to make learning with a neural network easier.

A scatterplot of 'normal' x1 and x2 is shown in figure A.2. In this plot it is clear that the transitionpath from state 3 to 2 is through points of state 1.

The novelty in this time series is a new state with new transitions. The novelty state has the following parameters:

state \ parameter	a	b	c
4	4	8	1

The transition matrix changes also slidely due to the new transitionpaths to the novelty state. This results in the following matrix:

from state \ to state	1	2	3	4
1	0.7	0.1	0.1	0.1
2	0.14	0.7	0.08	0.08
3	0.1	0.2	0.5	0.2
4	0.15	0	0.15	0.7

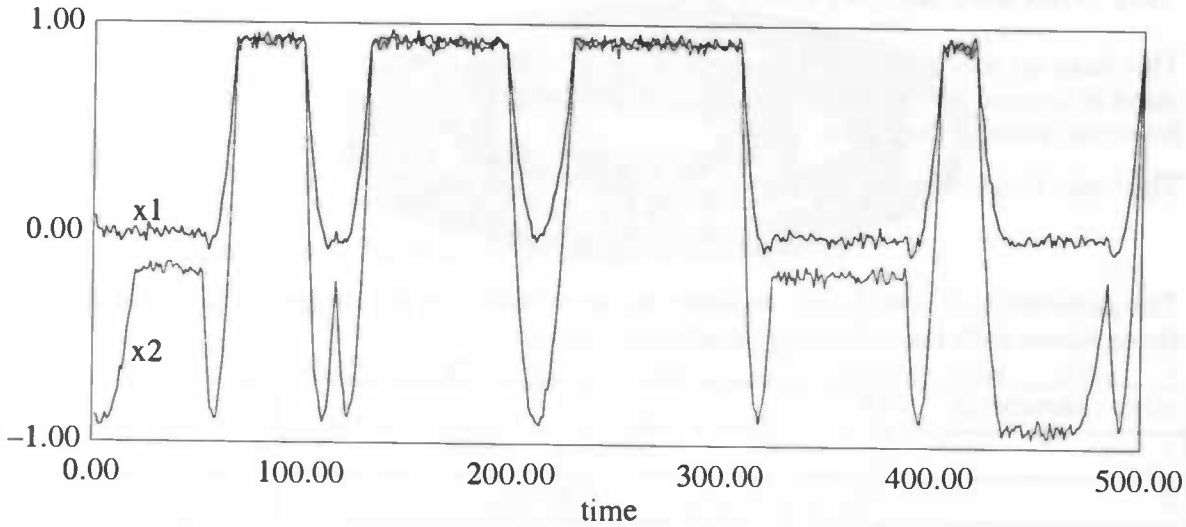

Figure A.1 *First 500 values of x1 and x2*

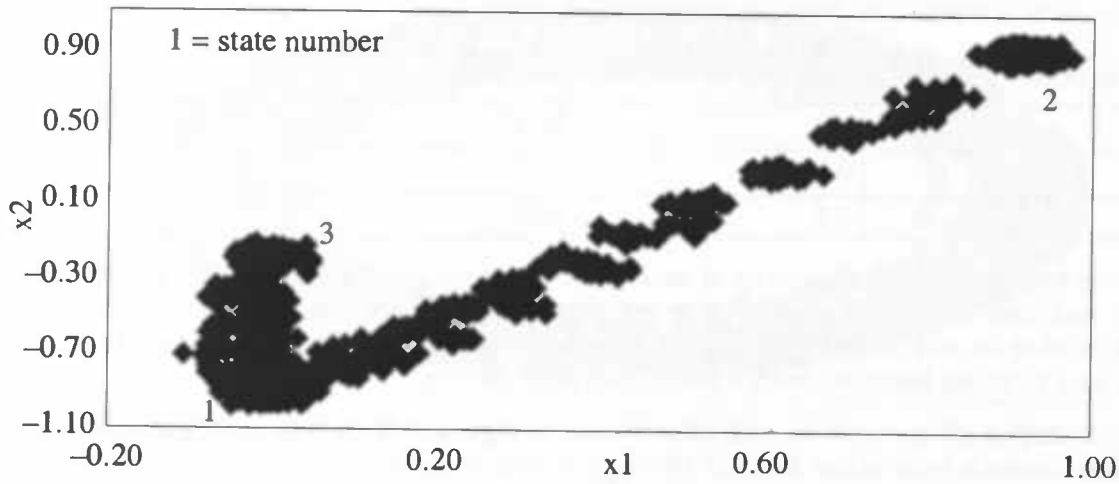

Figure A.2 *scatterplot of normal data set*

The novelty in this time series is an incipient novelty (see paragraph 1.3). A scatterplot of the novelty state and its transitions is shown in figure A.3. Some points of the transition from state 3 to the novelty state cross state 1.

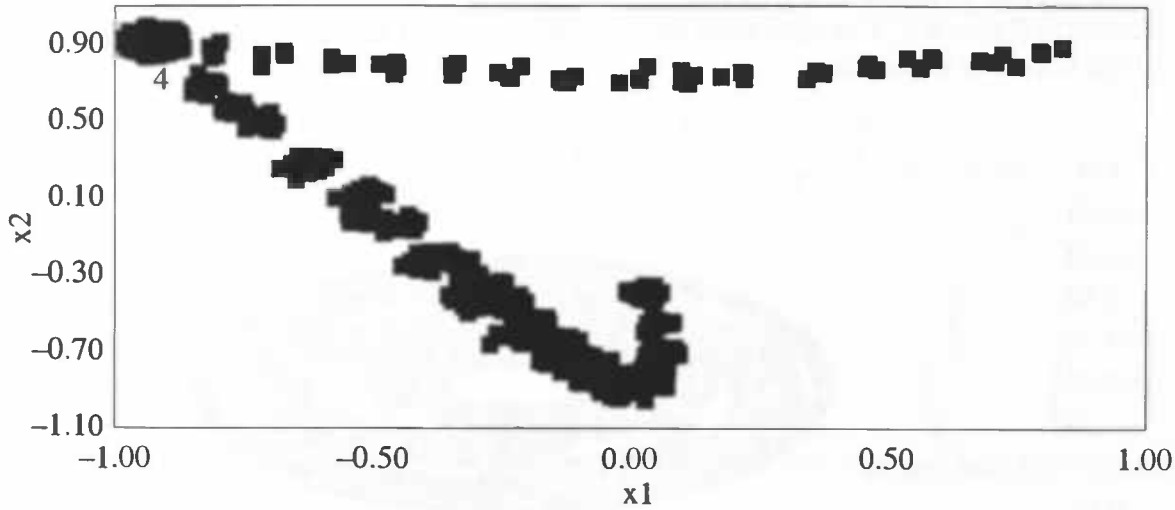

Figure A.3 *scatterplot x1 and x2 in the novelty state and its transitions*

A.2 Circle data set

This data set is also a bivariate data set generated by a Markov model. The two basic functions are a cosinus function and a sinus function with amplitude a and sampled at the same sample rate (eq. (A.2)). In scatterplot form this results in a circle.

$$x1 = a \cdot \cos\left(\frac{t}{10}\right) \wedge x2 = a \cdot \sin\left(\frac{t}{10}\right) \wedge t = 0, 1, \ldots, N \qquad \text{(A.2)}$$

The normal time series exists of two states, the states are represented as circles with different radians. As in the data set from section A.1, in this data set the parameters of the functions are interpolated towards each other when switching from one state to the other. In this series this results, due to the time−parameter and the Markov chance model, in a uniform distributed set of transitionpaths between the states. The number of transition steps used is 50.

The transition matrix for normal behavior is as follows:

from state \ to state	1	2
1	0.7	0.3
2	0.2	0.8

In the first state a has value 0.5 and in the second state 2.0. In a scatterplot this is represented by circles with respectively radians of 0.5 and 2.0. The novelty in this time series is a new state with a is 3.0. The transition matrix is slidely different then in the normal situation, the transition matrix is printed below.

from state \ to state	1	2	3
1	0.7	0.3	0.0
2	0.2	0.6	0.2
3	0.0	0.4	0.6

The novelty is an incipient novelty. When classifying the data by the state in which the data is produced then the classes are non−linear separable. The scatterplots of the normal data-

set is printed in figure A.4 and the scatterplot of data generated in the novelty state is printed in figure A.5. The data set is scaled to the range −1 to 1 to make the modelling with a neural network possible.

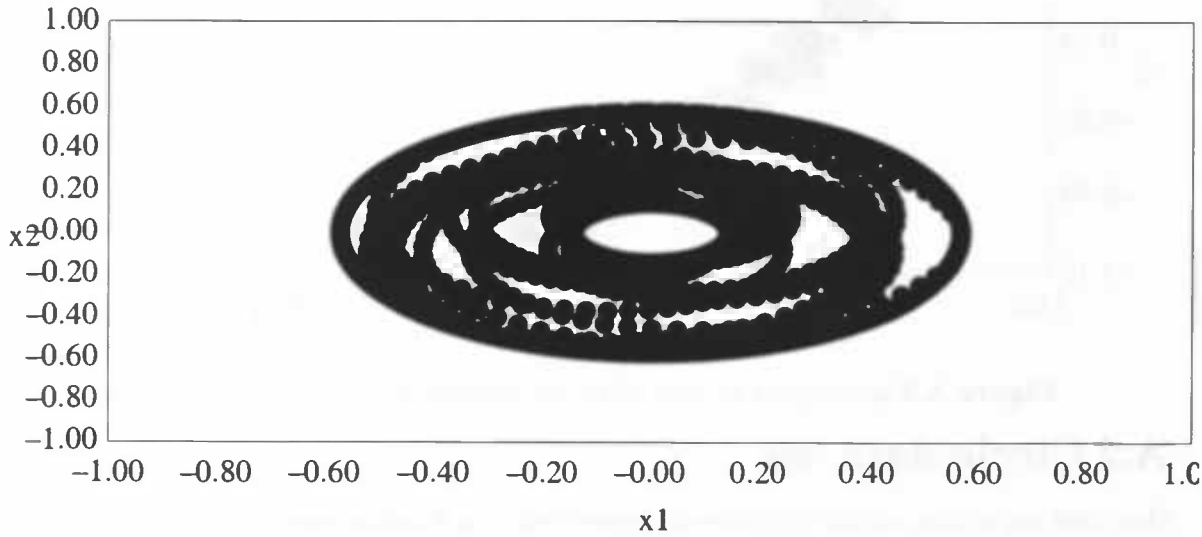

Figure A.4 *scatterplot of normal data set*

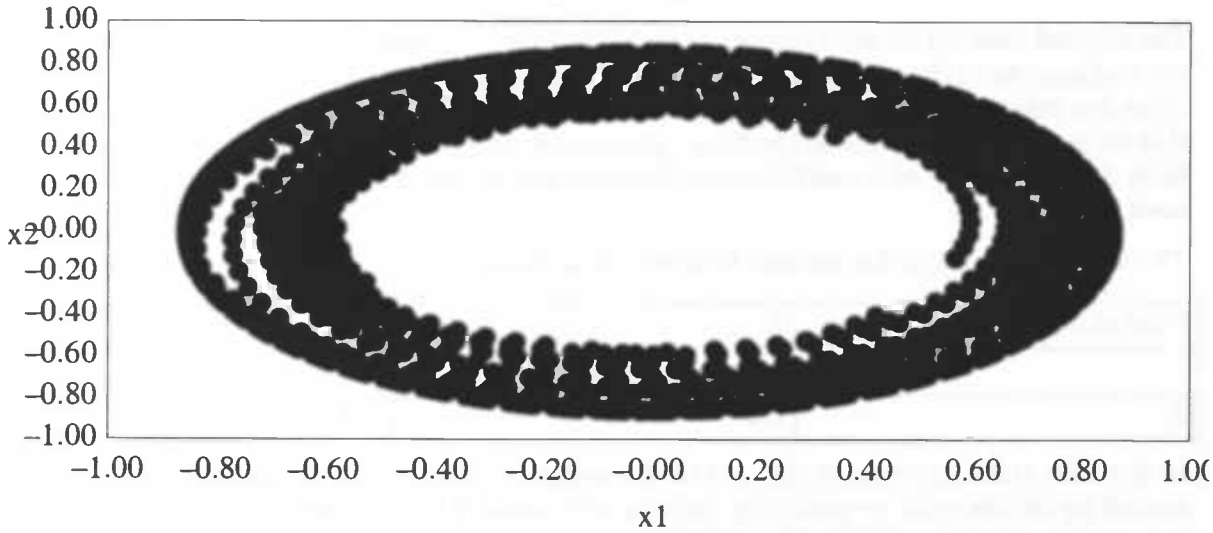

Figure A.5 *scatterplot of data generated in the novelty state*

A.3 Sinus with two frequencies

This time series is a simple univariate sinus with amplitude 1 and mean 0. The frequency of the normal time series is 10 Hz sampled at 190 Hz. The novelty in this time series is a sudden change to a sinus with a frequency of 11 Hz sampled also at 190 Hz. Thus it is a abrubt novelty. One second of the normal time series is printed in figure A.6.

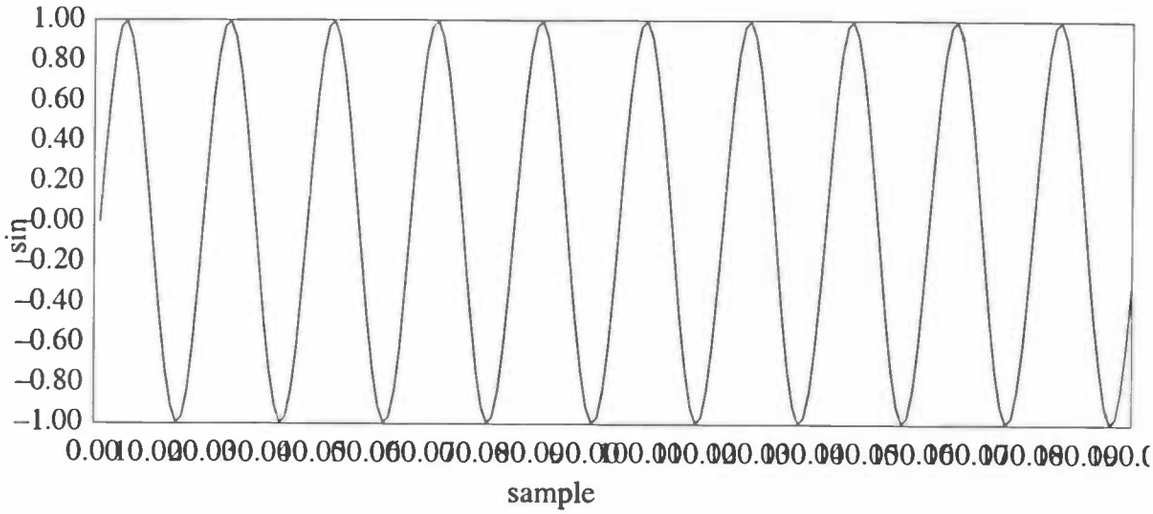

Figure A.6 *Sinus with frequency 10 Hz and samplefrequency 190 Hz*