NON-LINEAR NEURAL NETWORKS FOR MODELING AND ONLINE SEGMENTATION OF NON-STATIONARY VENTILATORY SIGNALS

By

HELENE L. CHINI

A THESIS PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2002

To my family

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Principe, for providing me with the opportunity to work on such a wide and challenging subject. He allowed me a great freedom to experiment and find my own way, while at the same time giving me valuable insights and supervision. I am grateful to Dr. Harris for being the inspiring teacher who awakened my motivation and interest for signal processing at my arrival in the graduate program; and to Dr. van Oostrom for his availability and patience during the past few months.

Special thanks go to my family back home and all my friends, who always supported and strengthened me, and to the best roommate ever this side of the Atlantic.

TABLE OF CONTENTS

page

ACKNOWLEDGMENTS iv

LIST OF FIGURES vii

ABSTRACT x

CHAPTER

1 INTRODUCTION 1

1.1 Goal 1
1.2 Of The Importance of Parameter Settings 3
1.3 Outline 3

2 SEGMENTATION METHODS 4

2.1 Statistical Model of a Signal and Definitions 4
2.1.1 Random Processes and Time Series 4
2.1.2 Probabilistic Averages and Stationarity 5
2.1.3 Ergodicity and Piecewise Stationarity 6
2.2 Methods of Segmentation 7
2.2.1 Classical Sequential Supervised Approach 8
2.2.2 Classical Sequential Unsupervised Approach 8
2.2.3 Competitive Experts Approach 9
2.2.4 Implementation Concerns 15

3 DATA SETS AND MODELS FOR THE DATA 17

3.1 Real World Non-Stationary Data 17
3.1.1 Description of the Data 17
3.1.2 Concerns 21
3.2 Synthetic Data Set Generation 21
3.3 Parameters Associated with the Models 24
3.3.1 Number of Experts 24
3.3.2 Dynamic Modeling 24
3.4 Data Models 29
3.4.1 Degrees of Freedom 29
3.4.2 Linear Experts 31
3.4.3 Radial Basis Function Network Experts 32

3.4.4 Multilayer Perceptron Network Experts 34
3.4.5 Concerns 35

4 SIMULATIONS WITH SYNTHETIC DATA SET 36

4.1 Linear Experts 36
4.1.1 Approach I: Offline Training 36
4.1.2 Other Approaches: Online Adaptation 37
4.2 Radial Basis Function Network Experts 39
4.2.1 Offline Training: Approach I 39
4.2.2 Other Approaches: Online Adaptation 42
4.3 Multilayer Perceptron Network Experts 48
4.3.1 Approach I: Offline Training 48
4.3.2 Other Approaches: Online Adaptation 49

5 SIMULATIONS WITH REAL DATA SET 52

5.1 Segmentation Results 52
5.1.1 Linear Model 52
5.1.2 RBF Network Models 53
5.1.3 MLP Network Models 55
5.2 Results Comparison and Analysis 56
5.2.1 Comparison of the Models 56
5.2.2 Practical Considerations 57
5.3 Simulations with Another Set 59
5.3.1 Description of the data 59
5.3.2 RBF Network Models 60
5.3.3 MLP Network Models 63

CONCLUSION 64

REFERENCES 67

BIOGRAPHICAL SKETCH 70

LIST OF FIGURES

Figure page

2-1 Gated competition of experts 10

2-2 Segmentation example with instantaneous error: black is signal to be predicted, red and blue are two models, the dotted line below is segmentation. 12

3-1 Real world data set: (a) Flow; (b) Pressure; (c) Volume 18

3-2 Presentation of the data: (a) Flow; (b) Pressure; (c) Volume 19

3-3 Cycles used for generating the test set: (a) Flow; (b) Pressure; (A) Mandatory breath; (B) Spontaneous breath. 22

3-4 Synthetic test set: (a) Flow; (b) Pressure 23

3-5 Returns maps of the data for $\tau = 1$: (a) Flow; (b) Pressure; (I) Original data; (II) Test Set. For the test set, spontaneous breath is in red, mandatory in blue. 27

3-6 Returns maps of the data for $\tau = 2$: (a) Flow; (b) Pressure; (I) Original data; (II) Test Set. For the test set, spontaneous breath is in red, mandatory in blue. 28

3-7 Second data set: (a) Flow; (b) Pressure; (c) Volume 30

4-1Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure. 36

4-2 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure. 37

4-3 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure. 38

4-4 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure. 39

4-5 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure. 40

4-6 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure. 41

4-7 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure. 42

4-8 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure. 43

4-9 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure. 43

4-10 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure is for. 44

4-11 History of the gate: full line is for expert 1, dotted line is for expert 2 45

4-12 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure. 46

4-13 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure. 47

4-14 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure. 47

4-15 History of the gate: full line is for expert 1, dotted line is for expert 2 48

4-16 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure 49

4-17 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure. 49

4-18 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure. 50

4-19 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure. 51

5-1 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure. 53

5-2 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure. 54

5-3 History of the gate: full line is for expert 1, dotted line is for expert 2 54

5-4 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure. 55

5-5 History of the gate: full line is for expert 1, dotted line is for expert 2 55

5-6 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure. 56

5-7 Data set: (a) Flow; (b) Pressure. 60

5-8 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure. 61

5-9 History of the gate: full line is for expert 1, dotted line is for expert 2.......................61

5-10 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure. 62

5-11 History of the gate: full line is for expert 1, dotted line is for expert 2.....................62

5-12 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure. 63

5-13 History of the gate: full line is for expert 1, dotted line is for expert 2.....................63

Abstract of Thesis Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Master of Science

NON-LINEAR NEURAL NETWORKS FOR MODELING AND ONLINE SEGMENTATION OF NON-STATIONARY VENTILATORY SIGNALS

By

Hélène L. Chini

December 2002

Chair: José C. Principe
Major Department: Electrical and Computer Engineering

Many real-world time series are multi-modal, where the underlying data generating process switches among several different subprocesses. The difficulty in analyzing such systems is to discover the underlying switching process, which entails identifying the number of subprocesses, the dynamics of each subprocess and the switching pattern among them. Unfortunately, real world time series usually are non-stationary. The goal of this work is to perform online segmentation of a real-world time series, i.e., tracking of the slow evolution as well as detecting the sudden changes. Several algorithms in the framework of gated experts are implemented and tested on biomedical data, using non-linear neural models.

These approaches are first tested on a synthetic data set, to understand the effect of the system and model parameters, then on real data, yielding encouraging results.

CHAPTER 1
INTRODUCTION

1.1 Goal

Good data analysis and forecast are essential for the scientific, industrial and even economic communities. Recent years have seen an ever-increasing interest in analyzing time series, which are collections over time of measurements gathered from a dynamic process. Norbert Wiener initiated the field broadly referred to as “signal processing” in the 1940’s when he started considering a time series as a stochastic process. Areas of time series collection range from meteorology to banking, telecommunications, earthquakes, biology or medicine; and main objectives for time series analysis include describing the data (usually by a time plot or statistics), modeling the data (i.e., identifying a statistical model able to describe the data given its past values) and forecasting (or predicting future values of the series). Good predictions then allow taking actions to control the process generating the time series, particularly when it exhibits abrupt statistical changes.

A basic requirement for traditional time series modeling is stationarity [1], since only one global model is usually used to explain a whole process. But real world time series data are hardly ever stationary. Non-stationarity can be classified in two groups: quasi-stationary signals, and piecewise stationary signals. In the first case, the process’ parameters changes occur gradually over time, and an appropriate way to deal with this problem is by “tracking” the data, or adapting a chosen model to fit the local incoming values, forgetting at the same time other previously learned properties. In the latter case, the process' parameters change abruptly after a certain period of time. Although a single adaptive system theoretically could model any function, such sudden change renders this task very hard. An interesting approach to counter this problem is then to divide the time series in several stationary or quasi-stationary regions, and identify the different models corresponding to those regions [2]. This approach, called segmentation, is more comprehensive than tracking, since it does not necessarily require forgetting past properties to learn new ones, and a past model can be re-used on a new segment belonging to the same regime. Many algorithms attempting to perform such tasks have been developed in recent years, under the framework gated experts models [3], but have not been extensively tested.

The goal of this thesis is to assess the performance and practicality of this framework approach on modeling and performing online segmentation of a particular type of real world non-stationary time series: data recorded from patients under assisted ventilation. Medical time series constitute indeed a great challenge as far as data analysis goes, because they are inherently non-stationary, often presenting impulse-type or complicated shapes, and they depend on so many parameters that it is often difficult to model them: in our case, a lung's behavior can be modeled in analogy with an RC electrical circuit, R being the lung resistance and C the lung compliance (equivalent to the capacity), and those are only two of the basic parameters that can change over time creating non-stationarity. There are many directions of research in this field, but a lot of the latest ones share a common point: instead of being fitted with linear models, biomedical signals are analyzed through segmentation [4, 5] with non-linear or chaotic modeling [6, 7] for 1-dimensional signals such as respiratory ones, or 3 to 4-dimensional signals such as MR images [8].

1.2 Of the Importance of Parameter Settings

The complexity in using the gated experts approach stems from the number of parameters that have to be set specifically for the particular data before the system can work, and the potentially negative impact that one parameter change can have. Key parameters can be divided in two groups: the global system parameters such as performance measure and memory, which account for the accuracy in segmentation and adaptation of the experts (most of which will be addressed in chapter 2), and the local experts parameters, such as the kind of experts (linear vs. non-linear) or the type of basis for non-linear experts (global or local), which determine the prediction by the experts (they will be addressed in chapter 3).

1.3 Outline

This thesis is comprised of six chapters. Chapter 2 presents more in-depth definitions concerning time series, stationarity, and the segmentation methods; chapter 3 focuses on describing data sets and problems of stationarity that entail generating a synthetic test set, then presents the different models used for modeling the data. Chapter 4 covers the results of segmentation by the different methods on the test set. In chapter 5 real data is used, and the results are reviewed and compared. Finally, chapter 6 summarizes the results of the simulations analysis and conclusions, and then opens doors for future work.

CHAPTER 2
SEGMENTATION METHODS

2.1 Statistical Model of a Signal and Definitions

2.1.1 Random Processes and Time Series

A random variable *X* realizes a mapping from the sample space on the real line. It is usually described by its probability density function (PDF) $f_X(x)$, which can be represented roughly as a "histogram" of the probability for a realization of the random variable *X* to take the value *x*, described with respect to all possible values of the random variable . Random variables usually describe systems that are not time dependent. On the contrary, a random (or stochastic) process can describe a system that changes with time. It realizes a mapping from the sample space onto an ensemble of time functions: each realization *X(t,p)* of a random process *X* consists in a well-defined function of time (in the continuous case), or a sequence of values indexed by time (in the discrete case). At a given time *t*, the realization of a random process is a random variable represented by its own PDF.

$$F(x,t) = P\left[X(t) \le x\right] \tag{2.1}$$

$$f(x,t) = \frac{dF(x,t)}{dx} \tag{2.2}$$

The first-order cumulative density function of a random process is defined by (2.1), and its first-order probability density function is defined by (2.2). These definitions generalize to nth order.

2.1.2 Probabilistic Averages and Stationarity

In order to be able to perform a study of the system, the randomness factor has to be overcome: the expected value operator *E[.]* performs a statistical average on the random process, producing the following moments: the first-order moment of the random process called mean (2.3), and its second-order moment called autocorrelation (2.4).

$$m(t) = E[X(t)] = \int_{-\infty}^{+\infty} x\, f(x,t)\, dx \tag{2.3}$$

$$R(\tau,t) = E[X(t)X(t+\tau)] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x_1 x_2\, f(x_1, x_2, \tau)\, dx_1 dx_2 \tag{2.4}$$

A major goal in time series analysis is to be able to estimate the unknown structure of the process by using the available data. This model can only be valid to perform prediction on future data if the process is stable, or stationary. A process is said to be stationary if its statistical properties do not change over time, i.e., for any time delay c, it verifies (2.5), or more precisely, any change of origin in the time domain does not affect any of the n^{th} order cumulative density function.

$$F(x_1, x_2,..., xn, t_1, t_2,...tn) = F(x_1, x_2,..., xn, t_1 + c, t_2 + c,...tn + c) \tag{2.5}$$

That implies that the first-order statistics (mean) are independent of time, and that second-order statistics (autocorrelation), as well as higher order-ones, depend only on the time difference τ but not on absolute time. This definition is very restraining, and therefore a broader sense of stationarity is defined focusing only on the first two order moments: wide-sense stationarity (WSS) or weak stationarity [1]. A process is said to be WSS if its mean is constant and its autocorrelation depends only on the time difference.

All definitions given so far concern continuous-time random processes, but they all have a discrete time equivalent in replacing the integration by a summation. A time series is a sequence of observations that are ordered in time, usually sampled for computational analysis, so the times series that we are interested in is a realization of a discrete-time random process. Yet we aim to study the properties of the random process through only one realization, this is allowed by the fundamental property of ergodicity.

2.1.3 Ergodicity and Piecewise Stationarity

Most of the time, a process cannot be stopped and reset to its initial values, or not enough times to get a sufficient number of realizations to perform statistical averaging. Instead of performing averages on the number of realizations, it is more practical to average over time. A process is called ergodic if all orders statistical averages of the process are equal to the time averages (assuming there is an infinite number of data points, or $t \to \infty$). This property allows us to use one realization of a random process, i.e., one time series, to find its statistical properties (usually just a few realizations are available, but they contain a lot of data points). Practically, in all time series analysis methods, the assumption of ergodicity is assumed to hold, because it would not be possible to study the statistics of the process otherwise. It is interesting to note that ergodicity infers stationarity, and therefore the converse is true too: non-stationarity infers non-ergodicity. Therefore in many cases (particularly this one) the results derive from a known wrong assumption!

Most of the times real data is non stationary but can be considered multi-modal with stationary modes, and this is the basic assumption in segmentation problems: the time series is assumed to be not stationary but piecewise stationary. A piecewise stationary signal can be described in two different manners: both have in common the state space representation of a model including a parameter vector q. The first model, called "parameter switching" assumes that from one stationary segment to another, q is changed but the state of the system is kept intact, and creates transient effects in the case when the system has memory. The second model, called "external switching", assumes that not only the parameter q but the whole state of the system switches from one subprocess to another, therefore entailing no transient effects in the case of system with memory. Those differences are in general of no great importance if the switching period is greater than the memory depth of the system, and in the framework of competition of experts, we will consider that the process is submitted to external switching, which is perfectly represented as each expert tries to model one subprocess.

2.2 Methods of Segmentation

Segmentation is a method of detection of change in a dynamical system that can be used as integral part of the modeling process, as an alarm during the monitoring of that system, and as a tool for improving the tracking capability of an adaptive algorithm in the presence of non-stationarity [2]. A key issue is the parameterization of the segmentation structure: it is indeed dependent on the choice of several parameters, either by decision of the designer or experimental manual tuning.

A first global parameter is to decide what is to be detected: given a data set, one could want to detect sudden changes in signal amplitude, jumps in mean, or more radically a change in the whole waveform, which is our goal in this thesis. This parameter is crucial because it impacts most of the other parameters.

Another obvious parameter is the number of experts: it can be set a priori if enough information about the data is known, or by using techniques such as tree growing or pruning algorithms, which are not examined further in this thesis.

Other important parameters exist that are specific to each segmentation method, explored in the following sections.

2.2.1 Classical Sequential Supervised Approach

The first step for understanding segmentation is to understand supervised change detection in a signal presenting switching among a finite number K of known processes. In that case, predictive models are available (or developed for each process using measured time series), and represented by their PDF's $p_k(X(n))$, $X(n)$ being the history of the measurements $x(n)$ over n time steps.

Then the time series is monitored online by computing at every step for any two processes the log-likelihood ratio of the processes PDF's values at that time step, as shown in (2.6).

$$L_{ij}(n) = \log\left[\frac{p_i(X(n))}{p_j(X(n))}\right] \text{with i,j=1....K} \qquad (2.6)$$

The log-likelihood ratio is a very important tool in sequential change point detection. Indeed, it serves as the decision function: basically, once a starting regime i has been identified, all $L_{ij}(n)$ are monitored for $j \neq i$, and whenever one $L_{ij}(n)$ goes above a set threshold, it means there has been a change of regime at that point. This method needs therefore offline training of the experts, and then an online segmentation is performed.

2.2.2 Classical Sequential Unsupervised Approach

In the case when the number of states (or subprocesses) is very large or completely unknown, as well as the change points between the regions, the classical approach to segmentation focuses on a sequential detection of change points in the data by developing a local model of the data and monitoring its dynamics for change, as for the supervised case. Contrary to the supervised case, no a priori information about the regimes can be used, so only two PDFs are used: at each time step, one PDF is estimated assuming there was no change in the data stationarity (hypothesis H0), and one assuming there was a change at a certain time step inside the window (hypothesis H1). Then the log-likelihood ratio of those two PDFs is monitored until it reaches a threshold, indicating a change from H0 to H1, and records the position of the change point and starts again. This approach [9, 10], implemented by the generalized likelihood ratio (GLR) test does not keep a memory of past regions since all information is discarded after detection of a change point. The processes we are interested in switch between a small number of subprocesses, therefore it seems more appropriate to consider approaches that can re-use previously learned information. Although the GLR test is usually considered as a benchmark to compare other segmentation algorithms to, it has a high computational complexity, especially with non-linear predictors, and will not be used in this thesis.

2.2.3 Competitive Experts Approach

2.2.3.1 The mixture of experts

Another approach can be taken when some information about the data is known: more specifically when the data is known to switch among a finite number P of subprocesses, called regimes, it is indeed interesting to be able to identify a new segment of data to an already trained expert. One obvious way to preserve the information learned about each subprocess is to train one predictor (or expert) for each subprocess. An application of this idea is the Mixture of Experts (MOE) model [11], where several experts and a gate see the same input: the output is a sum weighted by the gate of the experts' outputs, and the system attempts to model the overall PDF of the data as a mixture of the experts' PDFs, assumed to be Gaussian. During the training phase, the overall output of the system can be chosen as a mixture of the outputs of the experts as described in (2.7), where g_p is the output of the gate for the pth expert.

$$output(n) = \sum_{p=1}^{P} g_p(n) \cdot output_p(n) \tag{2.7}$$

The gate ensures that most successful experts receive the most update, and the system is trained on the entire data set, then segmentation is performed using a classical supervised sequential method.

2.2.3.2 Gated competitive experts

The MOE model fits in a larger framework called gated competitive experts (not necessarily with Gaussianity assumption). The systems all fit the same general architecture described in Figure 2-1, but segmentation can be performed simultaneously with the modeling by using local-in-time performance of the experts.

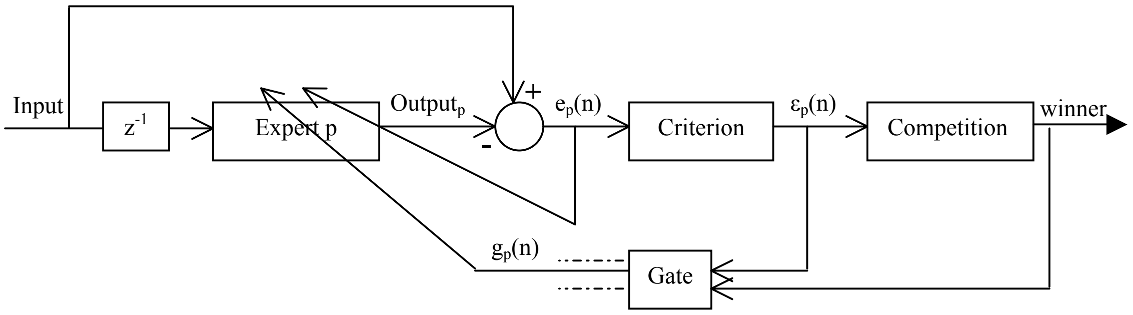

Figure 2-1 Gated competition of experts

Several adaptable experts compete to explain the same input data, presented through a tap delay input line. A gate and a competition mechanism monitor their performance, measured by a criterion, decide which expert is the best predictor (winner) at any time step, and provide a moderation for the amount of update experts receive.

The criterion. As with all time series modeling, it is necessary that we be able to predict the data a step in advance in the future given previous data, which means that some measure of the prediction error has to remain small. Some adaptive systems use the instantaneous error $e_p(n)$ as error measure. Unfortunately, for our application $e_p(n)$ cannot be used since it is shown that the sequence e_p is an insufficient statistic [12], which means that different experts could produce identical error statistics (or there is a many to one mapping from different regimes to identical prediction error statistics). This also explains why segmentation cannot be performed by only tracking the data with one adaptive system and monitoring $e_p(n)$. Besides, in cases where the return maps of several regimes tend to be close and where the data is noisy, using the instantaneous prediction error as measure for the performance of the experts will probably cause the segmentation to exhibit spurious switching (or false alarm), even when it is known that the data switches less frequently.

As can be seen in Figure 2-2 on a trivial example, if the segmentation is performed using $e_p(n)$ as criterion, even though Model 1 is not a good predictor at all for the data it is still seen as the best predictor in areas A and B. In order to avoid this kind of "rattling" from a model to another, a solution that brings memory to the performance criterion is introduced [1].

The criterion has to be a measure of an average performance at that point in time, so an good criterion would be the expected value of the square error $E[e_p^2]$. With the useful lie of assuming that the system is ergodic, it can be expressed by the mean square error (MSE).

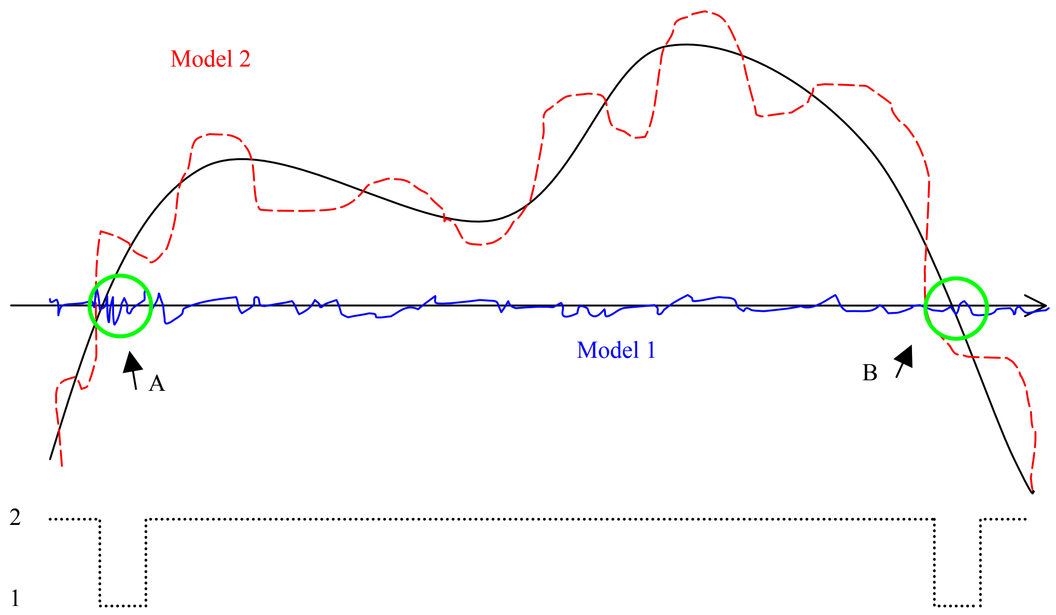

Figure 2-2 Segmentation example with instantaneous error: black is signal to be predicted, red and blue are two models, the dotted line below is segmentation.

The criterion used is the instantaneous mean squared error $\varepsilon_p(n)$, computed with a recursive estimate on the instantaneous squared error $e_p^2(n)$, using a gamma delay (2.8). The parameter γ_{eps} is the inverse of the memory depth of the filter in samples [13], and taking the expected values on both sides of (2.8) shows that $E[\varepsilon_p] = E[e_p^2]$, or ε_p is an unbiased estimate of estimate of the MSE.

$$\varepsilon_p(n) = \gamma_{eps} e_p^2(n) + (1 - \gamma_{eps})\varepsilon_p(n-1) \qquad 0 < \gamma_{eps} < 1 \tag{2.8}$$

Now γ_{eps} is another parameter to set, effectively controlling the “window” over which the MSE is calculated. If the memory depth of $\varepsilon_p(n)$ is too short, it is not really an average anymore, and some spurious switching might appear; if it is too long, the error is integrated over too much data, and the change detection might not be as sharp. In fact, the $\varepsilon_p(n)$ has the regretful property of building up when the instantaneous error is high (when the expert is not a winner), and often the decrease in criterion of the expert that should be a winner does not compensate fast enough, the other expert stays a winner and adapts to the new regime, so that there the system barely or never detects a switch (Figure 4-7). Therefore the determination of the memory depth is a compromise between unwarranted switches and detection delays or misses.

The gate. The gate is the part of the model that decides the assignment of a specific regime to a particular expert, by ensuring that most successful experts receive the most update. The gate can be input-based (i.e., an adaptable function of the input, that learns to forecast which expert will perform best), or output-based (i.e., a calculated function of the output of the experts, or of their performance). All gates studied here are output based.

In this framework, the competition mechanism and the gate can be coupled: the history of the output of the gate does provide segmentation information, but since we are attempting online segmentation concurrent with the training of the experts, the gate's output is not necessarily binary, so for the sake of clarity the competition mechanism for the segmentation output is defined separately.

The winner is chosen with respect to its performance as shown in (2.9), $\varepsilon_p(n)$ being the value of the performance criterion for the pth expert at sample n.

$$winner(n) = \arg\min_p \lfloor \varepsilon_p(n) \rfloor \tag{2.9}$$

In Figure 2-1 is depicted the common structure used along this thesis, with several types of gates and competition, and several types of experts such as linear filters, multilayer perceptron, or radial basis function network (another example of interesting experts is PCA networks [14], where then the competition is based on how well the experts can compress and decompress the data rather than predict it).

2.2.3.3 Approaches used for simulations

For simulations presented in this thesis, the winner is always the expert with the best performance (i.e., lowest criterion value), but the gate and the way the experts are adapted and monitored vary. Most of the algorithms in the gated competitive experts framework require repeated training of the non-linear experts through the whole data set. In this thesis, we try to develop an online segmentation method, with only one run through the data. Therefore the training is done in an online fashion, at the same time as the segmentation, and the output of the system is chosen to be only the output of the winner at that particular time step (2.10).

$$output(n) = output_{winner(n)}(n) \tag{2.10}$$

This would ensure that the experts actually track the regimes, and are updated as more information comes. We can assume that if some process parameters exhibit a slow change (like the lung compliance), this approach will adapt the experts so that they still recognize each type of breath. Following are the approaches chosen for the simulations, the first is supervised, and the other ones are unsupervised.

- Approach I: For every type of expert, first a sequential change point detection is performed, where the experts are trained on a sample data set of the regime they are supposed to explain, and then by processing all the data through the experts, the criterion is monitored, and a winner is chosen at every time step without adapting the experts. Since the data we consider is not even piecewise stationary (there are changes in any regime from one segment to the other), this approach is not expected to perform well.

- Approach II: The simplest gate model is the identical to the hard competition (2.10) and will be the first tested in the simulations. In this approach the experts are trained before the segmentation, as in approach I, but at every time step the winner is updated.

$$\begin{aligned} g_{winner}(n) &= 1 \\ g_p(n) &= 0 \qquad for\ t \neq winner(n) \end{aligned} \tag{2.10}$$

- Approach III: In cases where the return maps of the different experts overlap, simulations show that often with the hard competition (approach II) an expert starts to win, gets adapted and starts fitting the other regime. In fact, since the regimes only differ for the part of the breathing cycle after the end of inspiration, there are not a lot of samples that allow the criterion to show a change in regime, which creates a problem associated to the "build up" property of the criterion. So instead of just monitoring the MSE estimate, we also monitor its slope for detection of unusual spikes indicating that there might have been a change of regime, in which case the winning expert is not adapted, but the second one is (the threshold on the slope is determined visually, depending on prior simulations for the same data)

- Approach IV: This last approach borrows to the annealed competition of experts (ACE) [3] and the self-annealing competitive prediction (SACP) [15] algorithms. The algorithm is identical to the ACE in the fact that the gating function is formed under the assumption that all experts have equal prediction error variance, but not assuming a Gaussian distribution of the error: the gating function is defined accordingly to the SACP as in (2.11), normalized in (2.12).

$$\varphi_k(n) = \left[\varepsilon_k(n)\right]^{\frac{-M}{2}} \tag{2.10}$$

$$g_g(n) = \frac{\varphi_k(n)}{\sum_K \varphi_k(n)} \tag{2.12}$$

M being the parameter that determines the degree of competition: the higher M, the harder the competition. Since the real data is not even piecewise stationary and might exhibit some slow changes, M will not be annealed to a very high value (we cannot assume that after several breaths in the same regime we have more information about it: this is true for the test set, but not for real data). Also, the experts are adapted at every time step with a learning rate proportional to the gate output like for the online ACE.

2.2.4 Implementation Concerns

Most of the difficulties in using these methods lay in the number of parameters that need to be set, and their effect on the performance of the system. For example the issue of the size and initialization of all the experts or their training parameters is addressed in the next chapter, but the competition and memory parameters also play a great role in achieving a good segmentation. While the segmentation methods are designed to be completely unsupervised, we are tackling a particular problem with its own specificities, and they should be included in the design of the system.

One particularity is our goal to achieve online segmentation, which means only one epoch through the data and training at the same time as segmentation. Choosing the MSE estimate helps preventing spurious switching during segmentation by adding memory to the performance criterion, its memory depth has to be set for each expert. Since we have no reason for a priori distinguishing one expert or from another in our case, and considering the expert's return map have close trajectories as we show in the next chapter, γ_{eps} is chosen to be the same for each expert, and since the real data is non-stationary, it will not be annealed. Therefore, after testing several values on the test set defined in the next chapter, γ_{eps}=0.01 is chosen to define a memory depth of one hundred sample points, which seems logical because it corresponds roughly the length of a breath cycle. But such a long memory depth also accounts for the "error buildup" issue evoked in approach III. Also, the more memory about the experts' performance is available, the more information the gate has about the experts' performance, but the less accurate the segmentation becomes, because the relative importance of a change point in the criterion decreases, so the switch might come a few samples late.

Lastly, concerning approach IV, the same competition parameter is set for all models, for a really soft competition at M=2. This allows the non-winning expert to still adapt a little to the data without loosing its specialization (the learning rates are chosen so that the experts do not adapt too much).

CHAPTER 3
DATA SETS AND MODELS FOR THE DATA

The data used in this thesis belongs to the biomedical field, but this application is only one of many that can come to mind when thinking about real-world signals. Most of the simulations results are obtained on a particular time series presented below.

3.1 Real World Non-Stationary Data

3.1.1 Description of the Data

Ventilators are used for patients who's respiratory muscles are too weak to breathe consistently on their own. A ventilator's goal is to optimize the patient's work of breathing (i.e., the ventilator has to be set so that the patient is producing just the right amount of effort to take a breath). In order to optimize the parameter settings of the ventilator and the comfort of the patient, a physician needs to have a good model of the patient's lung and to be able to monitor the dynamics of the respiratory system.

The simplest model is called the One Compartment Model, because it does not make a differentiation between the two lungs. This model is defined by (3.1), where P is the driving pressure in cmH_20, R is the resistance to flow (or variation in volume) in $cmH_20.L^{-1}.sec^{-1}$, $\dot{V}$ is the airflow in $L.sec^{-1}$, E is the elastance in $cmH_20.L^{-1}$ and V the volume in L.

$$P(t) = R \cdot \dot{V}(t) + E \cdot V(t) \qquad (3.1)$$

The lung resistance R and its compliance $C = \frac{1}{E}$ are considered constant, although in reality they are not and their variations actually are one source of non-stationarity.

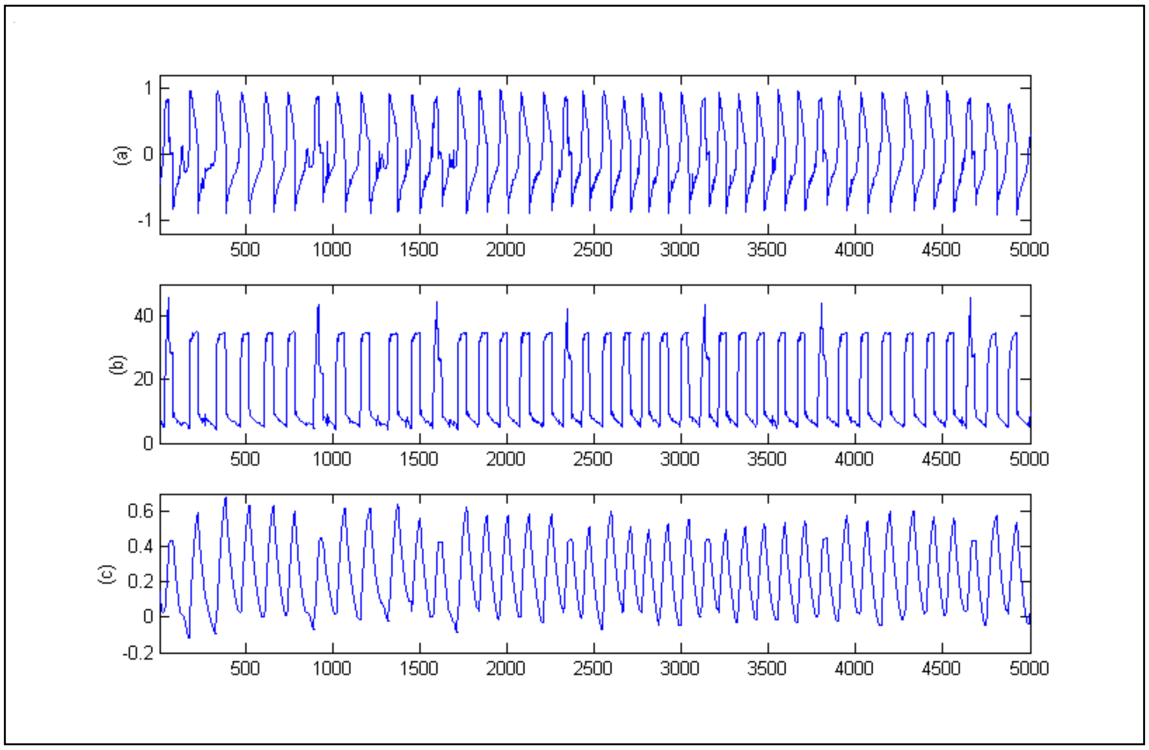

Figure3-1 Real world data set: (a) Flow; (b) Pressure; (c) Volume

Figure 3-1 shows the four basic descriptive parameters of mechanical ventilation: flow, pressure, volume and time. The data set is not shown in its entirety (30,000 samples) but the extract is representative. The series was composed of two sensors' recordings (flow and pressure) and a computed quantity (volume) performed at a rate of 100 Hz, which can be an oversampling when the "periodicity" of the signal is often less than a breath per second. The original data was oversampled for a fitting by a linear filter, therefore we chose to downsample it to 50 Hz (to be able to perform linear modeling and still have enough data for non-linear model training). The volume graph will only be shown in this chapter, since it is usually not a measurement but a calculated integration of the values shown in the flow graph, and therefore it does not bring any more information.

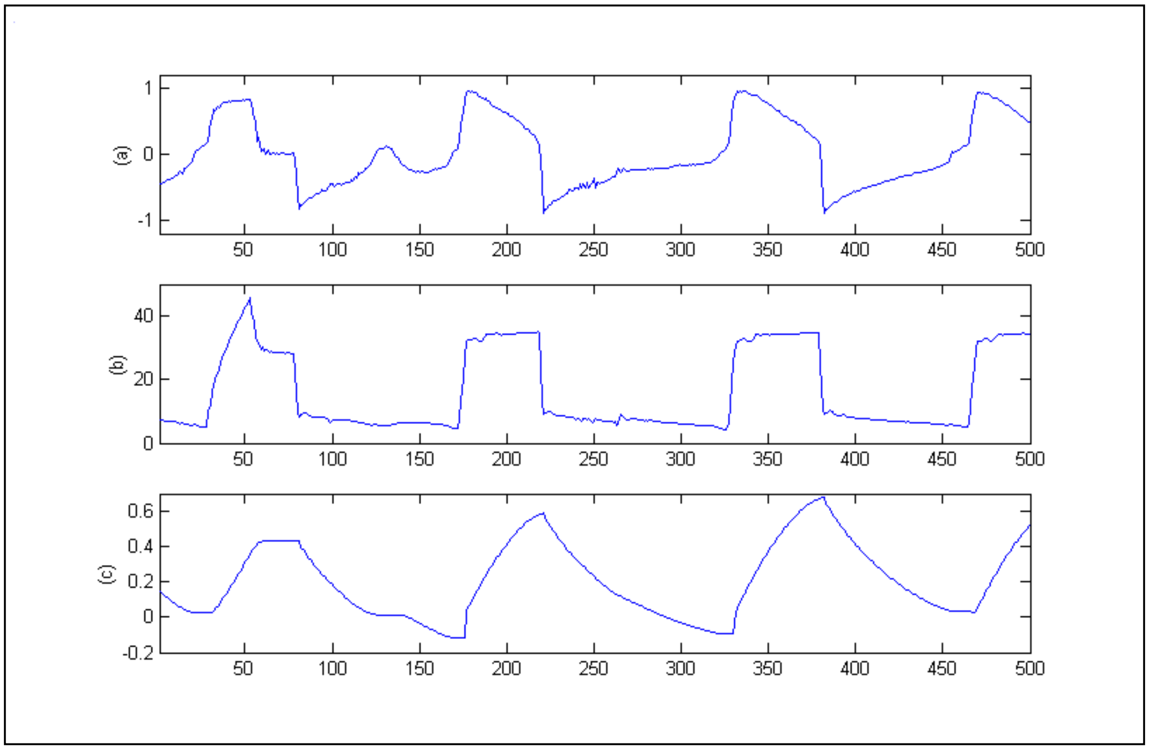

Figure3-2 Presentation of the data: (a) Flow; (b) Pressure; (c) Volume

A close up on a few breaths in Figure 3-2 allows us to analyze the data: the time series exhibits two different types of breath. As it was recorded from a patient under assisted ventilation, it is composed of a sequence of segments belonging to two different regimes: switching between the two regimes occurs rapidly: the patient either breathes in one mode or the other.

The first breath shown has a plateau in pressure and a flow at zero after the end of the inspiration: it means that the inspiratory phase is finished, but the volume of air delivered to the lungs is held inside by blocking the exhalation valve shut. The expiratory phase is halted to create a distinctive shape that allows physicians to measure the patient’s lungs parameters. Indeed, as can be seen from (3.1), an analogy can be made between this model and a simple electrical RC circuit. Therefore, the same methods used for estimating values of components in an RC circuit from voltage waveforms can be used to estimate the lung parameters from the pressure waveform: the value of the pressure at the plateau indicates the pressure needed to inflate the lungs and hence their compliance C; the amount of drop in pressure before the plateau indicates the pressure due to flow resistance of the patient and the endotracheal tube, and can be used to estimate the lungs' resistance R [16]. Such breaths are called "mandatory" because the patient does not initiate them: the ventilator is set to administer a minimum number of breath cycles per minute, and they are used at the pressure pattern we saw to provide the physician with more information, even though ventilator measurement and calculations are made mostly outside the patient's body, so it is difficult to obtain accurate estimates. Also, this data set has been chosen for a particular feature: the plateau in pressure is perfectly readable, which means that the patient is not fighting the ventilator. All patients indeed do not react the same way under artificial respiration, and the data is not always so close to the theoretic waveforms.

The second and third breaths reflect a different pattern: they exhibit a slight negative deflection immediately before the rise in pressure. This deflection indicates that the patient is trying to inhale, and triggers a breath, which is why they are called "spontaneous" breaths. They do not actually represent a breath that the patient takes on its own, just a breath that was spontaneously triggered by the patient. Lastly, for both kind of breath, the exhalation phase is performed by the patient since it is passive leveling of the lung pressure to the outside pressure, and depends only on the resistance of the airway and the compliance of the lung [17].

Knowing just how many spontaneous and mandatory breaths the patient takes per minute is important information for a physician, but ventilators usually do not record how each breath was initiated. The goal of this thesis is then to “classify” each breath, or segment the times series between the two different modes: mandatory or spontaneous breath.

3.1.2 Concerns

Considering the data and the task we propose to study, several problems appear.

To perform the segmentation, we need to use one of the times series and try to model it. The pressure exhibits a near square waveform, with very high and very low frequencies, and this would be hard for any adaptive system to model, so the flow waveform is chosen (and since it is centered around zero and has an amplitude of two L.sec^{-1} whereas the pressure spikes at forty cmH2 0., is does not need any preprocessing before being used in a non-linear model). But the segmentation is visually more easily done on the pressure graph, so it is kept as a visual reference.

Another problem is the obvious non-stationarity of the real data sample, due to several factors: as said previously, the lung parameters are not constant in reality, the patient might become agitated and breathe too fast for the ventilator settings, there is noise in the recording that can be non-stationary, and other parameters could change. In order to compare the performance of the several models and segmentation methods, we need to have some control over the data. For that purpose, we create a synthetic test set using the real samples, so that some control is gained over the variability of the original data while its specificity is preserved

3.2 Synthetic Data Set Generation

As just stated, a synthetic test set is important because it allows us to test the systems in a controlled environment, to know that there are only two regimes, meanwhile keeping some of the nonstationarity and versatility of the original data, which still gives insight of performance of the segmentation methods with real world applications, not just on computer generated random processes.

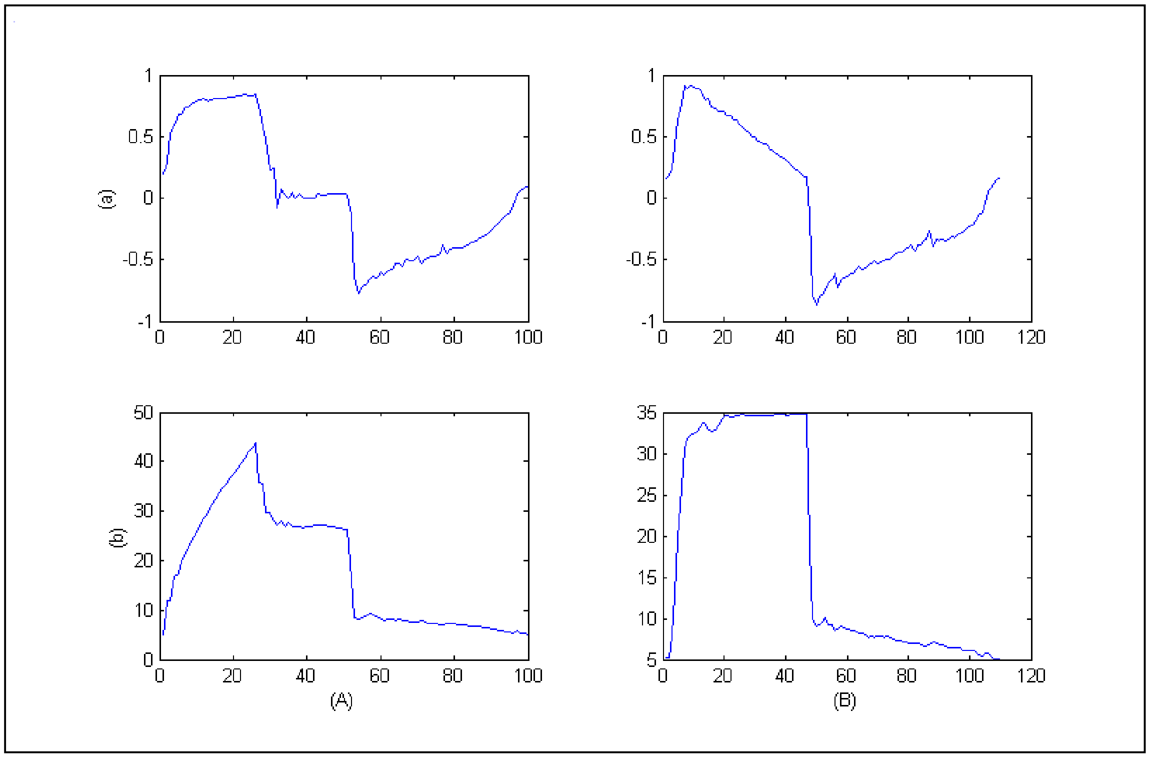

Figure3-3 Cycles used for generating the test set: (a) Flow; (b) Pressure; (A) Mandatory breath; (B) Spontaneous breath.

The first step to generate this data set is to excerpt two representative breath cycles from the original signal, which serve as components for the test set. Although creating the test set this way implies neglecting the fact that every breath has a different length in the real set, it will allow us to study the performance of the segmenting algorithm for the particular case when breaths have the same length, and to focus on the effect of other waveform particularities such as shape and amplitude. The effect of breath length will be seen in chapter 5 in the simulation results with the original set. Samples 3108 to 3207 complete a full mandatory breath labeled A, and samples 3318 to 3427 complete a spontaneous breath labeled B, as shown in Figure 3-3. In both cases the breath starts and ends with the same low pressure, and the chosen cycles' waveforms are representative of the general waveforms of the breaths.

The second step consists in sequencing the two sample cycles to create the synthetic set. The sequence is chosen to be as in (3.2), as shown in Figure 3-4.

$$[B\ A\ B\ B\ B\ B\ A\ B\ B\ B\ A\ B\ B\ A\ B\ A\ A\ B\ B\ B\ B\ A\ A\ A\ B\ B\ A\ B\ B] \qquad (3.2)$$

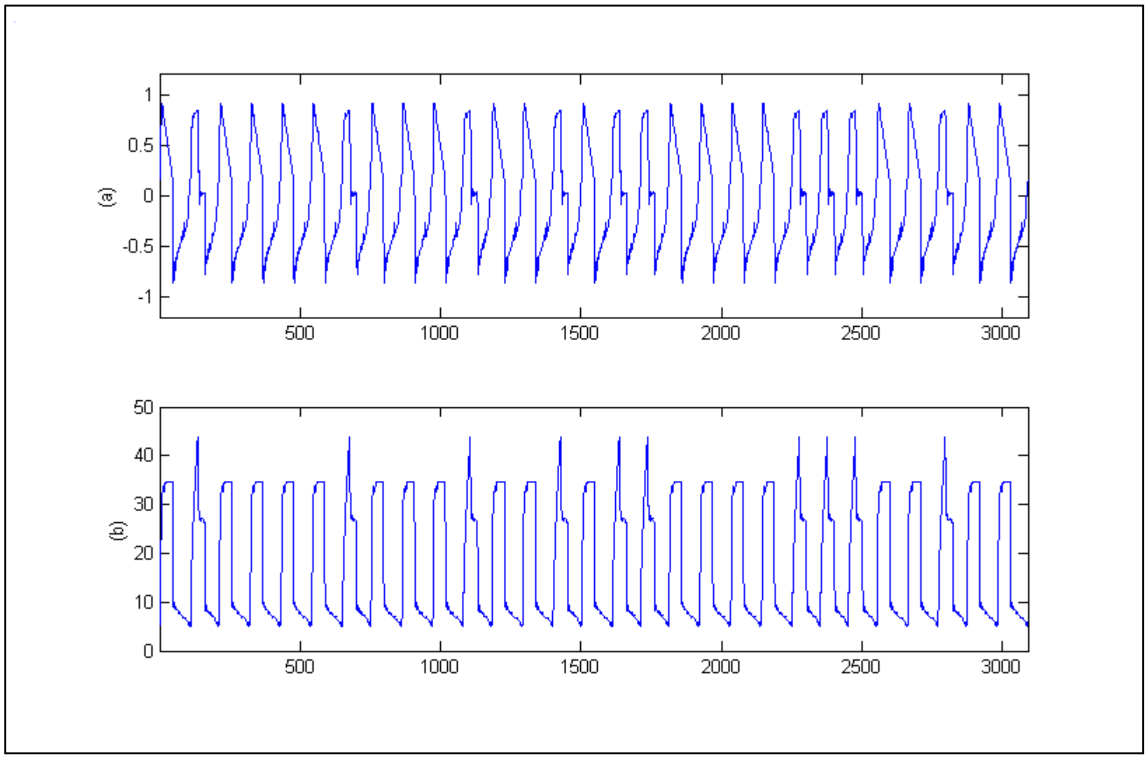

Figure 3-4 Synthetic test set: (a) Flow; (b) Pressure

The beginning part of the sequence is faithful to the type of data considered, with one mandatory breath followed by several spontaneous breaths, whereas the end of the sequence is more unrealistic with two or three mandatory breaths in a row followed by a few spontaneous ones. This was done to create some diversity, test the system to the maximum switching speed, and beyond the particularities of this data to give more insight into the performance of the systems for segmentation of other dynamical behaviors or other switching patterns.

3.3 Parameters Associated with the Models

3.3.1 Number of Experts

Before discussing any model parameter, a choice has to be made of how many experts are going to compete to predict the data. The simplest choice in our case is to consider two experts, since we are trying to discriminate between two different regimes, but the system could also be designed with three experts for example (one for the expiration phase, since it belongs to the same regime in both case, and one for each inspiration mode).

3.3.2 Dynamic Modeling

The data is going to be used as input to the different types of experts: linear filters, and non-linear networks such as radial basis function networks (RBFN) or multilayer perceptron (MLP). These are static neural networks, in that they do not depend on time: the order of presentation of the data would not change their final performance.

Obviously, in time series analysis, time is an essential parameter for the modeling process. Since it is not explicitly used as parameter, it has to be provided to the experts by an implicit representation, to give the static networks dynamic properties. In order for a network to be dynamic, it has to have short-term memory. Through supervised learning, long-term memory is achieved by storing in the synaptic weights the information contained in the training data. But short-term memory can only be achieved by embedding memory into the structure of the network. One easy way to do so is to use time delays lines at the input layer of the network, because the temporal structure of the data is then embedded in the spatial structure of the input signal, and therefore of the network. Delay lines are actually a physical implementation of Taken's embedding theorem.

The embedding theorem. Taken's embedding theorem [18] allows for states to be reconstructed from observations. Let a vector time series defined by (3.3), $x(n)$ being the sample at time *n* of the time series. The times series described by $z(n)$ creates a path of observations in the Euclidian space of size k.

$$z(n) = \{x(n), x(n-\tau), \ldots, x((n-(k-1)\tau)\} \tag{3.3}$$

The embedding theorem states that for a certain value K of k sufficiently large (and the in the absence of noise), the trajectories of the data never intersect in that state space, and k>K is a sufficient condition for the existence of a diffeomorphism that maps the current state $z(n)$ to the next $z(n+1)$. Assuming the underlying dynamics of the system are smooth enough to be modeled from the observable, this one-to-one mapping is the systems dynamic description that the network will try to model; K is called the embedding dimension, and τ is called the lag.

Note that the standard embedding uses as simple time delay embedding, but quantities other than delayed observations can be useful in some applications, such as: the time since the last local maximum or minimum of the time series; the result of a change of base in the reconstruction space obtained by multiplying each state vector $z(n)$ by a weighting matrix of full rank K (guaranteeing the existence of the mapping); or the trajectory of the data in a memory space created by the implementation of the embedding with a gamma delay line.

Theoretically, if the embedding is performed perfectly, any nonlinear adaptive system can identify the mapping in the reconstruction space, and therefore would not be helpful in determining the best modeling strategy [19]. All these considerations of course assume stationarity of the data, which is not our case and therefore we can still expect different behaviors from the predictors. Also the simplifying assumption is made that the vector observations $z(n)$ exactly describe the underlying system dynamics, when they are actually computed with observed values $x(n)$ that could be corrupted by noise, so any disturbance is actually considered part of the system's dynamics to be modeled.

Choice of embedding. Arguably "the question of what is a good embedding cannot be answered separately from the question of what kind of model one is trying to build" [20], but for coherence purposes, the embedding is chosen to be the same for all models. It is also set to be the order of the linear filter, since the output of a linear filter is nothing else than a linear projection of the state space input trajectory on the hyperplane defined by the filter weights. Two parameters have then to be determined: the embedding dimension K and the lag τ .

The return maps of the data ($x(n)$ plotted versus $x(n-\tau)$), for $\tau = 1$ (Figure3-5) and $\tau = 2$ (Figure3-6), show how the lag impacts the reconstruction space: the higher the lag, the further the trajectories are from each other, but they still intersect at the same points. The choice of the lag parameter is only limited by the fact that τ has to be such that $x(n)$ and $x(n-\tau)$ are sufficiently independent from each other to define distinct dimensions in state space, but still somehow correlated so as to carry information about the systems dynamics.

In this thesis the focus is set on segmenting the data, with models accurate and specific enough to discriminate between them, more than achieving a perfect embedding and prediction, so we chose the commonly used lag $\tau = 1$ and we will not further consider methods for finding the perfect embedding, but the possibility of fine tuning this parameter should be kept in mind.

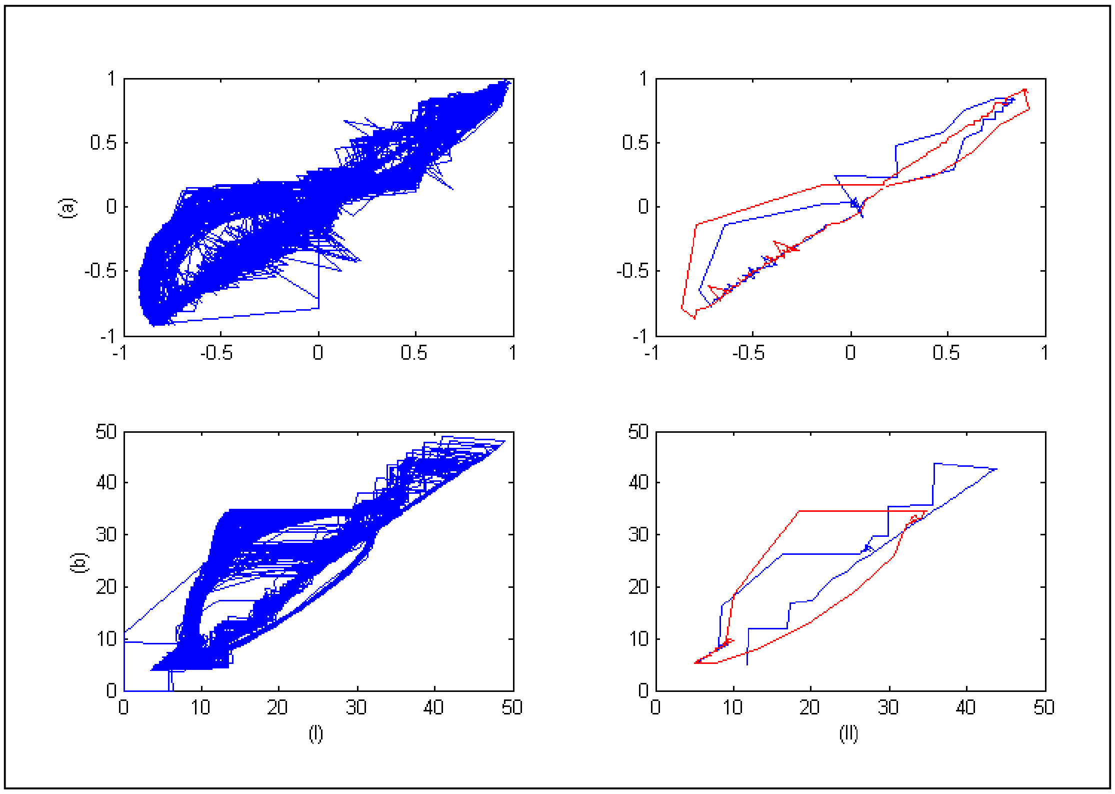

Figure 3-5 Returns maps of the data for $\tau = 1$: (a) Flow; (b) Pressure; (I) Original data; (II) Test Set. For the test set, spontaneous breath is in red, mandatory in blue.

Several trial and error tests have allowed us to determine a good embedding dimension K=10 for the nonlinear models (see 3.4.1), allowing both acceptable prediction result (enough depth at the input) and low enough input dimensionality (to allow for acceptable computational constraints: every added dimension means another input weight vector). The exact dimension required by the embedding theorem is not computed, because it usually overestimates the minimal order of the model, and it is dependent on the sampling rate [21].

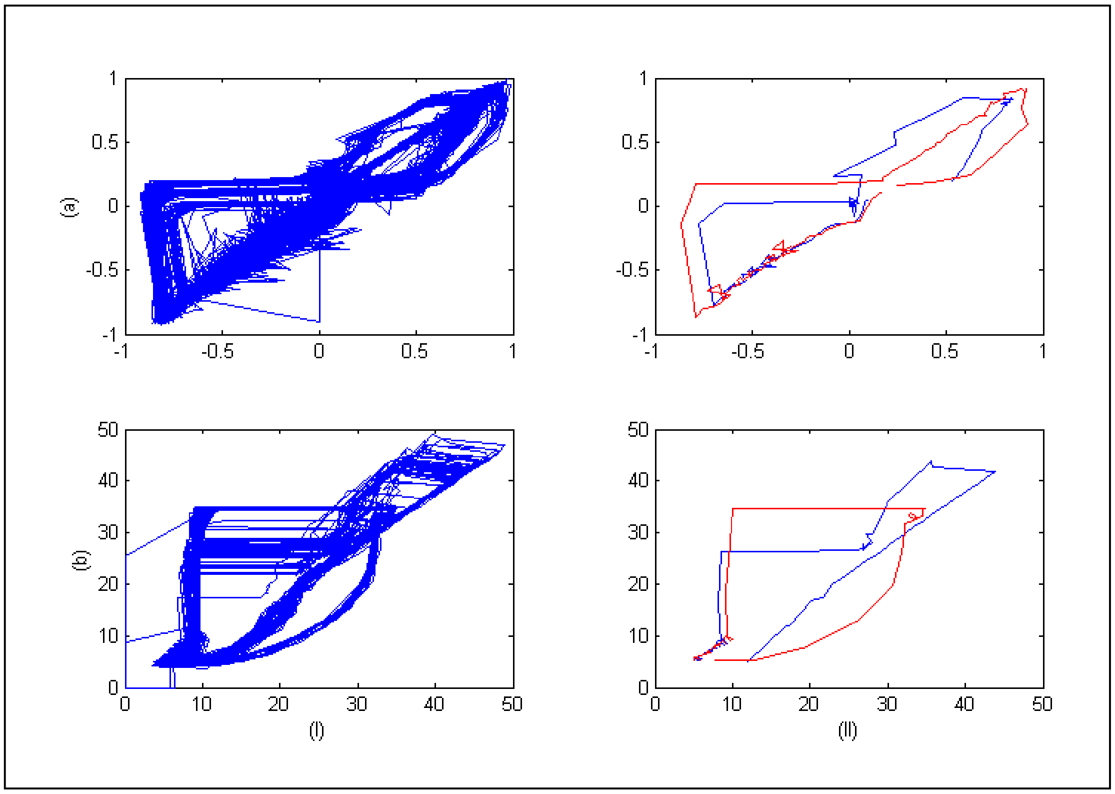

Figure 3-6 Returns maps of the data for $\tau = 2$: (a) Flow; (b) Pressure; (I) Original data; (II) Test Set. For the test set, spontaneous breath is in red, mandatory in blue.

As can be seen in Figure 3-5, the shapes of the trajectories for the chosen samples replicate quite well the shapes of the whole data set (with slight shifts), and one can immediately see the difference between the two breathing patterns, but also the similarities: for the flow, signal on which is the segmentation is performed, the first and third quarter of the map are very close, and the fourth quarter is a complete overlap. This means that on the last part at least, the experts might not be able to discriminate between the two modes, because they are not separable: it is the part where the exhaling takes place, and since it is a passive mode for the patient and the ventilator, it follows the same model for the two types of breath.

It is also important to notice in Figure 3-3 that the data is a little noisy, and that noise induces variations around the return map that can cause the instantaneous error of a predictor to spike (assuming the predictor did not model the noise) and might create segmentation problems.

3.4 Data Models

Dynamic modeling implies a two-step process: transforming the observed data into a path in the state space (i.e., determine an embedding), and then from this path build the predictive model [22]. Now that the embedding is chosen, we consider several design options for the experts: the traditional linear approach and two nonlinear ones using global dynamic models that are universal mappers, one with global basis (MLP) the other with local basis (RBFN).

3.4.1 Degrees of Freedom

Whereas all structural parameters are now set for the linear model, there is still another parameter to consider for the nonlinear models. As explained in the next sections, both models have a hidden layer of processing elements (PE), and the number of PEs determines the number of degrees of freedom of the experts. Minimizing the network size is a important issue in signal modeling, because a network of smaller size has less risks of learning the noise in the data (and thus generalizes better), and also because a smaller size means fewer weights and a smaller computational load. The optimal network size can be determined by using “network growing or pruning” methods, but it is also dependent on the embedding dimension.

A second data set recorded from a patient under assisted ventilation is presented in Figure 3-7. This data set shares the waveform shapes and other particularities with the set used for the simulations, so the expert’s structure should be similar for both sets, and it is interesting to consider designing some of the expert’s structure from this second set, which was downsampled to 50 Hz too. The segmentation goal on that set was to detect changes in amplitude of the signal instead of changes in type of breath.

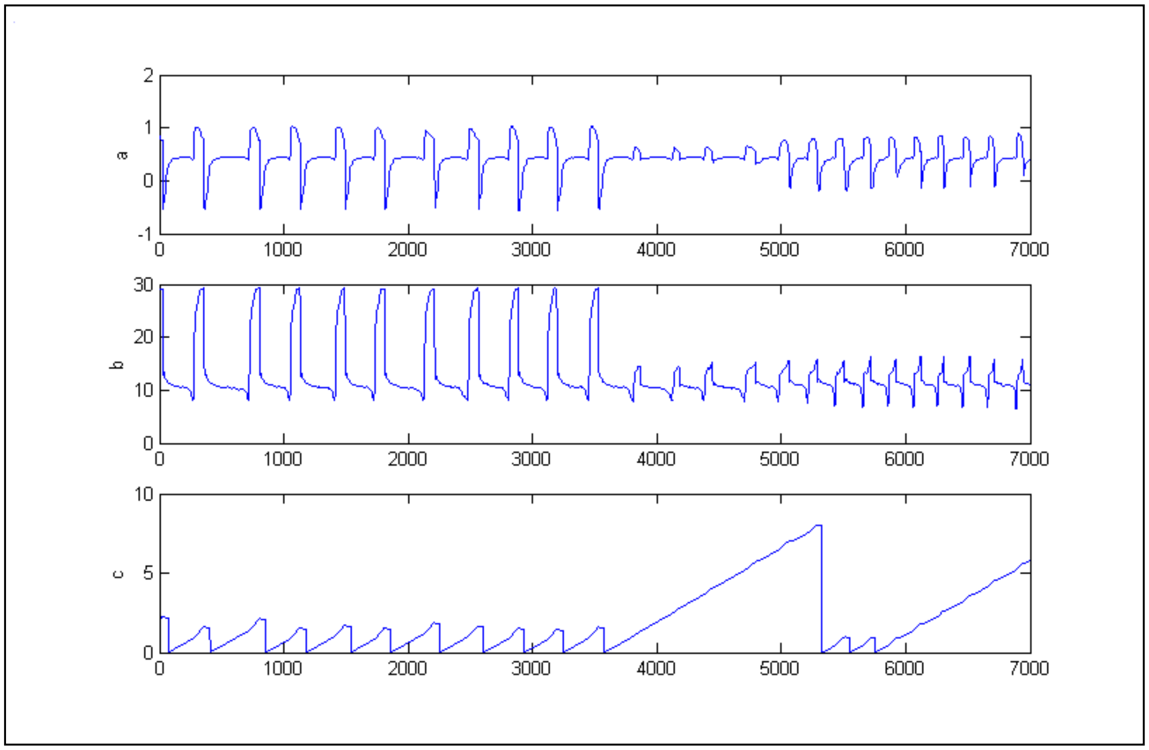

Figure3-7 Second data set: (a) Flow; (b) Pressure; (c) Volume

A trial and error method is used in this case to determine a good choice for both the embedding dimension and the network size. The non-linear models (RBFN) considered were chosen to have an input delay line of length thirty, and more than a hundred processing elements in the hidden layer. Such systems predicted the data accurately (low prediction error), but no specialization could be made for the experts on the amplitude of the data. Also the size of the hidden layer was too high to achieve online training of the output layer, and its size was reduced to thirty processing elements with an input embedding dimension of ten without much change in the performance. Segmentation could not be achieved on the amplitude feature to detect a change in the breathing capacity of the patient.

These parameters are kept for the modeling of the data set presented previously (since the waveforms of the data are similar), where the amplitude of the flow is quite constant but there are two different kinds of breaths. It can be noted that in the second data set, the flow is not centered around zero but around a positive offset (likely caused by a measurement error), this might be a reason which the non-linear models did not perform well, and outlines the high dependency of modeling and segmentation methods to data preprocessing.

3.4.2 Linear Experts

In most system identification cases, the data is non-linear, but can be fitted to a linear model within an acceptable error range. As noted before though, the presence of both sharp changes and flat parts in the data will make it harder for a linear model to perform a good prediction, because the first would require a high filter order whereas the latter would on the contrary require a low order. Therefore non-linear models might be a better option.

To train the experts, we use a Wiener filter [23]. With this approach, the final weights of the model are computed in one step. The correlation matrix of the input vector is computed using a sample breath sequence (mandatory or spontaneous). The Wiener optimal filter coefficients W_{opt} are computed by simply inverting the autocorrelation matrix, as shown in (3.4).

$$W_{opt} = R^{-1}P \tag{3.4}$$

Then the least mean square algorithm (LMS) [23] is used for the online update of the experts. Usually the recursive least squares (RLS) algorithm is faster and more robust than the LMS, but in our framework of online segmentation, using the LMS gives a computationally efficient way to update the weights of the experts, and the LMS is known for its efficient tracking properties, which will be useful for the real data is known to be nonstationary (property that the RLS does not handle well). For the choice of the learning rate, the inequality (3.5) and the need of a sufficiently small learning rate to track slow changes in one regime yielded η_{LMS}=0.0007.

$$0 < \eta_{LMS} < \frac{2}{tap - inputpower} \tag{3.5}$$

3.4.3 Radial Basis Function Network Experts

We then turn to nonlinear neural networks for experts. An interesting form of feedforward neural networks is a Radial Basis Function Network (RBFN) [23, 24].

A RBF network is a feedforward neural network with a single hidden layer of N nonlinear processing units φ_i input weights t_i and an output layer of linear units and weights w_i. The hidden layer computes the distance between the input vector and the center of the RBF, defined by the weight vector t_i on that processing unit. The input vectors are processed through the non linear radial basis function, chosen to be a Gaussian defined in (3.6), and through the linear layer as shown in (3.7).

$$output = \sum_{i=1}^{N} w_i \varphi_i(input, t_i) + w_0 \tag{3.6}$$

$$\varphi(input, t_i) = \exp\left(-\frac{\|input - t_i\|^2}{2\sigma_i^2}\right) \tag{3.7}$$

The distance measure between vectors is the Euclidian distance (inner product), and is the spread of the RBF.

A RBF network is composed of two kinds of layers, and they are usually trained separately. As one may view the training of a multilayer perceptron as a “curve fitting” process, the training of an RBF can be linked to an “interpolation” process. Indeed, the RBF centers cover the span of the input space, but are located in the highest data density zones in this input space. The process of optimizing the location of the RBF centers is the first in training the network. To that goal, several approaches can be taken:

1. The RBF centers can be positioned at equal distance from each other in a gridlike fashion. With a 10 delays input line, if we choose to divide the spread of the data into 3 points, there will be 3^{10} =59049 processing elements in the hidden layer, which would render the training computationally explosive if not impossible.

2. Another idea to fix the RBF centers is to use the some input vectors randomly taken in the data set, since that would distribute the centers according to the probability density function of the training data. Such an approach still does not produce very good prediction performance, due to the non-stationarity of the data. Therefore the centers of the RBF need to be clustered adaptively.

3. The first approach is to use the k-means clustering algorithm [25]. For each input vector, it finds the closest RBF center according to the Euclidian norm, and updates the location of that winning center proportionally to the distance from the input point to that center. The learning rate is annealed during the training. This is performed online (update center after each input), but can be performed offline (run one epoch and do the updates in batch). Also the spread of each processing element is chosen to be half the mean of the distance to its ten closest neighbors (in order to have a good coverage of the space between neighbors without too much overlap). A limitation of the k-means algorithm is that it can get stuck in local optima, dependent on the initial center locations.
4. The second clustering algorithm is the orthogonal least squares (OLS) algorithm [26]. For every presentation of the whole set of training vectors, it defines a new RBF center by optimizing the variance of the explained data. It actually uses the training vector space and defines an orthogonal basis using a method analog to the Gram-Schmidt orthogonalization algorithm. Contrary to the k-means algorithm, OLS provides an optimum clustering of the training data, but it does so by favoring the directions of maximum variance, so if the chosen hidden layer dimension is not high enough, and or if the two regimes are too much alike in those directions (if the differences are in directions of smaller variance), this method might lack specialization.

Once the RBF layer is clustered, the linear layer has to be trained: we use the RLS algorithm [23] for initial training, and the LMS algorithm for online adaptation. Again the step size and other parameters were set after looking at the power at the output of the first layer, and by trial and error method. Only the second layer is updated in the online adaptation phase, because the return maps should not change enough for the clustered RBF not to cover them.

3.4.4 Multilayer Perceptron Network Experts

Another type of expert is a multilayer perceptron (MLP) [27]. The ones used here have one hidden layer of sigmoid processing elements. The network size is chosen to be the same as the RBF network's in order to compare performance, but although both types of networks are universal approximators, the power of representation for the two are intrinsically different and so the same number of hidden processing elements might not mean the same accuracy in representing all the data features: the RBF functions construct localized approximations, whereas the MLPs conduct global approximations and therefore might need fewer PEs to achieve the same degree of accuracy as the RBFN [24]. We choose to have the same number of degrees of freedom for both networks; we realize however that this gives an edge to the MLPs in terms of modeling power.

The MLPs are trained with the Levenberg-Marquadt algorithm (generally for 5 epochs or less to achieve a prediction error comparable to that of the RBFNs). In order to have a fair comparison, only the second layer of the MLP is adapted during the online adaptation phase, with the gradient descent.

3.4.5 Concerns

We know the return maps of the data are close, and simulation with all methods perform poorly in specializing the experts. Since we want to achieve online segmentation and we have some information about the data (we can recognize a spontaneous from a mandatory breath visually), we can include this information in our system by positioning the experts in interesting initial values: each MLP expert is trained for a few epochs its corresponding breath sample, or each RBF expert is clustered on its corresponding breath sample. Expert 1 is decided to be the spontaneous breath expert; expert 2 is the mandatory breath expert.

CHAPTER 4
SIMULATIONS WITH SYNTHETIC DATA SET

This chapter presents the simulation results of the monitoring approaches with the several types of experts, using the synthetic data set generated in the previous chapter

4.1 Linear Experts

4.1.1 Approach I: Offline Training

Figure 4-1 shows the result of approach I simulation for a linear filter of order 10: each expert is a Wiener filter modeling one regime, and online change detection is performed.

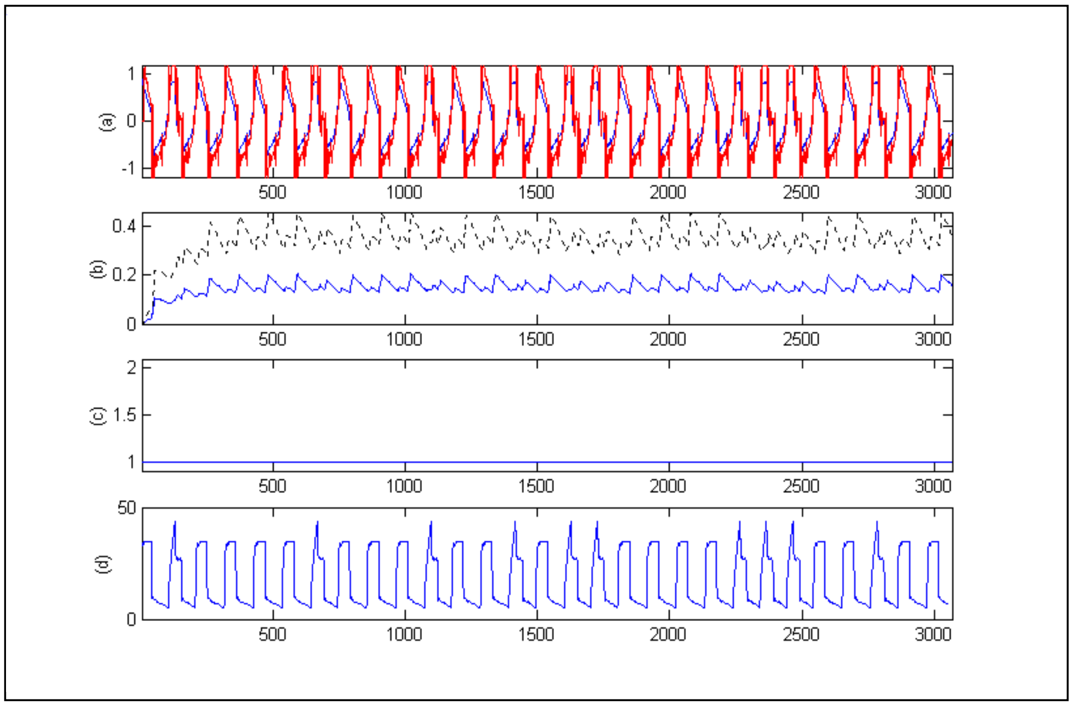

Figure 4-1Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure.

It is obvious on the flow graph (cropped: the actual values of the predicted flow peak out of the window!) that the data is not modeled well, as can be seen on also by considering the high values of the error estimate. Besides, segmentation is not performed, only expert 1 wins. An order of ten is clearly not enough to model with such complex shapes with a range of different frequencies, but linear models with a high order act as smooth predictors averaging over the whole input window, and are not able to present the fast regime changes that have to be detected. Besides, each of the training sequences only has 110 samples, so the "rule of thumb" demanding at least ten times more data points than coefficients to estimate is therefore not respected for orders of eleven or more.

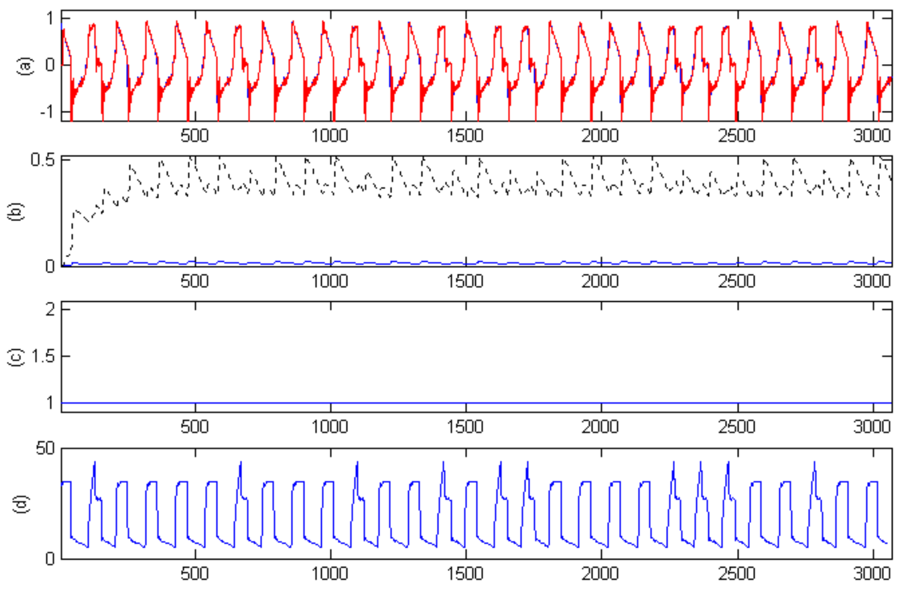

Figure 4-2 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure.

4.1.2 Other Approaches: Online Adaptation

Figure 4-2 shows the results of simulation of approach II, where the experts are trained on their respective regimes, and then the winner is adapted online with the LMS at a learning rate η=0.0007 verifying (3.5).

Figure 4-3 shows results for approach IV, where both experts are adapted with a learning rate dependent on the output of gate at each time step.

In both cases, only expert 1 wins and keeps winning, even with approach IV when the two networks are adapted: this proves further that a linear model is not a good expert in this case: there is a conflict between the high dimension of the reconstruction space needed to be able to reproduce the data accurately in order to segment on shape discrimination, and the necessity for a short input size to achieve sharp detection of the changes.

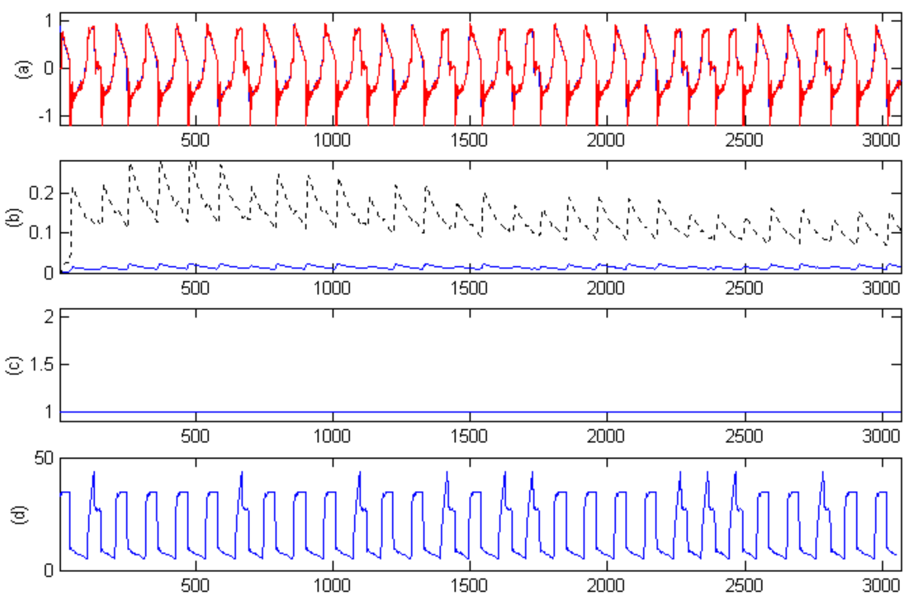

Figure 4-3 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure.

4.2 Radial Basis Function Network Experts

4.2.1 Offline Training: Approach I

K-means clustering. Figure 4-4- shows the simulation for a RBFN using k-means clustering for the RBF layer, the RLS algorithm for the output layer training, and online change detection is performed.

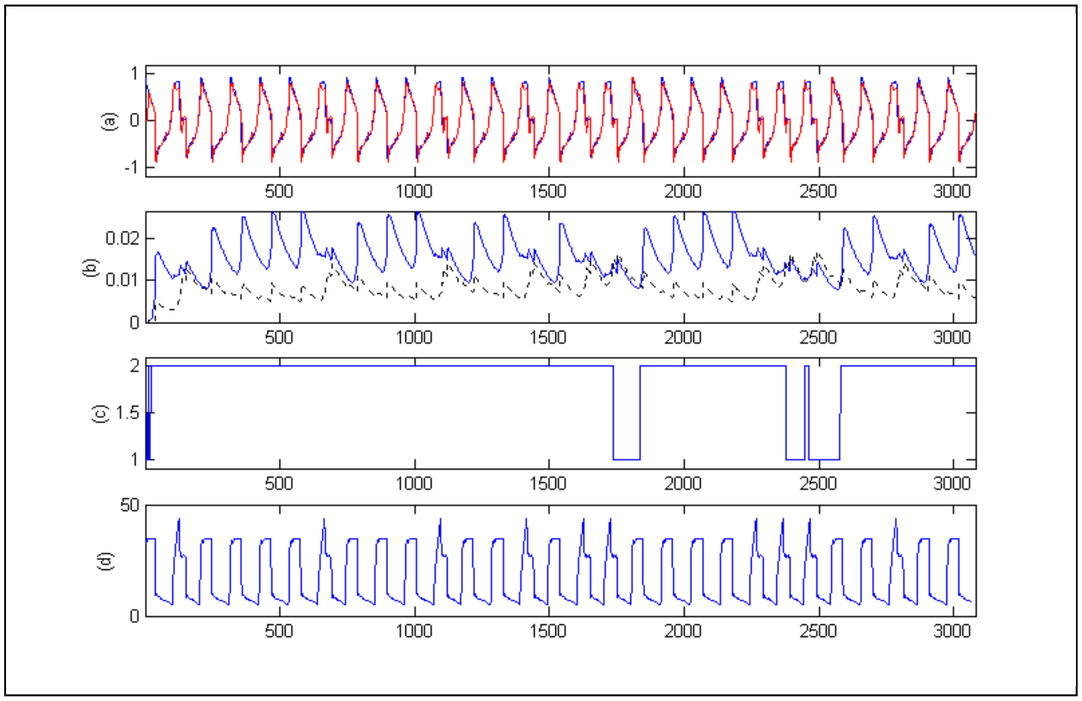

Figure 4-4 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure.

An important comment to be made is that the value of the error criterion is ten times lower than for the linear model, and it can be seen on the flow graph that the predicted series is a lot smoother which means that the experts are modeling the data more accurately, but the segmentation is not performed well because of the “build-up” of the criterion on expert 1: it only decreases below the criterion on expert2 when there are two mandatory breaths in a row, so single mandatory breaths are not detected. There are two ways this could be improved: decrease the memory depth of the criterion, but then we would run the risk of having unwarranted switches (as can be seen on Figure 4-5 showing the results for γ_{eps} =0.02, or a memory depth of 50 samples, where the system detects a switch back to regime 2 at the end of inspirations in regime 1, before switching immediately back to regime 1), or even no switches at all (in Figure 4-6, with γ_{eps}=0.03 or a memory depth of 33 samples, the mandatory breaths are barely detected, because the criterion values are smoothed less because of lesser samples for the average).

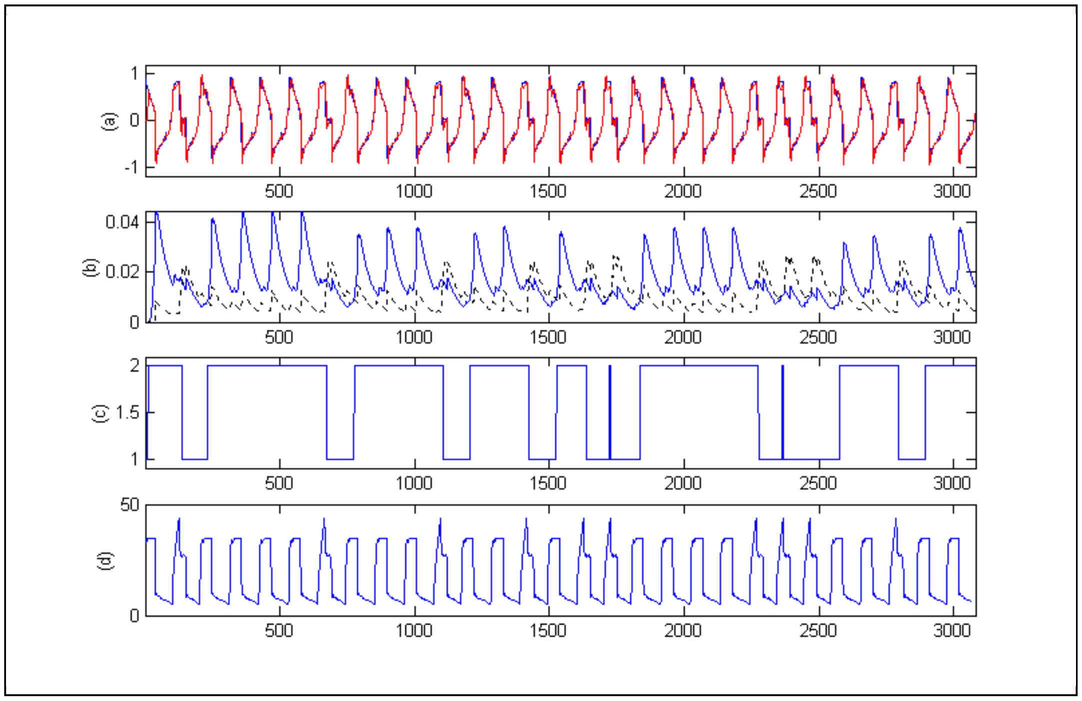

Figure 4-5 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure.

Also with only a memory depth of 33 samples, the two experts compete in the expiration phase without memory of what kind of inspiration has just taken place, and expert 2 seems to model that phase better.

These examples show the effect of varying the memory depth of the criterion, but from now on, the same value of γ_{eps}=0.01 will be used for all simulations, for it ensures no unwarranted switches in all cases.

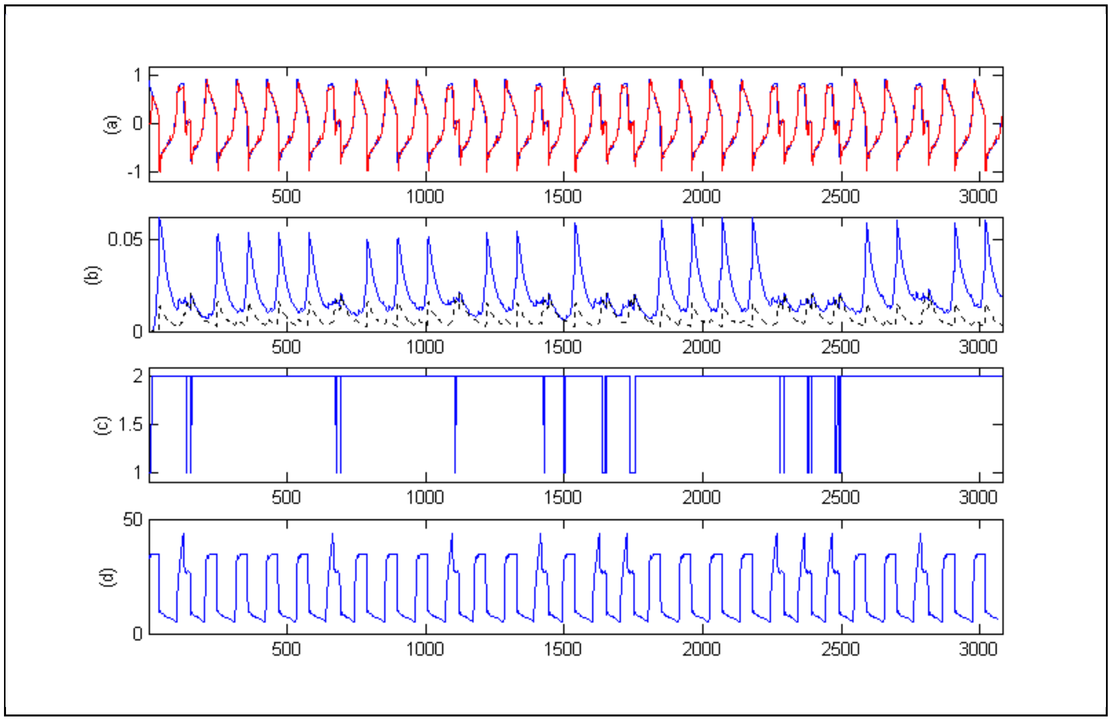

Figure 4-6 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure.

OLS clustering. Figure 4-7- shows the simulation results for a RBFN, this time using OLS clustering for the RBF layer. The values of the error criterion are in the same range as the ones for the k-means clustered experts, but the segmentation performed a lot better: all mandatory breaths are detected. This is due to the fact that the criterion value for expert 2 increases more when there is a change in regime than for the other clustering method, therefore compensating faster for the “buildup” (the decrease of the other criterion value has essentially the same exponential rate for all models, determined by the memory depth).

With such encouraging results on the segmentation without adaptation, we now consider the results for other segmentation approaches with online adaptation of the experts.

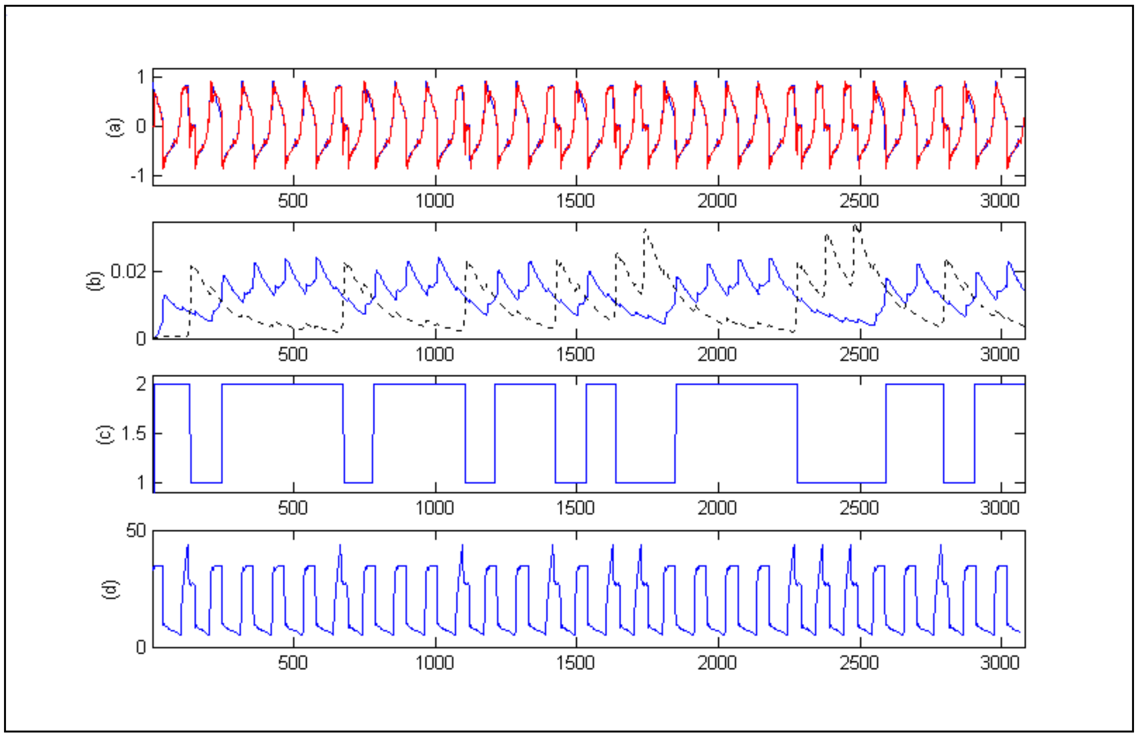

Figure 4-7 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure.

4.2.2 Other Approaches: Online Adaptation

For the two types of RBFN, the linear layer is updated with the gradient descent, at the same learning rate η=0.01 determined experimentally (except for approach IV, where η=0.002, because each expert is adapted even it is not the winner).

K-means clustering. For approach II, Figure 4-8 presents the results for k-means clustering. Obviously adapting the second layer weights helps in detecting the previously undetected mandatory breaths, but the “buildup” still appears, allowing the detection only within the last few samples, but the memory depth could not be shorter, since there is already an unwarranted switch on the set of three mandatory breaths (the first switch is only detected accurately because it occurs within 100 data samples and the criterion, initialized at zero, has reached its full memory length yet, sot it has not build up yet).

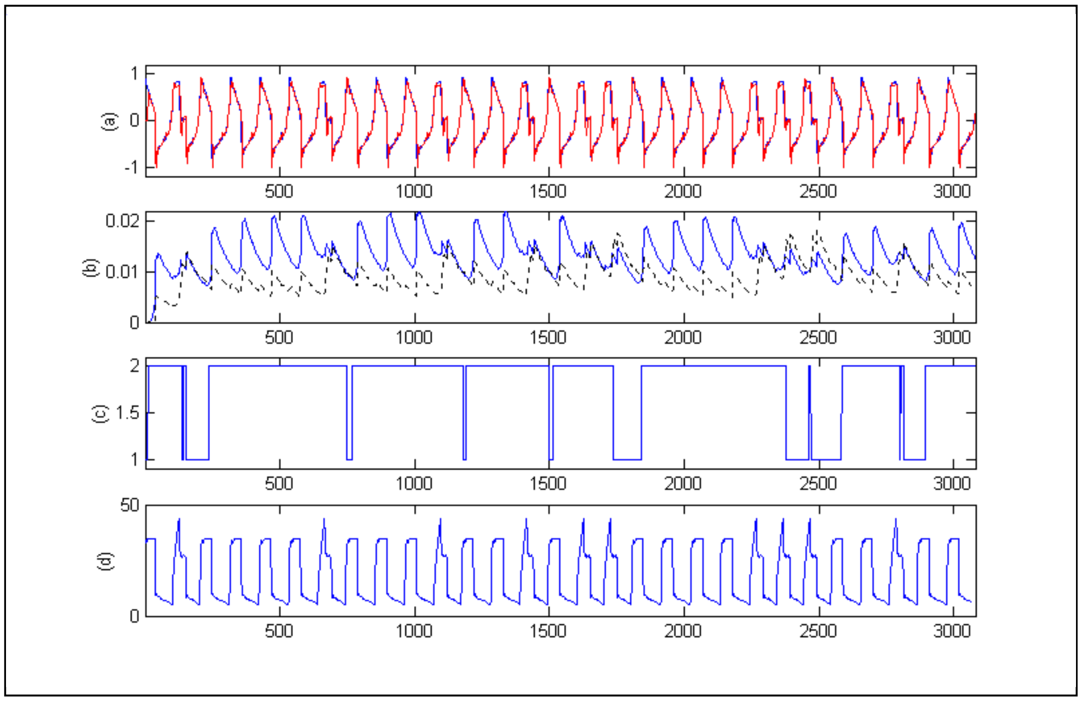

Figure 4-8 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure.

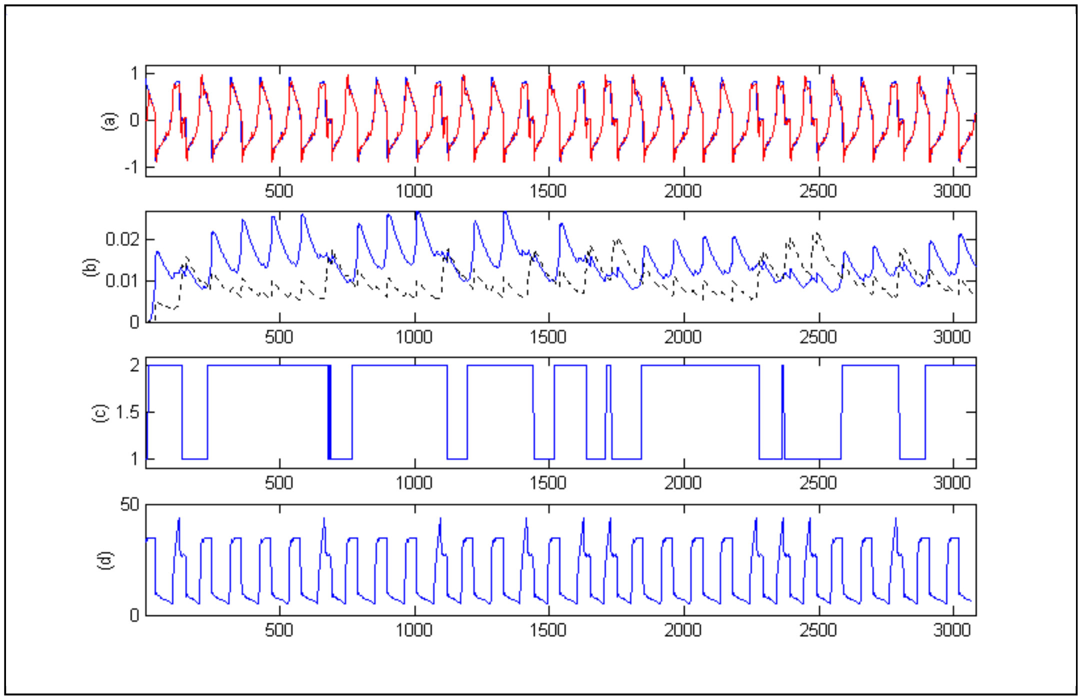

Figure 4-9 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure.

Figure 4-9 shows the results for approach III, and the effect of the spike detection of the winner is important: all mandatory breaths are now detected accurately, but there are still two unwarranted switches, due to the fact that expert 2 seems to attain better prediction of data (the amplitude of the criterion for expert 2 decreases slightly).

Figure 4-10 presents approach IV, which does not work well for that model: the two experts are adapted proportionally to the gate output and the winner gets most of the update, but expert 1 is still updated for several consecutive spontaneous breaths, which could explain why the criterion does not decrease as fast as expected when a switch occurs (the first one as previously explained is detected only because it is at the very beginning of the series). Also, expert 2 is updated during that first mandatory breath, so it starts to explain regime 1 (we can notice that the spikes in criterion for expert 2 are not as high for a switch to regime 1 as for the other approaches).

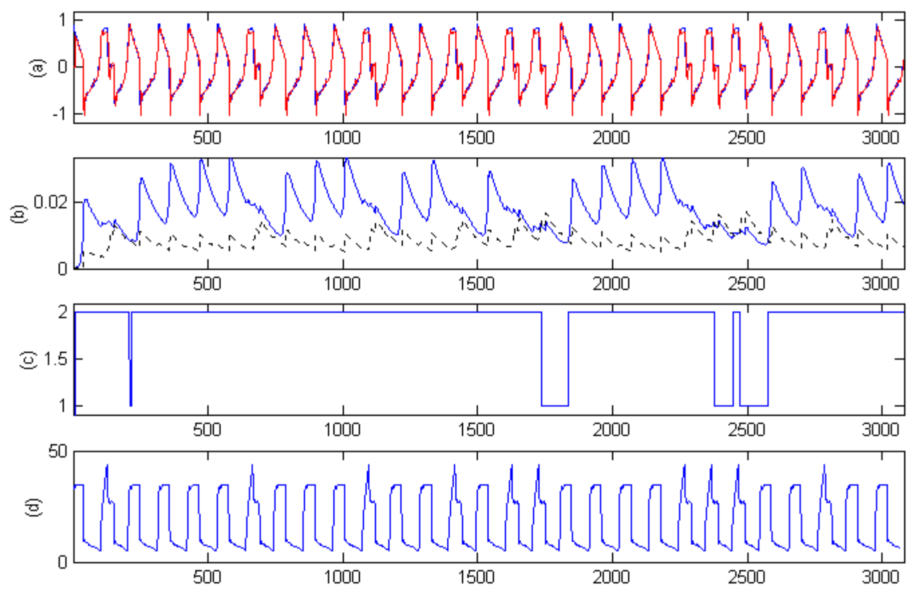

Figure 4-10 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure is for.

These results may indicate that more competition is needed. Another solution for segmentation is to look at the output history of the gate in Figure 4-11: the maximum value is for the winner, but the waveform gives good segmentation information.

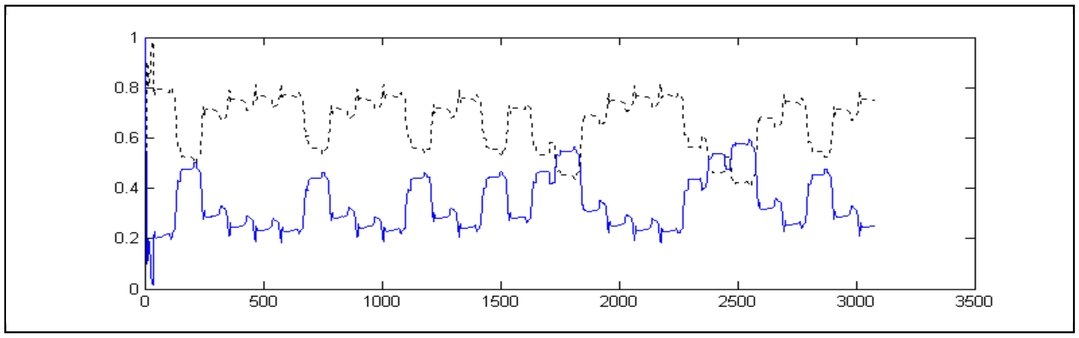

Figure 4-11 History of the gate: full line is for expert 1, dotted line is for expert 2

OLS clustering. Figure 4-12 shows the results of approach II for OLS clustering. The prediction error is in the exact same range as with k-means clustering, but the segmentation on that approach is performed far better: all mandatory breaths are detected well. Actually, this model exhibits a behavior opposite that of the previous model: the mandatory breaths are well detected, but the detection of return to regime 2 takes longer (10 to 90 samples more): this is a first hint that the switching mechanism is not symmetric, an issue we will further develop in chapter 5.

Also, there is a fundamental difference between those two clustering algorithms: the k-means algorithm tries to find similarities by assigning a data point to the closest cluster, whereas the OLS algorithm tries to finds each added center with respect to how much more than the others it explains, or dissimilar from the others it is.

One can then expect that with OLS clustering the particularities of each regime is better identified and modeled even if they are represented by a few points only, when those few points might not have as much weight with k-means and be “blended” to a behavior occurring statistically more often.

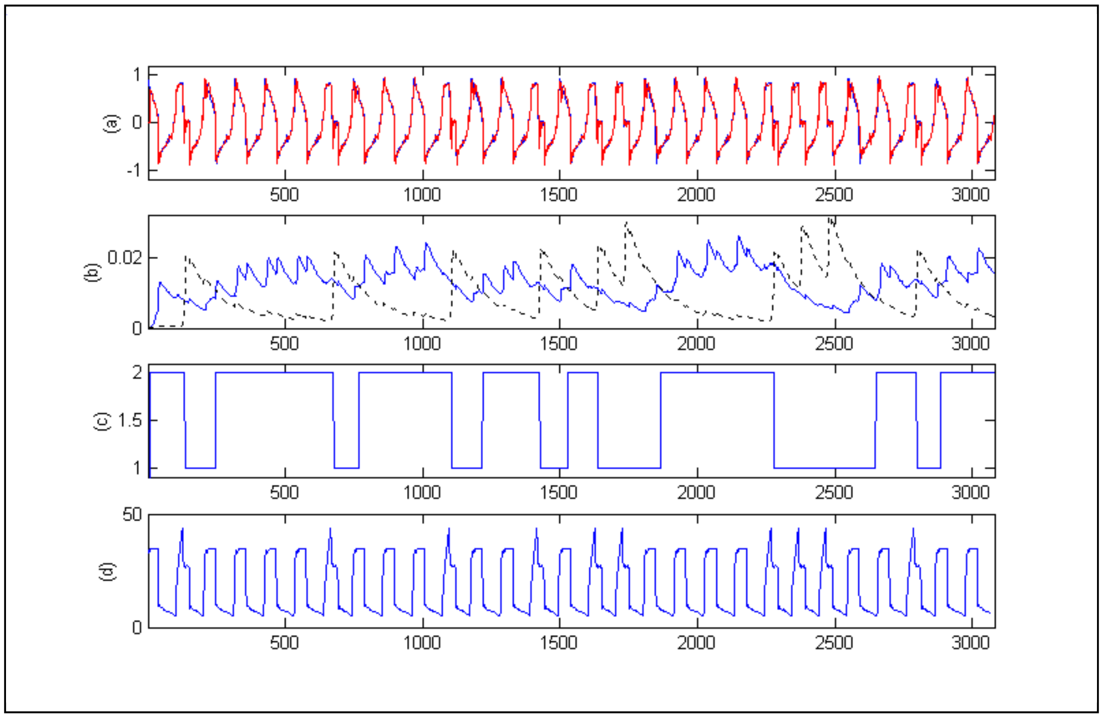

Figure 4-12 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure.

Figure 4-13 shows results for approach III, but since all mandatory breaths were detected accurately in approach II, the spike detection does not change much the history of segmentation, and the waveforms of the criterion are a lot similar to those in approach II. Also the spike detection mechanism does not help for a sooner detection of the switch back to regime 2, because expert 1 does not show such big spikes when a change occurs as expert 2.

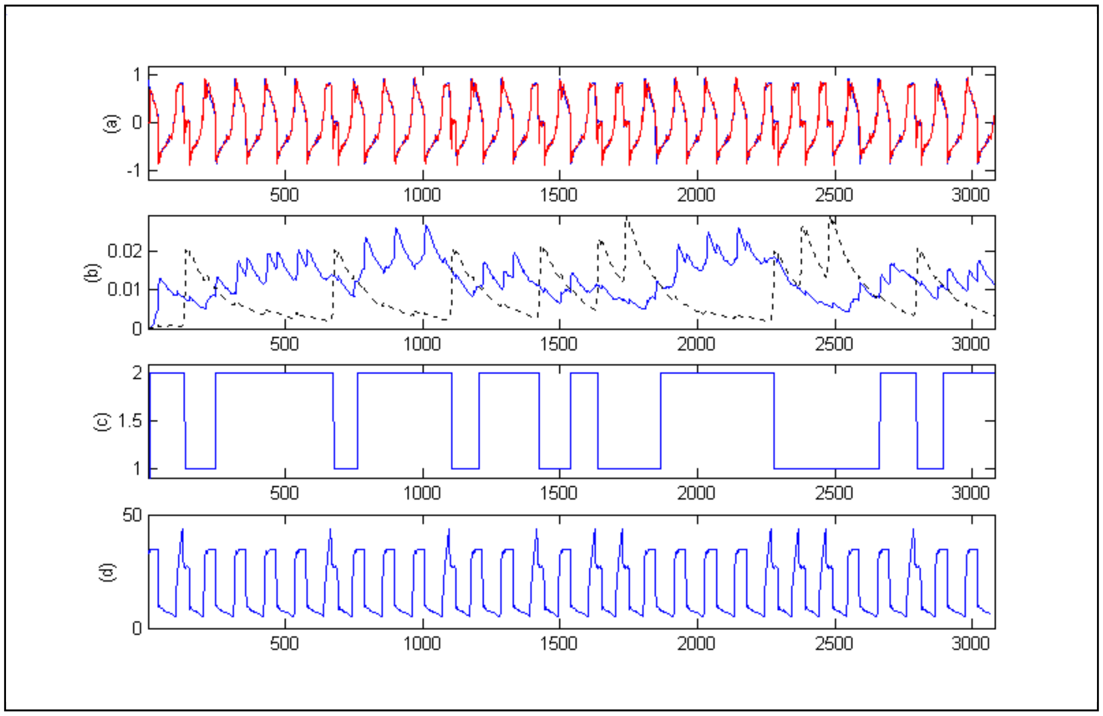

Figure 4-13 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure.

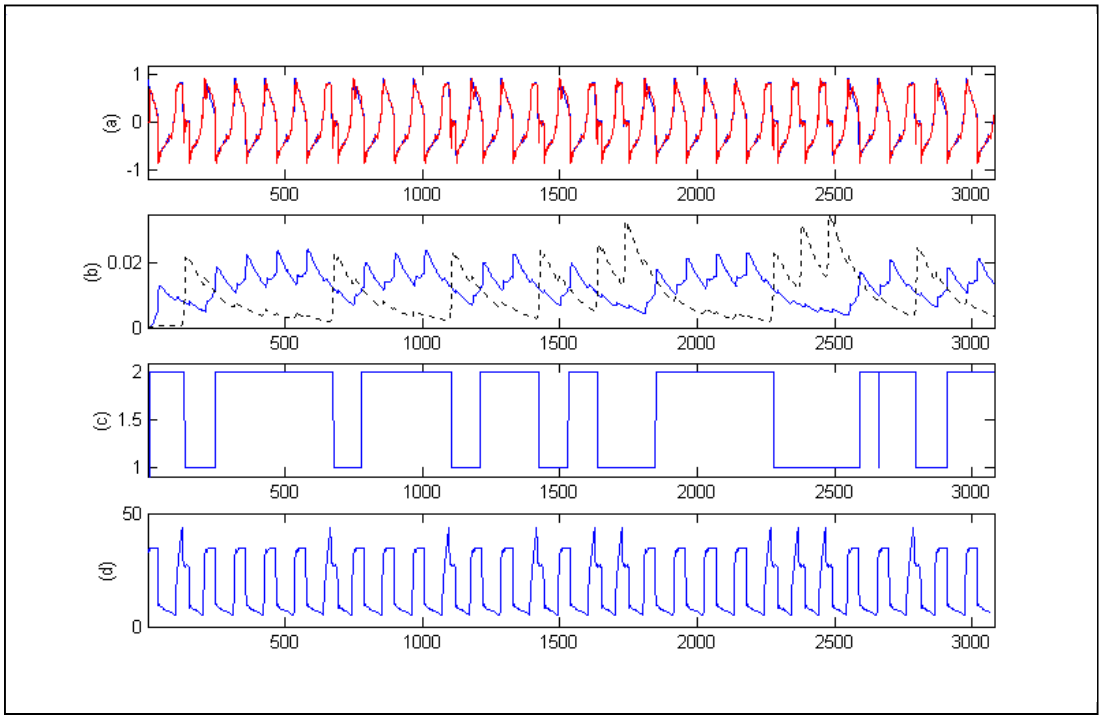

Figure 4-14 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure.

Figure 4-14 shows the results for approach IV, which detects well the mandatory breaths, but still has a delay in detecting the switch back to spontaneous breaths, especially at the end of the series after three mandatory ones.

This delay is explained by the criterion "buildup" that prevents the change detection the same way as before, only on regime 1 this time, when the buildup of error on expert 1 is smaller than with the other approaches. It also explains the short unwarranted switch after the three mandatory breaths, and shows the complexity in balancing the segmentation parameters. Figure 4-15 shows the output of the gate, where the segmentation history can clearly be seen.

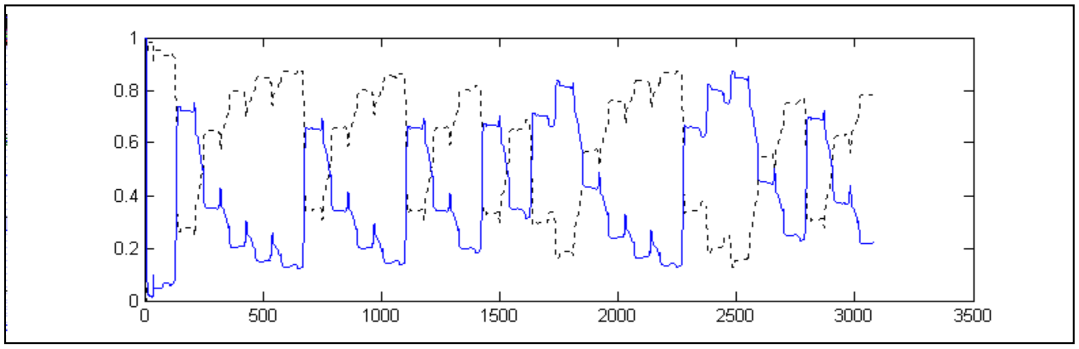

Figure 4-15 History of the gate: full line is for expert 1, dotted line is for expert 2

4.3 Multilayer Perceptron Network Experts

4.3.1 Approach I: Offline Training

Figure 4.16 shows the results for approach I with the MLP experts trained with the LM algorithm. An important remark is that the amplitude of the criterion is half of the criterion amplitude for the RBF networks. It can be explained by the fact that an MLP is a global basis network, and has more generalization power than a RBF for the same number of degrees of freedom. Still the criterion "buildup" problem is present, and delays the detection of the change back to regime 2 after the three mandatory breaths.

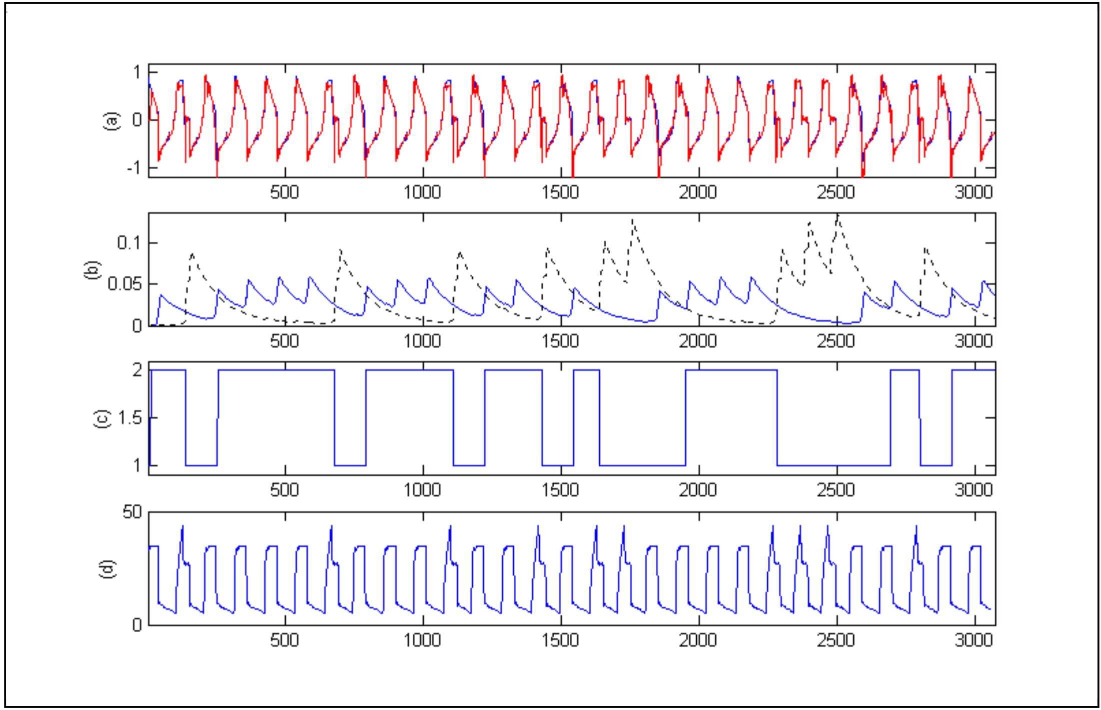

Figure 4-16 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure

4.3.2 Other Approaches: Online Adaptation

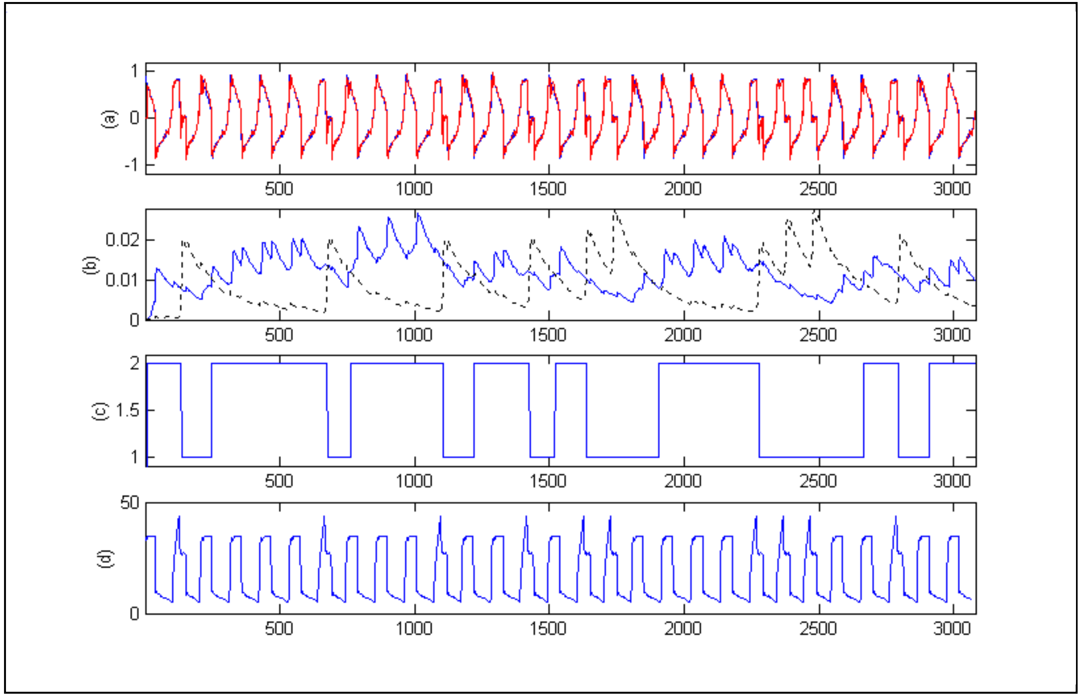

Figure 4-17 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure.

For all simulations, the only the output layer of the experts are adapted (with the gradient descent at the learning rate η=0.01) in order to keep some fairness in the comparison with the RBFs. Figure 4-17 presents the simulation for approach II with MLP experts, which detect all the mandatory but only follows the switch back to spontaneous breaths with a little delay, always because of the same "buildup" problem of the criterion.

Figure 4-18 shows the results for approach III, that should not present any difference from approach II since all switches were detected. It can be seen on the criterion graph (b) that the error for expert 1 is a lot lower than in Figure 4-17, enhancing the effect of the "buildup" to the point that single breath of regime 2 between breaths in regime 1 is not detected. The reason for the value of the criterion to change so much from one simulation to the other is due to the fact that the MLPs are only trained for a few epochs before starting the online segmentation and adaptation.

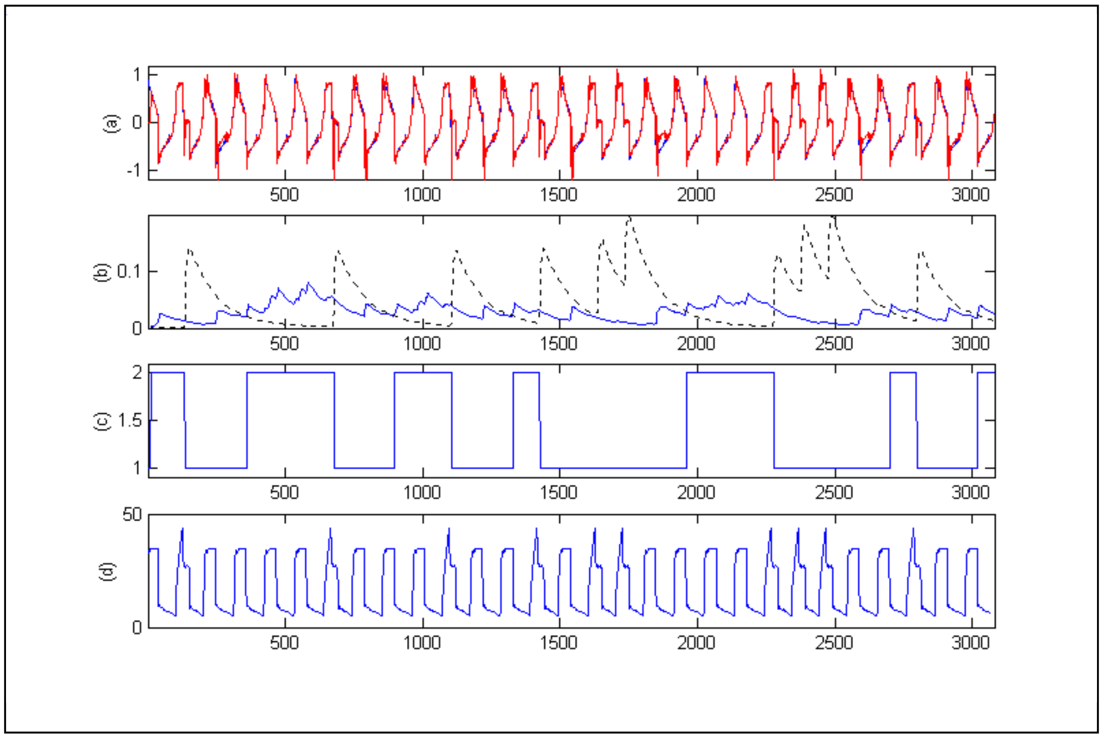

Figure 4-18 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure.

For approach IV, the learning rate was chosen to be equal to 0.02, as for the RBFs. Figure 4-19 presents the simulation results, and it is interesting to note how the criterion values tend to decrease: both experts actually adapt to perform better on each regime. The segmentation history is not perfect, for even if all switches are detected, there are also some unwarranted switches. Fine-tuning even more the system parameters could probably refine the performance in this approach.

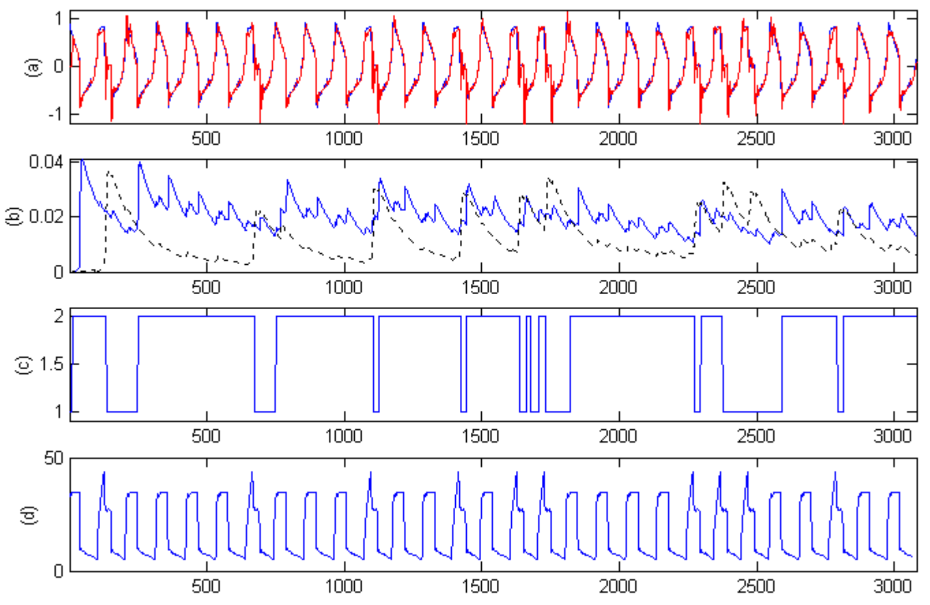

Figure 4-19 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure.

Having analyzed the results for the synthetic test set, it seems that the MLPs predict the data with more accuracy than the RBFs, but they do not necessarily provide better segmentation. The next chapter provides some insight into the performance of the system on the real data set, before further comparing all methods.

CHAPTER 5
SIMULATIONS WITH REAL DATA SET

In this chapter we present the results of segmentation on the original data set with the different models and some parameter tuning, and we analyze the system in the light of all these results.

5.1 Segmentation Results

From the results obtained previously, approach IV is the one that showed the most promise, providing that the gate implements harder competition, which is done by increasing the competition parameter to M=5, except for the MLP experts for which M=2 appears to be sufficient. Also, with the real non-stationary data, at first simulation showed that the memory depth might be too long, so for all simulations below, for γ_{eps}=0.02 is used and helps in shortening the delays before a switch is detected.

5.1.1 Linear Model

Figure 5-1 shows simulation with linear experts, which quite expectedly does not perform better than on the test set, and still has a high prediction error.

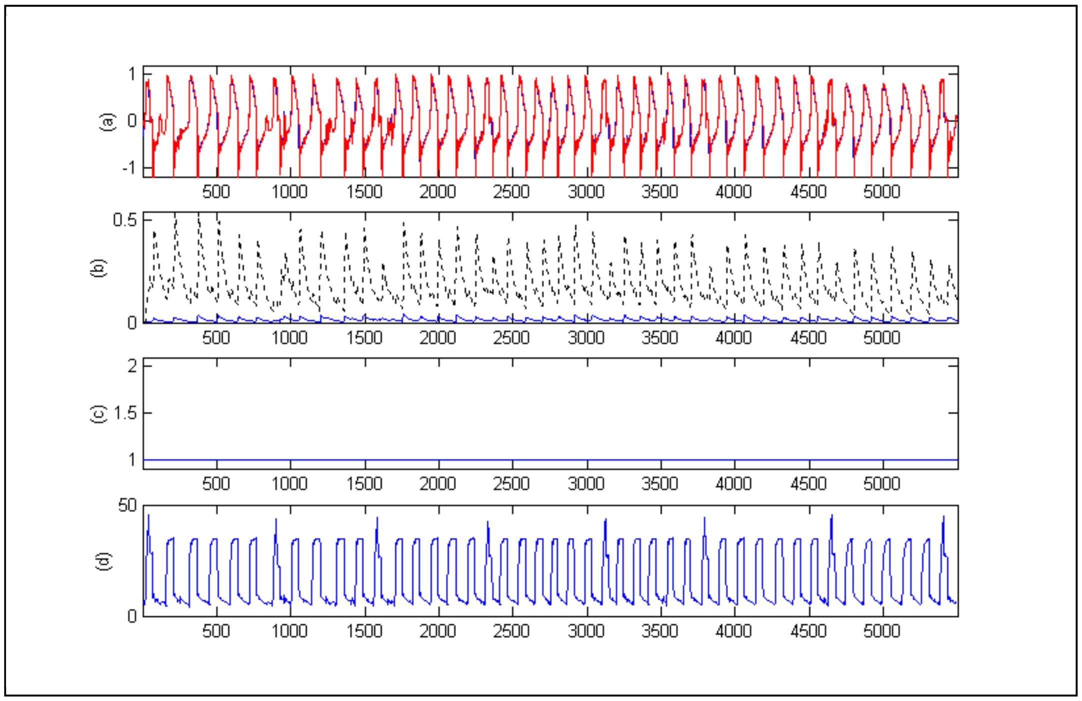

Figure 5-1 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure.

5.1.2 RBF Network Models

K-means clustering. Figure 5-2 shows results of simulation with RBFN experts that are clustered with the k-means algorithm. The segmentation detects without too much delay the switches from and to both regimes. Although there are some spurious switches at the start, the experts seem to be adapting well to their respective regimes as the amplitude and the spikes of the criterion seem to decrease, and this is verified by looking at the gate value in Figure 5-3: from "spiky", the gate gets smoother and nearly square, attesting that each winner gets updated more than before, and specializes even if the data is not stationary.

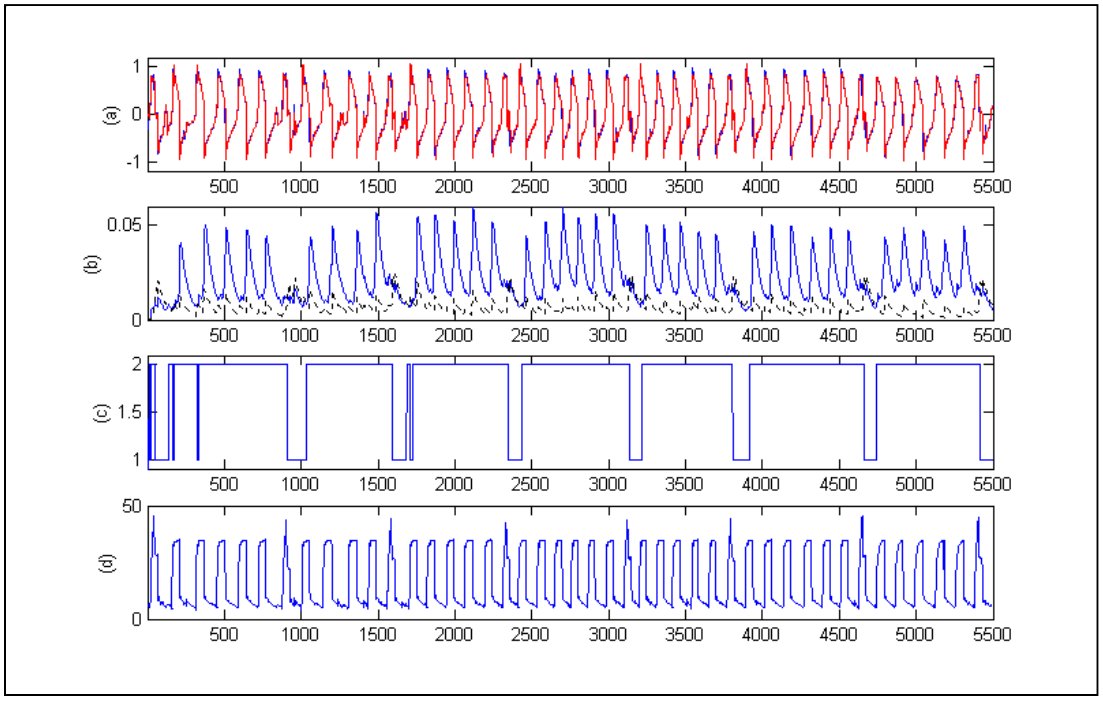

Figure 5-2 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure.

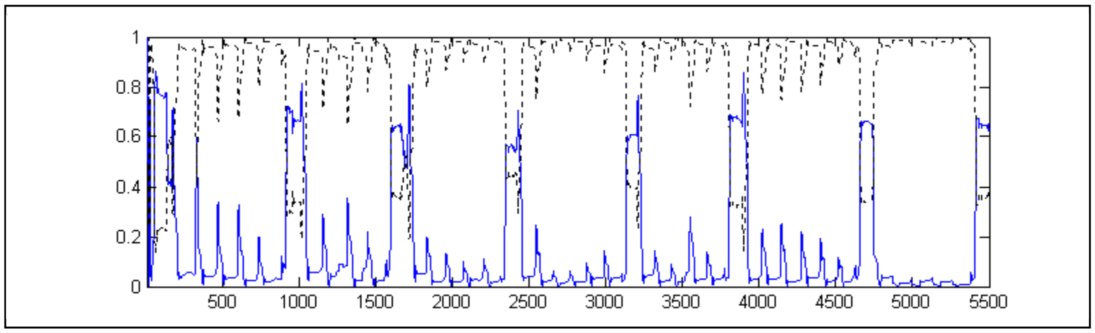

Figure5-3 History of the gate: full line is for expert 1, dotted line is for expert 2

OLS clustering. Figure 5-4 shows results of simulation with RBFN experts that are clustered with the OLS algorithm. The segmentation exhibits some spurious switching in the beginning, even more than the other RBFs, but the experts adapt to their respective regime like previously, and six switches (3 mandatory breaths) are detected without one unwarranted switch. Again by looking at the gate value in figure 5-5, one can verify that each expert has adapted to its regime.

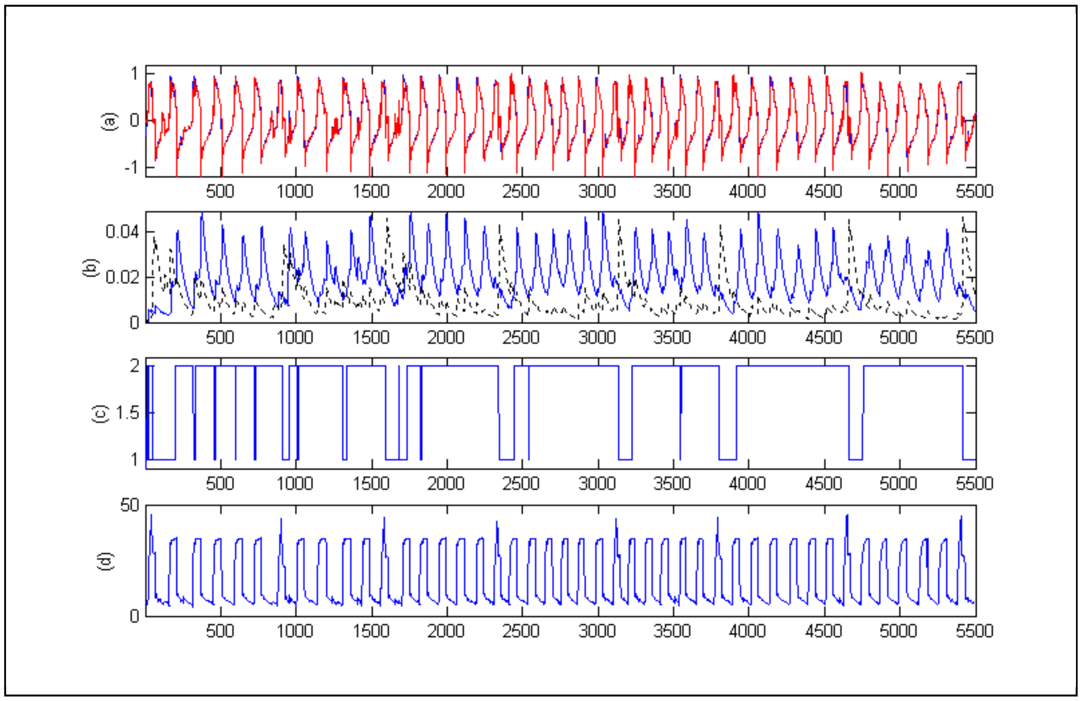

Figure 5-4 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure.

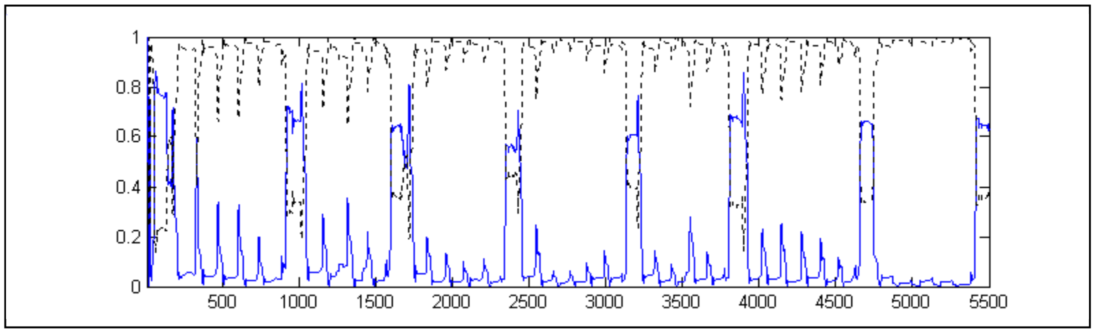
Figure 5-5 History of the gate: full line is for expert 1, dotted line is for expert 2

5.1.3 MLP Network Models

Finally in Figure 5-6 is presented the segmentation by the MLP experts. It is very obvious in this case too that each network adapts to its regime, although here a consequence of this adaptation is that the effect of the criterion “buildup” is increased, and the delay in detection of switch from regime 1 to regime 2 is delayed a little more every time.

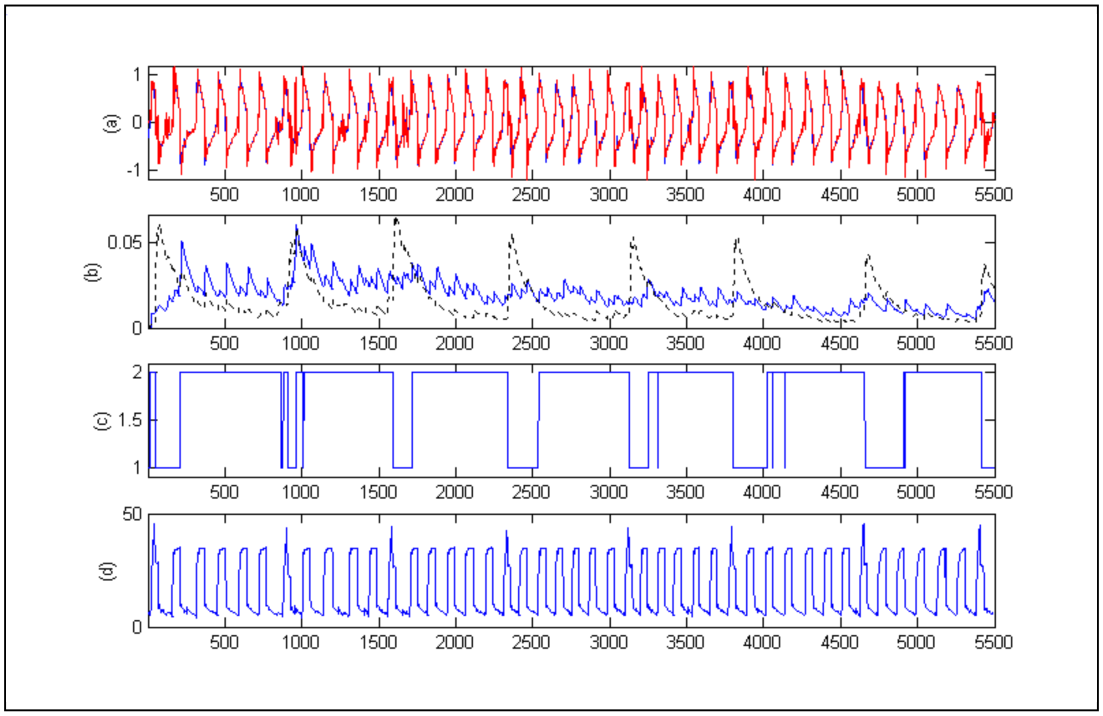

Figure 5-6 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure.

5.2 Results Comparison and Analysis

First an important note to be made is that the results of the simulations, in this chapter and the previous one, were somewhat erratic and not always consistent: for example sometimes one type of expert performed really well and other times it exhibited trends where one expert's criterion. Also, if the initial values of the networks are not "chosen" to be near the desired ones, oftentimes the networks do not converge. Still the results are encouraging for online segmentation from competitive experts: they have had only one presentation of the data set, whereas the framework called for several epochs, and segmentation is performed accurately for the three models.

5.2.1 Comparison of the Models

RBFNs and MLPs are both examples of feedforward neural networks, and the difference between them lies in the local versus global basis for approximation of the time series. This implies that MLPs need less coefficients than the RBFs to return the same accuracy, but it also means that it is a lot more difficult for an MLP to track change (one slight evolution in the time series could require that all the weights of the network be retrained, whereas only one RBF center would be moved). Besides, we only adapted the output layer online, which makes it even more difficult for the MLP to adapt. This could explain why the MLP does not achieve such better prediction (i.e., smaller criterion) than the RBF although they have the same number of weights, and the MLP would be expected to perform a lot better.

Interestingly, the three types of experts perform good segmentation, each with some particularities (each RBF seems to predict better one of the regimes for example): the importance in the choice of the type of model for segmentation is emphasized, and for this data the RBF network is the most satisfying one.

5.2.2 Practical Considerations

Non-linear dynamics modeling is still part art part science, because there is no universal technique that works for all data. The key in designing all these systems, indeed, is to find the best set of parameters for the application at stake: the number of experts, size of hidden layer of those experts and embedding size and lag, the memory depth of the criterion, but also the learning rates for the adaptation algorithms, the competition parameter of the gate and most of all the initial values of the networks. All those parameters have to be set optimally by hand, and external knowledge about the data is extremely useful in doing so. One important missing piece of knowledge is the actual switching history that would allow us to numerically assess the accuracy of segmentation in terms of delay to detect a change, and length of each detected regime. There is no clear-cut switch on the data between each breath, so the performance of the segmentation for this type of data can only be qualitative.

Several assumptions we made for this data could be reconsidered in order to further the results. For example, we decided that there are only two regimes, but using three experts can make sense: two types of inspiration, one of expiration. That solution would also provide a sure alternation in the online update between at least two experts, so the "buildup" would be more even.

We also assumed that both regimes could be modeled by same size networks, but the fact that expert 2 seems to always have a lower average prediction error, or that expert 1 does not present such big spikes when there is a change of regime clearly indicates some differences, and accounts for the different detection delays for a switch from 1 to 2 than fro 2 to 1. Trying to customize the size and parameter of the model to the different regimes if there is enough prior information about them can improve the segmentation performance, by increasing the robustness of the system with regards to the errors due to identification. Also, all regimes do not necessarily have the same memory depth to account for an expert's performance, or the same embedding dimension, so the system could be improved by using different values for these parameters, but this can only be done by knowing more about the regimes themselves.

Lastly we chose to use a recursive estimate of the MSE because it fits in the online framework (the MSE estimate is updated with every new instantaneous error value), and the last value of the instantaneous error is given the most weight, which would emphasize a change and ease its detection. Although the memory depth is fixed (all values beyond the memory depth have a negligible impact), the estimate uses an infinite window that creates the “build up” problem when the instantaneous error is too high. Another way to compute an estimate of the MSE is to use a boxcar average of the instantaneous error: this approach would give as much weight to all past instantaneous error values used to compute it (so the change detection might not be as fast), but within one window length all previous error values are forgotten so the buildup problem would be restricted to that length. Unfortunately, looking at the memory depth used for the models (one hundred samples on the test set, fifty samples on the real data), it appears that using a boxcar average of the instantaneous error in this case would solve the problem: most of the difference in waveform between the two regimes is represented by the samples gathered right after the end of the inspiratory phase, and lasts for less than forty samples, then the expiratory phase follows the same model in the two regimes. So in a two-experts system, changing the error estimate would not help, but with a three-experts system modeling the two different inspiratory phases and the expiratory phase, the memory depth could maybe be decreased, so would a window size for the boxcar estimate that could then become more interesting.

5.3 Simulations with Another Set

5.3.1 Description of the data

The data set previously used was a particular case of respiratory signal under assisted ventilation, but many different ones exist: indeed, the set waveform of the delivered flow can be different (square or sine for example), individual breathing patterns can be different, and also the patient might fight the ventilator.

Figure 5-7 shows another data set at 50 Hz with two different kinds of breaths, and we can see that the waveforms are not similar to the ones of the former data set.

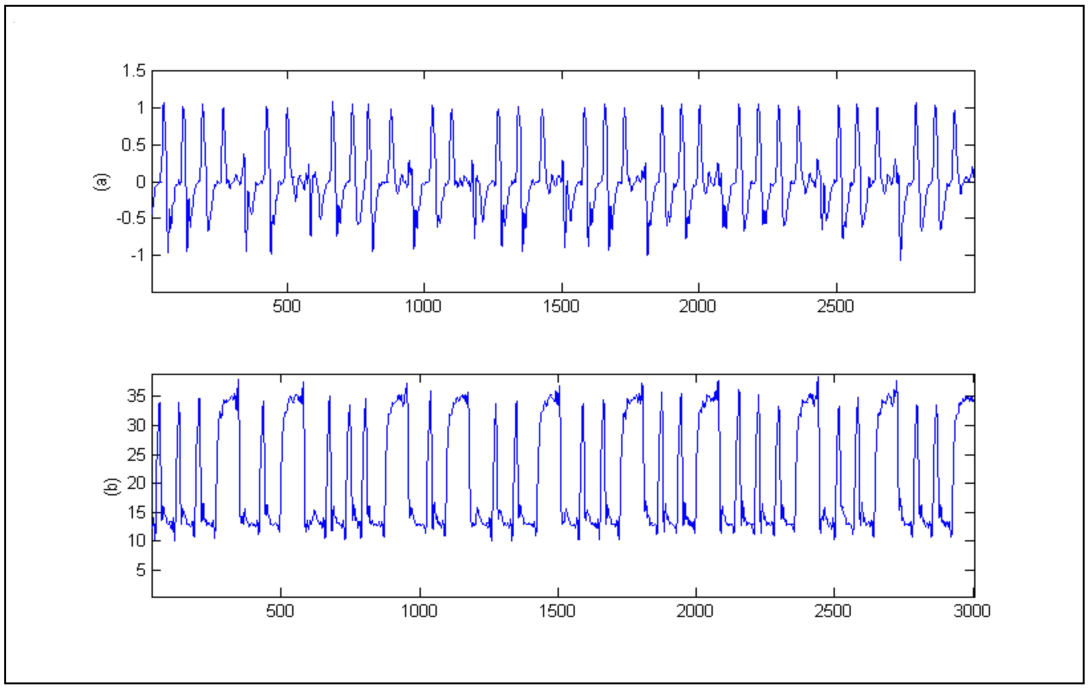

Figure5-7 Data set: (a) Flow; (b) Pressure.

We then test the same algorithms and settings as before on this data set, keeping the network dimensions and embedding the same, but fine tuning the learning, competition and performance parameters to achieve segmentation of the data: in the end, the following results were achieve with the same parameters as for the first data set: γ_{eps}=0.02 and M=5.

5.3.2 RBF Network Models

Figures 5-8 /5-9 show the result for the RBFs' k-means clustering; Figures 5-10/5-11 show the results for the OLS clustering. In both cases, all regime changes are detected, but the latter is less accurate, detects the changes later and is more susceptible to spurious switches than the former. Also on the gate graph, one can see that the competition is not as hard with OLS clustering, which means the models are less specialized.

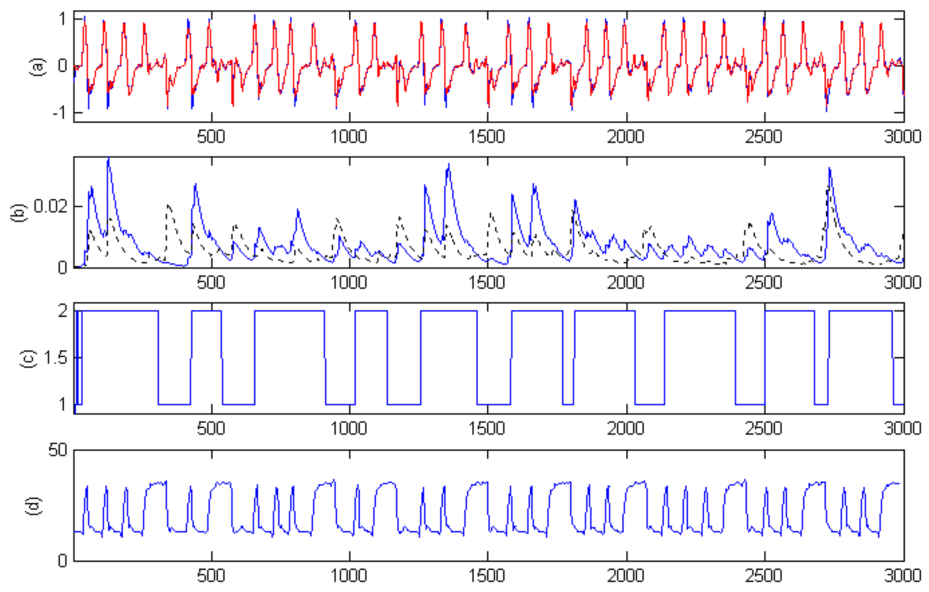

Figure 5-8 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure.

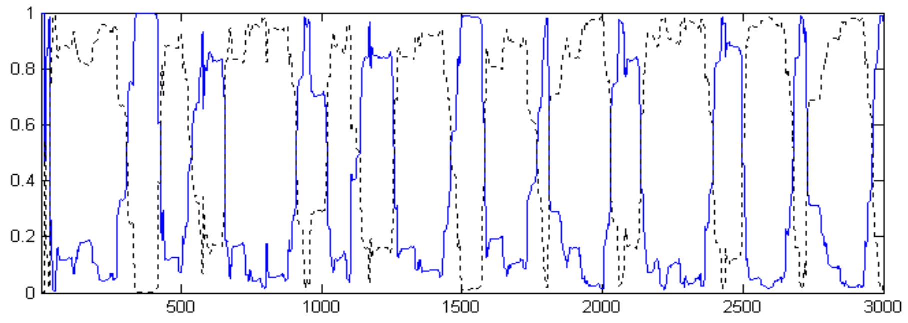

Figure5-9 History of the gate: full line is for expert 1, dotted line is for expert 2

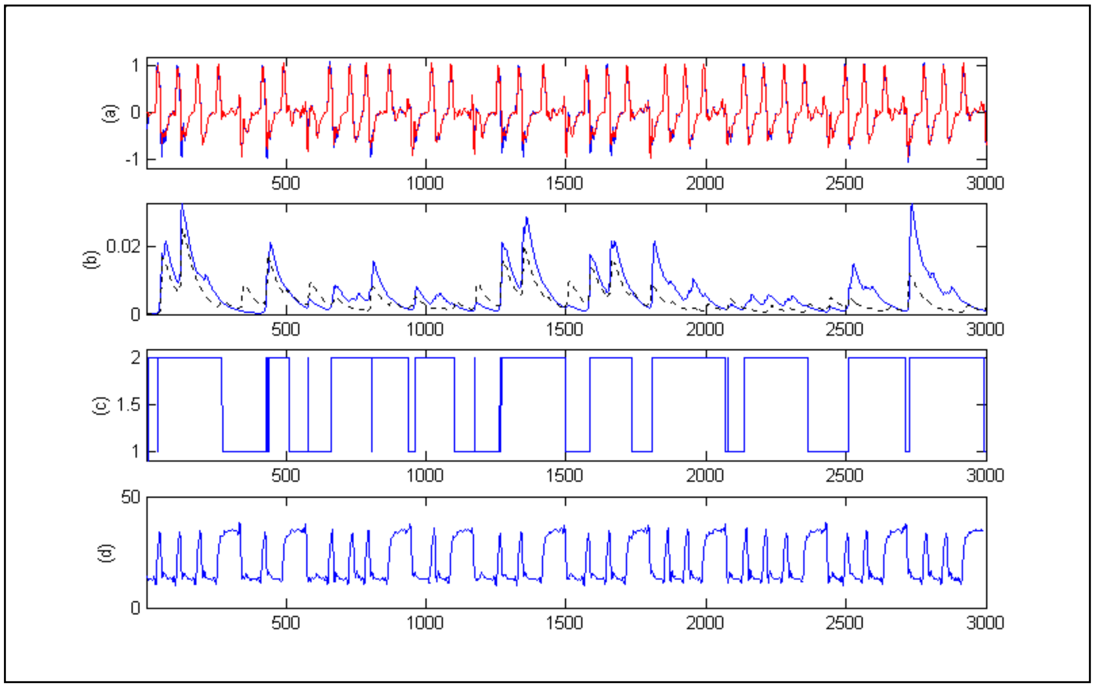

Figure 5-10 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure.

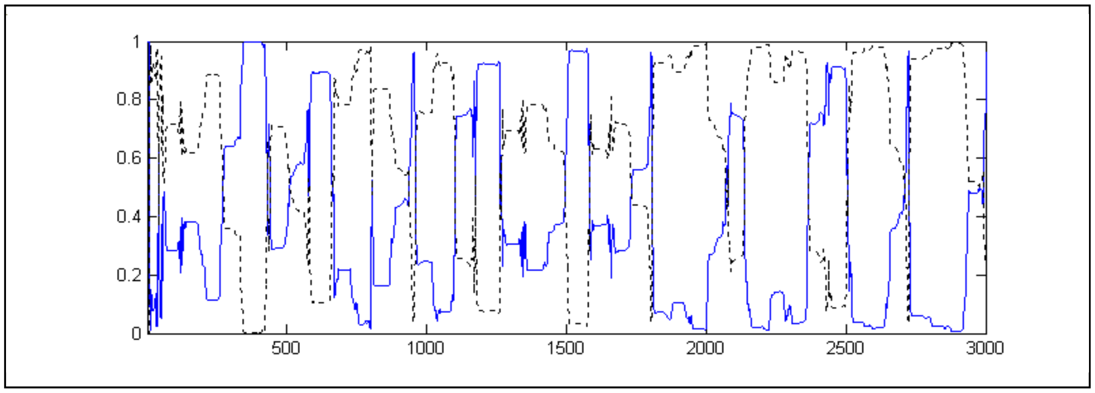

Figure5-11 History of the gate: full line is for expert 1, dotted line is for expert 2

5.3.3 MLP Network Models

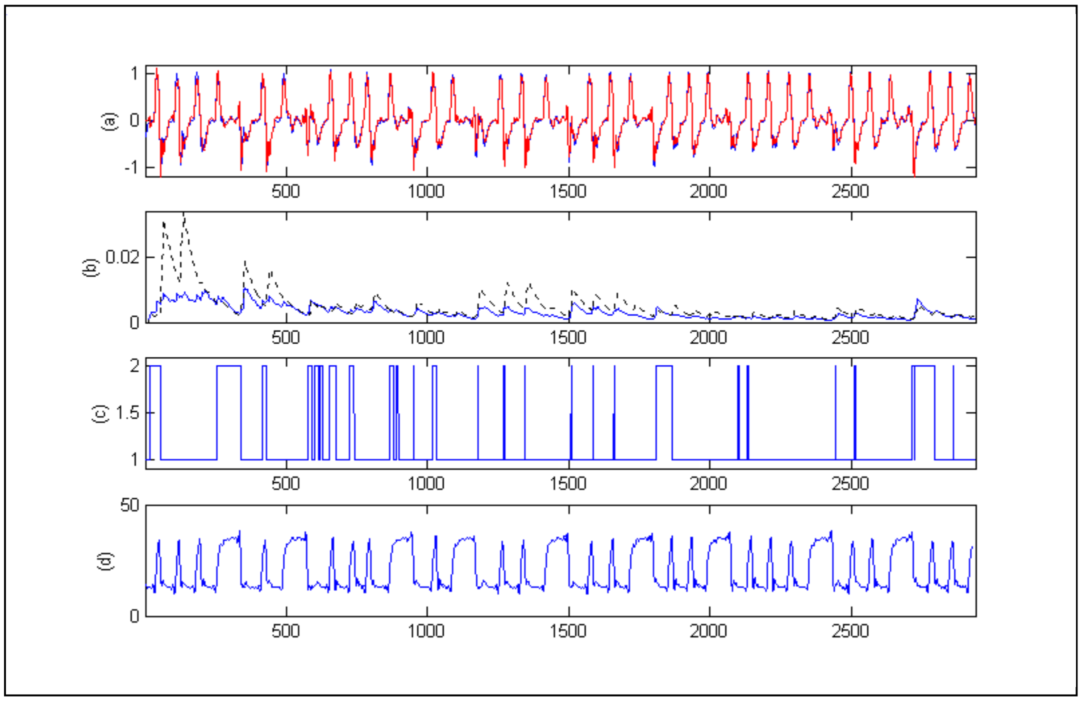

Figure 5-12 Results: (a) Flow: desired in blue, predicted in red; (b) MSE criterion, full line is for expert 1, dotted line is for expert 2; (c) Winner; (d) Pressure.

As an be seen on Figure 5-12, the MLP is actually adapting well to the models, but not with enough specialization that could differentiate the two experts: the competition shown in Figure 5-13 is not as fierce as for the RBFN, for the reasons evoqued in part 5.2. Also the simulations with the MLP are a lot longer because computationaly more extensive thant for the MLP, so for this data set the RBFs'are the best choice of experts.

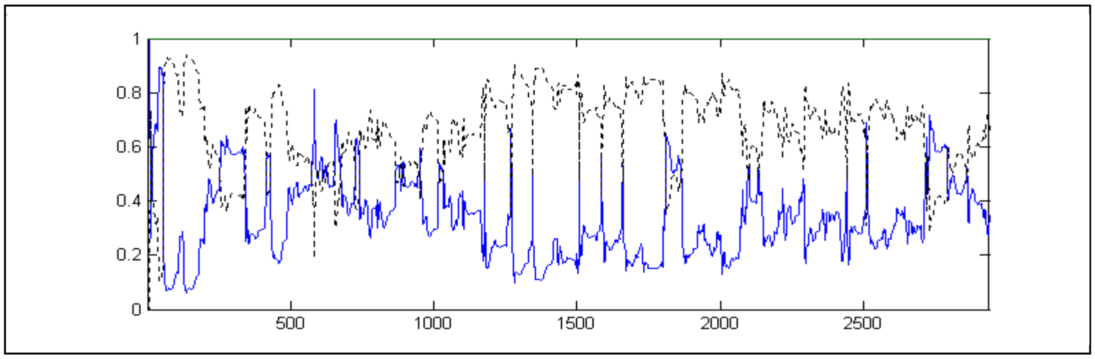

Figure5-13 History of the gate: full line is for expert 1, dotted line is for expert 2

CHAPTER 6
CONCLUSION

Many real-world time series are multi-modal, where the underlying data generating process switches between several different subprocesses. The difficulty in analyzing such systems is to discover the underlying switching process, which entails identifying the number of subprocesses, the dynamics of each subprocess and the switching pattern between them. The two different kinds of changes in stationarity (sudden or over time), can be addressed in two ways (segmenting or tracking), and real world time series usually are non-stationary in both regards. The goal of this thesis was to perform online segmentation of a real-world time series, i.e., tracking of the slow evolution as well as detecting the sudden changes.

To that aim several algorithms have been tested: in approach I the experts are trained offline, then the data is processed through the system, performing the segmentation without being adapted. While the method worked to some extent with the test set, it is not an acceptable approach when the time series is known to be non-stationary. Therefore more approaches were considered, based on unsupervised algorithms (none was completely unsupervised since the experts were always trained a little on the sample breaths to provide “good” initial conditions allowing the experts to specialize). Approach II and III implement hard competition, where only the winner is gets an updated. Finally in approach IV a softer competition is implemented, where both experts are updated proportionally to the output of the gate.

These approaches were first tested on a synthetic data set, to understand the effect of the system and model parameters, then approach IV was applied to real data, yielding good results especially with the RBFs. Overall, the results displayed in chapter 5 are very encouraging and show that these segmentation methods can work well even if the data is not piecewise stationary, but at the price of a lot of human parameter settings and design choices. The main challenge resides in the interdependence of all these parameters.

As previously explained, optimizing several parameters could improve the results. One of these is the embedding: since the data is oversampled for some segments of the data, the choice of a simple delay for the lag is probably not an optimum value, and some simulations with a lag of two or three should be performed. Future work should also include considering using more experts (at least three: one for each type of inspiration, one for expiration) so that there is more opportunity of specialization and differentiation between the experts, and allowing those different experts to be structurally different: since they are modeling processes that might not require the same state space dimension, the same memory depth on the criterion, or might not even be accurately modeled by the same kind of experts! The self-organizing map of competiting predictors [28] might perform better than the algorithms tested, but it seemed awkward to create a Kohonen map for only two predictors. Improvement on the models could derive from using a linear fit of the data complemented by RBFs explaining the residual variations, and from examining different estimates for the criterion (as discussed, using a boxcar average would give too much weight to past values of the instantaneous error for the purpose of change detection, but the a ramp or triangular window would have the advantage of being of finite memory like the boxcar, and give more weight to the latest prediction error values like the recursive gamma average).

Also, a quantitative analysis of the segmentation history could be performed by computing a Receiver Operating Characteristic [29], differentiating false alarm from detection by a threshold on the number of concurrent samples without change after a switch, and assessing the performance of the system in terms of accurately detected switches with respect to spurious switches.

Lastly, it is important to note that for the particular case of ventilatory data where the segmentation is based on differences in waveforms, a more straightforward approach might be to use some feature detection or template matching system, that focus on the shape of the signal without depending on the length of a breath or its amplitude like the competitive experts do.

LIST OF REFERENCES

[1] Box G.E.P. and Jenkins G.M., *Time Series Analysis: Forecasting and Control*, revised edition, Holden-Day, San Francisco, 1976.

[2] Basseville M. and Benveniste A., Detection of Abrupt Changes in Signals and Dynamical Systems, *Lecture Notes in Control and Information Sciences*, vol. 77, Springer-Verlag, Berlin, 1986

[3] Sandberg I.W., Lo T. J., Fancourt C.L., Principe J.C., Katagiri S., Haykin S., *Nonlinear Dynamical Systems: Feedforward Neural Network Perspectives (Adaptive and Learning Systems for Signal Processing, Communications and control)*, Wiley-Interscience, New York, 2001.

[4] Haan-Go Choi, Principe J.C., Hutchison A.A, Wozniak, J., A. Multiresolution segmentation of respiratory electromyographic signals, *IEEE Trans. Biomedical Engineering,* vol.41, pp 257-266, 1994

[5] Bernard M., Bouchoucha M., Cugnenc P, Analysis of medical signals by an automatic method of segmentation and classification without any a-priori knowledge, *Proc. of Third Annual IEEE Symposium on Computer-Based Medical Systems*, pp 381–388, 1990.

[6] Bhattacharya J., Detection of weak chaos in infant respiration, *IEEE Trans. On Systems, Man and Cybernetics, Part B*, vol. 31, no 4, pp 634-642, 2001

[7] Cohen M.E., Hudson D.L., Anderson M.F., Deedwania P.C., Using chaotic measures to summarize medical signal analysis, *Proceedings of the First Joint Annual Fall Meeting of the Biomedical Engineering Society*, vol. 2, p 904, 1999

[8] Wang, Y., Adali T., Kung S.-Y, Szabo Z., Quantification and segmentation of brain tissues from MR images: A probabilistic neural network approach, *IEEE Trans. Image Processing*, vol. 7, pp.1165-1181, 1998

[9] Appel U. and Brandt A.V., Adaptive sequential segmentation of piecewise stationary time series*, Inf. Sci.*, vol. 29, no.1, pp 27-56, 1982.

[10] Fancourt C.L. and Principe J.C, On the use of neural networks in the generalized likelihood ratio test for detecting abrupt changes in signals, *Proc. of the IEEE-INNS-ENNS Int. Joint Conf. on Neural Networks (IJCNN),* vol. 2, pp. 243-248, 2000.

[11] Fancourt C.L. and Principe J.C, Modeling Time Dependencies in the Mixture of Experts, *Proc. of the Int. Joint Conf. on Neural Networks (IJCNN)*, vol. 3, pp. 2324-2328, 1998.

[12] Basseville M. and Nikiforov I.V., *Detection of Abrupt Changes, Theory and Application*, Prentice Hall, Upper Saddle river, 1993.

[13] Principe J.C., deVries B., Guedes de Oliveira P., The gamma filter: a new class of adaptive IIR filters with restricted feedback, *IEEE Trans. Signal Processing*, vol. 41, no. 2, pp.649-656, 1993.

[14] Rao Y.N., *Algorithms for Eigendecomposition and Times Series Segmentation*, Master of Science Thesis, University of Florida, 2000.

[15] Fancourt C. L. and Principe J.C., Temporal self-organization through competitive prediction, *Proc. of the IEEE Int. Conf on Acoustics, Speech and signal Processing (ICASSP)*, vol. 4, pp. 3325–3328, 1997.

[16] Macintyre N.R. and Hagus C. K., *Graphical Analysis of Flow, Pressure, and Volume*, Bear Medical systems, Riverside, 1989.

[17] Waugh J.B., Deshpande V.M., Harwood R.J., *Rapid Interpretation of Ventilator Waveforms*, Prentice Hall, Upper Saddle River, 1999.

[18] Takens F, Detecting strange attractors in turbulence, in Rand D.A. and Young L., Dynamical systems and Turbulence (Springer Lecture Notes in Mathematics), Springer, Berlin, vol. 898, pp 366-381, 1980.

[19] Principe J.C., Wang L., Motter M.A., Local dynamic modeling with self-organizing maps and applications to linear system identification and control, *Proc. of the IEEE*, vol. 86, pp. 2240-2258, 1998.

[20] Mees A., Useful Lies: Dynamics from data; *Non Linear and Nonstationary Signal Processing*, Fitzgerald W.J., Smith R.L., eds., Cambridge University Press, Cambridge, 2000.

[21] Kuo J. and Principe J.C., Reconstructed dynamics and chaotic signal modeling, *Proc. of the IEEE Workshop on the Neural Networks for Signal Processing,* pp.661-670, 1994.

[22] Haykin S. and Principe J.C., Making sense of a complex world [chaotic events modeling], *IEEE Digital Signal Processing Mag*., pp 66-81, 1998.

[23] Haykin S., *Adaptive Filtering Theory*, Prentice Hall, Upper Saddle River, 1996

[24] Haykin S., *Neural Networks: A Comprehensive Foundation*, Mac Millan, New York, 1994.

[25] Duda R. O. and Hart P. E., *Pattern Classification and Scene Analysis*, Wiley, New York, 1973

[26] Chen S., Cowan C.F.N., Grant P. M., Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks, *IEEE Trans. on Neural Networks*, vol. 2, no. 2, pp. 302-309, 1991.

[27] Principe, J., Euliano, N., Lefebvre, C*., Neural and Adaptive Systems*, John Wiley & Sons, New York, 2000.

[28] Fancourt C. L. and Principe J.C., A neighborhood map of competing one-step predictors for piecewise segmentation and identification of time series*, IEEE Trans. Conf. On Neural Networks (ICNN)*, vol. 4, pp. 1906-1911, 1996.

[29] Helstrom C. W., *Elements of Signal Detection and Estimation*, Prentice Hall, Englewoods Cliffs, 1995

BIOGRAPHICAL SKETCH

Helene Chini was born in Talence, France, on May 8, 1976. She graduated high school with honors in June 1994 and attended the Lycée Pierre de Fermat in Toulouse for the two-year science prep school program (necessary to take the national competitive exams for admission into French engineering schools) at the issue of which she was admitted to her second school of choice: ENSIEG, the electrical engineering department of the INPG National Polytechnic Institute in Grenoble, France. She graduated from ENSIEG in September 2000 with the degree of Engineer, a major in industrial and computer science, and a comprehensive background in signal processing, electrical power and process control. In fall 1999, she was awarded the University Alumni Fellowship from the University of Florida and began her graduate studies by earning a minor in nuclear and radiological engineering. She joined the Computational NeuroEngineering Laboratory in May 2001, and ever since has been working as a research assistant under the supervision of Dr. Jose C. Principe on her thesis to complete her M.S. in electrical and computer engineering. Her primary area of interest is the processing of nonstationary, non-Gaussian signals with dynamic neural networks.