Ath. Kehagias and V. Petridis.
"Time Series Segmentation using Predictive Modular Neural Networks".

This paper has appeared in the journal:
Neural Computation, Vol.9, pp.1691-1710, 1997.

Manuscript Nr. 1406

Time Series Segmentation using Predictive Modular Neural Networks

Ath. Kehagias
Dept. of Electrical Engineering
Aristotle Univ. of Thessaloniki
and
American College of Thessaloniki

Vas. Petridis
Dept. of Electrical Eng.
Aristotle Univ. of Thessaloniki

January 9, 1997

Abstract

A predictive modular neural network method is applied to the problem of unsupervised time series segmentation. The method consists of the concurrent application of two algorithms: one for source identification, the other for time series classification. The source identification algorithm discovers the sources generating the time series, assigns data to each source and trains one predictor for each source. The classification algorithm recursively computes a credit function for each source, based on the competition of the respective predictors, according to their predictive accuracy; the credit function is used for classification of the time series observation at each time step. The method is tested by numerical experiments.

This paper appeared in *Neural Computation*, Vol.9, pp.1691-1710, 1997

1 Introduction

For a given time series, the *segmentation* process should result in a number of models each of which best describes the observed behavior for a particular segment of the time series. This can be formulated in a more exact manner as follows. A time series y_t, $t = 1, 2, ...$ is generated by a source $S(z_t)$, where z_t is a time varying *source parameter*, taking values in a finite *parameter* set $\Theta = \{\theta_1, \theta_2, ..., \theta_K\}$. At time t, y_t depends on the value of z_t, as well as on y_{t-1}, y_{t-2}, ... However, the nature of this dependence, the number K of parameter values as well as the actual values θ_1, θ_2, ..., θ_K are unknown. What is required is to find the value of z_t, for t=1, 2,...; or, in other words, to *classify* y_t for t=1, 2, ... Typical examples of this problem include speech recognition (Rabiner, 1988), radar signal classification (Haykin and Cong, 1991), electroencephalogram processing (Zetterberg et al. 1981); other applications are listed in (Hertz et al., 1991).

Under the assumptions that (a) the parameter set $\Theta = \{\theta_1, \theta_2, ..., \theta_K\}$ is known in advance and (b) a predictor has been trained on source specific data before the classification process starts, we have presented a general framework for solving this type of problem in (Petridis and Kehagias, 1996a, 1996b; Kehagias and Petridis, 1997), using *Predictive Modular Neural Networks* (PREMONNs). The basic feature of the PREMONN approach is the use of a decision module and a bank of prediction modules. Each prediction module is tuned to a particular source $S(\theta_k)$ and computes the respective prediction error; the decision module uses the prediction errors to recursively compute certain quantities (one for each source) which can be understood as the Bayesian posterior probabilities (probabilistic interpretation) or simply as credit functions (phenomenological interpretation). Classification is performed by maximization of the credit function.

Similar approaches have been proposed in the control and estimation literature (Lainiotis, 1968, 1969, 1971) and in the context of *predictive hidden Markov models* (Kenny et al., 1990). These approaches also assume the sources to be known in advance.

On the other hand, several papers which have appeared recently (Jacobs et al. 1991, Jordan and Jacobs 1994; Jordan and Xu, 1995; Xu and Jordan, 1996) use a similar approach but provide a mechanism for *unsupervised source specification*. The term *mixtures of experts* is used in these papers; each expert is a neural network specialized in learning the required input - output relationship for a particular region of the pattern space. There is also a *gating* network that combines the output of each expert to produce a better approximation of the desired input - output relationship. The activation pattern of the local experts is the output of the gating network, and it can be considered as a classifier output: when a particular expert is activated, this indicates that the current observation lies in a particular section of the pattern space.

The emphasis in the mixture of experts method, is on *static patterns* classification, rather than classification of dynamic patterns such as time series. However, there are some applications of the mixture of experts method to dynamic problems of time series classification. For example, in (Prank et al., 1996), the mixture of experts method is applied to a problem of classifying the "pulsatile" pattern of growth hormone in healthy subjects and patients. The goal is to discriminate the pulsatile pattern of healthy subjects from that of patients. In general the time series corresponding to a healthy subject is more predictable than that corresponding to a patient. Classification is based on the level of prediction error for a particular time series (low error indicating a healthy subject and high error indicating a patient). The selection pattern of local experts (i.e. which expert is activated at any given time step), can also be used for classification, since the frequency distribution of most probable experts significantly differs between patients and healthy subjects. However, the activation of experts by the gating network, is performed at every time step independently of previous activation patterns. This may result in the loss of important correlations across time steps, which characterize the dynamics of the time series.

This static approach to time series classification may be problematic in cases where there is significant correlation between successive time steps of the time series; this has been recognized and discussed in (Cacciatore and Nowlan, 1994), where it is stated that "The original formulation of this [local experts] architecture was based on an assumption of statistical independence of training pairs ... this assumption is inappropriate for modelling the causal dependencies in control tasks". Cacciatore and

Nowlan proceed to develop the *mixture of controllers architecture*, which emphasizes "the importance of using recurrence in the gating network". The mixture of controllers is similar to the method we present in this paper. However, the method presented in (Cacciatore and Nowlan, 1994) requires several passes through the training data; our method, on the other hand can be operated on-line as it does not require multiple passes. In addition, in some cases Cacciatore and Nowlan need to explicitly provide the active sources (for some portion of the training process) for the mixture of controllers network to accomplish the required control task; in our case learning is completely unsupervised and the active sources always remain hidden.

Finally, a very recent paper (Pawelzik et al., 1996) presents an approach using *Annealed Competition of Experts*. The method presented therein is similar to the work cited earlier; it is applied to time series segmentation and does *not* assume the sources to be known in advance. This method is mainly suitable for off-line implementation, due to the heavy computational load incurred by the annealing process.

In this paper we present an unsupervised time series segmentation scheme, which combines two algorithms. The first is the *source identification* algorithm: it detects the number of active sources in the time series, performs *assignment* of data to the sources and uses the data to train one predictor for each active source. The second is the *classification* algorithm: it uses the sources / predictors specified by the identification algorithm and implements a PREMONN which computes credit functions. A probabilistic (Bayesian) interpretation of this algorithm is possible, but not necessary.

The combination of the two algorithms, the source identification one and the time series classification one, constitutes the combined time series segmentation scheme; the two algorithms can be performed concurrently and on-line. A crucial assumption is that of *slow switching* between sources. Our approach is characterized by the use of a modular architecture, with predictive modules competing for data and credit assignment. A decision module is also used, to assign the credit in a recursive manner which takes into account previous classification decisions. It should be mentioned that a convergence analysis of the classification algorithm is available elsewhere (Kehagias and Petridis, 1997).

2 The Time Series Segmentation Problem

Consider a time series z_t, $t = 1, 2, ...$, taking values in the *unknown* set $\Theta = \{\theta_1, ..., \theta_K\}$; further consider a time series y_t, $t = 1, 2, ...$, which is generated by some source $S(z_t)$: for every value of z_t a source $S(z_t)$ is activated, which produces y_t using y_{t-1}, y_{t-2}, ... We call z_t the *source parameter* and y_t the *observation*. For instance, if $z_t = \theta_1$, we could have $y_t = f(y_{t-1}, y_{t-2}, ...; \theta_1)$, where $f(\cdot)$ is an appropriate function with parameter θ_1; θ_1 can be a scalar or vector quantity or, simply, a label variable.

The task of *time series segmentation* consists in estimating the values z_1, z_2, ... , based on the observations y_1, y_2, ... The size K of the *source set* Θ and its members $\theta_1, \theta_2, ..., \theta_K$, are initially unknown. This original task can be separated into two subtasks: *source identification* and *time series classification*. The task of source identification consists in determining the active sources or, equivalently, the source set $\Theta = \{\theta_1, ..., \theta_K\}$. Having achieved this, the time series classification task consists in corresponding each z_t (t=1, 2, ...) to a value θ_k ($k = 1, 2, ..., K$) or, equivalently, to determine at each time step t the source $S(\theta_k)$ that has generated y_t.

The combination of the two subtasks yields the original *time series segmentation* task. Two algorithms are presented here, one to solve each subtask. The *source identification algorithm* distributes data y_1, y_2, ... , y_t among active sources and uses the data to train one *predictor* per source. For instance, a data segment y_1, y_2, ... , y_M may be assigned to source $S(\theta_1)$ and used to approximate some postulated relationship $y_t = f(y_{t-1}, y_{t-2}, ...; \theta_1)$. [1] The *time series classification algorithm* uses the predictors trained in the source identification phase, to compute predictions of incoming observations. The respective prediction errors are used to compute a credit function for each predictor / source; an observation y_t is classified to the source of maximum credit.

Each of the two algorithms has been developed using a different approach. The source identification algorithm has an heuristic motivation; its justification is experimental. On the other hand, the classification algorithm is motivated by probabilistic considerations. While in this paper its performance is evaluated only experimentally, a detailed theoretical analysis is available (Kehagias and Petridis, 1997).

3 The Time Series Segmentation Scheme

We first present the two algorithms that constitute the time series segmentation scheme; then we proceed to describe the manner in which they are combined.

3.1 The Source Identification Algorithm

Other authors have tackled the same problem using similar methods (Pawelzik et al., 1996). There are also connections to k-means (Duda, 1979) and Kohonen's learning vector quantizer (Kohonen, 1981); however note that these have been applied mainly to classification of *static* patterns. We now give a description of the algorithm. The following parameters will be used in what follows; L, the length of a *data block*, M, the length of a *data segment*, where $M < L$, K the number of active predictors and K_{max} the *maximum allowed* number of active predictors.

SOURCE IDENTIFICATION ALGORITHM

Initialization

Set $K = 1$. Use data block y_1, y_2, ... , y_L to train the first predictor, of the form $y_t^1 = f(y_{t-1}, y_{t-2}, ...; \theta_1)$, where $f(\cdot)$ is a (sigmoid) neural network and θ_1 is taken to be the connection weights.

Main Routine

Loop for $n = 1, 2, ...$

[1] $y_t = f(y_{t-1}, y_{t-2}, ...; z_t)$ reflects the dependence of y_t on past observations, as well as on the parameter z_t. In general the true input-output relationship will be unknown and $f(\cdot)$ will be an approximation, such as furnished by a sigmoid neural network. In fact, due to the competitive nature of the algorithms presented, even a crude approximation suffices, as will be explained later.

1. Split *data block* y_t, $t = n \cdot L + 1, n \cdot L + 2, ..., (n+1) \cdot L$ (that is, L time steps) into L/M *data segments* (that is, M time steps, $M < L$).

2. Loop for $k = 1, ..., K$.

   (a) Use each segment of y_t's as input to the k-th predictor to compute a prediction $y_t^k = f(y_{t-1}, ...; \theta_k)$ for each time t.

   (b) For every segment, compare the predictions y_t^k with the actual observations y_t ($t = n \cdot L + 1, n \cdot L + 2, ..., (n+1) \cdot L$) to compute the *Segment-Average Square Prediction Error* (henceforth SASPE).

   End of k loop.

3. Loop for every segment in the block (a total of L/M segments).

   (a) For the current segment, find the predictor that produces the minimum SASPE.

   (b) For this predictor, compare minimum SASPE with a respective threshold [2].

   (c) In case SASPE is above the threshold, and $K < K_{max}$, set $K = K + 1$ and assign the current segment to a new predictor; otherwise, assign the segment to the predictor of minimum SASPE.

   End of segment loop.

4. Loop for $k = 1, ..., K$.

   (a) Retrain the k-th predictor for J iterations, using the L/M most current data segments assigned to it.

   End of k loop.

End of n loop.

It is assumed that the switching rate is slow enough, so that the initial segment will mostly contain data from one source; however some data from other sources may also be present. The modular and competitive nature of the algorithm is evident. A data segment is assigned to a predictor even if the respective prediction is poor; what matters is that it is better than the remaining ones. Hence, every predictor is assigned data segments on the assumption that they were generated by a particular source and the predictor is trained (*specialized*) on this source. Training can be performed independently, in parallel, for each predictor.

Numerical experimentation indicates that the values of several parameters may affect performance; these are now discussed.

[2] This threshold is computed separately for each predictor.

1. L is the length of data block. If L is too large, then training for each data block requires a long time. If L is too small, then not enough data are available for the retraining of the predictors. For the experiments reported in Section 4, we have found that values around 100 are sufficient to ensure good training of the predictors without incurring excessive computational load.

2. M is the length of data segment; it is related to the switching rate of z_t. Let T_s denote the minimum number of time steps between two successive source switches. While T_s is unknown, we operate on the assumption of *slow* switching, which means that T_s will be relatively large. Now, since the M data points included in a segment will all be assigned to the same source (and used to train the same predictor), it is obviously desirable that they have been truly generated by the same source; otherwise the respective predictor will not specialize on one source. In reality, however, this is not guaranteed to happen. In general, a small value of M will increase the likelihood that most segments contain data from a single source. On the other hand, it has been found that small M leads to an essentially random assignment of data points to sources, especially in the initial stages of segmentation, when the predictors have not specialized sufficiently. The converse situation holds for large M. In practice, one needs to guess a value for T_s and then take M somewhere between 1 and T_s. This choice is consistent with the hypothesis of slow switching rate. The practical result is that most segments in a block contain data from exactly one source, and a few segments contain data from two sources. It should be stressed that the exact value of T_s is not required to be known; a rough guess suffices.

3. J, the number of training iterations, should be taken relatively small, since the predictors must not be overspecialized in the early phases of the algorithm, when relatively few data points are available.

4. The error threshold is used to determine if and when a new predictor should be added. At every re-training epoch the training prediction error, E_k, is computed for $k = 1, 2, ..., K$. Then the threshold, D_k, is set equal to $a \cdot E_k$, where a is a scale factor. A high a value makes the introduction of new predictors difficult, since large prediction errors are tolerated; conversely, a low a value facilitates the introduction of new predictors.

5. In practice a value K_{max} is set to be the maximum number of predictors used. As long as this is equal to or larger than the number of truly active sources, it does not affect the performance of the algorithm seriously.

A further quantity of interest is T, the length of time series. Theoretically this is infinite, since the algorithm is on-line, with new data becoming continuously available. However, it is interesting to note the minimum number of data required for the algorithm to perform the source identification succesfully. In the experiments presented in Section 4 it is usually between 1000 and 3000. This is the information that the algorithm requires to discover the number of active sources and to train respective predictors. In addition, the time required to process the data is short, because only a few training iterations are performed per data segment; these computations are performed while waiting for the new data segment to be accumulated. The short training time and the small amount of data required, make the source identification algorithm very suitable for on-line implementation.

3.2 The Classification Algorithm

The classification algorithm has already been presented in (Petridis and Kehagias 1996b; Kehagias and Petridis, 1997) under the name of *Predictive Modular Neural Network* (PREMONN). A brief summary is given here.

It is assumed that the source parameter time series z_t, $t = 1, 2, ...$, takes values in the *known* set $\Theta = \{\theta_1, ..., \theta_K\}$ (this has been specified by the source identification algorithm). Corresponding to each source $S(\theta_k)$, there is a predictor (e.g. a neural network trained by the source identification algorithm) which is characterized by the following equation ($k = 1, 2, ..., K$):

$$y_t^k = f(y_{t-1}, y_{t-2}, ...; \theta_k). \tag{1}$$

At every time step t, the classification algorithm is required to identify the member of Θ which *best* matches the observation y_t. Using a probabilistic approach, one can define $p_t^k = \Pr(z_t = \theta_k | y_1, y_2, ..., y_t)$. In (Petridis and Kehagias, 1996a) the following p_t^k update is obtained:

$$p_t^k = \frac{p_{t-1}^k \cdot e^{-\frac{|y_t - y_t^k|^2}{\sigma^2}}}{\sum_{l=1}^K p_{t-1}^l \cdot e^{-\frac{|y_t - y_t^l|^2}{\sigma^2}}} \tag{2}$$

Eq.(2) is a recursive implementation of Bayes' rule. The recursion is started using p_0^k, the prior probability assigned to θ_k at time $t = 0$ (before any observations are available). In the absence of any prior information, all models are assumed equally credible:

$$p_0^k = \frac{1}{K} \quad for \quad k = 1, 2, ..., K. \tag{3}$$

Finally, $\hat{z}_t$ (the estimate of z_t) is chosen as

$$\hat{z}_t \doteq arg \max_{\theta_k \in \Theta} p_t^k. \tag{4}$$

Hence the PREMONN classification algorithm is summarized by eqs.(3), (1), (2), (4). This is a recursive implementation of Bayes' rule; the value p_t^k reflects our belief (at time t) that the observations are produced by a model with parameter value θ_k;

The probabilistic interpretation of the algorithm, presented above, is not necessary. A different, *phenomenological*, interpretation goes as follows. p_t^k is a *credit function* which evaluates the performance of the k-th source in predicting the observations y_1, y_2, ... , y_t. This evaluation depends on two factors: $e^{-\frac{|y_t - y_t^k|^2}{\sigma^2}}$ reflects how accurately the current observation was predicted, while p_{t-1}^k reflects past performance. Hence eq.(2) updates the credit function p_t^k recursively; the update takes into account the previous value p_{t-1}^k; this makes p_t^k implicitly dependent on the entire time series history. The final effect is that correlations between successive observations are captured. Note also the competitive nature of classification, which is evident in eq.(2): even if a predictor performs poorly, it may still receive high credit, as long as the remaining predictors perform even worse.

In (Kehagias and Petridis 1997) it is proved that eq.(2) converges to appropriate values, that ensure

classification to the most accurate predictor. In the case of slow switching (as assumed in this paper), convergence to the active source is guaranteed for the time interval between two source switchings, *provided that the convergence rate is fast, compared to switching rate.* Again, this requirement is in accordance with the hypothesis of slow switching.

The performance of the classification algorithm is influenced by two parameters, σ and h, which are discussed next.

1. σ is a smoothing parameter. Under the probabilistic interpretation, σ is the standard deviation of prediction error. A phenomenological interpretation is also possible. For a fixed error $|y_t - y_t^k|$, note that a large σ results in a large value of $e^{-\frac{|y_t - y_t^k|^2}{\sigma^2}}$. This is of interest particularly in the case where a predictor performs better than the rest over a large time interval, but occasionally generates a bad prediction (perhaps due to random fluctuation). In this case, a very small σ will result in a very small value of $e^{-\frac{|y_t - y_t^k|^2}{\sigma^2}}$ in eq.(2). This, in turn, will decrease p_t^k excessively. On the other hand, a very large σ results in sluggish performance (a long series of large prediction errors is required before p_t^k is sufficiently decreased). For the experiments reported in this paper, σ is determined at every time step t by the following *adaptive* estimate: $\sigma = \frac{\sqrt{\sum_{k=1}^{K} |y_t - y_t^k|^2}}{K}$.

2. A threshold parameter h is also used in the classification algorithm. Suppose that z_t remains fixed for a long time at, say, θ_1; then at time t_s a switch to θ_2 occurs. For $t < t_s$, $e^{-\frac{|y_t - y_t^1|^2}{\sigma^2}}$ is large and $e^{-\frac{|y_t - y_t^2|^2}{\sigma^2}}$ small (much less than one). Considering eq.(2) one sees that p_t^2 decreases exponentially; after a few steps it may become so small that computer underflow takes place and p_t^2 is set to 0. It follows from eq.(2) that after that point p_t^2 will remain 0 for all subsequent time steps; in particular, this will also be true after t_s, despite the fact that now the second predictor performs well ($e^{-\frac{|y_t - y_t^2|^2}{\sigma^2}}$ will be large, but multiplied by p_{t-1}^2 will still yield $p_t^2 = 0$). Hence the currently best predictor receives zero credit and is excluded from the classification process. What is required is to ensure that p_t^2 will never become *too* small. Hence the following strategy is used: whenever p_t^k falls below h, it is reset to h; if h is chosen small (say around 0.01), then this resetting does not influence the classification process, but also gives predictors with low credit the chance to quickly recover, once the respective source is activated.

3.3 The Time Series Segmentation Scheme

The time series segmentation scheme consists in the concurrent application of the two algorithms. Namely, the classification algorithm is run on the current data block, using the predictors discovered and trained by the source identification algorithm in the *previous* epoch [3] (operating on the *previous* data block). Concurrently, the identification algorithm operates on the current block and provides an update of the predictors; but this update will only be used in the next epoch of the segmentation scheme. Hence, the classification algorithm uses information provided by the source identification algorithm, but not vice versa.

[3] An epoch is defined as the interval of$L$ time steps which correspond to a data block.

The identification algorithm in effect provides a crude classification, by assigning *segments* to predictors; the role of the classification algorithm is to provide segmentation at the level of *individual time steps*. Comparatively speaking, in the initial stages of the segmentation process, the identification algorithm is more important than the classification algorithm and the predictors change significantly at every epoch; at a later stage the predictors reach a steady state and the source identification algorithm becomes less important. In fact, a reduced version of the segmentation scheme could use *only* the identification algorithm initially (until the predictors are fairly well defined) and *only* the classification algorithm later (when the predictors are not expected to change significantly). However, for this reduced version to work, it is required that new sources are not activated at a later stage of the time series evolution.

4 Experiments

Two sets of experiments are presented in this section, so as to evaluate the segmentation scheme.

4.1 Four Chaotic Time Series

Consider four sources, each generating a chaotic time series:

1. for $z_t = 1$, a logistic time series of the form $y_t = f_1(y_{t-1})$, where $f_1(x) = 4x(1-x)$;
2. for $z_t = 2$, a tent-map time series of the form $y_t = f_2(y_{t-1})$, where $f_2(x) = 2x$ if $x \in [0, 0.5)$ and $f_2(x) = 2(1-x)$ if $x \in [0.5, 1]$;
3. for $z_t = 3$, a double logistic time series of the form $y_t = f_3(y_{t-1}) = f_1(f_1(y_{t-1}))$ and
4. for $z_t = 4$, a double tent-map time series of the form $y_t = f_4(y_{t-1}) = f_2(f_2(y_{t-1}))$.

The four sources are activated consecutively, each for 100 time steps, giving an overall period of 400 time steps. Ten such periods are used, resulting in a 4000-steps time series. The task is to discover the four sources and the switching schedule by which they are activated. Six hundred time steps of the composite time series (encompassing the source transition $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 1 \rightarrow 2$) are presented in Figure 1. Hence the first hundred time steps correspond to $z_t = 1$; the second hundred time steps correspond to $z_t = 2$ and so on.

A number of experiments are performed, using several different combinations of block and segment lengths; also the time series is observed at various levels of noise, i.e. at every step y_t is mixed with additive white noise uniformly distributed in the interval $[-A/2, A/2]$. Hence an experiment is determined by a combination of L, M and A values. The predictors used are 1-5-1 sigmoid neural networks; the maximum number of predictors is $K_{max} = 6$; the number of training iterations per time step is $J = 5$; the credit threshold is set at $h = 0.01$. [4]

[4] It should be noted that, for training the sigmoid predictors, a Levenberg-Marquardt algorithm was used; this was implemented by Magnus Norgaard, Technical University of Denmark, and distributed as part of the Neural Network System Identification Toolbox for Matlab.

In every experiment performed, all four sources are eventually identified. This takes place at some time T_c, which is different for every experiment. After time T_c, a classification figure of merit is computed. It is denoted by $c = T_2/T_1$, where T_1 is the *total* number of time steps after T_c, and T_2 is the number of *correctly classified* time steps after T_c. The results of the experiments (i.e. T_c and c) are listed in Tables 1 and 2. The evolution of a representative credit function, in this case p_t^3 (the remaining five credit functions evolve in a similar manner), is presented in Fig. 2.A. This figure was obtained from a noise-free experiment (A=0.0), with L=105, $M = 15$. Note that classification beyond time T_c is very stable. The classification decisions (namely the source / predictor selected at every time step) for the same experiment are presented in Fig.2.B. It can be observed that the four sources correspond to predictors 1, 2, 3 and 6. Predictors 4 and 5 have received relatively few data, have been undertrained and do not correspond to any of the truly active sources; generally these predictors play no significant role in the segmentation process.

4.2 Mackey-Glass Time Series

Now consider a time series obtained from three sources of the Mackey-Glass type. The original data evolves in continuous time and satisfies the differential equation :

$$\frac{dy}{dt} = -0.1y(t) + \frac{0.2y(t-t_d)}{1+y(t-t_d)^{10}}, \tag{5}$$

For each source a different value of the delay parameter t_d was used, namely t_d= 17, 23 and 30. The time series is sampled in discrete time, at a sampling rate $\tau = 6$, with the three sources being activated alternately, for 100 time steps each. The final result is a time series with a switching period of 300 and a total length of 4000 time steps. Six hundred time steps of this final time series (encompassing the source transition $1 \rightarrow 2 \rightarrow 3 \rightarrow 1 \rightarrow 2 \rightarrow 3$) are presented in Figure 3. Hence the first hundred time steps correspond to $z_t = 1$; the second hundred time steps correspond to $z_t = 2$ and so on.

Again, a particular experiment is determined by a combination of L, M and A values. The predictors used are 5-5-1 sigmoid neural networks; the maximum number of predictors is $K_{max} = 6$; the number of training iterations per time step is $J = 5$; the credit threshold is set at $h = 0.01$.

Again all three sources were correctly identified in every experiment. The results of the experiments, T_c and c (which have been described in Section 4.1) are listed in Tables 3 and 4. The evolution of three representative credit functions, p_t^1, p_t^2, p_t^3, are presented in Figs. 4.A, 4.B and 4.C, respectively. This figure was obtained from an experiment with A=0.15, L=105, $M = 15$. Note that classification beyond time T_c is quite stable. The classification decisions (namely the source / predictor selected at every time step) for the same experiment are presented in Fig.4.D. It can be observed that the three sources correspond to predictors 1, 2 and 3. Predictors 4, 5 and 6 have received relatively few data, have been undertrained and do not correspond to any of the truly active sources; generally these predictors play no significant role in the segmentation process.

4.3 Discussion

It can be observed from Tables 1-4 that the segmentation scheme takes between 1000 and 3000 steps to discover the active sources. After that point classification is quite accurate for low to middle noise levels and gradually drops off as noise level increases. [5] The scheme is fast: source identification and classification of the 4000 time steps (run concurrently) require a total of approximately 4 minutes. The algorithms are implemented in MATLAB and run on a 486 120 MHz IBM compatible machine; a significant speedup would result from a C implementation (consider that MATLAB is an interpreted language).

The same segmentation tasks have been treated by the method of *Annealed Competition of Experts* (ACE) in (Pawelzik et al., 1996). Regarding accuracy of segmentation, both methods appear to achieve roughly the same accuracy; however it should be pointed out that the ACE method was applied either to noise-free or low-noise data; we do not know how well the ACE method performs in the presence of relatively high noise. Execution time is not cited in (Pawelzik et al., 1996) for the two tasks considered here. On a third, prediction, task, the ACE method requires 2.5 hr on a SPARC 10/20 GX. In general, we expect the ACE method to be rather time consuming, since it depends on annealing, which requires many passes through the training data.

Regarding the required number of time series observations, the ACE method uses 1200 data points for the first task; 400 data point for the second; several annealing passes are required. Our method requires a single pass through at most 3000 data points. We do not consider this an important difference: because all time series involved are ergodic, we conjecture that processing 3000 data points once should yield more or less the same information as cycling thirty times over 100 data points. [6]

5 Conclusions

An unsupervised time series segmentation scheme has been presented, which combines a source identification and a time series classification algorithm. The source identification algorithm determines the number of sources generating the observations, distributes data between the sources and trains one predictor for each source. Then, predictors are used by the classification algorithm as follows. For every incoming data segment, each predictor computes a prediction; comparing these to the actual observation, segment prediction errors are computed for each predictor. The prediction errors are used to recursively and competitively update a credit function for each predictor / source. Namely, sources which correspond to more succesful predictors receive higher credit. Each segment is classified to the source which currently has highest credit. The combined scheme consists of the identification and classification algorithms running concurrently, in a competitive and recursive manner.

The scheme is accurate, fast and requires few training data, as has been established by numerical experimentation; hence it is suitable for on-line implementation. It is also modular: training and pre-

[5] To evaluate the effect of noise, one should keep in mind that the chaotic time series take values in the $[0,1]$ interval and the Mackey-Glass time series in the $[0.7,1.3]$ interval.

[6] It should also be mentioned that using the same data for training and testing, makes the task considered in (Pawelzik et al., 1996) somewhat easier than the one considered in this paper, where segmentation is evaluated on previously unseen data. But, again, we believe that the difference is not great, due to the ergodic nature of the time series.

diction are performed by independent modules, each of which can be removed from the system, without affecting the properties of the remaining modules. This results in short training and development times. Finally, a convergence analysis of the classification algorithm has been published elsewhere (Kehagias and Petridis, 1997); this guarantees correct classification.

It should be mentioned that, while we have not made any assumptions about the switching behavior of the source parameter, a Markovian assumption can be easily incorporated in the update equation (2). A convergence analysis for this case is also available. This modification affects the classification algorithm only, leaving the source identification algorithm unaffected. However, because of lack of space, these developments will be presented in the future.

References

[1] T.W. Cacciatore and S.J.Nowlan. 1994. "Mixtures of Controllers for Jump Linear and Nonlinear Plants", in *Advances in Neural Information Processing Systems 6 (NIPS 93)*, eds. J.D. Cowen, G. Tesauro and J. Alspector, pp. 719-726, San Francisco, CA, Morgan Kaufmann.

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

[3] S. Haykin and D. Cong. 1991. "Classification of radar clutter using neural networks", IEEE Trans. on Neural Networks, vol.2, pp.589-600.

[4] J. Hertz, A. Krogh and R.G. Palmer. 1991. *Introduction to the Theory of Neural Computation*, Addison-Wesley, Redwood City, CA.

[5] C. G. Hilborn and D.G. Lainiotis. 1969. "Unsupervised Learning Minimum Risk Pattern Classification for Dependent Hypotheses and Depedent Measurements", *IEEE Trans. on Systems Science and Cybernetics*, vol.5, pp. 109-115.

[6] R.A. Jacobs, M.I. Jordan, S.J. Nowlan and G.E. Hinton. 1991. "Adaptive mixtures of local experts", *Neural Computation*, vol.3, pp.79-87.

[7] M.I. Jordan and R.A. Jacobs. 1994. "Hierarchical Mixtures of Experts and the EM Algorithm", *Neural Computation*, vol.6, pp. 181-214.

[8] M.I. Jordan and L. Xu. 1995. "Convergence Results for the EM Algorithm to Mixtures of Experts Architectures", *Neural Networks*, vol.8, pp. 1409-1431.

[9] A. Kehagias and V. Petridis. 1997. "Predictive Modular Neural Networks for Time Series Classification", *Neural Networks*, vol.10, pp.31-49.

[10] P. Kenny, M. Lennig ands P. Mermelstein. 1990. "A linear predictive HMM for vector-valued observations with applications to speech recognition", *IEEE Trans. on Acoustics, Speech and Signal Proc.*, vol.38, pp.220-225.

[11] T. Kohonen. 1988. *Self Organization and Associative Memory*, Springer, New York.

[12] D.G. Lainiotis. 1971. "Optimal adaptive estimation: structure and parameter adaptation", *IEEE Trans. on Automatic Control*, vol.16, pp. 160-170.

[13] V. Petridis and A. Kehagias. 1996a. "A Recurrent Network Implementation of Time Series Classification", *Neural Computation*, vol.8, pp.357-372.

[14] V. Petridis and A. Kehagias. 1996b. "Modular Neural Networks for MAP Classification of Time Series and the Partition Algorithm", *IEEE Trans. on Neural Networks*, vol.7, pp.73-86.

[15] K. Pawelzik, J. Kohlmorgen and K.R. Muller. 1996. "Annealed Competition of Experts for a Segmentation and Classification of Switching Dynamics", *Neural Computation*, vol.8, pp.340-356.

[16] K. Prank et al. 1996. "Self-organized quantification of hormone pulsatility based on predictive neural networks: Separating the secretory dynamics of growth hormone in acromegaly from normal controls", *Biophysical Journal*, Vol.70, pp.2540-2547.

[17] L.R. Rabiner. 1988. "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", *Proc. IEEE*, vol. 77, pp.257-286.

[18] F.L. Sims, D.G. Lainiotis and D.T. Magill. 1969. "Recursive algorithm for the calculation of the adaptive Kalman filter coefficients", *IEEE Trans. on Automatic Control*, vol.14, pp.215-218.

[19] L. Xu and M.I. Jordan. 1996. "On Convergence Properties of the EM Algorithm for Gaussian Mixtures", *Neural Computation*, vol.8, pp. 129-151.

[20] L.H. Zetterberg et al. 1981. "Computer analysis of EEG signals with parametric models", Proc. IEEE, vol. 69, pp.451-461.

Fig.1

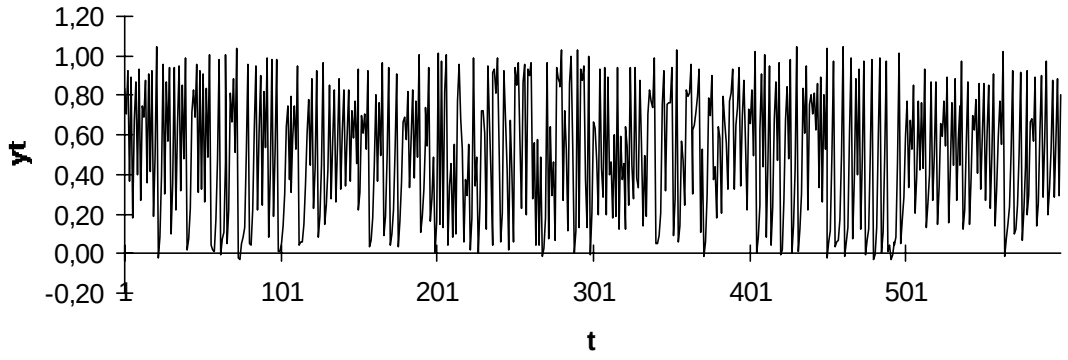


Fig.2A

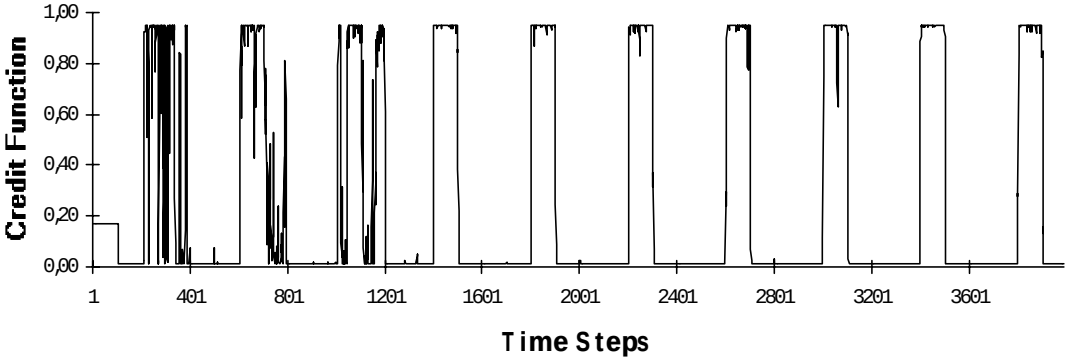


Fig.2B

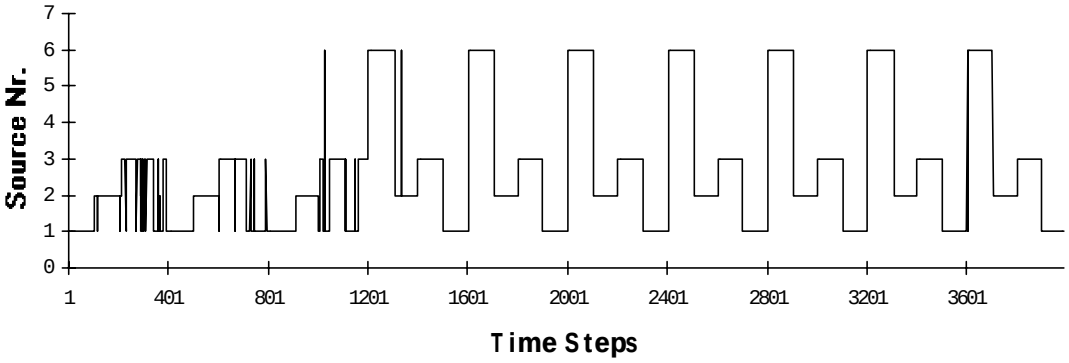


Fig.3

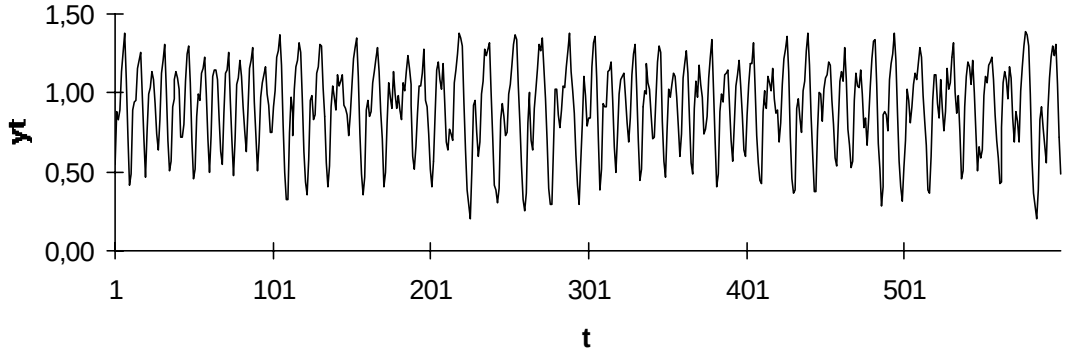


Fig.4A

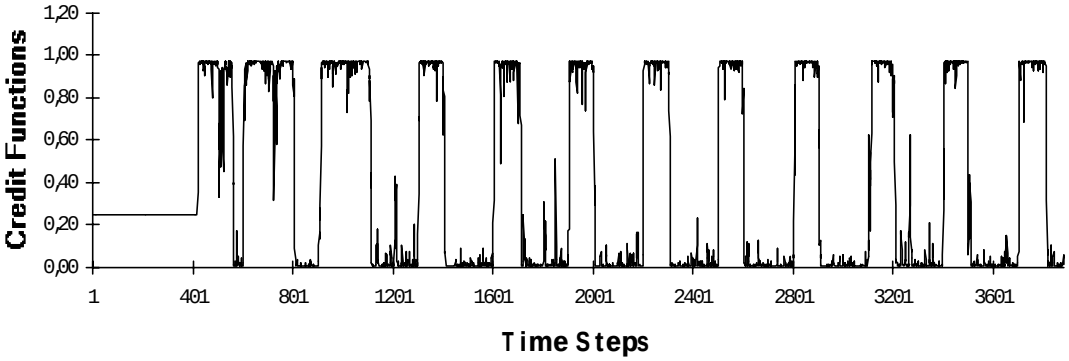


Fig.4B

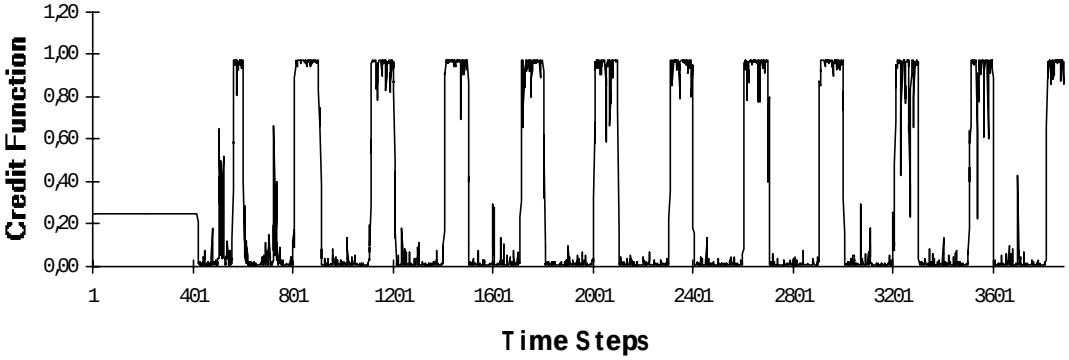


Fig.4C

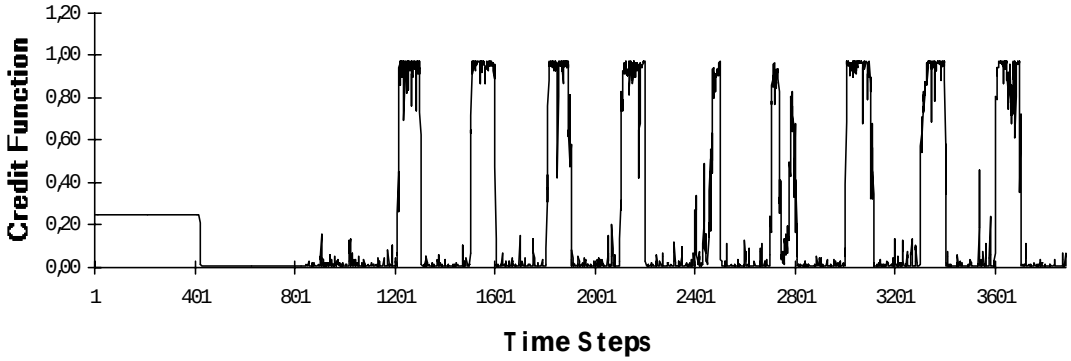
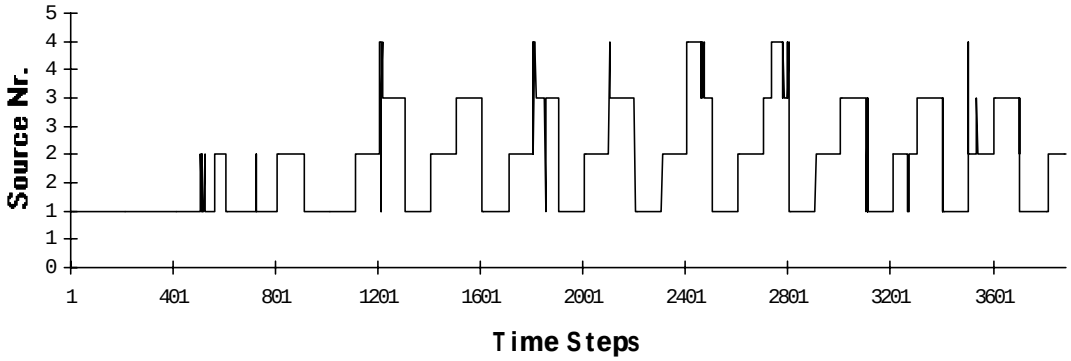

Credit Function
1,20
1,00
0,80
0,60
0,40
0,20
0,00
1
401
801
1201
1601
2001
2401
2801
3201
3601
Time Steps

Fig.4D


Source Nr.
5
4
4
3
3
2
2
1
1
0
1
401
801
1201
1601
2001
2401
2801
3201
3601
Time Steps