Exploiting Multi-Modality for Segmentation and Modeling of Non-Stationary Time Series

Craig L. Fancourt, Jose C. Principe

Computational NeuroEngineering Laboratory
University Florida

1 Introduction

1.1 Overview

In recent years, there has been a growing interest in fields as diverse as statistics, engineering, biology, geology, and economics to soften the requirement of stationary required by the traditional stochastic modeling of time series (Box and Jenkins, 1976). It seems that some systems, rather than exhibiting a slow evolution, may exist in stable regimes for long periods of time and then, suddenly, exhibit a rapid, abrupt change to another regime. The goal of this chapter is to examine a new class of algorithms for detecting and characterizing abrupt changes in time series.

According to Klein's (1997) work on the history of time series, Darwin and Wallace's natural selection theory (1858) spawned some of the first mathematical investigations of non-stationarity, long before Khinchin (1932) and Wold (1938) defined it in modern statistical terms. Weldon (1890) observed that the distribution of the breadth of the forehead of Naples crabs formed a double peak and he, as well as Wallace, Pearson, and Galton, hypothesized that the presence of such bi-modal distributions might be evidence of a bifurcation occurring over time. Although these measurements were made at a single moment in time, their problem bears a resemblance to the central problem considered in this paper. Given a series of measurements over time that, in the aggregate, are observed to be multi-modal, when does multi-modality imply non-stationarity, or more precisely, piecewise stationarity?

As a simple example, consider a time series that switches between two memoryless Gaussian random processes with unit variance but with means of one and minus one, respectively. Figure 1(a) shows a time series where the processes switch every fifty samples; this process is non-stationary but locally piecewise stationary. Figure 1(b) shows a time series produced by random switching with equal probability at every time step; this process is strictly stationary. And yet, both of these time series have identical multi-modal distributions, as shown in the histogram of Figure 1(c), and thus can be fit to the exact same Gaussian mixture model using a standard method such as the Expectation-Maximization algorithm (Dempster, 1972). However, if this mixture model is passed to a properly tuned segmentation algorithm, the first time series can be recognized as piecewise stationary while the second recognized as stationary but multi-modal. As we will see, the case of switching processes with memory must be analyzed differently, using a mixture of predictors to decompose the multi-variate process density function, but the net result is fundamentally the same.

The two fundamental issues in modeling and segmentation of time series are identification of a stationary segment and detection of a statistically significant change. We focus on a particular class of algorithms called mixture models that can model and segment piecewise stationary time series in a completely unsupervised fashion. They use an adaptive mixture of adaptive models that

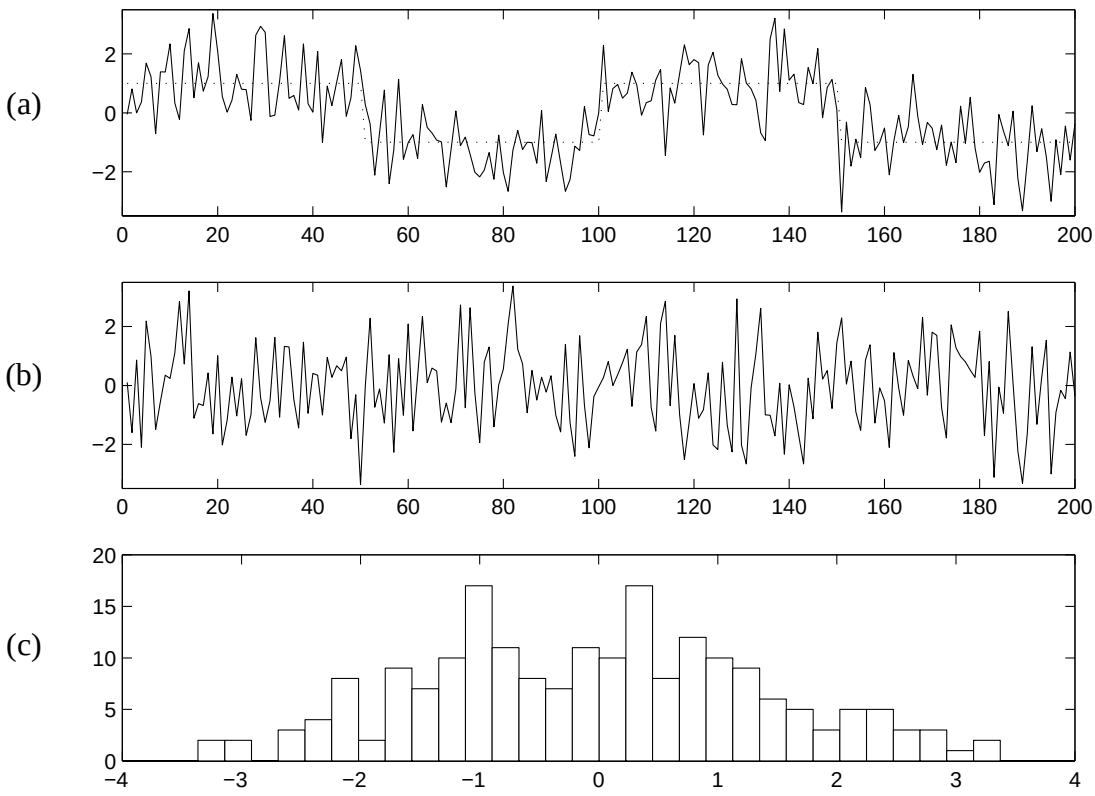

Figure 1. Multi-modal time series: (a) regime switching every fifty samples; (b) random regime switching; (d) histogram representing either time series.

are local in signal space. The models have short-term memory sufficient for local modeling but no long-term memory sufficient to recognize piecewise stationarity. That is, given a piecewise stationary time series, these mixture models will learn equally well irrespective of any random permutation of the stationary segments. This invariance allows the modeling to be done independently of the segmentation. Once trained, the mixture models are analyzed with supervised segmentation algorithms to detect changepoints and identify the models within the time series. The segmentation information can in turn be used to refine the models.

We need to give more precise definitions of some of the terms used, but first it is instructive to consider some real data that can be regarded as at least piecewise quasi-stationary. At a small enough time scale (5-100 msec) speech can be regarded as a sequence of quasi-stationary segments. These segments can be considered quasi-stationary because, although slowly evolving, they have been successfully modeled by simple linear auto-regressive (AR) systems with constant parameters, driven either by a pulse train, in the case of voiced speech, or noise, in the case of unvoiced speech. The goal of the analysis is the identification of the stationary regime. In the case of speech recognition, identification of the quasi-stationary segments can be passed on to a higher level processor for word or sub-word recognition.

There are many other applications of identification and segmentation of signals. In monitoring any process, such as the operation of a jet or car engine, or a machine tool, there may be several stationary regimes, some of which represent in-process states, and others which represent out-of-process states. The detection of a change in stationarity can be a possible indication of an out-of-

process state and identification of the new regime can help determine this, and whether additional steps need to be taken. In the area of control, a plant may have several stationary regimes, each of which requires a specially tuned controller (Narendra and Balakrishnan, 1997). Detection of a change in the plant and identification of the new regime can be used to switch to the appropriate controller.

1.2 Contrast with other approaches

The mixture model approach has several advantages. By isolating the modeling from the segmentation, it doesn't require global search techniques for potential changepoints and thus it is computationally tractable with non-linear models. By considering the entire time series simultaneously, the mixture model is capable of recognizing recurring regimes without the expense of modeling the transitions between those regimes. Moreover, it makes few assumption about the data and thus treat changepoints as deterministic events.

There are several other approaches in time series analysis to deal with the issues of modeling and segmentation. Hidden Markov models (HMM) (Baum, 1972) are probably the most widely utilized method today for modeling time series whose statistical properties evolve over time, as evidenced by their wide use in speech recognition (Rabiner and Juang, 1993). They postulate a set of hidden states, representing stationary regimes, to each of which is associated a probability density function (pdf) and a matrix of transition probabilities between the states. Thus, HMM's treat changepoints as implicitly stochastic. Generally, HMM's model stochastic "feature vectors" that describe a locally stationary block of data. Thus, the precision with which they can segment is limited by the size of the analysis window. For these reasons, HMM's require large amounts of data to train. In fact, to segment a single realization of a time series, each regime must be visited multiple times in order to reliably estimate the state pdf's and the transition probabilities. We will not further discuss this approach.

Another class of models proposes a hidden *deterministic* state which is determined by the value of a latent variable that is itself a function of a time series (not necessarily the same one that is being modeled). The simplest example of this is Tong's (1980) threshold auto-regressive (TAR) model. These techniques are capable of modeling very fast switching between regimes at the expense of requiring global search methods for fitting the model to data. Such global searches make non-linear modeling computationally expensive. There are also hybrid algorithms that combine these ideas with HMM's (Goldfeld and Quandt, 1973).

Finally, there is the generalized likelihood ratio (GLR) approach, which is a sequential change detection algorithm that does a global search for potential changepoints in an increasing window of time. Being sequential in nature, the algorithm essentially discards older modeling information and cannot readily recognize recurring regimes. Although the algorithm can theoretically accommodate non-linear models, it is computationally prohibitive to do so. We will describe this algorithm in greater detail in the second section to help us introduce in a graded fashion the two fundamental problems of modeling and segmentation. Moreover, it is a benchmark against which all other segmentation algorithms should be judged.

1.3 Contrast with other problems

The segmentation problem differs from the so-called blind source separation problem. There, the problem is one of unmixing multiple signals that have been *simultaneously* mixed to produce a new signal. If the component signals are Gaussian random processes, the difference between these two problems is significant. If Gaussian processes are simultaneously added, the resulting

process will also be Gaussian, and it is impossible to unmix the signals. Thus, the solubility of the blind source separation problem is related to the degree of non-Gaussianity of the signals. As we shall see, this is not the case for the segmentation problem. If several Gaussian processes are switched, the overall process will be multi-modal Gaussian and the segmentation problem is still soluble.

The segmentation problem also differs from the one of estimating the parameters of a single system, usually linear, driven by stationary but multi-modal noise (see Shawn et al., 1998). In this case the presence of multi-modality does not imply non-stationarity.

1.4 Outline

In the remainder of this section, we will define some of the terms and concepts laid out in the goal. In section two we review the classical algorithms for supervised segmentation when models are known apriori. This assumption solves one of the fundamental issues (the identification) and allows us to treat the case of change detection in a simplified setting. We will later marry these supervised algorithms with unsupervised mixture modeling in sections four and five. In section three we review unsupervised sequential approaches to segmentation and modeling for the case when little or nothing is known of the data. In sections four and five, we show how unsupervised mixture modeling can be combined with the classical supervised segmentation algorithms to produce a new hybrid algorithm.

For the remainder of the chapter we delve into experimental algorithms that are no longer likelihood based. They are derived by analogy with the working principles of the mixture model. They possess both short term memory sufficient for modeling and long term memory sufficient for segmentation, and thus attempt to segment and model concurrently during training. These algorithms employ a long term memory control that is either annealed or adapted during training to effect the granularity of segmentation. In section six, we generalize to a framework that we call unsupervised gated competitive systems, under which fall all the new algorithms presented in this chapter, as well as some recent ones in the literature. In section seven we develop gated competitive principal component analysis. In section eight we look at various ways of adding memory to gated competitive systems for time series segmentation without recourse to the classical supervised segmentation algorithms. In section nine we examine novel gate structures. Finally, in section ten we will draw conclusions on the present work and suggest possible lines of inquiry for future work.

1.5 Stationarity

Khinchin (1932) first defined a stationary random process and Wold (1938) extended the definition to discrete time processes. Roughly speaking, a stationary random process is one whose statistics are constant with absolute time. In particular, its distributions, and hence its moments, are invariant with respect to the time the random process began, or how long it has been “running”, assuming all transients have died out. Such a definition is very broad, and so let us break it down into degrees of stationarity. This partitioning is very important, for later on we will find that there is no single test for stationarity, but rather multiple tests which correspond to the various degrees of stationary.

Let us first review the concept of a discrete time random process. Imagine sampling some property of a dynamical system over a period of time, so that we have a series of measurements, $x(n)$. Now, imagine that somehow we can reset the dynamical system and repeat the measurements. If the results are different, then the process is random (or random with a deterministic com-

ponent). If we repeat the experiment an infinite number of times, we can find the first order probability density function (pdf) of the signal at each sampled time instant, $p(x(n))\ \forall n$. Now, $x(n)$ can be regarded as a random variable. A *first order stationary process* is one whose first order pdf is the same for all given sampled times: $p(x(m)) = p(x(n))\ \forall m,n$. Note that the term "first order stationarity" refers to the number of random variables only; if the pdf's are the same, all higher order moments must also be the same: $E[x^a(m)] = E[x^a(m)],\ \forall m, n$.

First order stationarity does not, however, say anything about relationships across time. For that we need the second order or joint pdf between any two sampled time instants, $p(x(m),x(n))$, $\forall m,n$. A *second order stationary process* is one where the joint pdf between two sampled time instants depends only on the time difference between them. That is, the joint pdf remains the same for all constant time differences: $p(x(m),x(n)) = p(x(i),x(j))$ when $m - n = i - j$. Second order stationarity implies that all joint moments will only be a function of the time difference: $E[x^a(m)x^b(n)] = f(m-n, a, b),\ \forall m, n$. The most common joint moment is the autocorrelation: $r(m, n) = E[x(m)x(n)]$.

Strict stationarity requires that all higher order pdf's be invariant to absolute time. However, estimating multivariate pdf's from data is subject to the so-called dimensionality explosion, requiring an exponentially increasing amount of data with each increase in pdf order, thus rendering impractical higher order stationarity tests. In addition, our interest often lies in the moments themselves which can be estimated directly from the data, thus bypassing the problem of estimating the pdf's. For these reasons, a lesser requirement known as *wide sense stationarity* is often employed, for which it is only necessary that the first moments of the first and second order pdf's be independent of absolute time: $E[x(m)] = E[x(n)]$ and $E[x(m)x(n)] = r(m-n),\ \forall m, n$. Note that the latter equation implies that the second moment of the first order pdf will also be independent of time: $E[x^2(m)] = E[x^2(n)],\ \forall m, n$. This fact has important implications for designing stationarity tests. In particular, tests for a change in autocorrelation, or spectrum, are also sensitive to changes in power, and must be countered by explicit design considerations.

Finally, because Gaussian processes are completely characterized by their first and second moments, a wide sense stationary Gaussian process is also strict stationary.

1.6 Ergodicity

There is another, more fundamental problem with these definitions of stationarity. They often cannot be tested in practice because many physical dynamical systems are uncontrolled free-running systems which cannot be reset, and thus there exists only one realization of the random process. We then have to hope that the time average statistics of the single realization are equivalent to the process average statistics, in which case the process is called *ergodic*. Note that an ergodic process is stationary but a stationary process is not necessarily ergodic. As for stationarity, there are also different degrees of ergodicity, but we shall not elaborate. We will always assume ergodicity of a random process throughout the following development.

1.7 Piecewise Stationarity

Having defined stationarity, we now define piecewise stationarity. Generally, there are two frameworks that describe piecewise stationary signals. Common to both are a generic state space representation of a dynamic model described by a parameter set, θ, and an internal state $\boldsymbol{x}$. The

model is driven by an excitation source, $u(n)$. A change in either the parameter set or the excitation source may create a new stationary regime.

The first framework is called the parameter switching framework and is diagrammed in Figure 2 (a). At the transition time, one parameter set or excitation is simply replaced by another. The second framework is called the external switching framework and is diagrammed in Figure 2 (b). At a well defined transition time, the switch connects to a different dynamic model, producing a new stationary regime.

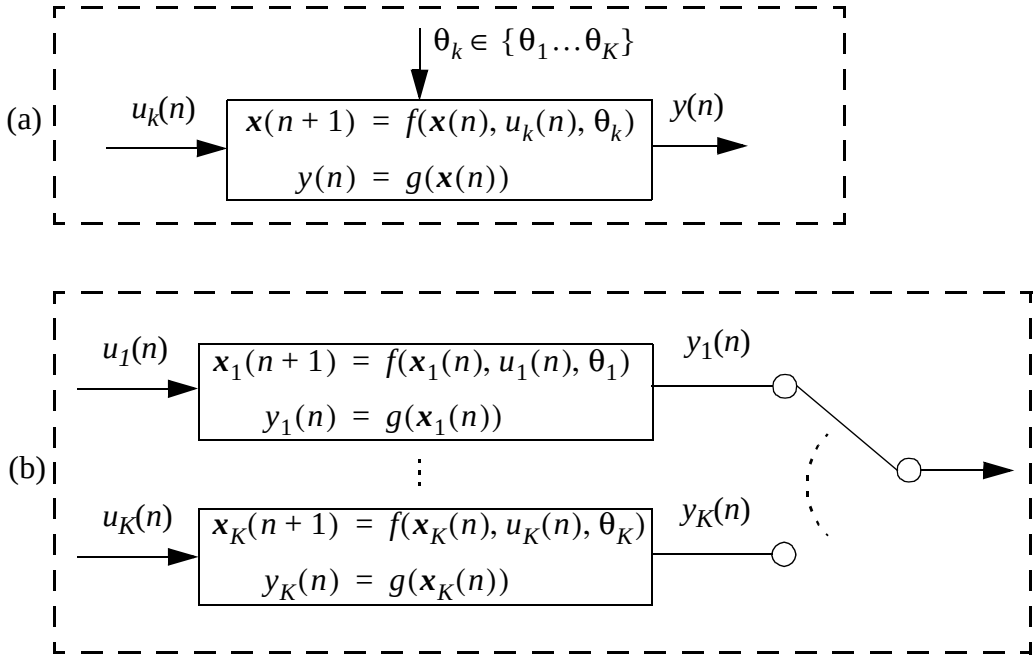

Figure 2. Piecewise stationary models: (a) parameter switching; (b) external switching.

The primary difference between these two models are the transient effects. If the dynamic model has memory, then the parameter switching model will exhibit transient effects for the effective memory depth of the system. The external switching model, on the other hand, does not exhibit transient effects. Clearly the external switching model is an idealization but it has the added benefit of allowing for models of differing complexity.

There are differences in the way these two production models are treated statistically but the differences are slight in practice. Nevertheless, we will primarily concern ourselves with the external switching model in this chapter.

1.8 Quasi-stationarity

There is another class of signals that are called quasi-stationary signals. In these signals the transition between regimes occurs gradually rather than instantaneously. For example, in a transition between two voiced or unvoiced speech segments the excitation remains nearly the same, but the shape of the vocal tract changes, resulting in a continual change in the parameter set of the corresponding dynamic model.

The external switching model can be altered to produce a quasi-stationary signal by replacing the switch with a gradually changing mixture during the transition period. The parameter switch-

ing model can be altered to allow the dynamic model parameters to gradually evolve over time from one set to another. In fact, an adiabatic transition between different parameter sets will generally exhibit less severe transient effects than a quantum switch. Kohlmorgen et al. (1997) have developed an algorithm for detecting drifts between regimes. We will not examine this algorithm, but rather their simpler ACE algorithm on which it is based.

1.9 Approaches to piecewise stationary signals: tracking versus dividing

The question now becomes, given our goal of segmenting and modeling piecewise stationary signals, how can this be accomplished? When confronted with a non-stationary signal, the standard approach is to employ a single adaptive system. An adaptive system has continually adjusting parameters that respond to the local (in time) properties of the signal. In order to be local in time, an adaptive system must have a finite effective memory. That is, it must eventually forget what it has learned in the past. The effective memory of an adaptive system may be governed by a learning rate, for example, in the case of the least means square (LMS) algorithm, or a "forgetting factor", in the case of the exponentially weighted recursive least squares (EWRLS) algorithm. However, this finite memory creates a classic trade-off between the speed of convergence and the misadjustment after convergence. Misadjustment describes the variance of the parameters about their nominal solution. In terms of the segmentation and modeling problem, this translates into a trade-off between the accuracy of the modeling and the delay in detecting a new stationary regime. If the effective memory is too short, the parameters exhibit a large variance about the mean, which can even masquerade as a transition. If the effective memory is too long, transition times become uncertain or worse, short stationary regions may be missed entirely.

There is, however, a more fundamental limitation in using a single adaptive system to monitor a non-stationary signal. The normal procedure for detecting a change involves monitoring the innovations of the adaptive system for deviation from whiteness, or a change in variance, the innovations being the part of the signal that cannot be explained by the model. Unfortunately, it can be shown that the sequence of innovations of a single system is an *insufficient statistic* (see Basseville and Nikiforov, 1993). That is, there is a many to one mapping from different stationary regimes to identical statistics of the innovations. For this reason, two or more adaptive models are required, and this forms the basis of the sequential segmentation algorithms.

These algorithms are called sequential because they only detect a *change* between two stationary but statistically different processes, and information from prior stationary regions is discarded. That is, they are local in both time and model. Under the production models of Figure 2, such methods are appropriate whenever the potential number of dynamical models is very large, so that there is a small probability of returning to the same model. In this case, it is appropriate to view the data as a sequence of stationary *regions*, and the segmentation problem becomes one of detecting the *transition* between two stationary regions.

There is another, fundamentally different, approach to dealing with piecewise stationarity. Instead of viewing the data as a sequence of stationary *regions*, we model the data under the assumption that it has been produced by switching among a finite set of stationary *regimes*. This approach is global in time and local in model but is only feasible when the number of potential regimes is reasonably small.

Modeling the regime instead of the region has a number of advantages. First and foremost, if several discontiguous regions belong to the same regime, information from all these regions can be used to improve the model estimate. Second, a segmentation system that models the regimes

can generalize to new piecewise stationary processes as long as they are generated by the same switching model.

The development of various algorithms that segment a signal by modeling the regimes forms the fundamental topic of this chapter.

2 Supervised Sequential Change Detection

Although we are interested in unsupervised segmentation algorithms, it is important to first understand sequential change detection among a finite set of *known* random processes. The task consists of two steps: first, the active process is identified, and then it is monitored for a possible change to a different process. The first step is accomplished by a generalization of Wald's (1947) sequential probability ratio test (SPRT), which was extended to the multi-hypothesis case by Armitage (1950). The second step is a generalization of Page's (1954) CUSUM test, which was extended to the multi-hypothesis case by Zhang (1989) and later Keresteciogl u (1993). Although the SPRT only serves as a seed for starting the CUSUM algorithm, which is then used to segment the entire time series, the SPRT is easier to understand and many of the principles carry forward to the CUSUM.

2.1 Multi-hypothesis SPRT

Let $X(n) \equiv \left[x(n)\ x(n-1)\ \ldots\ x(1)\right]^T$ represent a vector of a single realization (measurement) of a one-dimensional random process over n discrete time steps. We wish to assign $X(n)$ as belonging to one of K known random processes. A random process is completely characterized by its multivariate distribution across time. Therefore, let H_i be the hypothesis that the sequence belongs to the multivariate distribution $p_i(X)$, i=1...K. For the moment, we assume that there are no costs associated with incorrect decisions and that no prior information is available about the relative occurrence of the various regimes. Let i^* be the true process and let $\tilde{i^*}$ be our estimate of the true process. If we *must* make a decision within n_{max} time steps, then the best decision is to choose the hypothesis $H_{\tilde{i^*}}$ based on the most likely sequence at time n_{max}

$$\tilde{i^*} = \text{argmax}_i[p_i(X(n_{max}))] \tag{1}$$

If the goal is fast detection, at the expense of false alarms, then the multi-hypothesis sequential probability ratio test (SPRT) is appropriate. The log-likelihood ratio between any two candidate pdf's evaluated at the measured sequence is

$$L_{ij}(n) = \log\left[\frac{p_i(X(n))}{p_j(X(n))}\right] \tag{2}$$

where $L_{ii}(n)$ is trivially zero. As we will show later, if i=i^* is the true process, from information theory we know that $L_{i^*j}(n)$ will increase with time for all $j \neq i^*$. The multi-hypothesis SPRT exploits this through a set of parallel Neyman-Pearson tests for deciding among the K hypotheses.

That is, a hypothesis $H_{\tilde{i}^*}$ is chosen if there exists an index $\tilde{i}^* = i$ for which $L_{ij}(n)$, called the decision function in this context, is greater than a set of thresholds

$$L_{ij}(n) \quad \underset{\text{monitor}}{\overset{H_{\tilde{i}^* = i}}{\geq}} \quad \beta_{ij} \quad \forall j \neq i \,. \tag{3}$$

The notation in (3) is such that if the test passes, the upper action is taken and testing stops, otherwise the lower action is taken, which in this case is continued monitoring. The detection thresholds, β_{ij}, are positive constants.

Although there are $K(K\text{-}1)$ separate likelihood ratios to evaluate, only K of them are actually independent. It follows directly from (2) that $L_{ji} = -L_{ij}$. In addition, if L_{k1} is known for all k, the others can be evaluated from

$$L_{ij}(n) = L_{i1}(n) - L_{j1}(n) \,. \tag{4}$$

This can be exploited to simplify the test in (3) by defining a modified decision function, L_i,

$$L_i(n) = \sum_{j=0}^{K-1} L_{ij}(n) \,. \tag{5}$$

and then choosing the hypothesis $H_{\tilde{i}^*}$ if *any one* of the L_i exceed a single predefined threshold

$$L_i(n) \quad \underset{\text{monitor}}{\overset{H_{\tilde{i}^* = i}}{\geq}} \quad \beta \quad \exists i \tag{6}$$

or, if the test fails, no decision is made and monitoring continues. To show that this test works, first note that it can easily be shown that

$$L_j(n) = L_i(n) - K \cdot L_{ij}(n) \,. \tag{7}$$

Now, let $i=i^*$ be the true process. Then (7) becomes

$$L_j(n) = L_{i^*}(n) - K \cdot L_{i^*j}(n) \tag{8}$$

Again, information theory tells us that, on average, $L_{i^*}(n)$ and $L_{i^*j}(n)$ will both increase with time so that $L_j(n) < L_{i^*}(n)$ and thus L_{i^*} should be the first to exceed the threshold. To the best of our knowledge, this modified multi-hypothesis SPRT appears here for the first time.

Kehagias & Petridis (1994) and independently Plataniotis et al. (1996) have proposed using the normalized cumulative likelihood to directly indicate the probability of the various models at each instant of time, and provide a way of calculating these probabilities recursively using Bayes' rule. While such an approach is valid, they apparently never recognized the simple connection with the SPRT (the log of the ratio of the absolute probability in their approach is simply the log-

likelihood ratio in (2)) and thus they don't provide a stopping rule. By not testing for a change in regime they never reset the decision function, and thus the cumulative likelihood decreases without bound, causing numerical precision problems and long detection delay. The CUSUM algorithm we are about to describe continually resets the cumulative log-likelihood as long as a test indicates no change in the current process.

2.2 Multi-hypothesis CUSUM test

Having established the active process, the next step is to detect a *change* to a different random process. Assume that the current active process has been correctly identified as belonging to $p_{i^*}(X)$, the data is arriving one sample at a time, and we wish to determine if and when a change occurs to $p_{j^*}(X)$. We know that under i^*, the process likelihood $L_{i^*j}(n)$ will increase with time. However, for historical reasons, the CUSUM algorithm tracks $L_{ji^*} = -L_{i^*j}$, which decreases with time. The main idea behind the CUSUM algorithm is to continually reset the log-likelihood for the active process relative to its minimum value

$$\Delta L_{ji^*}(n) = L_{ji^*}(n) - \min[L_{ji^*}(n-1), L_{ji^*}(n-2), \ldots L_{ji^*}(1)] \qquad \forall j \neq i^* \tag{9}$$

which can also be written recursively

$$\Delta L_{ji^*}(n) = \max[0, \Delta L_{ji^*}(n-1) + l_{ji^*}(n)] \qquad \forall j \neq i^* . \tag{10}$$

where l_{ji^*} is the log-likelihood increment, which we shall discuss in the next section. The other values are determined relative to the active process, similar to (4)

$$\Delta L_{jk}(n) = \Delta L_{ji^*}(n) - \Delta L_{ki^*}(n) . \tag{11}$$

Under i^*, both $\Delta L_{ji^*}(n)$ and $\Delta L_{ji^*}(n)$ will be repeatedly reset to zero, causing $\Delta L_{jk}(n)$ to fluctuate around zero. If a change occurs to j^*, then $\Delta L_{j^*k}(n)$ will tend to increase. This forms the basis for a test that can be used to decide if a change has occurred, and if so, the new active process.

$$\Delta L_{jk}(n) \underset{\text{monitor}}{\overset{H_{\tilde{j}^* = j}}{\geq}} \beta_{jk} \quad \forall k \neq j \tag{12}$$

If the test fails, monitoring continues. Once a detection occurs, the actual changepoint can be estimated by choosing the time at which $L_{i^*j^*}$ was maximum

$$T = \arg\max_n [L_{i^*j^*}(n)] . \tag{13}$$

The CUSUM algorithm is then reset and started anew from the changepoint.

Just as for the SPRT, we prefer a simplified decision function

$$\Delta L_j(n) = \sum_{k=0}^{K-1} \Delta L_{jk}(n) \tag{14}$$

that has only a single threshold, such that if any one of the ΔL_j exceed the threshold,

$$\Delta L_j(n) \underset{j^* = j}{\overset{H}{\gtrless}} \beta \quad \exists j \tag{15}$$

monitor

the test stops and a decision is made in favor of a change.

Keresteciogl u (1993) has proposed a multi-hypothesis CUSUM test where the process log-likelihood ratio differences in (10) are reset to zero all together or not at all.

To summarize, in order to segment a time series using a set of candidate known random processes, the initial active process at the beginning of the time series is identified using the multi-hypothesis SPRT. Then, the time series is monitored for change to a different process using the multi-hypothesis CUSUM algorithm. Once a change is detected, the CUSUM algorithm is reset and monitoring continues anew.

2.3 Recursive estimation of the likelihood ratio

It would clearly be advantageous to calculate the log-likelihood ratio recursively instead of having to know multivariate distributions that increase in dimensionality at every time step. This is easily accomplished for memoryless random processes, more commonly known as independent identically distributed (i.i.d.) sequences. Then, a multivariate pdf factors into a product of identical univariate pdf's

$$p(X(n)) = \prod_{m=1}^{n} p(x(m)). \tag{16}$$

Applying this to the multi-hypotheses case, the log likelihood ratio can be written

$$L_{ij}(n) = \sum_{m=1}^{n} l_{ij}(m) \tag{17}$$

where $l_{ij}(m)$ is the instantaneous log-likelihood, often called the log-likelihood increment

$$l_{ij}(n) = \log\left[\frac{p_i(x(n))}{p_j(x(n))}\right]. \tag{18}$$

The process log-likelihood can now be calculated recursively

$$L_{ij}(n) = L_{ij}(n-1) + l_{ij}(n). \tag{19}$$

If the processes are not memoryless, we can use a result from conditional probability theory to simplify the likelihood of a sequence,

$$p(X(n)) \;=\; p(x(n), X(n-1)) \;=\; p(x(n)|X(n-1)) \cdot p(X(n-1)) \tag{20}$$

and then through recursion

$$p(X(n)) \;=\; \prod_{m\,=\,1}^{n} p(x(m)|X(m-1)) \tag{21}$$

where it is understood that $p(x(1)|X(0)) \equiv p(x(1))$. Applying this to the multi-hypothesis case, the process log-likelihood ratio is again given by (17), but the log-likelihood increment is now given by

$$l_{ij}(n) \;=\; \log\left[\frac{p_i(x(n)|X(n-1))}{p_j(x(n)|X(n-1))}\right] \tag{22}$$

We can use prediction as a tool to evaluate this likelihood ratio. If we can find models under each hypothesis that can predict $x(n)$ from prior data, up to a random term, $\varepsilon_i(n)$,

$$x(n) \;=\; f_i(X(n-1)) + \varepsilon_i(n) \qquad i \;=\; 1 \ldots K \tag{23}$$

then we can evaluate the conditional probability in terms of the pdf of the innovations, $\varepsilon_i(n)$,

$$p_i(x(n)|X(n-1)) \;=\; p_{\varepsilon_i}(\varepsilon_i(n)) \qquad i \;=\; 1 \ldots K. \tag{24}$$

and the log-likelihood increment becomes

$$l_{ij}(n) \;=\; \log\left[\frac{p_{\varepsilon_i}(\varepsilon_i(n))}{p_{\varepsilon_j}(\varepsilon_j(n))}\right]. \tag{25}$$

However, only if the innovations are i.i.d. can the process log-likelihood ratio be written

$$L_{ij}(n) \approx \sum_{m\,=\,1}^{n} l_{ij}(m)\,. \tag{26}$$

Equation (26) is written as an approximation because we have ignored initial conditions, and for additional reasons that are discussed below. Once again, this achieves the goal of recursive computation of the process log-likelihood exactly as in (19).

This expression of the likelihood of a sequence in terms of prediction error is known as the prediction error decomposition. In practice, however, the entire past sequence is not needed to predict the next sample. For example, for an AR(p) process, only p past samples are required to predict the next sample and any additional history does not improve the prediction. In terms of the statistics of the random process, the conditional distribution of the random process given its past becomes independent of its distant past. If the *effective memory depth* is M for a given process, then we can say

$$p(x(n)|X(n-1)) \approx p(x(n)|\chi(n-1))\,, \tag{27}$$

$$x(n) \approx f(\chi(n-1)) + \varepsilon(n) \tag{28}$$

where $\chi(n)$ is a vector of the M most recent samples of the process

$$\chi(n) \equiv [x(n), x(n-1), \ldots, x(n-M+1)]^T \tag{29}$$

in effect the output of a tapped delay line with M taps.

One must be careful in assuming independence of the innovations. For example, the optimal minimum mean square error (MMSE) predictor can be shown to be the conditional expectation

$$f(X(n-1)) \;=\; E[x(n)|X(n-1)] \tag{30}$$

In this case, the most that can be said about the innovations is that they are uncorrelated, $E[\varepsilon(m)\varepsilon(n)] \propto \delta_{m,n}$. If they were not, then one could find a better model that would exploit the correlations and improve the prediction. However, this does not tell us whether the innovations are independent. The exception is if they are Gaussian distributed, in which case uncorrelation implies independence. There is an additional problem in evaluating the process likelihood for predictors that do not represent the current true process, for then the errors are not innovations at all but are the residuals, and are likely to be correlated. In spite of all this, in practice, we will always assume independence of the errors.

2.4 Example: detecting a change in an i.i.d. vector Gaussian process

Let H_i be the hypothesis of an i.i.d. vector Gaussian process with distribution $N(\mu_i, \boldsymbol{R}_i)$. The log-likelihood increment is then

$$l_{ij}(n) \propto [\boldsymbol{x}(n) - \mu_j]^T \boldsymbol{R}_j^{-1} [\boldsymbol{x}(n) - \mu_j] - [\boldsymbol{x}(n) - \mu_i]^T \boldsymbol{R}_i^{-1} [\boldsymbol{x}(n) - \mu_i] + \log[|\boldsymbol{R}_j|] - \log[|\boldsymbol{R}_i|]. \tag{31}$$

The first two terms represent the difference between the Mahalonobis distances of the current sample to the two models.

2.5 Example: detecting a change in a scalar random process using predictive models under the assumption of Gaussian innovations

Let H_i be the hypothesis that the random process can be modeled as $x(n) \;=\; f_i(\chi(n-1)) + \varepsilon_i(n)$, where the residuals are i.i.d. Gaussian $\varepsilon_i \sim N(0, \sigma_i^2)$. The log-likelihood increment is then

$$l_{ij}(n) \propto \frac{\varepsilon_j^2(n)}{\sigma_j^2} - \frac{\varepsilon_i^2(n)}{\sigma_i^2} + \log\left[\frac{\sigma_j^2}{\sigma_i^2}\right] \tag{32}$$

The first two terms represent the weighted difference between the squared innovations of the two predictive models.

2.6 Information theoretic interpretation

As eluded to earlier, there is an information theoretic interpretation to the change detection algorithms. For a memoryless random process, if i^* is the true active process, the expected value of the process log-likelihood ratio, L_{i^*j}, is

$$E_{i^*}[L_{i^*j}(n)] = \sum_{m=1}^{n} E_{i^*}[l_{i^*j}(m)] = nE_{i^*}\left[\log\left[\frac{p_{i^*}(x)}{p_j(x)}\right]\right] = nK(p_{i^*}, p_j) \tag{33}$$

where $K(p_{i^*}, p_j)$ is the Kullback-Leibler mutual information of p_j with respect to p_{i^*}.

$$K(p_{i^*}, p_j) = \int \log\left[\frac{p_{i^*}(x)}{p_j(x)}\right] p_{i^*}(x)dx \geq 0\,. \tag{34}$$

Since the mutual information is always positive, the expected value of the process log-likelihood *prior* to change increases linearly with time at a rate determined by the mutual information $K(p_{i^*}, p_j)$. Likewise, if j^* is the true active process *after* the changepoint, the expected value of the process log-likelihood decreases linearly with time at rate determined by $K(p_i, p_{j^*})$, which is a negative quantity for all i.

This suggests that the *detectability* of a change will be related to the similarity of the processes as measured by the Kullback-Leibler mutual information. By detectability, we mean both the false alarm rate before the changepoint, and the detection delay after the changepoint. Intuitively, one would expect the false alarm rate to be inversely related to the magnitude of the slope of $L_{i^*j}(n)$ before the changepoint. That is, the smaller the mutual information $K(p_{i^*}, p_j)$, the larger the false alarm rate. Likewise, one would expect the detection delay to be inversely related to the slope of $L_{ij^*}(n)$ after the changepoint. In other words, the smaller the mutual information $K(p_i, p_{j^*})$, the longer the detection delay. For the off-line segmentation problem, of which we are primarily concerned here, the detection delay represents the minimum length stationary region that can be detected, since a second change may occur before the first one is detected.

The standard multi-hypothesis SPRT and CUSUM algorithms are affected by the smallest mutual information between the active and non-active processes because of the use of separate thresholds in (3) and (12). For our modified decision functions in (5) and (14), the average slope of the decision function with respect to time is a sum of positive mutual information measures

$$E_{i^*}[L_{i^*}(n)] = n\sum_{j=1}^{K} K(p_{i^*}, p_j) \tag{35}$$

which is dominated by the *largest* mutual information.

Because the Kullback-Leibler information is not necessarily symmetric, detecting a change from process A to process B may be more or less difficult than detecting a change from B to A. For example, in the case of detecting a change in the variance of an i.i.d. process, the non-symmetry of the mutual information results in it being easier to detect an increase in variance compared to a decrease in variance.

This interpretation also applies to random processes with memory, but in this case the appropriate distance measure is the mutual information between two random processes, which is a more difficult quantity to calculate.

In the literature, the average false alarm rate is called the operating characteristic (OC), while the mean detection delay is called the average sample number (ASN). While these have been thoroughly investigated for the two hypothesis case, less is known about the properties of the multi-hypothesis case.

3 Unsupervised Sequential Segmentation

The algorithms of the previous section were based on prior knowledge of the candidate random processes. What if no prior information is available? Is it still possible to segment a single realization of a switching random process? That is, given a single realization of a piecewise ergodic random process, consisting of a finite ordered set of vectors, can we segment the data into contiguous stationary regions? This defines the off-line segmentation problem. We do not know how many stationary regions are contained in the data, nor the change points between regions. Because of this, the problem cannot be easily formulated as a global test between known hypotheses. The only possible approach then is to perform *sequential detection* of changepoints, which makes the off-line segmentation problem similar to the on-line case, reducing the problem to one of detecting changepoints between successive pairs of stationary regions. This problem was first addressed by Page (1957).

Basically, there are two approaches to unsupervised sequential segmentation. In the first approach, the detection algorithm is temporarily "turned off" while a model is developed for the incoming data. It is then "turned on" and the incoming data is monitored for a minimum change from the reference model, using some appropriate distance measure.

The second approach, and the one that we will describe in some detail, is called the Generalized Likelihood Ratio (GLR), first applied to time series by Willsky and Jones (1976) and later Appel and Brandt (1982). It is a Neyman-Pearson test for deciding between two hypotheses: the null hypothesis, H_0, that no change occurred within n samples, and the posited hypothesis, H_1, that a change did occur at the intermediate changepoint time T. It differs from the standard likelihood ratio test previously described in that the pdf's are unknown and must be estimated directly from the data within their respective regions, as defined by the hypothesized changepoint time. In addition, a search must be done for the most likely changepoint within the block of n samples. The algorithm then proceeds analogously as for the supervised case. When the decision function exceeds a preset threshold, a change is detected. The exact transition time is then determined, and the algorithm is reset and started anew.

3.1 Tests for a change in a memoryless process

Consider the problem of detecting a change in the first order pdf of an independently identically distributed (i.i.d.) process We assume that the data both before and after change can be described by a parametric family of pdf's, characterized by the parameter set θ. Since we are interested in an unsupervised algorithm, we do not know the parameters of the pdf before the change point, θ_0, or after the change point, θ_1, nor the changepoint time, T. The likelihood of the null hypothesis, H_0, that no change occurred within n samples is

$$L_{H_0}(n) = \prod_{m=1}^{n} p(x(m);\theta_0)\,. \tag{36}$$

Likewise, the likelihood of the hypothesis H_1 that a change in the pdf occurred at time T is given by

$$L_{H_1}(n) = \left[\prod_{m=1}^{T-1} p(x(m);\theta'_0)\right] \cdot \left[\prod_{m=T}^{n} p(x(m);\theta_1)\right]. \quad (37)$$

Note that we differentiate between the parameter set under the null hypothesis, θ_0, and the parameter set before change under the change hypothesis, θ'_0, since they are estimated over different times. The log-likelihood ratio between the two hypotheses is then

$$L(n) = \log\left[\frac{L_{H_1}(n)}{L_{H_0}(n)}\right] = \log\left[\frac{\prod_{m=1}^{T-1} p(x(m);\theta'_0) \prod_{m=T}^{n} p(x(m);\theta_1)}{\prod_{m=1}^{n} p(x(m);\theta_0)}\right], \quad (38)$$

which can also be written

$$L(n) = \sum_{m=1}^{T-1} \log\left[\frac{p(x(m);\theta'_0)}{p(x(m);\theta_0)}\right] + \sum_{m=T}^{n} \log\left[\frac{p(x(m);\theta_1)}{p(x(m);\theta_0)}\right]. \quad (39)$$

A decision function is then formed by replacing the parameters θ_0, θ'_0, and θ_1, in $L(n)$ by their maximum likelihood (ML) estimates, $\hat{\theta}_0$, $\hat{\theta}'_0$, and $\hat{\theta}_1$, over their respective regions and then maximizing with respect to the change time, T

$$Q(n) = \max_T [L(n)|_{\hat{\theta}_0, \hat{\theta}'_0, \hat{\theta}_1}]. \quad (40)$$

A change is deemed to have occurred within the block of n data if the decision function exceeds some predefined threshold

$$Q(n) \underset{H_0}{\overset{H_1}{\geq}} \beta \quad . \quad (41)$$

From (39) and (40), we see that under H_0, any two blocks of data are equally likely and the decision function will tend to fluctuate near zero, while under H_1 the decision function increases with increasing n.

The algorithm starts from the last detected change point, from which a small block of n points is examined. To find $Q(n)$, the two hypotheses are compared using (39) for all possible changepoints, T, within the block, and the maximum value of $L(n)$ is assigned to $Q(n)$. If $Q(n)$ is less than the threshold, no changepoint is detected, n is incremented, a new sample $x(n)$ is introduced, and the process repeats. If $Q(n)$ is greater than the threshold, the algorithm is reset and started anew. A symbolic diagram of this is shown in Figure 3 (a). Regions where the number of samples is

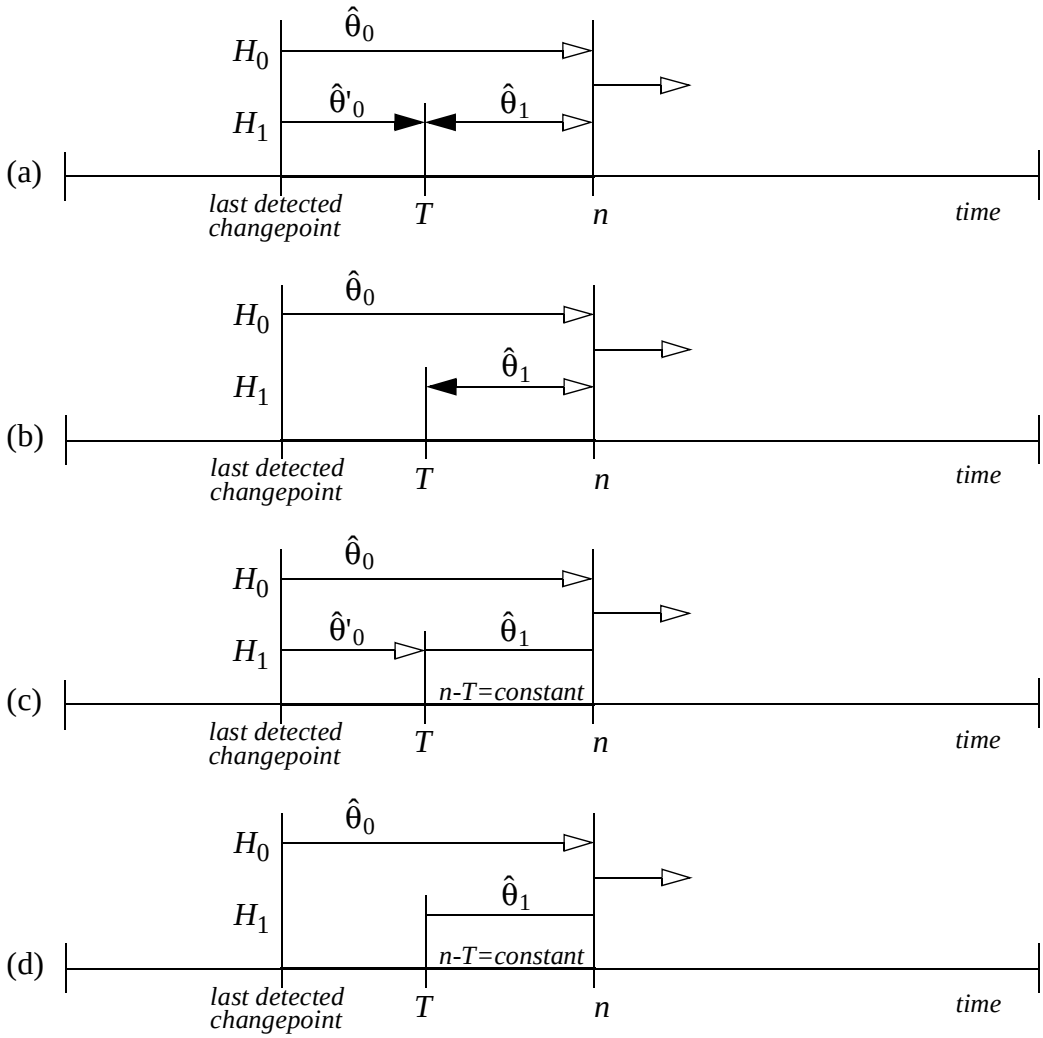

Figure 3. GLR change point detection: (a) three model; (b) two model; (c) three model with fixed length region for θ_1; (d) two model with fixed length region for θ_1.

affected by T are shown with solid arrows, while regions effected by n are shown with hollow arrows.

It is important to note that in order to keep the notation simple, the time index here is relative to the last detected changepoint. This is indicative of the sequential nature of the algorithm and the fact that all acquired knowledge is discarded after the detection of a changepoint.

3.2 Tests for a change in a process with memory

The hypotheses are exactly the same as in the memoryless case, except now we must evaluate the likelihood recursively. Skipping the intermediate steps, there results

$$L(n) = \sum_{m=1}^{T-1} \log\left[\frac{p'_0(x(m)|X(m-1))}{p_0(x(m)|X(m-1))}\right] + \sum_{m=T}^{n} \log\left[\frac{p_1(x(m)|X(m-1))}{p_0(x(m)|X(m-1))}\right] \tag{42}$$

where for convenience, we have temporarily used a non-parametric representation for the pdf's.

One can question why the second term is conditioned on all prior data even though we are testing for a changepoint at the intermediate time *T*. The answer is closely related to the two production models discussed previously. Under the parameter switching model, the system state memory remains intact at the changepoint, and thus the history of the process before the changepoint can influence its future and (42) is appropriate. Under the external switching model, the history of the process before the changepoint cannot influence its future. In this case, the second term in (42) should be conditioned only on the data since the changepoint, *T*. In practice, the difference between these two formulations is slight. This is due to the fact that any random process has a finite effective memory, as previously discussed. In particular, if the maximum effective memory of the process is *M*, then (42) can be written

$$L(n) \approx \sum_{m=1}^{T-1} \log\left[\frac{p'_0(x(m)|\chi(m-1))}{p_0(x(m)|\chi(m-1))}\right] + \sum_{m=T}^{n} \log\left[\frac{p_1(x(m)|\chi(m-1))}{p_0(x(m)|\chi(m-1))}\right]. \tag{43}$$

where it is understood that $p(x(1)|\chi(0)) \equiv p(x(1))$. The differences between the two production models now exists only in a region of *M* samples after the hypothetical changepoint *T*.

Assume we can develop predictors for each of the three regions up to a random term

$$x(n) = f(\chi(n-1);\theta_i) + \varepsilon_i(n) \qquad i = 0, 0', 1 \tag{44}$$

where θ now represents the parameter set of a parametric function. The conditional pdf of $x(n)$ can now be written in terms of the pdf of the prediction errors

$$p_i(x(n)|(\chi(n-1))) = p_{\varepsilon_i}(\varepsilon_i(n)). \tag{45}$$

However, only if the prediction errors are i.i.d. can we rewrite (42) as

$$L(n) \approx \sum_{m=1}^{T-1} \log\left[\frac{p_{\varepsilon'_0}(x(n) - f(\chi(n-1);\theta'_0))}{p_{\varepsilon_0}(x(n) - f(\chi(n-1);\theta_0))}\right] + \sum_{m=T}^{n} \log\left[\frac{p_{\varepsilon_1}(x(n) - f(\chi(n-1);\theta_1))}{p_{\varepsilon_0}(x(n) - f(\chi(n-1);\theta_0))}\right] \tag{46}$$

where the approximation ensues due to our ignoring the details of initial conditions, transitions, and possible non-independence of residuals.

Just as in the memoryless case, the decision function is then formed by replacing the functional parameters, θ_0, θ'_0, and θ_1, in (44) by their maximum likelihood (ML) estimates, $\hat{\theta}_0$, $\hat{\theta}'_0$, and $\hat{\theta}_1$, respectively, over their respective regions and then maximizing with respect to the change time, *T*, exactly as in (40) but repeated here for convenience

$$Q(n) = \max_T[L(n)|_{\hat{\theta}_0, \hat{\theta}'_0, \hat{\theta}_1}]. \tag{47}$$

It is important to note that the pdf's of the prediction errors $p_{\varepsilon_i}(\varepsilon_i(n))$ may contain "nuisance parameters" that themselves may have to be replaced by their maximum likelihood estimates in (47), however they can all be lumped together with the predictor parameters as part of a larger parameter set.

3.3 Simplifications

The GLR algorithm is computationally intensive. For each block of n points, on the order of n+2 new ML estimates must be performed. Extensive use of recursive ML estimates can significantly reduce the computational load but require significant memory. A minor simplification can be made by assuming that the null hypothesis maximum likelihood estimates are approximately equal, $\hat{\theta}'_0 \approx \hat{\theta}_0$, resulting in the two model GLR algorithm shown in Figure 3(b). The first term in (39) is then zero, and for the memoryless case results in

$$L(n) = \sum_{m=T}^{n} \log\left[\frac{p(x(m);\theta_1)}{p(x(m);\theta_0)}\right], \tag{48}$$

while for the memory case

$$L(n) = \sum_{m=T}^{n} \log\left[\frac{p_{\varepsilon_1}(x(m) - f(\chi(m-1);\theta_1))}{p_{\varepsilon_0}(x(m) - f(\chi(m-1);\theta_0))}\right]. \tag{49}$$

Note that $\hat{\theta}_0$ is still estimated over all n samples but the GLR is only evaluated over the last n-T+1 samples. The advantages of estimating $\hat{\theta}_0$ from the entire block of n samples, rather than the first T-1 samples, is that this need only be done once for each evaluation of the decision function in (47) and, using more data, it is also more reliable.

A more dramatic reduction can be achieved by eliminating the maximization over T in (40) by fixing the length of the data segment associated with the parameter set θ_1. This results in the symbolic diagram shown in Figure 3(c). Finally, these two simplifications can be combined, resulting in the symbolic diagram of Figure 3(d). Eliminating the maximization over T also allows for on-line recursive estimation of the parameters.

3.4 Example: testing for a change in the mean of a Gaussian i.i.d. process

As an example, consider the problem of detecting a change in the mean, μ, of a one-dimensional Gaussian process under the assumption of constant variance. Maximization with respect to the parameters in (40) is explicit, resulting in the following decision function

$$Q(n) \propto \max_T[(T-1)(\mu'_0)^2 + (n-T+1)(\mu_1)^2 - n(\mu_0)^2]. \tag{50}$$

Using the relationship between the means, $n\mu_0 = (T-1)\mu'_0 + (n-T+1)\mu_1$, this can be further simplified to

$$Q(n) \propto \max_T\left[\frac{n(n-T+1)}{T-1}(\mu_1 - \mu_0)^2\right]. \tag{51}$$

Note that the relationship between the model parameters allowed reformulation of the three model decision function in terms of two models.

3.5 Example: testing for a change in an AR process under the assumption of constant Gaussian variance.

Recall that if a process is modeled as purely auto-regressive (AR), then the optimal predictor is a finite impulse response (FIR) filter. It can be shown that maximization with respect to the parameters in (40) results in the decision function

$$Q(n) \propto \max_T[(T-1)\boldsymbol{w}'^T_0\boldsymbol{R}'_0\boldsymbol{w}'_0 + (n-T+1)\boldsymbol{w}_1^T\boldsymbol{R}_1\boldsymbol{w}_1 - n\boldsymbol{w}_0^T\boldsymbol{R}_0\boldsymbol{w}_0]. \quad (52)$$

where $\boldsymbol{R}$ is the autocorrelation matrix and $\boldsymbol{w}$ is the vector of optimal FIR predictor coefficients. Using the relationship between the autocorrelations, $n\boldsymbol{R}_0 = (T-1)\boldsymbol{R}'_0 + (n-T+1)\boldsymbol{R}_1$, the decision function can be written

$$Q(n) \propto \max_T[(n-T+1)(\boldsymbol{w}_1^T\boldsymbol{R}_1\boldsymbol{w}_1 - \boldsymbol{w}'^T_0\boldsymbol{R}_1\boldsymbol{w}'_0) - n(\boldsymbol{w}_0^T\boldsymbol{R}_0\boldsymbol{w}_0 - \boldsymbol{w}'^T_0\boldsymbol{R}_0\boldsymbol{w}'_0)]. \quad (53)$$

Unfortunately, there is no such simple relationship between the predictor coefficients. Further simplification can only be made if we use the two model GLR algorithm which assumes that $\boldsymbol{w}'_0 = \boldsymbol{w}_0$, in which case the decision function becomes

$$Q(n) \propto \max_T[(n-T+1)(\boldsymbol{w}_1^T\boldsymbol{R}_1\boldsymbol{w}_1 - \boldsymbol{w}_0^T\boldsymbol{R}_1\boldsymbol{w}_0)]. \quad (54)$$

Note that this is a test for a change in the spectral properties of an AR process and is insensitive to pure changes in power, a direct result of assuming that the noise driving the Gaussian AR process is constant, even as the AR parameters change. Finally, it is interesting to note that the term in brackets is closely related to the Kullback information between two AR processes.

3.6 Discussion

The GLR algorithm is very powerful and its primary strength is that it makes no assumptions about the data, other than that it should be piecewise ergodic. Its primary limitation is in detecting rapid switching between regimes. In practice, the maximization over the changepoint T in (40) must be restricted, in order to avoid unreliable estimates of small samples. This creates a deadzone upon starting the algorithm after detection of the last changepoint. This deadzone inevitably increases with the complexity of the parametric models. For detecting a change in the mean, the deadzone can be relatively short, on the order of five or ten samples. However, for detecting a change in a process with memory, such as an AR process, the deadzone needs to be much larger in order to reliably estimate correlations. In short, it takes more time to reliably estimate correlations as opposed to a simple mean. The situation is even worse when the process requires non-linear predictors.

Closely related, another drawback is that, being a sequential detector, it discards information from previous regions. If we suspect apriori that the number of potential regimes is limited, and that some regimes may be revisited, we can potentially improve the estimate of our parametric models, and potentially reduce the deadzone by using global information. This is clearly a non-causal approach but is not a serious drawback since we are interested in the off-line segmentation problem.

Finally, for the memory case, while the GLR is directly applicable to the use of non-linear predictors, it is computationally infeasible. Non-linear regressors cannot be trained in a single pass through a block of data. Thus, even for the two model GLR algorithm, a single non-linear model

must be trained approximately *n* (ignoring the deadzone) separate times just to evaluate the decision function for a block of *n* samples.

4 Memoryless Mixture Models

We now seek an unsupervised segmentation algorithm that overcomes some of the drawbacks of the GLR algorithm. As a starting point, we assume that the data is generated by a finite number of switching regimes. Some of the regimes may be re-occurring, and therefore the algorithm should make use of the *entire* time series for improved model estimation *prior to or in conjunction with* the segmentation stage. In addition, some of the regimes may involve non-linear processes, and therefore the algorithm must be computationally tractable for parametric non-linear predictors that cannot be trained in a single step.

Taking all these constraints into consideration, we propose to first train a global mixture model for the entire random process, and then use the global information to deduce information about local changepoints. The local changepoint information can then be used in turn to refine the global models.

We will first consider mixture models appropriate to memoryless random processes, and then processes with memory.

4.1 Memoryless models

Let us consider modeling a vector i.i.d. process that may be multi-modal. That is, a vector $\boldsymbol{x}$ may be generated by one of K different distributions. Let $p(\boldsymbol{x},k)$ be the joint probability of a vector $\boldsymbol{x}$ belonging to the discrete mode k. We can't determine this distribution directly from data because we don't know which modes are active at which times. We can, however, observe the marginal density

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} p(\boldsymbol{x}, k) = \sum_{k=1}^{K} P(k) \cdot p(\boldsymbol{x}|k) = \sum_{k=1}^{K} g_k \cdot p(\boldsymbol{x}|k) \tag{55}$$

where the kth mixing coefficient, g_k, can be viewed as the apriori probability of the kth mode. The modes are usually modeled as a Gaussian distribution

$$p(\boldsymbol{x}|k) = \frac{1}{(2\pi)^{D/2}\left|\Sigma_k\right|^{1/2}} \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{m}_k)^T \Sigma_k^{-1} (\boldsymbol{x}-\boldsymbol{m}_k)\right] \tag{56}$$

where Σ_k is the covariance and D is the dimension of the vector process.

Given a time series $x(n)$, n=1...N, the standard approach to fitting the data to the model is to maximize the likelihood

$$L = \prod_{n=1}^{N} p(\boldsymbol{x}(n)) = \prod_{n=1}^{N} \sum_{k=1}^{K} g_k \cdot p(\boldsymbol{x}(n)|k) \tag{57}$$

with respect to the free parameters. Unfortunately, this is difficult to maximize with using standard techniques such as gradient descent. The Expectation-Maximization (EM) algorithm of Dempster (1977), on the other hand, is ideally suited to estimating joint distributions when some

of the variables are not directly observable. A complete description of the EM algorithm is beyond the scope of this chapter. Therefore, we present an intuitive development due to Weigend (1995) that hilites the outcomes of the full application of the EM algorithm.

If we knew to which mode each sample belonged, the problem would easily decouple into K separate maximum likelihood problems. Therefore, we postulate a latent binary variable, $I_k(n)$, which indicates which mode is active at each time step. This allows the "complete" likelihood to be written

$$L = \prod_{n=1}^{N} \prod_{k=1}^{K} [g_k \cdot p(\boldsymbol{x}(n)|k)]^{I_k(n)} . \tag{58}$$

Unfortunately, we do not know which mode is active at which times. However, we can estimate this from the current values of the mixture model using Bayes' law, in what is known as the expectation (E) step

$$h_k(n) \equiv E[I_k] = P(k|\boldsymbol{x}(n)) = \frac{P(k) \cdot p(\boldsymbol{x}(n)|k)}{p(\boldsymbol{x}(n))} = \frac{g_k \cdot p(\boldsymbol{x}(n)|k)}{\sum_{j=1}^{K} g_j \cdot p(\boldsymbol{x}(n)|j)} . \tag{59}$$

The h_k, called the posterior probabilities for obvious reasons, are then used in place of the indicator variables in (58), which after taking the logarithm becomes our cost function

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} \left\{ -h_k(n) \log[g_k] + \frac{1}{2} h_k(n) \{ (\boldsymbol{x}(n) - \boldsymbol{m}_k)^T \Sigma_k^{-1} (\boldsymbol{x}(n) - \boldsymbol{m}_k) + \log[|\Sigma_k|] \} \right\} . \tag{60}$$

Minimization of this cost function with respect to the free parameters forms the maximization (M) step:

$$\boldsymbol{m}'_k = \frac{1}{\sum_{n=1}^{N} h_k(n)} \sum_{n=1}^{N} h_k(n) \boldsymbol{x}(n) , \tag{61}$$

$$\Sigma'_k = \frac{1}{\sum_{n=1}^{N} h_k(n)} \sum_{n=1}^{N} h_k(n) (\boldsymbol{x}(n) - \boldsymbol{m}'_k)(\boldsymbol{x}(n) - \boldsymbol{m}'_k)^T , \tag{62}$$

$$g'_k = \frac{1}{N} \sum_{n=1}^{N} h_k(n) , \tag{63}$$

where the primes indicate values to be used in the next iteration, beginning with the E-step. The algorithm is iterated until the parameters converge.

Note that using the posterior probabilities as an estimate of the indicator variables has completely decoupled the problem. The means and covariances of the modes are estimated independently of each other and the mixing coefficients are estimated independent of the modes and covariances.

4.2 Training Issues

While the EM algorithm is guaranteed to reduce the cost function in (60) with each iteration, it does not guarantee finding the global minimum. In fact, even for the simplest of problems, there may be many local minima. For this reason, annealing the model parameters during training can greatly improve the repeatability of the results, although not necessarily increase the chances of finding the global minimum. Annealing can take two forms. First, the complexity of the model can be annealed. For example, the covariances can be constrained to be diagonal or even equal to each other early in the training, and then relaxed to maximum freedom. Second, parameters that reflect model confidence can be annealed. For example, early in the training the mixture coefficients can be restricted to a certain range within the value $1/K$, or the covariances can be kept artificially large to reflect model uncertainty, and then relaxed to full freedom.

A much more difficult question is how to choose the number of modes. There are basically two approaches. One can start from a small order, K, train the system to completion, and then start over and repeat the training at the next model order $K+1$. Training stops when the likelihood reaches a plateau with respect to model order. The other approach involves annealing the model order during training, using techniques such as tree growing and pruning algorithms. There are numerous examples of this in the literature but they are beyond the scope of this chapter.

4.3 Obtaining a segmentation from the global model

Note that the mixture model is memoryless. That is, one could randomly reorder the input and the algorithm would converge to the same final parameters, given the same initial conditions. Clearly, there is no reason to expect a global mixture model to correctly recognize that a given sequence is piecewise stationary, unless there is absolutely no overlap of the clusters in the input space. How can we use the mixture model after training to segment a time series? Assuming we have trained a mixture model with the entire data set, we now have a set of candidate models for the data and can proceed as for the supervised sequential detection and identification. To do so, we first identify which mode is "active" using the SPRT, and then set up a change detection threshold test using the CUSUM. Assuming that the last detected changepoint was at time step T-1, then the process log-likelihood ratio between modes i and j from time T to n is

$$L_{ij}(n) = \log\left[\frac{\prod_{m=T}^{n} p(\boldsymbol{x}(m), i)}{\prod_{m=T}^{n} p(\boldsymbol{x}(m), j)}\right] = \sum_{m=T}^{n} \log\left[\frac{g_i \cdot p(\boldsymbol{x}(m)|i)}{g_j \cdot p(\boldsymbol{x}(m)|j)}\right] = \sum_{m=T}^{n} \log\left[\frac{h_i(m)}{h_j(m)}\right]. \quad (64)$$

This can be written in a recursive manner as

$$L_{ij}(n) = L_{ij}(n-1) + l_{ij}(n) \quad (65)$$

where the log-likelihood increment, $l_{ij}(n)$, is expressed in terms of the posterior probabilities

$$l_{ij}(n) \;=\; \log\left[\frac{h_i(n)}{h_j(n)}\right]. \tag{66}$$

We can now apply the multi-hypothesis SPRT and CUMSUM algorithms to segmenting the data. The only difference is that we use the ratios of the posterior probabilities instead of the pdf's themselves. Once the data has been segmented and identified, we can make a final pass through the data to improve the models.

Let's take a closer look at the effect of the priors. The log-likelihood increment between the true mode i^* and any other mode j can be decomposed as

$$l_{i^*j}(n) \;=\; \log\left[\frac{h_{i^*}(n)}{h_j(n)}\right] \;=\; \log\left[\frac{p(\boldsymbol{x}(n)|i^*)}{p(\boldsymbol{x}(n)|j)}\right] + \log\left[\frac{g_{i^*}}{g_j}\right]. \tag{67}$$

The expected value of this with respect to the true mode will be the slope of $L_{i^*j}(n)$

$$E_{i^*}[l_{i^*j}(n)] \;=\; K(p_{i^*}, p_j) + \log\left[\frac{g_{i^*}}{g_j}\right] \tag{68}$$

where $K(p_{i^*}, p_j)$, a positive quantity, is the Kullback-Leibler information (34) between the conditional pdf's for modes i^* and j. The priors can have several effects on the detectability of a mode. In order to guarantee detectability for the multi-hypothesis CUSUM, the slope of $L_{i^*j}(n)$ must be positive or $\log[g_{i^*}/g_j] > -K(p_{i^*}, p_j)$, for all j. Since this is difficult to guarantee, it is better to use the modified log-likelihood in (5), which is less sensitive to the individual ratios. As long as detectability is guaranteed, then the effect of the priors will be to either increase or decrease the detection delay. Alternatively, one can simply ignore the priors in (67) in order to guarantee that every model has the potential for becoming active.

5 Mixture Models for Processes with Memory

We now turn to modeling a process with memory using a mixture model. We will assume a one-dimensional time series, but the results can easily be generalized to the multi-dimensional case. Recall that the multivariate pdf of a random process, with an effective memory depth of M, can be decomposed as

$$p(\chi(n)) \;=\; \prod_{n=1}^{N} p(x(n)|\chi(n-1)). \tag{69}$$

As before, let us entertain the possibility that the random process is produced by K switching regimes. Therefore, we propose the joint conditional density $p(k, x(n)|\chi(n-1))$, where the discrete variable $k = 1 \ldots K$ indicates the regime. Once again, we can't observe this pdf directly because we don't know which regime is active at which times, but we can observe its marginal distribution

$$p(x(n)|\chi(n-1)) = \sum_{k=1}^{K} p(k, x(n)|\chi(n-1)) =$$
$$\sum_{k=1}^{K} P(k|\chi(n-1)) \cdot p(x(n)|\chi(n-1), k) = \sum_{k=1}^{K} g_k(\chi(n-1)) \cdot p(x(n)|\chi(n-1), k) \quad (70)$$

where $g_k(\chi(n-1)) \equiv P(k|\chi(n-1))$ can be regarded as the apriori probability of the kth regime. If we can find predictors for each of the sub-processes, up to a random term ε_k,

$$x(n) = f(\chi(n-1);\theta_k) + \varepsilon_k(n) \qquad k = 1 \ldots K \quad (71)$$

then we can evaluate the conditional pdf in terms of the innovations

$$p(x(n)|\chi(n-1), k) = p_{\varepsilon_k}(\varepsilon_k(n)). \quad (72)$$

The residuals are usually modeled as a Gaussian distribution, which for a one dimensional time series becomes

$$p_{\varepsilon_k}(\varepsilon_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left[-\frac{(\tilde{x}_k(n) - x(n))^2}{2\sigma_k^2}\right] \quad (73)$$

where we have defined $\tilde{x}_k(n) \equiv f(\chi(n-1);\theta_k)$ to be kth predictor's estimate of the next value of the time series and σ_k, a "nuisance parameter", is the variance of the kth predictor.

Taking the expected value of both sides of (70) gives the best MMSE prediction of the next value of the time series

$$\tilde{x}(n) \equiv E[x(n)|\chi(n-1)] = \sum_{k=1}^{K} g_k(\chi(n-1)) \cdot E[x(n)|\chi(n-1), k] = \sum_{k=1}^{K} g_k(\chi(n-1))\tilde{x}_k(n) \quad (74)$$

which is a weighted sum of outputs of the individual predictors. We can regard this as the total system output.

5.1 The mixture of experts

Unlike the memoryless case, the mixing coefficients, $g_k(\chi(n-1))$, are a function of the time series itself due to the dependence on the recent past. In the "mixture of experts" algorithm[1] of Jordan and Jacobs (1991, 1994), this mapping is provided by an adaptable function called the "gate". The particular variation of the algorithm we will examine is Weigend's (1995) non-linear gated experts, which implements the gate using a single multilayer perceptron. This architecture

1. The mixture of experts was originally developed in the context of regression and was first applied to time series by Weigend (1995).

is shown in Figure 4. Note that the individual predictors, called “experts” in this context, and the

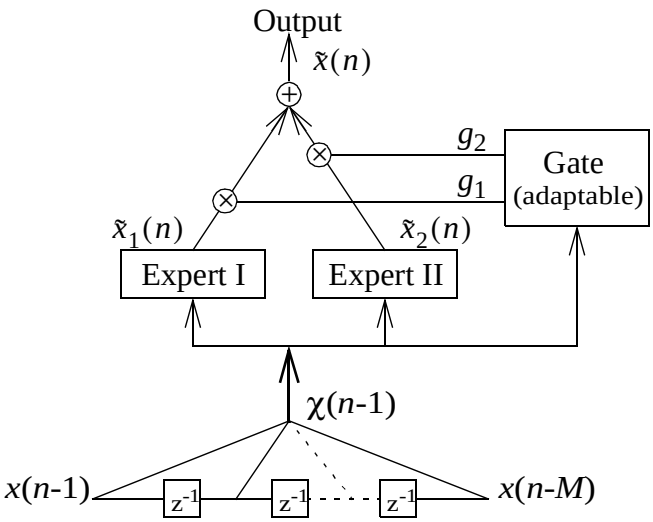

Figure 4. Mixture of experts during activation for a network with two experts.

gate all see the same input.

We now turn to the question of how to train the gate and experts. In the following development, for ease of reading, we leave out explicit time dependence whenever possible. Also, there is an implicit iteration index in all the following equations. Given a time series of length N, we choose the free parameters of the predictors and gate that maximize the process log-likelihood. If the innovations are i.i.d, we can re-write the process likelihood as

$$p(\chi(n)) = \prod_{n=1}^{N} \sum_{k=1}^{K} g_k(\chi(n-1)) \cdot p_{\varepsilon_k}(\tilde{x}_k(n) - x(n)) . \tag{75}$$

This is difficult to maximize directly. Therefore, just as for the memoryless case, we propose a latent binary indicator, I_k, indicating which expert is valid, allowing the likelihood to be written

$$L = \prod_{n=1}^{N} \prod_{k=1}^{K} [g_k(\chi(n-1)) \cdot p(x(n)|\chi(n-1), k)]^{I_k(n)} . \tag{76}$$

The indicator variable is “hidden”, in the sense that we do not know apriori which expert is valid at any time step. In the E step of the EM algorithm, for a given set of free parameters of the experts and gate, the entire data set is evaluated, holding the free parameters constant, to determine $p(x(n)|\chi(n-1), k)$ and $g_k(\chi(n-1))$ for all k and n. We then replace the indicator variables, I_k, at every time step, by their expected value:

$$h_k(n) \equiv E[I_k(n)] = P(k|x(n), \chi(n-1)) = \frac{P(k|\chi(n-1)) \cdot p(x(n)|\chi(n-1),k)}{p(x(n)|\chi(n-1))} = \frac{g_k(\chi(n-1)) \cdot p(x(n)|\chi(n-1),k)}{\sum_{k=1}^{K} g_k(\chi(n-1)) \cdot p(x(n)|\chi(n-1),k)} . \tag{77}$$

Thus, h_k is the posterior probability of expert k, given both the current value of the time series and the recent past. For the M step, (76) is maximized or equivalently, the negative log-likelihood,

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} \left\{ -h_k(n) \cdot \log[g_k(\chi(n-1))] + \frac{1}{2} h_k(n) \left[\frac{(\tilde{x}_k(n) - x(n))^2}{\sigma_k^2} + \log[\sigma_k^2] \right] \right\} \tag{78}$$

is globally minimized over the free parameters. The process is then repeated. If, in the M step, (78) is only decreased and not minimized, then the process is called the generalized EM (GEM) algorithm. This is necessary when either the experts or gate is non-linear, and a search for the global minimum is impractical.

The first term in the summation of (78) can be regarded as the cross entropy between the posterior probabilities and the gate. It has a minimum when only one expert is valid, and thus encourages the experts to divide up the input space. In order to ensure that the outputs of the gate sum to unity, the output layer of the multilayer perceptron has a "softmax" transfer function,

$$g_k(\chi) = \frac{\exp[s_k(\chi)]}{\sum_{j=1}^{K} \exp[s_j(\chi)]} \tag{79}$$

where s_k is the k^{th} input to the softmax. For a gate implemented as a multilayer perceptron, the cross entropy term in (78) cannot be minimized in a single step and the GEM algorithm must be employed. If the gate is trained through gradient descent (backpropagation), the error backpropagated to the *input* side of the softmax, from (78) and (79), at each time step is

$$\frac{\partial J}{\partial s_k} = g_k(\chi) - h_k . \tag{80}$$

This is the same backpropagated error that would result for a mean square error with the posterior probabilities acting as the desired signal. Thus, the posterior probabilities act as targets for the gate. For each EM iteration, several training iterations may be required for the gate, since it is implemented using a multi-layer perceptron.

There is an analytical solution for the experts, at each iteration, when they are linear predictors, $\tilde{x}_k(n) = \boldsymbol{w}_k^T \chi(n-1)$,

$$\boldsymbol{w}_k = \boldsymbol{R}_k^{-1} \boldsymbol{p}_k \tag{81}$$

where $\boldsymbol{R}_k$ and $\boldsymbol{p}_k$ are weighted autocorrelation and cross-correlation matrices, respectively

$$\boldsymbol{R}_k = \frac{1}{\sum_{n=1}^{N} h_k(n)} \sum_{n=1}^{N} h_k(n) \chi(n-1) \chi^T(n-1), \qquad \boldsymbol{p}_k = \frac{1}{\sum_{n=1}^{N} h_k(n)} \sum_{n=1}^{N} h_k(n) \chi(n-1) x(n) \tag{82}$$

Note that the solution represents a weighted Weiner filter, where the data is weighted by the posterior probabilities.

If the experts are non-linear, then the error used to iteratively train each expert is simply interpreted as being weighted by the posterior probabilities, $h_k(n)$. However, Zeevi et al. (1997) have shown that the non-linearity of the gate allows the mixture of experts to be a universal approximator, even if the expert predictors are linear.

No matter whether the experts are linear or non-linear, there is an exact solution for the Gaussian covariance parameter Σ_k at the end of each epoch

$$\sigma_k^2 = \frac{1}{\sum_{n=1}^{N} h_k(n)} \sum_{n=1}^{N} h_k(n)[\tilde{x}_k(n) - x(n)]^2. \tag{83}$$

5.2 Training issues

For a given data set, there are many local minima of the performance surface. Tips that were given for training memoryless mixture models also hold here. Annealing the experts' model complexity, however, is clearly a more difficult task if the experts are non-linear. In order not to dramatically increase training time, the complexity should be increased without completely discarding the training results of the previous iteration. If the experts are multi-layer perceptrons, this can be accomplished with a processing element growing algorithm such as Fahlman's (1990) cascade correlation. In the area of annealing confidence parameters, Weigend anneals the posterior probabilities that are used to update the variance parameters, and found that it led to better repeatability.

Strangely enough, this architecture has difficulties when the predictors are linear. In particular, the gate must learn to predict the appropriate expert independent of the amplitude of the input. Amplitude independent mappings are notoriously difficult for a neural network. For this reason, Xu (1995) uses a simplified gate for linear predictors such that the mixing coefficients are static, exactly as for memoryless processes (63), at the expense of no longer representing the true likelihood of the process.

We wish to mention again that the number of experts can be grown or pruned during training. There is an abundance of literature on this topic but Ramamurti and Gosh (1999) have addressed it as it applies to a mixture of expert predictors.

5.3 Obtaining a segmentation from the trained mixture of experts

Let us now return to the application of the mixture of experts to time series segmentation. Recall that for the memoryless case, the mixture model parameters would converge to the same state, given the same initial conditions, for any random ordering of the time series. This is not the case here. Once vectorized and viewed as a regression problem, where $\chi(n\text{-}1)$ is the input and $x(n)$ is a desired signal, the input and desired signals can be randomly sorted and the algorithm will still converge to the same state, given the same initial conditions. However, the input vectors would no longer represent the same time series. In addition, the gate, being a multi-layer perceptron, possesses a weak memory in the sense that it must learn to predict the apriori probability of the experts by learning from their performance over the entire data set.

Clearly the architecture has memory, but does it have sufficient memory to achieve a perfect segmentation? The answer is no for two reasons. First, there may be overlap between the return maps of the dynamical systems in signal space or, in the language of non-linear dynamics, the manifolds of the various dynamical regimes may intersect in phase space. Second, measurement noise can "smear" manifolds that ordinarily would never intersect. Where there is overlap in signal space, the gate cannot perform perfect segmentation.

In short, the predictive experts have short term memory that by itself is insufficient for performing longer term segmentation. Therefore, we propose to use the supervised segmentation algorithms in conjunction with the unsupervised mixture of experts, exactly as for the memoryless mixture model case. That is, the mixture of experts are trained to completion on the entire data set. The "active" mode at the beginning of the data set is then identified using the multi-hypothesis SPRT. Then the multi-hypothesis CUSUM algorithm is used to perform sequential change detection and identification. Once a change is detected, the CUSUM algorithm is reset.

Let time T-1 be the last detected changepoint, and consider the process log-likelihood ratio between modes i and j from time T to n

$$L_{ij}(n) = \log\left[\frac{\prod_{m=T}^{n} p(i, x(m)|\chi(m-1))}{\prod_{m=T}^{n} p(j, x(m)|\chi(m-1))}\right] = \sum_{m=T}^{n} \log\left[\frac{h_i(m)}{h_j(m)}\right] \quad . \tag{84}$$

Once again, this can be written in a recursive manner as

$$L_{ij}(n) = L_{ij}(n-1) + l_{ij}(n) \tag{85}$$

where $l_{ij}(n)$ is the log-likelihood increment

$$l_{ij}(n) = \log\left[\frac{h_i(n)}{h_j(n)}\right]. \tag{86}$$

Just as in the memoryless case, the change detection algorithms are applied to the posterior probabilities. However, the argument for ignoring the priors is less compelling here than it was for the memoryless mixture model case because the priors are a function of the data itself. For example, if i* is true current active process, the log-likelihood ratio increment is

$$l_{i^*j}(n) = \log\left[\frac{h_{i^*}(n)}{h_j(n)}\right] = \log\left[\frac{p(\chi(n-1)|i^*)}{p(\chi(n-1)|j)}\right] + \log\left[\frac{g_{i^*}(\chi(n-1))}{g_j(\chi(n-1))}\right]. \tag{87}$$

Taking the expectation with respect to $p(\chi(n-1)|i^*)$ of the first term on the right leads to the Kullback information between two random *processes*. The expectation with respect to the second term is more difficult to characterize because $g_i(\chi(n-1))$ is mapped by a multi-layer perceptron.

The generalization properties of neural networks suggest the possibility of segmenting and identifying a time series without the use of the models themselves. Since the gate ideally learns to

predict the posterior probabilities, it should be possible to obtain a segmentation based only on the priors

$$l_{ij}(n) \approx \log\left[\frac{g_i(\chi(n-1))}{g_j(\chi(n-1))}\right]. \tag{88}$$

If the models are available, however, the only apparent advantage to this is speed of computation.

5.4 Summary

In summary, we are proposing to use a completely unsupervised algorithm to train a mixture of parametric models over the entire time series, and then apply a supervised change detection algorithm to the trained models. It is natural to stop and ask what we have gained relative to the unsupervised GLR algorithm.

First, recall that the GLR algorithm has a dead time that is determined by how many samples are required to reliably estimate the parametric models over the various subregions. While this dead time is small for memoryless systems, it can be quite large for processes with memory. By using a global mixture model, we have essentially eliminated this dead time.

An even greater benefit accrues when non-linear experts are required. The GLR is too computationally demanding for non-linear models. The mixture of experts, however, are well suited to the use of non-linear experts.

In fact, even time series for which the concept of piecewise stationarity no longer applies can be segmented. Weigend (1995) has shown how the mixture of experts can be used to segment chaotic dynamical systems. He did not, however, apply the CUSUM algorithm to the posterior probabilities, as we do.

6 Gated Competitive Experts

We now consider some experimental unsupervised models that are no longer derived from the multivariate pdf of the time series, but are a generalization of mixture models that fall within a single framework that we call gated competitive experts. Although some of the algorithms can be applied to memoryless processes, we will concentrate on processes with memory for the remainder of this chapter.

In this section we will develop a common framework for several seemingly disparate algorithms that have appeared in the literature recently, including the mixture of experts. In the following sections we will show how some experimental algorithms fall within the framework. In particular, in the next section we will consider functional mappings other than prediction that can be used to provide segmentation information. Next, we will consider algorithms that can directly segment a time series, without resorting to an external segmentation algorithm like the CUSUM, through the use of memory. Finally, we will consider novel gate structures that exist on a spatial grid similar to the Kohonen Map.

6.1 Overview

A gated competitive expert model is characterized by several adaptable “experts” that compete to explain the same data. That is, they all see the same input, and all attempt to produce the same output. Their performance is both monitored and mediated by a gate, whose goal is to determine the relative validity of the experts for the current data, and then to appropriately moderate their learning. The history of the gate’s decisions can be used to segment the time series.

In the activation phase, the total system output is a simple weighted sum of the experts' outputs

$$\boldsymbol{y}(n) = \sum_{k=1}^{K} g_k(n) \cdot \boldsymbol{y}_k(\chi(n)) \tag{89}$$

where $\boldsymbol{y}_k$ is the output of the kth expert with χ as input, and g_k is the kth output of the gate. The input, $\chi(n)$ is a vector of delayed versions of the time series, in essence the output of a tapped delay line, repeated here for convenience

$$\chi(n) = \left[x(n)\ x(n-1)\ \ldots\ x(n-M+1)\right]^T. \tag{90}$$

In order for the mixture to be meaningful, the gate outputs are constrained to sum to one

$$\sum_{k=1}^{K} g_k(n) = 1. \tag{91}$$

Such a linear mixture can represent either a competitive or cooperative system, depending on how the experts are penalized for errors, as determined by the cost function. In fact, it was in the context of introducing their mixture of experts model that Jacobs et al. (1991) first presented a cost function that encourages competition among gated expert networks, which we generalize to

$$J(n) = \sum_{k=1}^{K} g_k(n) \cdot f(\boldsymbol{d}(n) - \boldsymbol{y}_k(\chi(n))) \tag{92}$$

where $\boldsymbol{d}$ is the desired signal and $f(\boldsymbol{d}(n) - \boldsymbol{y}_k(\chi(n)))$ is a function of the error between the desired signal and the kth expert. Since the desired signal is the same for all experts, they all try to regress the same data, and are always in competition. This alone, however, is not enough to foster specialization. The gate uses information from the performance of the experts to produce the mixing coefficients. There are many variations of algorithms that fall within this framework. Let's discuss the important components one at a time, starting with the design of the desired signal.

The formalism represented by (92) is a supervised algorithm, in that it requires a desired signal. However, we are interested in a completely unsupervised algorithm. A supervised algorithm becomes unsupervised when the desired signal is a fixed transformation of the input: $\boldsymbol{d} \Rightarrow \boldsymbol{d}(\chi)$. Although many transformations are possible, the two most common transformations involve the delay operator and the identity matrix, resulting in *prediction* and *auto-association*, respectively. We have already used prediction in conjunction with the mixture of experts in the previous section. In the next section, we will consider auto-association.

Gates can be classified into two broad categories, which we designate as input or output based and are shown in Figure 4. For output based gating, the gate is a direct *calculated* function of the performance, and hence, the *outputs*, of the experts. The simplest scheme for the gate that falls within this category is hard competition, for which the gate chooses the expert with the smallest magnitude error in a winner take all fashion. Other output based gates possess memory to keep

track of past expert performance, the simplest example of which is the mixture model of section four where the gate is the average of the posterior probabilities over the data set (63). The gate in the annealed competition of experts (ACE) of Muller et al. (1995) implements memory in the form of a local boxcar average squared error of the experts. The self-annealing competitive prediction of Fancourt and Principe (1997a) also uses the local squared error, but using a recursive estimator. Both of these algorithm will be discussed later.

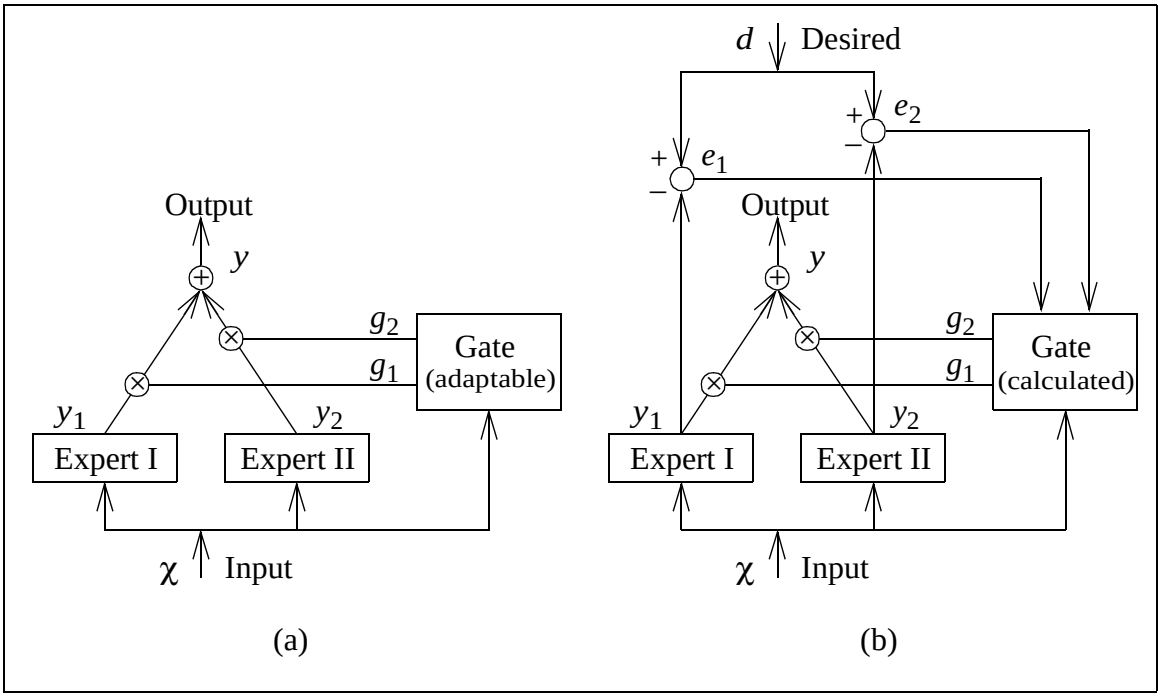

Figure 5. Two classes of gates during activation: (a) input based; (b) output based.

With input based gating, the gate is an *adaptable* function of the *input*, $g_k \Rightarrow g_k(\chi)$, that learns to forecast which expert will perform the best. The best example of input based gating is the mixture of experts algorithm, which we have previously examined in some detail when the experts were predictors. In the next section we consider the mixture of experts algorithm when the desired signal is the input itself.

Finally, the total cost function can be constructed in several ways from the instantaneous cost, depending on the error function $f(\cdot)$. If $f(\cdot)$ is a quadratic function of the error, the total cost over a data set of N samples is the sum of the instantaneous costs

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} g_k(n) \cdot [\boldsymbol{d}(n) - \boldsymbol{y}_k(\chi(n))]^2 . \tag{93}$$

If $f(\cdot)$ is a probability density function in the error, the appropriate total cost function is a product of the instantaneous costs

$$J = \prod_{n=1}^{N} \sum_{k=1}^{K} g_k(n) \cdot p(\boldsymbol{d}(n) - \boldsymbol{y}_k(\chi(n))) \tag{94}$$

and returns us towards the mixture of experts when the gate is input based.

7 Competitive Temporal Principal Component Analysis

As mentioned previously, any supervised algorithm can be converted to an unsupervised algorithm by making the desired signal a fixed transformation of the input. In this section we explore the case when the desired signal is the input itself

$$J(n) = \sum_{k=1}^{K} g_k(n) \cdot f(\chi(n) - \hat{\chi}_k(n)) \tag{95}$$

where $\tilde{\chi}_k \equiv \boldsymbol{y}_k$ is the kth expert's estimate of the input.

Of course, any expert can immediately reconstruct the input by implementing an identity function. In order to place the experts in competition, they must be given a task that they cannot perform perfectly. One way to achieve this is to for the experts to first reduce the dimensionality of the input, and then attempt to reconstruct the input from the reduced dimensional space. The experts essentially compete on their ability to compress and then uncompress the input. Although they are many compression schemes, we will employ a linear one called principal component analysis (PCA). When PCA is applied to a vector of delayed versions of a time series, it is sometimes called the discrete Karhunen-Loeve transform. However, we prefer temporal PCA.

We also have to choose the gate and overall cost function. Dony and Haykin (1995) used hard competition with multiple PCA experts for image compression and segmentation. However, hard competition is extremely sensitive to initial conditions. We will utilize the non-linear gated experts, where the gate is a non-linear function of the input and $f(\cdot)$ is an appropriate pdf. In other words, we seek to combine principal component analysis (PCA) with the mixture of experts formalism. We first give a brief introduction to the terminology and notation and main results of temporal PCA, without proof.

7.1 Temporal principal component analysis

Consider expanding the vector $\chi(n)$, of dimension M, in terms of some fixed but complete basis in the M dimensional space

$$\chi(n) = \sum_{i=0}^{M-1} z_i(n)\boldsymbol{u}_i = \boldsymbol{U}\boldsymbol{z}(n) \tag{96}$$

where the matrix $\boldsymbol{U} = \left[\boldsymbol{u}_0\ \boldsymbol{u}_1\ \ldots\ \boldsymbol{u}_{M-1}\right]$ is deterministic and full rank. If the basis is constrained to be orthonormal

$$\boldsymbol{u}_i^T \boldsymbol{u}_j = \delta_{ij} \tag{97}$$

multiplying both sides of (96) from the left by $\boldsymbol{U}^T$ immediately yields the coefficients of the expansion

$$z_i(n) = u_i^T \chi(n) \Rightarrow \quad \boldsymbol{z}(n) = \boldsymbol{U}^T \chi(n). \tag{98}$$

If the expansion is performed over a subset of the basis functions, the result is an estimate of χ

$$\tilde{\chi}(n) = \sum_{i=0}^{P-1} z_i(n)\boldsymbol{u}_i = \boldsymbol{U}_p \boldsymbol{z}_p(n) \qquad P < M \tag{99}$$

where $\boldsymbol{U}_p$ is a matrix formed from a subset of the columns of $\boldsymbol{U}$. Likewise, the subset of the expansion coefficients are given by

$$\boldsymbol{z}_p(n) = \boldsymbol{U}_p^T \chi(n). \tag{100}$$

Given a set of random vectors, $\chi(n)$, n=1...N, what is the basis that minimizes the mean square reconstruction error over the data set, $E[\|\chi - \tilde{\chi}\|^2]$, under the same orthonormality constraints as in (97)? It is not difficult to show that the basis must satisfy the eigenvalue equation

$$\boldsymbol{R}\boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i \tag{101}$$

where λ_i and $\boldsymbol{u}_i$ are the eigenvalues and eigenvectors, respectively, of the autocorrelation matrix $\boldsymbol{R} = E[\chi\chi^T]$. The P eigenvectors are chosen on the basis of the P largest corresponding eigenvalues. The z_i are called the principal components. The minimum mean square reconstruction error is given by the sum of the discarded eigenvalues

$$Min[E[\|\chi - \tilde{\chi}\|^2]] = \sum_{i=P}^{M-1} \lambda_i. \tag{102}$$

This results in the temporal PCA architecture shown in Figure 6 below, which will become a distinct expert within the context of a gated competitive system.

7.2 Temporal PCA in the frequency domain

When viewed as a set of FIR filters, referred to as eigenfilters, the eigenvectors of the autocorrelation matrix have very interesting properties. Szegö (see Thomson, 1982) showed that the eigenvalues are asymptotically equal to the power spectrum sampled at equal frequency intervals. To illustrate this in the infinite case, consider the Fourier transform of the eigenvalue equation (101),

$$\lambda_i U_i(f) = \int_{-1/2}^{1/2} U_i(f')S(f')\frac{\sin\pi M(f-f')}{\sin\pi(f-f')}df' \tag{103}$$

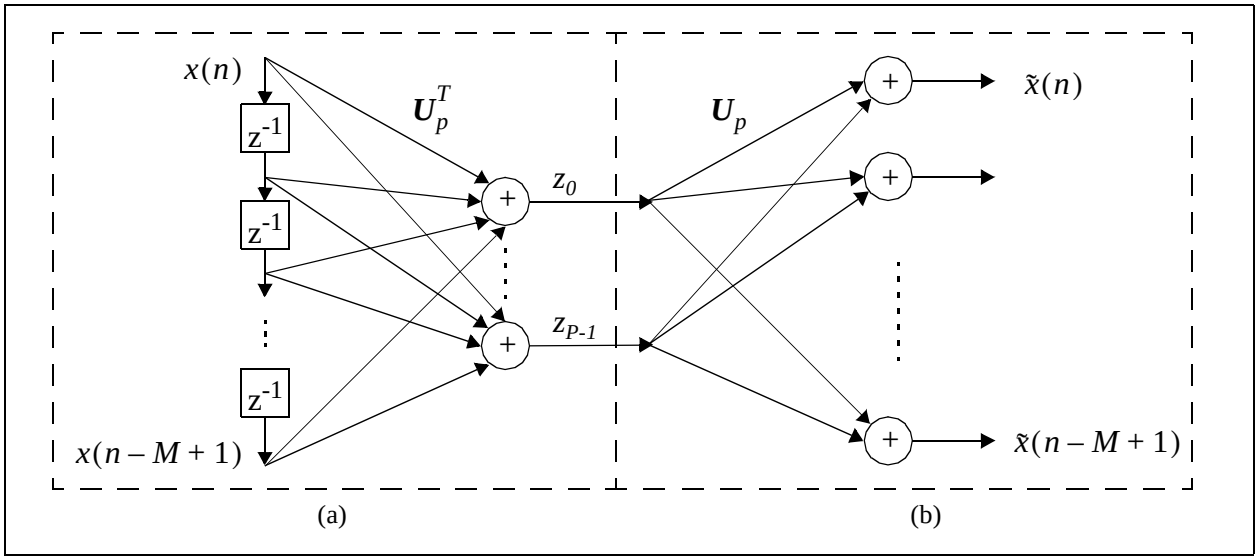

Figure 6. Temporal PCA architecture: a) PCA network. b) reconstruction network.

where $S(f)$ is the power spectrum associated with the input signal autocorrelation function, $U_i(f)$ is the Fourier transform of the ith eigenfilter, and M is the eigenfilter dimension as defined previously. As $M \to \infty$, the Dirichlet function becomes a Dirac delta function and (103) becomes,

$$\lambda_i U_i(f) \;=\; U_i(f) S(f) \tag{104}$$

the solution of which is

$$U_i(f) \;=\; \delta(f - f_i)\;, \quad \lambda_i \;=\; S(f_i)\,. \tag{105}$$

That is, the eigenfilters form a fixed Fourier basis, and the eigenvalues become the power spectrum. Since the autocorrelation matrix is real and symmetric, the eigenvalues are real, and we can enforce the constraint that the eigenfilters be real. Then, each eigenvalue will be doubly degenerate, associated with two real sinusoidal eigenvectors separated in phase by 90 degrees. Assuming that the eigenvalues have been ordered from largest to smallest, so that λ_0 is the largest eigenvalue, then λ_0 is the maximum value of the power spectrum: $\lambda_0 \;=\; Max_f[S(f)]$.

For a finite dimensional autocorrelation matrix, the basis is no longer fixed but depends on the signal itself. However, as implied by Szegö's theorem, for a finite but sufficiently large eigenfilter dimension, M, the eigenfilters act as bandpass filters centered around peaks in the signal power spectrum, subject to orthogonality constraints. The passband becomes increasingly narrow as the number of taps is increased. Thus, eigenfilters can be approximately regarded as FIR filters matched to peaks in the signal spectrum. We can estimate the bandwidth by noting that since $\boldsymbol{R}$ is Toeplitz, the largest eigenfilter has all its zeros on the unit circle (Makhoul, 1981) and thus forms the passband through the suppression of a zero near the peak frequency. Thus, the bandwidth is approximately

$$\Delta f \approx (M+1)^{-1}\,. \tag{106}$$

7.3 System architecture

We now bring together the mixture of experts architecture and temporal PCA. The basic architecture is shown in Figure 7, where each expert is a temporal PCA network as in Figure 6. The input to the overall system is a tap delay line with M-1 delays, which creates an input vector χ of dimension M. Each expert has two outputs: the principal components, $\boldsymbol{z}_k$, which we call the *expert system output*, and the reconstructed input, $\tilde{\chi}_k$, which we call the *expert training output*. It is important to understand why this choice was made. In PCA the goal is to either utilize the principal components or the eigenvectors themselves. But it is not the principal components that drive the competition; rather, it is the reconstructions of the input by the experts. This is the reason the block diagram of Figure 7 differs from the more conventional mixture of experts which display a single output. The gating outputs, g_k, weigh *both* the system outputs, $\boldsymbol{z}_k$, and the training outputs, $\tilde{\boldsymbol{x}}_k$. The total system and training outputs are thus given by

$$\boldsymbol{z} = \sum_{k=1}^{K} g_k(\chi)\boldsymbol{z}_k(\chi) , \quad \tilde{\chi} = \sum_{k=1}^{K} g_k(\chi)\tilde{\chi}_k(\chi) . \tag{107}$$

The gate is a multilayer perceptron which receives its inputs from the same delay line as the PCA experts.

Each PCA expert is capable of perfectly reconstructing the input (the PCA transformation is full rank), so there is no information to drive the competition. In order to create a difference in performance between the PCA experts, to drive the adaptation of the gate, one has to perform the reconstruction with fewer than the total number of eigenvectors. That is, *the reconstruction has to be done from a subspace*. However, one can still utilize the full set of eigenvectors or principal components as the overall system output, as may be required by PCA.

7.4 The pdf of reconstruction error

In the mixture of experts (92) is viewed as a statistical mixture model

$$p(\boldsymbol{d}|\chi) = \sum_{k=1}^{K} p(k|\chi) \cdot p(\boldsymbol{d}|\chi, k) = \sum_{k=1}^{K} g_k(\chi) \cdot p(\boldsymbol{d}|\chi, k) \tag{108}$$

where the kth gate output, $g_k(\chi)$, represents the apriori probability of the kth expert and $p(\boldsymbol{d}|\chi, k)$ is the conditional pdf of the desired signal, given the input and the parameters of the kth expert. It is tempting to try and model $p(\boldsymbol{d}|\chi, k)$ as a Gaussian distribution in the error between the desired signal and the kth expert's output, which for auto-association becomes

$$p(\boldsymbol{d}|\chi, k) = \frac{1}{(2\pi)^{M/2}\left|\Sigma_k\right|^{1/2}} \exp\left[-\frac{1}{2}(\chi - \tilde{\chi}_k)^T \Sigma_k^{-1} (\chi - \tilde{\chi}_k)\right] \tag{109}$$

where Σ_k is the weighted covariance matrix of the reconstruction error of the kth expert. Equation (109) is perfectly valid when the experts are non-linear auto-associators. However, as we will now show, when the experts are linear, Σ_k is not full rank and thus not invertible. Consider a single

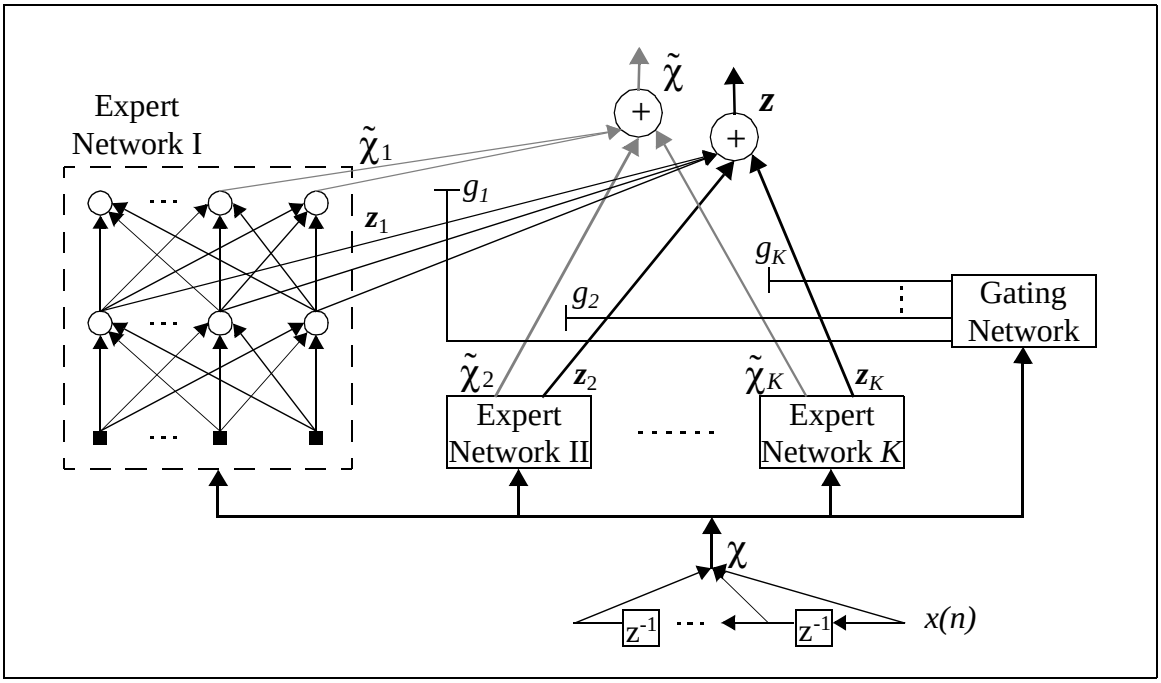

Figure 7. Gated competitive temporal PCA architecture.

PCA system with eigenvectors $\boldsymbol{U} = \left[\boldsymbol{U}_p \; \boldsymbol{U}_s\right]$ as previously defined. Noting that $\boldsymbol{U}\boldsymbol{U}^T = \boldsymbol{U}_p\boldsymbol{U}_p^T + \boldsymbol{U}_s\boldsymbol{U}_s^T = \boldsymbol{I}$, the reconstruction error of a single input can be written as $\chi(n) - \tilde{\chi}(n) = [\boldsymbol{I} - \boldsymbol{U}_p\boldsymbol{U}_p^T]\boldsymbol{x}(n) = \boldsymbol{U}_s\boldsymbol{U}_s^T\boldsymbol{x}(n)$, and the covariance of the reconstruction error is given by

$$\boldsymbol{\Sigma} = E[(\chi - \tilde{\chi})(\chi - \tilde{\chi})^T] = \boldsymbol{U}_s\boldsymbol{U}_s^T\boldsymbol{R}\boldsymbol{U}_s\boldsymbol{U}_s^T = \sum_{i=P}^{M-1} \lambda_i \boldsymbol{u}_i\boldsymbol{u}_i^T. \tag{110}$$

Thus, the rank of the covariance matrix is given by the number of secondary eigenvectors, $S = M - P$. Since the covariance matrix is square and of dimension *P*, it is not full rank for any dimensionality reduction. Ultimately, all such direct attempts at formulating PCA in a statistical framework fail because *PCA is not a model for the data but merely a transformation of the data based on a decomposition of the covariance matrix*. This is an undesirable situation because we would like to use the same statistical framework for both linear and non-linear auto-association. One way around this is to borrow some results from factor analysis (see Morrison, 1976 or Jolliffe, 1986). Factor analysis is a *model* that attempts to explain an observed high dimensional random vector in terms of an unobserved lower dimensional random vector. It can be shown that, under certain simplifying assumptions, factor analysis defaults to a PCA expansion. Factor analysis suggests that, for the linear PCA case, instead of modeling the reconstruction error, as in (108), we should directly model the pdf of the input as the weighted sum of the conditional pdf of the input given the *K* PCA models:

$$p(\chi) = \sum_{k=1}^{K} g_k(\chi) \cdot p(\chi|k) \tag{111}$$

However Fancourt and Principe (1998a) show that the conditional probability of the input given the PCA parameters of the kth expert can be approximated as,

$$p(\chi|k) \approx \frac{1}{(2\pi\sigma_k^2)^{S/2}} \exp\left[-\frac{\left\|\chi - \tilde{\chi}_k\right\|^2}{2\sigma_k^2}\right] \tag{112}$$

where the variance is given by the mean value of the secondary eigenvalues,

$$\sigma_k^2 = \frac{1}{S}\sum_{i=P}^{M-1} \lambda_{ki} = \frac{1}{S} E[\left\|\chi - \tilde{\chi}_k\right\|^2] \tag{113}$$

and λ_{ki} is the ith eigenvalue of the kth expert. We will see shortly how to determine the eigenvectors and eigenvalues of the experts. Note that the approximate factor analysis model, consisting of (111), (112), and (113), is equivalent to the model based on reconstruction error in (108) and (109) if we set the covariance of the reconstruction error to have the diagonal form

$$\Sigma_k = \sigma_k^2 \boldsymbol{I} \tag{114}$$

and assume *S* degrees of freedom. Thus, using (114), we can use the same reconstruction error model independent of whether the experts are linear or non-linear. It is natural then to question the consequences of not using the exact pdf in (112). First, even the exact formulation makes a Gaussian assumption, which may be violated by the data itself. Second, any approximation in the conditional pdf's of the experts can be accommodated by a small change in the mixture coefficients. Third, as we shall soon see, the approximation leads to a weighted mean square error derivation of the linear PCA parameters similar to the derivation for a single PCA system. Finally, the model works sufficiently well in practice.

7.5 Training the model

We now have all the pieces required to train the model, given a time series, $x(n)$, n=1...N. Again, we will use the EM algorithm but this time we skip a detailed description, since the principles are very much the same as in sections 3 and 4. In the E step, for an initial set of model parameters, the posterior probabilities are evaluated using

$$h_k(n) \equiv \frac{g_k(\chi(n)) \cdot p(\chi(n)|k)}{\sum_{k=1}^{K} g_k(\chi(n)) \cdot p(\chi(n)|k)} \tag{115}$$

where $p(\chi|k)$ is given by the approximation in (112). In the M step, the cost function

$$J \approx \sum_{n=1}^{N} \sum_{k=1}^{K} \left\{ -h_k(n) \cdot \log[g_k(n)] + \frac{1}{2} h_k(n) \left[\frac{\left\| \chi(n) - \tilde{\chi}_k(n) \right\|^2}{\sigma_k^2} + S \cdot \log[\sigma_k^2] \right] \right\} \tag{116}$$

is globally minimized over the free parameters. We have written this as an approximation because the input vectors, $\chi(n)$, are *not* i.i.d. as a result of being delayed versions of the time series.

We now minimize (116) with respect to the free parameters. We skip many intermediate steps, but see Fancourt and Principe (1998a) for a more complete development. Recall that if the gate is a multi-layer perceptron, the error backpropagated to the input side of the softmax is given by (80), and (116) is usually only decreased and not minimized. However, since PCA is a linear transformation, we can globally minimize the second part of (116) with respect to the experts' weights. Furthermore, since the experts have been decoupled in (116), we can this for each expert separately. Let the weights of the kth expert be given by some unknown $P \times M$ matrix $\boldsymbol{W}_k$ so that the reconstruction error is given by

$$\chi(n) - \tilde{\chi}_k(n) \;=\; (\boldsymbol{I} - \boldsymbol{W}_k \boldsymbol{W}_k^T)\chi(n) + \boldsymbol{b}_k \tag{117}$$

where we have included the unknown constant vector $\boldsymbol{b}_k$ to allow for the possibility that χ is not zero-mean. Then the cost function of the kth expert is

$$\begin{aligned} J_k \;&=\; \frac{1}{2} \sum_{n=1}^{N} h_k(n) \left\| \boldsymbol{x}(n) - \tilde{\boldsymbol{x}}_k(n) \right\|^2 \\ &=\; \frac{1}{2} \sum_{n=1}^{N} h_k(n) \{ \boldsymbol{x}^T(n)(\boldsymbol{I} - \boldsymbol{W}_k \boldsymbol{W}_k^T)\boldsymbol{x}(n) + 2\boldsymbol{x}^T(n)(\boldsymbol{I} - \boldsymbol{W}_k \boldsymbol{W}_k^T)\boldsymbol{b}_k + \boldsymbol{b}_k^T \boldsymbol{b}_k \} \end{aligned} \tag{118}$$

Finding the constant $\boldsymbol{b}_k$ first,

$$\boldsymbol{b}_k \;=\; -(\boldsymbol{I} - \boldsymbol{W}_k \boldsymbol{W}_k^T)\bar{\chi}_k \tag{119}$$

where $\bar{\chi}_k$ is the mean of χ_k weighted by the posterior probabilities

$$\bar{\chi}_k \equiv \frac{1}{\sum_{n=1}^{N} h_k(n)} \sum_{n=1}^{N} h_k(n)\chi(n) \,. \tag{120}$$

where $\boldsymbol{R}_k$ is a weighted autocorrelation matrix

$$\boldsymbol{R}_k \equiv \frac{1}{\sum_{n=1}^{N} h_k(n)} \sum_{n=1}^{N} h_k(n)[\chi(n) - \bar{\chi}_k][\chi(n) - \bar{\chi}_k]^T \,. \tag{121}$$

The $\boldsymbol{w}_{kj}$ turn out to be the eigenvectors of $\boldsymbol{R}_k$,

$$\boldsymbol{R}_k \boldsymbol{w}_{kj} = \lambda_{kj} \boldsymbol{w}_{kj} \qquad j = 1 \ldots P \tag{122}$$

but chosen so that the eigenvectors correspond to the largest eigenvalues. That is to say, for the optimal reconstruction of the kth expert at each *M* step, we should choose the *P* largest eigenvectors of $\boldsymbol{R}_k$. *This clearly shows that the competitive PCA architecture is still solving an eigenvalue problem*. Furthermore, if the posterior probabilities converge to binary values indicating the stationary regions of the time series, then each experts' autocorrelation matrix and eigenvectors will correspond to a different stationary regime.

Finally, at each iteration, the variance of each experts' pdf is set to the average of the discarded eigenvalues, according to (113). We now summarize the algorithm in pseudo-code:

For each iteration, *i*:

E-Step:

For each vector in the data set, $n = 1 \ldots N$

1. Calculate the gate outputs: $g_k(\chi(n))$.

For each expert, $k = 1 \ldots K$

2. Calculate the reconstruction estimate: $\tilde{\chi}_k(n) = \boldsymbol{W}_k \boldsymbol{W}_k^T \chi(n) + (\boldsymbol{I} - \boldsymbol{W}_k \boldsymbol{W}_k^T)\bar{\chi}_k$.

3. Evaluate the pdf of each expert: $p(\chi(n)|k) = \frac{1}{(2\pi\sigma_k^2)^{S/2}} \exp\left[-\frac{\left\|\chi(n) - \tilde{\chi}_k(n)\right\|^2}{2\sigma_k^2}\right]$.

4. Calculate the posterior probabilities: $h_k(n) = \frac{g_k(n) \cdot p(\chi(n)|k)}{\sum_{j=1}^{K} g_j(n) \cdot p(\chi(n)|k)}$

M-Step:

5. Train the gate using the posterior probabilities, $h_k(n)$, as targets at the input side of the soft-max output layer.

For each expert, $k = 1 \ldots K$

6. Calculate the conditional means: $\bar{\chi}_k = \left[\sum_{n=1}^{N} h_k(n)\right]^{-1} \sum_{n=1}^{N} h_k(n)\chi(n)$.

7. Calculate the autocorrelation matrices: $\boldsymbol{R}_k = \left[\sum_{n=1}^{N} h_k(n)\right]^{-1} \sum_{n=1}^{N} h_k(n)[\chi(n) - \bar{\chi}_k][\chi(n) - \bar{\chi}_k]^T$.

8. Calculate the eigenvectors, $j = 1 \ldots P$: $\boldsymbol{R}_k \boldsymbol{w}_{kj} = \lambda_{kj} \boldsymbol{w}_{kj}$.

9. Calculate the variances, $\sigma_k^2 = \frac{1}{S} \sum_{i=P}^{M-1} \lambda_{ki}$.

The overall output, as defined by (107), can be calculated after the algorithm has converged.

Once again, the gate only provides local segmentation information. Thus, the posterior probabilities can be used in conjunction with the classical segmentation algorithms of section two, exactly as was done for the mixture models of sections three and four.

8 Output Based Gating

We've already discussed how the output of the gate provides is insufficient to provide long term segmentation information, and for this reason the unsupervised mixture models are used in conjunction with supervised change detection algorithms, resulting in a modeling-segmentation-modeling iteration. We now consider architectures that are capable of segmenting signals without requiring the use of an external segmentation algorithm. Clearly, such architectures will require additional memory over and above that encapsulated in the experts, in order to monitor the longer term performance of the experts. Through the use of memory, an expert that has performed well in the recent past can be given an advantage in the present. In general, memory can be added either to the gate or the experts themselves. Weigend (1995) briefly discusses adding memory to the gate in the mixture of experts, in the form of a time delayed feedback of the gate outputs. Here, we will restrict ourselves to output based gates.

8.1 Local performance estimates

One way of adding memory to the gate is to have it monitor the local performance of the experts. The simplest short term performance measure is the local mean square error

$$\varepsilon(n) = \frac{1}{\tau}\sum_{m=0}^{\tau-1} e^2(n-m) \tag{123}$$

where the instantaneous error $e(n)$ is the difference between the system output and the desired output and τ is the window width. This can also be estimated recursively as

$$\varepsilon(n) = \lambda e^2(n) + (1-\lambda)\varepsilon(n-1) \qquad 0<\lambda<1\,. \tag{124}$$

The parameter λ controls the memory depth of the filter. There are several ways to define the effective memory of a recursive system, but the simplest is to calculate the average of the temporal index weighted by the impulse response (Principe et al., 1993). According to this definition, it is easy to show that the effective memory depth of (124), in samples, is given by λ^{-1}. Taking the expected value of both sides of (124) and simplifying shows that it is an unbiased estimate of the mean square error: $E[\varepsilon] = E[e^2]$.

For an on-line adaptive system, the change of some adaptable weight, *w,* is proportional to the instantaneous gradient

$$\frac{\partial}{\partial w}\varepsilon(n) = 2\lambda\frac{\partial}{\partial w}e(n) + (1-\lambda)\frac{\partial}{\partial w}\varepsilon(n-1)\,. \tag{125}$$

Viewed in this way, (125) is equivalent to *learning with momentum*. In the neural network literature, momentum learning is presented as an ad-hoc way of speeding up learning. Here, we see it is the natural consequence of a recursive mean square error cost function.

We now examine ways to incorporate these local performance measures into the gate.

8.2 Annealed competition of experts

Perhaps the most successful examples of output based gating is the annealed competition of experts (ACE) algorithm of Muller et al. (1995). A set of *K* expert neural predictors operate in

parallel, and a gating function determines the probabilities of the experts. The total cost function is based on (93) which uses the standard squared error for the overall cost function. The competitive framework provided by the gate is a Gaussian function of the local mean square error,

$$g_k(n) = Ae^{-\beta\varepsilon_k(n)} \tag{126}$$

where A is a normalization constant, β is an annealing term which determines the degree of competition, and ε_k is the local mean squared error of the kth expert, calculated using a *non-causal* boxcar filter

$$\varepsilon_k(n) = \sum_{m=-\tau}^{\tau} e_k^2(n-m) \tag{127}$$

where e_k is prediction error of the kth expert. Furthermore, the gradient of the cost function with respect to some weight, w_k, in the kth expert, is calculated treating the gate as a constant

$$\Delta w_k = -\eta\frac{\partial J}{\partial w_k} = -\eta\sum_{n=1}^{N} g_k(n)e_k(n)\frac{\partial y_k}{\partial w_k} \tag{128}$$

where η is the learning rate. Thus, the gate effectively represents a weighting term to the learning rate of the individual experts.

The algorithm must be annealed during training. By starting the learning process with a small value of β, all the predictors see the entire data set and converge to an average of all the dynamical systems. Then as β is increased and the competition becomes harder, each predictor begins to specialize on a single stationary region of the input, which then puts it at a disadvantage for other stationary dynamical regions, allowing other predictors to win those regions. Muller et al. report "phase transitions", specific values of β for which a dramatic change in the cost occurs as the experts begin to specialize. There may be several such phase transitions in the course of training. As $\beta\rightarrow\infty$, the competition becomes hard, and only the winning predictor earns the right to update its prediction, at which point the experts are "fine tuning" their modeling of their respective regimes.

Using non-linear neural predictors, they have successfully applied this algorithm to the segmentation and identification of switching chaotic maps and speech. However, the algorithm is still dependent on manual fine tuning to experimentally determine the memory depth of the moving average filter, and the annealing rate.

There are two key issues that are essential for the successful application of the algorithm. The first is the memory term, in the form of a lowpass "boxcar" filter, for the instantaneous squared error (127). In cases where there is a large overlap between the return maps of the various dynamical regimes, or a small signal to noise ratio, a single time instance is insufficient to estimate the probabilities of the experts. This can result in rapid switching between predictors. However, under the assumption of a relatively slow switching rate, a single predictor would be expected to win for a longer time period. Increasing the memory depth gives a greater advantage to the predictor that has won in the recent past. The acausal nature of the filter is not an hindrance because of the off-

line nature of the algorithm, but ensures that regime switches indicated by the gating function are aligned with the data.

8.3 Self-annealing competitive prediction

Recall that the ACE algorithm uses an output based gate with memory. The memory is in the form of a boxcar moving average (MA) of the local squared error of the experts, to which a Gaussian function is applied. The variance of the Gaussian function determines the degree of competition. In the ACE algorithm, memory is used in the gating function only. Here we propose an instantaneous cost function that imbeds memory in both the gate and the experts' mean square error

$$J(n) = \sum_{k=1}^{K} g_k(n)\varepsilon_k(n) \tag{129}$$

where ε_i is the local recursive mean square error of the i^{th} predictor, repeated here for convenience

$$\varepsilon_k(n) = \lambda e_k^2(n) + (1-\lambda)\varepsilon_k(n-1) \qquad 0 < \lambda < 1\,. \tag{130}$$

Again, $e_k(n)$ is the instantaneous error of the k^{th} expert. All the experts use the same value for the memory parameter, λ, but the algorithm could be modified to allow for differing memories.

Turning to the gate, in the ACE algorithm, the degree of competition within the gate, as expressed through the annealing parameter, and the memory depth of the squared error are uncoupled. And yet, it is intuitive that the longer the time period over which we collect information about the performance of the experts, the better should be our judgement about which expert is valid. Such a coupling eliminates the need for separate annealing of the memory depth and the competition. Niedzwiecki (1992) has in fact formulated such a relationship. Assuming that the variances of the errors of the experts are unknown but distributed according to a so-called non-informative prior distribution, Niedzwiecki's formula for the probability of the i^{th} expert is given by

$$g_k(n) = \frac{\varphi_k(n)}{\sum_{j=1}^{K} \varphi_j(n)} \tag{131}$$

where the unnormlized gate output is given by

$$\varphi_k(n) = [\varepsilon_k(n)]^{\frac{-\tau}{2}} \tag{132}$$

and ε_i is the causal boxcar MA of the squared error

$$\varepsilon_k(n) = \frac{1}{\tau}\sum_{m=0}^{\tau-1} e_k^2(n-m)\,. \tag{133}$$

The unnormalized gate given by equation (132) embodies the coupling between memory depth and degree of competition. When τ=1, the unnormalized gate is just the inverse of the absolute value of the instantaneous error $\varphi_i(n) = |e_i(n)|^{-1}$. As $\tau \to \infty$, the gate implements hard competition such that $g_i(n)$=1 when i represents the expert with the smallest mean square error, and $g_i(n)$=0 for all others.

Niedzwiecki developed his formulas for evaluating the performance of *known* experts. Here, we borrow Niedzwiecki's results and incorporate it into our cost function (129) for adaptable experts. We use (131) as the gating function, except that we replace the moving average calculation of the mean square average in (133) by our recursive calculation in (130). Likewise, we replace the memory depth, τ, in (132) by the memory parameter, λ^{-1},

$$\varphi_k(n) = [\varepsilon_k(n)]^{\frac{-1}{2\lambda}}. \tag{134}$$

We then do gradient descent on *both* the calculated gate and the mean square errors of (129). Taking the partial derivative with respect to any free parameter, w_k, in the kth system, results in

$$\Delta w_k = -\eta \frac{\partial}{\partial w_k} E(n) = -\eta g_k(n) \left[\frac{J(n)}{\varepsilon_k(n)} + 2\lambda - 1 \right] z_k(n), \tag{135}$$

$$z_k(n) = e_k(n) \frac{\partial}{\partial w_k} y_k(n) + (1-\lambda) z_k(n-1), \tag{136}$$

and with respect to the λ parameter

$$\Delta\lambda = \frac{-\eta'}{2\lambda} \sum_{k=1}^{K} g_k(n) \left\{ \left(\frac{J(n)}{\varepsilon_k(n)} + 2\lambda - 1 \right) v_k(n) - \frac{(J(n) - \varepsilon_k(n)) \log[\varepsilon_k(n)]}{\lambda} \right\}, \tag{137}$$

$$v_k(t) = \frac{\varepsilon_k(n) - \varepsilon_k(n-1)}{\lambda} + (1-\lambda) v_k(n-1). \tag{138}$$

Equation (136) for z_k is the standard weight update equation with momentum for the kth expert operating independently. Thus, momentum learning, which has been previously presented as an ad-hoc way to speed up and stabilize neural network training, falls out naturally as a result of using the mean square error in the cost function (129) instead of the instantaneous error.

The competitive nature of the algorithm is evident in (135), where the total weight update is given by the product of z_k with the probability of the kth expert, g_k, and another term which depends on the inverse of the mean square error of the kth expert. Thus, the expert with the smallest mean square error will have the largest weight update. Furthermore, (135) also shows how the annealing is coupled with the memory depth. For λ>0.5, the term in brackets is always positive, and thus all the experts move towards the data to improve their predictions. However, For λ<0.5, the sign of the term in brackets can be either positive or negative, depending on whether the mean square error of the kth predictor is less or greater than, respectively, the total cost. Thus, experts

that perform poorly can actually be pushed away from the data, although at a small learning rate due to the gating function.

8.4 Neighborhood map of competing predictors

Here we present a novel competitive structure, shown in Figure 8. The experts, in this case predictors, are located on a spatial grid, analogous to a Kohonen (1982) Map, and the gate moderates the learning of the experts based on their distance from the best performing expert. The best predictor is the one with the smallest local mean squared error

$$winner(n) = \arg\min_k[\varepsilon_k(n)] \tag{139}$$

where the local mean square error is computed using the recursive estimate for each expert, repeated here for convenience

$$\varepsilon_k(n) = \lambda e_k^2(n) + (1-\lambda)\varepsilon_k(n-1) \qquad 0 < \lambda < 1 \tag{140}$$

where e_k is the instantaneous squared error of the kth predictor. The memory term, λ, is identical for all experts, but can be manually adjusted during training. The gating function, which moderates the learning rate of the predictors, is determined by the distance from the winning predictor, k^*, to the other predictors

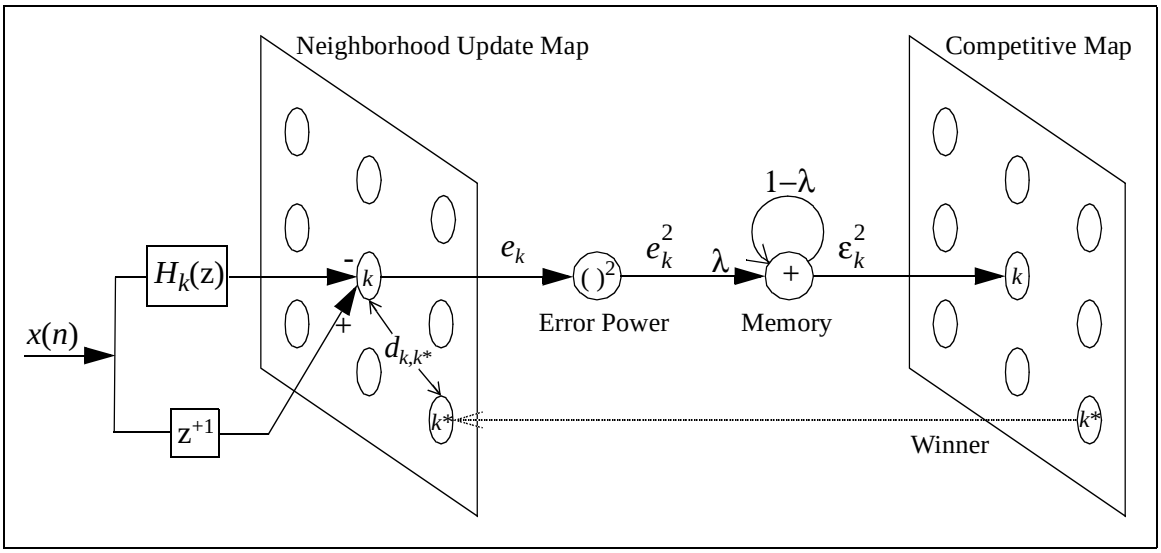

Figure 8. Neighborhood map of competing predictors.

$$g_k(n) = \exp\left[-\frac{d_{k,\,k^*}^2(n)}{2\sigma^2(n)}\right] \tag{141}$$

where k is the predictor to be updated, k^* is the winning predictor, d_{k,k^*} is the neighborhood distance between them, and σ is an annealing parameter which controls the neighborhood width. This results in the model shown in Figure 8. The signal flow graph is shown for only one predictor.

In exact analogy with training a Kohonen map, both the neighborhood width and the global learning rate are annealed during training according to an exponentially decreasing schedule

$$\sigma(t) = \sigma_0 e^{\frac{-i}{\tau}} \qquad \mu(t) = \mu_0 e^{\frac{-i}{\tau}} \tag{142}$$

where τ is the annealing rate and *i* is the iteration number. The overall learning rate for the k^{th} predictor at the i^{th} iteration is

$$\eta_k(i) = \mu(i) \cdot g_k(i). \tag{143}$$

9 Simulations

10 Conclusions

We decomposed the multivariate pdf for a random process with finite memory into a product of conditional densities that are local in time, showed how prediction can be used to evaluate the conditional densities, and then expressed them as a mixture of pdf's that were local in signal space (the predictors). The mixing coefficients were necessarily a function of the recent past of the signal and this led naturally to the concept of a gate and the mixture of experts algorithm. We showed how the model could be trained using the EM algorithm, and how the posterior mixture probabilities after training could be used to segment the time series using the CUSUM algorithm.

Etc.

11 References

Andersson P., Adaptive forgetting in recursive identification through multiple models, *Int. J. Control*, vol. 42, no. 5, pp. 1175-1193, 1985.

Andre-Obrecht R., A new statistical approach for the automatic semgnetation of continuous speech signals, *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 36, no. 1, pp. 29-40, 1988.

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

Arfken G., *Mathematical Methods for Physicists*, Academic Press, Orlando, 1985.

Basseville M., Detecting changes in signals and systems - A Survey, *Automatica*, vol. 24, no. 3, pp. 309-326, 1988.

Basseville M. & Benveniste A., Sequential detection of abrupt changes in spectral characteristics of digital signals, *IEEE Trans. on Information Theory*, vol. IT-29, no. 5, 1983.

Basseville M. and Nikiforov I.V., *Detection of Abrupt Changes, Theory and Application*, Prentice-Hall, Englewood Cliffs, 1993.

Baum, L.E., An inequality and associated maximization in statistical estimation for probabilistic functions of markov processes, *Inequalities* **3**, 1-8, 1972

Bishop C.M., *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford, 1995.

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

Brandt A.V., Detecting and estimating parameter jumps using ladder algorithms and likelihood ratio tests, *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 1017-1020, 1983.

Brillinger D., *Time Series: Data Analysis and Theory*, Holden-Day, San Francisco, 1975.

Broomhead D.S. & King G.P., Extracting qualitative dynamics from experimental data, *Physica D*, vol. 20, no. 2-3, pp. 217-236, 1986.

Casdagli M., Eubank S., Farmer J.D., & Gibson J., State space reconstruction in the presence of noise, *Physica D*, vol. 51, pp. 52-98, 1991.

Chu C., Time series segmentation: A sliding window approach, *Information Sciences*, vol. 85, pp. 147-173, 1995.

Danilov D., Solnsev V., & Zhigljavsky A., Analysis and forecast of time series on the base of principal components, *Second Int. Conf. on Computing in Economics and Finance*, Geneva, Switzerland, 1996.

Dempster A.P., Laird N.M., & Rubin D.B., Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistical Society*, Series B, 39, 1-38, 1972.

Dony R. & Haykin S., Optimally adaptive transform coding, *IEEE Trans. on Image Processing*, vol. 4, no. 10, 1995.

Doutriaux A. & Zipser D., Unsupervised discovery of speech segments using recurrent nets, *Proc. of the 1990 Connectionist Models Summer School*, pp. 303-309, 1990.

Elman J.L., Finding structure in time, *Cognitive Science*, vol. 14, pp. 179-211, 1990.

Fahlman S.E. & C. Lebiere, The cascade correlation learning architecture, *Advances in Neural Information Processing Systems 2*, pp. 524-532, 1990.

Fancourt C. & Principe J., Competitive principal component analysis for locally stationary time series, *IEEE Trans. on Signal Processing*, vol. 46, no. 11, pp. 3068-3081, 1998a.

Fancourt C. & Principe J., Modeling time dependencies in the mixture of experts, *Proc. of the IEEE Int. Joint Conf. on Neural Networks (IJCNN)*, vol. 3, pp. 2324-2327, 1998b.

Fancourt C. & Principe J., Temporal self-organization through competitive prediction, *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 4, pp. 3325-3328, 1997a.

Fancourt C. & Principe J., Soft competitive principal component analysis using the mixture of experts, *Image Understanding Workshop*, vol. 2, pp. 1071-1075, 1997b.

Fancourt C. & Principe J., A neighborhood map of competing one step predictors for piecewise segmentation and identification of time series, *IEEE Int. Conf. on Neural Networks (ICNN)*, vol. 4, pp. 1906-1911, 1996a.

Fancourt C. & Principe J., Temporal self-organization through competitive prediction, *Proceedings of the Ninth Yale Workshop on Adaptive and Learning Systems*, pp. 8-12, 1996b.

Friedman, J.H., Multivariate adaptive regression splines, *Annals of Statistics* **19**(1), 1-67, 1991

Fukunaga K., *Statistical Pattern Recognition*, Academic Press, 1990.

Gibson J. F., Farmer J.D., Casdagli M., & Eubank S., *An Analytic Approach to Practical State Space Reconstruction*, Santa Fe Institute Report, 92-04-021, 1992.

Hamilton, J.D., A new approach to the economic-analysis of nonstationary time-series and the business-cycle, *Econometrica* **57**(2), 357-384, 1989.

Hamilton, J.D., Analysis of time-series subject to changes in regime, *Journal of Econometrics* **45**(1-2), 39-70, 1990.

Hathaway, R.J., A constrained EM-algorithm for univariate normal mixtures, *Journal of Statistical Computation and Simulation* **23**(3), 211-230, 1986.

Haykin S., *Neural Networks: A Comprehensive Foundation*, New York: Macmillan, 1994.

Jacobs, R.A., Bayesian approach to model selection in hierarchical mixtures-of-experts architectures, *Neural Networks*, vol. 10, no.2, pp. 231-241, 1997.

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

Jolliffe I., *Principal Component Analysis*, Springer-Verlag, New York, 1986.

Jordan M.I. & Jacobs R.A., Hierarchical mixtures of experts and the EM algorithm, *Neural Computation*, vol. 6, pp. 181-214, 1994.

Jordan M.I. & Xu L., Convergence results for the EM approach to mixtures of experts architectures, *M.I.T. Artificial Intelligence Lab*, A.I. Memo No. 1458, 1993.

Kadirkamanathan V., Recursive nonlinear identification using multiple model algorithm, *Neural Networks for Signal Processing*, pp. 171-180, 1995.

Kehagias A. & Petridis V., Predictive modular neural networks for time series classification, *Neural Networks*, vol. 10, no. 1, 31-49, 1997.

Keresteciogl u F., *Change Detection and Input Design in Dynamical Systems*, Research Studies Press, Somerset, England, 1993.

Klein J.L., *Statistical Visions in Time*, Cambridge University Press, Cambridge, United Kingdom, 1997.

Kligiene N. & Telksnys L., Methods of detecting instants of change of random process properites, *Automation and Remote Control*, vol. 44, no. 10, Part II, pp. 1241-1283, 1983.

Kohlmorgen J., Muller K.-R., & Pawelzik K., Improving short-term prediction with competing experts, *ICANN'95: Proc. of the Int. Conf. on Artificial Neural Networks*, EC2 & Cie, Paris, 2:215-220, 1995.

Kohlmorgen J., Muller K.-R., & Pawelzik K., Segmentation and identification of drifting dynamical system, *Proc. of the 1997 7th IEEE Workshop on Neural Networks for Signal Processing*, NNSP'97, pp. 326-335 , 1995.

Kohonen T., Self-organized formation of topologically correct feature maps, *Biological Cybernetics*, vol. 43, pp. 59-69, 1982.

Kohonen T., The self-organizing map, *Proc. IEEE*, vol. 78, no. 9, pp. 1464-1480, 1990.

Lakkis I. & McLernon D., Optimum eigenfilters and matched filters, *Electronics Letters*, vol. 32, no. 22, pp. 2068-2070, 1996.

Lawley D.N., A modified method of estimation in factor analysis and some large sample results, *Uppsala Symposium on Psychological Factor Analysis*, pp. 35-42, 1953.

Levin E., Modeling time varying systems using hidden control neural architecture, *Advances in Neural Information Processing Systems 3*, pp. 147-154, 1990.

Makhoul J., On the eigenvectors of symmetric Toeplitz matrices, *IEEE Tran. on Acoustics, Speech, and Signal Processing*, vol. ASSP-29, pp. 868-872, 1981.

Michalopoulou Z.H., Nolte L.W., & Alexandrou D., Performance evaluation of multilayer perceptrons in signal detection and classification, *IEEE Trans. on Neural Networks*, vol. 6, no. 2, pp. 381-386, 1995.

Morrison D., *Multivariate Statistical Methods*, McGraw-Hill, 1976.

Muller K.R., Kohlmorgen J., & Pawelzik K., Analysis of switching dynamics with competing neural networks, *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E78-A, no. 10, pp. 1306-1315, 1995.

Narendra K.S. & Balakrishnan J., Adaptive control using multiple models, *IEEE Trans. on Automatic Control*, vol. 42, no. 2, pp. 171-187, Feb 1997.

Niedzwiecki M., Identification of nonstationary stochastic systems using parallel estimation schemes, *IEEE Trans. on Automatic Control*, vol. 35, no. 3, pp. 329-334, 1990.

Niedzwiecki M., Multiple model approach to finite memory adaptive filtering, *IEEE Trans. on Signal Processing*, vol. 40, no. 2, pp. 470-473, 1992.

Niedzwiecki M., Identification of time-varying systems with abrupt parameter changes, *Automatica*, vol. 30, no. 3, pp. 447-459, 1994.

Page E.S., A test for a change in a parameter occurring at an unknown point, *Biometrika* **42**, 523-527, 1955.

Page E.S., On problems in which a change in a parameter occurs at an unknown point', *Biometrika* **44**, 248-52, 1957a.

Page E.S., Estimating the point of change in a continuous process, *Biometrika*, vol. 44, pp. 248-252, 1957b.

Pawelzik K., Kohlmorgen J., & Muller K.R., Annealed competition of experts for a segmentation and classification of switching dynamics, *Neural Computation*, vol. 8, no. 2, pp. 340-356, 1996.

Petridis V. & Kehagias A., Modular neural networks for MAP classification of time series and the partition algorithm, *IEEE Transactions on Neural Networks* **7**(1), 73-86, 1996.

Plataniotis K.N., Androutsos D., Venetsanopoulos A.N., & Lainiotis, D.G., New time series classification approach, *Signal Processing*, vol. 54, no. 2, pp. 191-199, 1996.

Priestlley M.B., *Non-linear and non-stationary time series analysis*, Academic Press, San Diego, 1988.

Principe J., deVries B., & Guedes de Oliveira P. The gamma filter: a new class of adaptive IIR filters with restricted feedback. *IEEE Trans. Signal Processing*, 41(2):649-656, 1993.

Quandt R.E., A new approach to estimating switching regressions, *Journal of the American Statistical Association* **67**, pp. 306-310, 1972.

Quandt R.E. & Ramsey J.B., Estimating mixtures of normal distributions and switching regressions, *Journal of the American Statistical Association* **73**(364), pp. 730-752, 1978.

Rabiner L. & Juang B.H., *Fundamentals of Speech Recognition*, Prentice Hall, Englewood Cliffs, 1993.

Ramamurti V. & Ghosh J., Structurally adaptive modular networks for nonstationary environments, *IEEE Transactions on Neural Networks*, vol. 10, no. 1, pp. 152-160, 1999.

Sanger T., Optimal unsupervised learning in a single-layer linear feedforward neural network, *Neural Networks*, vol. 2, pp. 459-473, 1989.

Shi S. & Weigend A.S., Markov gated experts for time series analysis: beyond regression, *IEEE Int. Conf. on Neural Networks*, ICNN97, vol. 4, pp. 2039-2044, 1997.

Slepian D., Prolate spheriodal wave functions, Fourier analysis, and uncertainty-V: the discrete case, *Bell System Technical Journal*, vol. 57, no. 5, 1978.

Srivastava A.N. & Weigend A.S., Improved time series segmentation using gated experts with simmulated annealing, *Proc. of the 1996 IEEE Int. Conf. on Neural Networks (ICNN)*, vol. 4, pp. 1883-1888, 1996.

Thomson D.J., Spectrum estimation and harmonic analysis, *Proc. of the IEEE*, vol. 70, no. 9, pp. 1055-1096, 1982.

Tipping M.E. & Bishop C.M., Mixtures of probabilistic principal component analyzers, *Neural Computation*, vol. 11, no. 42, 1999.

Tong H.& Lim K.S., Threshold autoregression, limit cycles, and cyclical data, *J. Roy. Stat. Soc. B*, vol. 42, pp. 245-292, 1980.

Van Trees H., *Detection, Estimation and Modulation Theory: Part I*, Wiley, New York, 1968.

Waterhouse S.R & Robinson A.J., Pruning and growing hierarchical mixtures of experts, *Proc. of the 4th Int. Conf. on Artificial Neural Networks*, n 409, pp. 341-346, 1995.

Weigend A.S & Gershenfeld N.A., ed., *Time series prediction: forecasting the future and understanding the past*, Addison-Wesley, Reading, MA, 1990.

Weigend A.S., Mangeas M., & Srivastava A.N., Nonlinear gated experts for time series: discovering regimes and avoiding overfitting, *International Journal of Neural Systems*, vol. 6, no. 4, 1995.

Widrow B. & Stearns S.D., *Adaptive Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1985.

Willsky A.S. & Jones H.L., A generalized likelihood ratio approach to detection and estimation of jumps in linear systems, *IEEE Trans. Automatic Control*, vol. AC-21, no.1, pp. 108-112.

Xu L., Signal segmentation by finite mixture model and EM algorithm, *Proc. Int. Symp. Artif. Neural Net*s, ISANN'94, pp. 453-458, 1994.

Zeevi A., Meir R. & Adler R., Time series prediction using mixtures of experts, *Advances in Neural Information Processing Systems* 9, pp. 309-315, 1997.