Ath. Kehagias and V. Petridis.
"Predictive Modular Neural Networks for Time Series Classification ".

This paper has appeared in the journal:
Neural Networks, Vol.10, pp.31-49, 1997.

Predictive Modular Neural Networks for Time Series Classification

Ath. Kehagias and Vas. Petridis
Div. of Electronics and Comp. Eng., Dept. of Electrical Eng.,
Aristotle University of Thessaloniki
540 06 Thessaloniki, Greece
e-mail: kehagias@vergina.eng.auth.gr

May 31, 1997

Abstract

A predictive modular neural network (PREMONN) architecture for time series classification is presented. The PREMONN has a hierarchical structure. The bottom level consists of a bank of linear or nonlinear predictor modules. The top level is a decision module which employs Bayesian or nonprobabilistic decision rules. For various choices of prediction and decision modules, convergence to correct classification is proven. Also it is shown that PREMONN is robust to noise and the speed / accuracy tradeoff is investigated. The analysis is mainly mathematical; however, to corroborate our conclusions we also present classification experiments.

This paper appeared in Neural Networks, Vol.10, No.1, pp.31-49, 1997.

1 Introduction

Let us consider the following time series classification problem. A time series y_t, $t = 1, 2, ...$ [1] is produced by a source $S(\theta_k)$, where θ_k is a parameter taking values in a *finite* set $\Theta = \{\theta_1, ..., \theta_K\}$. The source that produces the time series should be identified; in other words the "true" or "best" value of θ_k should be computed.

For example, the time series y_1, y_2, ... may be a speech signal, and the parameter θ_k a phoneme label taking values in the set Θ= { [ah], [oo], ... }. When the speech signal, for instance, corresponds to the phoneme [ah], y_1, y_2, ... is produced by S([ah]). This type of problem arises in phoneme recognition (Waibel et al., 1989), radar (Haykin & Cong, 1991) and sonar (Gorman & Sejnowski, 1988) signal classification, processing of seismic signals (Lainiotis et al., 1988), EEG (Zetterberg et al., 1981)and ECG (Zhu et al., 1989) analysis etc. A Bayesian approach to classification is quite common.

A significant trend of current connectionist research (for classification as well as other problems) is the development of training algorithms that partition the parameter space into regions, and fit a local model to each region. This approach is implicit in many earlier papers (Farmer & Sidorowich; 1988, Moody, 1989) but is recently being reexamined and formulated in the more suggestive language of *probability mixtures*, *local experts* (Jacobs et al., 1991; Jordan & Jacobs, 1992; Neal, 1991; Nowlan, 1990), *committees* (Schwarze & Hertz, 1992), *neural ensembles* (Perrone & Cooper, 1993) etc. More important than the change in vocabulary, is the development of new, adaptive methods for efficient combination of the local models, as in (Ayestaran & Prager, 1993; Baxt, 1992; Beyer and Smieja, 1993; Jacobs et al., 1991; Jordan & Jacobs, 1992; Kadirkamanathan & Niranjan, 1992; Schwarze & Hertz, 1992) etc. We find *Bayesian model combination* methods particularly interesting. Our work lies in this direction and has common elements with (Jacobs et al., 1991; Jordan & Jacobs, 1992; Neal, 1991; Nowlan, 1990; Shadafan & Niranjan, 1993).

Our first approach to time series classification is based on Bayesian combination of local predictors. This draws on the classical *Partition Algorithm* first used in the context of control theory (Hilborn & Lainiotis, 1969; Lainiotis, 1971). We have introduced its use to time series classification in a previous paper (Petridis & Kehagias, 1993). The space of time series sources is partitioned into sets and a local prediction module is trained for each set. It is assumed that the training data can be separated into well defined classes and used for offline training. Each source is assigned a Bayesian posterior probability, which is adaptively updated on-line, according to the predictive performance of the corresponding prediction module. [2] Finally, the time series is classified to the source of highest posterior probability.

This approach results to a modular, hierarchical neural network with a *prediction* level at the bottom and a *decision* level at the top (see Fig. 1). This general architecture can be modified at

[1] In what follows we use interchangeably the notations "y_t, $t = 1, 2, ...$" or "$y_1, y_2, ...$" or "$\{y_t\}_{t=1}^{\infty}$" to denote a time series.

[2] The computed value of the posterior probability also depends on a prior probability, selected before classification starts, according to our prior knowledge. However, we prove in the Appendix that asymptotic classification performance (i.e. convergence to correct classification) does not depend on the choice of prior probabilities.

both the prediction and decision levels. For instance, at the prediction level linear, sigmoid, radial basis, polynomial etc. modules can be used. At the decision level a variety of credit functions other than the probabilistically motivated Bayesian can be used. Thus a framework for the design of a family of classification networks is established, which we call PREMONNs (*PRedictive MOdular Neural Networks*). These can also be used for prediction: in this case the classification results are employed in an intermediate step.

This paper is mainly devoted to a mathematical analysis of the properties of PREMONNs. In Section 2 a Bayesian approach to time series classification is presented. In the following sections our main result, which is convergence of the posterior probabilities (or, in general, figures of merit), is given. This result is proven for Bayesian decision rule and nonlinear prediction modules in Section 3; for Bayesian decision rule and linear prediction modules in Section 4; and for general decision rule and nonlinear prediction modules in Section 5. In Section 6 the general principles of the PREMONN architecture are discussed. Further, in Section 7 a prediction network is presented, which is obtained from a classifier PREMONN by combining the output of local prediction modules. It is proven that this composite prediction is convergent and has minimum square error. Finally, in Section 8 we include an informal discussion of implementation issues and in Section 9 a number of numerical experiments, that corroborate our conclusions, are given.

2 Bayesian Time Series Classification

A time series y_t, $t = 1, 2, ...$ is produced by a source $S(\theta_k)$, where θ_k is a parameter taking values in a *finite* set $\Theta = \{\theta_1, ..., \theta_K\}$. The source that produces the time series should be identified; in other words the "true" or "best" value of θ_k should be computed.

A random variable Z which takes values in $\Theta = \{\theta_1, ..., \theta_K\}$ is introduced. The time series y_1, y_2, ... is produced by source $S(Z)$. For instance, if $Z = \theta_1$, then the time series y_1, y_2, ... is produced by $S(\theta_1)$. At every time t a decision rule produces an estimate of Z, denoted $\hat{Z}_t$. For instance, if *at time* t we believe that the time series y_1, ... y_t has been produced by θ_1, then $\hat{Z}_t = \theta_1$. Clearly, $\hat{Z}_t$ may change with time, as more observations become available.

The *conditional posterior probability* p_t^k for $k = 1, 2, ..., K$, $t = 1, 2, ...$ is defined by

$$p_t^k \doteq Prob(Z = \theta_k \mid y_t, ..., y_1);$$

also the *prior probability* p_0^k for $k = 1, 2, ..., K$ is defined by

$$p_0^k \doteq Prob(Z = \theta_k \mid at \;\; t = 0). \tag{1}$$

p_0^k reflects our prior knowledge of the value of Z. In the absence of any prior information we can just assume all models to be equiprobable: $p_0^k = 1/K$ for $k = 1, 2, ..., K$. p_t^k reflects our belief (after observing data $y_1, ..., y_t$) that the time series is produced by $S(\theta_k)$. It is natural then to choose $\hat{Z}_t \doteq \arg\max_{\theta_k \in \Theta} p_t^k$. In other words, at time t we consider that $y_1, ..., y_t$ has been produced by source $S(\hat{Z}_t)$, where $\hat{Z}_t$ maximizes the posterior probability. So the classification problem has been reduced to computing p_t^k, $t = 1, 2, ..., k = 1, 2, ..., K$. This computation can be performed recursively. We start

with Bayes' rule

$$p_t^k = Prob(Z = \theta_k \mid y_t, ..., y_1) = \frac{Prob(y_t, Z = \theta_k \mid y_{t-1}, ..., y_1)}{Prob(y_t \mid y_{t-1}, ..., y_1)} = \frac{Prob(y_t, Z = \theta_k \mid y_{t-1}, ..., y_1)}{\sum_{j=1}^{K} Prob(y_t, Z = \theta_j \mid y_{t-1}, ..., y_1)}. \tag{2}$$

Also

$$Prob(y_t, Z = \theta_k \mid y_{t-1}, ..., y_1) = Prob(y_t \mid y_{t-1}, ..., y_1, Z = \theta_k) \cdot Prob(Z = \theta_k \mid y_{t-1}, ..., y_1) =$$

$$Prob(y_t \mid y_{t-1}...y_1, Z = \theta_k) \cdot p_{t-1}^k. \tag{3}$$

Now (2), (3) imply the following recursion for k =1,2, ..., K, t =0, 1, 2, ... :

$$p_t^k = \frac{Prob(y_t \mid y_{t-1}, ..., y_1, Z = \theta_k) \cdot p_{t-1}^k}{\sum_{j=1}^{K} Prob(y_t \mid y_{t-1}, ..., y_1, Z = \theta_j) \cdot p_{t-1}^j}. \tag{4}$$

and we only need (for each t and k) to compute $Prob(y_t \mid y_{t-1}, ..., y_1, Z = \theta_k)$. This probability depends on the form of the predictor. In the next two sections it is computed for nonlinear and linear predictors respectively.

3 Nonlinear Prediction Modules

The predictors have a general parametric form $f(\cdot; \theta_k)$, $k = 1, ... , K$:

$$y_t^k = f(y_{t-1}, ..., y_{t-N}; \theta_k). \tag{5}$$

Typically, $f(\cdot; \theta_k)$ would be a feedforward (sigmoid, radial basis, polynomial) neural network trained on data from source $S(\theta_k)$. This predictor approximates y_t *when the time series is produced by* $S(\theta_k)$. It should be noted that the predictor uses only the finite past y_{t-1}, ... , y_{t-N}. For $k = 1, 2, ..., K$ the prediction error e_t^k, $k = 1$, ... , K, $t = 1, 2$, ... is defined by

$$e_t^k \doteq y_t - y_t^k. \tag{6}$$

It is *assumed* that e_t^k has a Gaussian conditional probability of the form

$$Prob(e_t^k \mid y_{t-1}, ..., y_1, Z = \theta_k) = C(\sigma_k) \cdot exp(- \mid \frac{e_t^k}{\sqrt{2}\sigma_k} \mid^2). \tag{7}$$

It then follows immediately from (5), (6) and (7) that

$$Prob(y_t \mid y_{t-1}, ..., y_1, Z = \theta_k) = C(\sigma_k) \cdot exp(- \mid \frac{y_t - y_t^k}{\sqrt{2}\sigma_k} \mid^2). \tag{8}$$

The probability assumption of (7) is entirely arbitrary, but works well in practice. The parameter σ_k^2 is the variance and $C(\sigma_k)$ is a normalizing constant. Extensions for vector valued y_t and e_t^k are obvious.

The posterior probability of every source $S(\theta_k)$, $k = 1, 2, ..., K$, for time $t = 1, 2, ..$, can be computed

by means of the above equations. At time t the time series is classified to the source that maximizes the posterior probability:

$$\hat{Z}_t \doteq \arg\max_{\theta_k \in \Theta} p_t^k. \tag{9}$$

The desired recursion for p_t^k is obtained from (1), (4), (5), (8) and (9). This recursion is inspired by the *partition algorithm*, first developed by Hilborn and Lainiotis (Hilborn & Lainiotis, 1969) and applied to control problems in (Lainiotis, 1971): Our recursion equations can be implemented by a PREMONN with nonlinear prediction modules and Bayesian decision module. Now we turn to the question of convergence; ergodicity of $\{y_t\}_{t=1}^{\infty}$ must be assumed.

Theorem 1 *Suppose that the following assumptions hold*

A1 $\{y_t\}_{t=1}^{\infty}$ *is ergodic, square integrable.*

A2 $f(z_1, ..., z_N, \theta)$ *is a measurable function of* $z_1, ..., z_N$ *for every value of* $\theta \in \{\theta_1, ..., \theta_K\}$.

A3 y_t^k *is square integrable for every value of* $\theta \in \{\theta_1, ..., \theta_K\}$.

A4 $p_0^k > 0$ *for* $k = 1, 2, ..., K$.

Now define $d \doteq \min_{1 \le k \le K} \frac{E|e_t^k|^2}{\sigma_k^2}$ *and the following sets:* $\mathcal{K}_0 \doteq \{k : \frac{E|e_t^k|^2}{\sigma_k^2} = d\}$, $\mathcal{K}_1 \doteq \{k : \frac{E|e_t^k|^2}{\sigma_k^2} > d\}$. *Then*

$$\forall k \in \mathcal{K}_1, l \in \mathcal{K}_0 \lim_{t \to \infty} \frac{p_t^k}{p_t^l} = 0 \ \ with \ \ prob.1, \tag{10}$$

$$\lim_{t \to \infty} \sum_{k \in \mathcal{K}_0} p_t^k = 1 \ \ with \ \ prob.1, \tag{11}$$

$$\lim_{t \to \infty} \sum_{k \in \mathcal{K}_1} p_t^k = 0 \ \ with \ \ prob.1. \tag{12}$$

$\mathcal{K}_0$ is the set of "good" predictors; in particular, if $\sigma_k = \sigma$ for $k = 1, 2, ..., K$ (all predictors have the same error variance) then $\mathcal{K}_0$ is the set of predictors that minimize prediction error. Judging on prediction performance only, any element of $\mathcal{K}_0$ is as good as any other, and they are all preferrable to any element of $\mathcal{K}_1$. Note that both $\mathcal{K}_0$ and $\mathcal{K}_1$ are time invariant, because of ergodicity. Theorem 1 states that the total posterior probability of the $\mathcal{K}_0$ predictors tends to one, with probability 1. Note that convergence is *not* affected by the values of the prior probabilities, as long as they are nonzero (Assumption **A4**). The theorem is proven in the Appendix.

4 Linear Prediction Modules

If the predictor modules are linear, convergence results can be obtained under less restrictive assumptions. For instance ergodicity can be proven (rather than assumed), the Gaussian assumption can be justified and less restrictive convergence conditions can be obtained.

Let us assume that the source produces a signal y_t, $t = 1, 2, ...$ by a linear mechanism of the form

$$y_t = A' \cdot Y_t + w_t$$

where $A = [a_1 ... a_N]'$, $Y_t = [y_{t-1} ... y_{t-N}]'$ and w_t is a Gaussian, zero-mean, white noise process with variance σ^2. We use matched linear predicting modules; for k =1,2, ..., K,

$$y_t^k = A_k' \cdot Y_t \tag{13}$$

where $A_k = [a_1^k ... a_N^k]'$ is estimated using data from source θ_k. (13) can be realized by a linear neural network. If the active source is θ_k, then we expect that A and A_k are close in some sense.

The recursive computation of posterior probabilities is the same as in the case of nonlinear modules. The only change is in the computation of y_t^k, which is now given by (13). So, (1), (4), (13), (8) and (9) can be implemented by a PREMONN with linear prediction modules and Bayesian decision module. The σ_k^2 in (8) is the variance of $y_t - y_t^k$, which depends on A, A_k and the variance of w_t. It can be computed explicitly, but this is not necessary; we just need to know that it is well defined and constant.

Let us define $R \doteq E(Y_t Y_t')$ and $\mid z \mid_R \doteq z'Rz$. R is a positive definite matrix and hence $\mid \cdot \mid_R$ is a norm on z vectors. In particular, $\mid A - A_k \mid_R$ measures the difference between A and A_k.

Theorem 2 *Suppose that the following assumptions hold*

B1 *The polynomial* $z^N - a_1 z^{N-1} - a_2 z^{N-2} - \ldots a_{N-1} z - a_N$ *has roots* $z_1, \ldots, z_N$ *such that* $\mid z_n \mid < 1$ *for* $n = 1, 2, \ldots, N$.

B2 $p_0^k > 0$ *for* $k = 1, 2, ..., K$.

B3 $\sigma_k = \sigma$ *for* $k = 1, 2, ..., K$.

Now define $d \doteq \min_{1 \le k \le K} \mid A - A_k \mid_R$ *and the sets:* $\mathcal{K}_0 \doteq \{k : \mid A - A_k \mid_R = d\}$, $\mathcal{K}_1 \doteq \{k : \mid A - A_k \mid_R > d\}$. *Then*

$$\forall k \in \mathcal{K}_1, l \in \mathcal{K}_0 \lim_{t \to \infty} \frac{p_t^k}{p_t^l} = 0 \ \ with \ \ prob.\ 1. \tag{14}$$

$$\lim_{t \to \infty} \sum_{k \in \mathcal{K}_0} p_t^k = 1 \ \ with \ \ prob.\ 1. \tag{15}$$

$$\lim_{t \to \infty} \sum_{k \in \mathcal{K}_1} p_t^k = 0 \ \ with \ \ prob.\ 1. \tag{16}$$

The theorem implies that convergence to the set of "good" predictors $\mathcal{K}_0$ is guaranteed as long as the predictors are well separated in parameter space (that is, $|A - A_l|_R$ is large relative to $|A - A_k|_R$) and all predictors have the same variance. Once again, convergence is *not* affected by the values of the prior probabilities, as long as they are nonzero (Assumption **B2**). The proof of the theorem is given in the Appendix.

5 Decision Module

So far we have been considering variations of the prediction modules. However, the decision module can also be modified. Rather than the Bayesian *credit function* (4), a number of other credit functions can be used; in this section we present some. The Bayesian credit function used so far, allows a probabilistic

interpretation. Some of the new credit functions presented can also be interpreted probabilistically, but this is not necessary for the solution of the classification problem; they can be considered as figures of merit. It should be noted that changing the credit function affects only the top (decision) level of the PREMONN, not the bottom (prediction) level.

5.1 Multiplicative Credit Function

Let us consider the usual predictors $y_t^k = f(y_{t-1}, ..., y_{t-N}; \theta_k)$ and errors $e_t^k = y_t - y_t^k$, for $k = 1, 2, ..., K$, $t = 1, 2$ ... ; the figure of merit p_t^k is given by

$$p_t^k = \frac{p_{t-1}^k \cdot e^{-F(e_t^k)}}{\sum_{l=1}^K p_{t-1}^l \cdot e^{-F(e_t^l)}}, \tag{17}$$

where $F(\cdot)$ is a continuous, increasing and nonnegative function. (Obviously, the Bayesian credit function is a special case of (17), where $F(z) =\mid z \mid^2$; more generally we could use $F(z) =\mid z \mid^M$.) At time t the time series is classified to the k^*-th source, where k^* maximizes p_t^k.

Theorem 3 *Suppose that the following assumptions hold*

C1 $\{y_t\}_{t=1}^{\infty}$ *is ergodic, bounded.*

C2 $f(z_1, ..., z_n, \theta)$ *is a continuous function of* $z_1, ..., z_n$ *for every value of* $\theta \in \{\theta_1, ..., \theta_K\}$.

C3 $F(z) \geq 0$ *for all* z, *increasing and continuous in* z.

C4 $p_0^k > 0$ *for* $k = 1, 2, ..., K$.

Now define $d \doteq \min_{1 \leq k \leq K} E \mid F(e_t^k) \mid$ *and the sets:* $\mathcal{K}_0 = \{k : E \mid F(e_t^k) \mid = d\}$, $\mathcal{K}_1 = \{k : E \mid F(e_t^k) \mid > d\}$. *Then*

$$\forall k \in \mathcal{K}_1, l \in \mathcal{K}_0 \lim_{t \to \infty} \frac{p_t^k}{p_t^l} = 0 \ \ with \ \ prob.\ 1.$$

$$\lim_{t \to \infty} \sum_{k \in \mathcal{K}_0} p_t^k = 1 \ \ with \ \ prob.\ 1.$$

$$\lim_{t \to \infty} \sum_{k \in \mathcal{K}_1} p_t^k = 0 \ \ with \ \ prob.\ 1.$$

The proof is omitted, since it is very similar to that of Theorem 1.

5.2 Additive Credit Function I

Here the figure of merit p_t^k is given by

$$p_t^k = \frac{\sum_{s=1}^t F(e_s^k)}{\sum_{l=1}^K \sum_{s=1}^t F(e_s^l)}$$

$F(z)$ is a continuous, decreasing and nonnegative function.

Theorem 4 *Suppose that the following assumptions hold*

D1 $\{y_t\}_{t=1}^{\infty}$ *is ergodic, bounded.*

D2 $f(z_1, ..., z_N, \theta)$ *is a continuous function of* $z_1, ..., z_N$ *for every value of* $\theta \in \{\theta_1, ..., \theta_K\}$.

D3 $F(z) \geq 0$ *for all* z, *decreasing and continuous in* z.

Then

$$\lim_{t\to\infty} p_t^k = \frac{E(F(e_t^k))}{\sum_{l=1}^{K} E(F(e_t^l))} \quad with \ \ prob.\ 1.$$

The proof is omitted, since it is a trivial consequence of the Ergodic Theorem. It should be noted that, in general, no p_t^k will tend to 1. However, the highest p_t^k is assigned to the module which maximizes the expected value of $F(e_t^k)$. Intuitively, such a model is "best", since $F(e_t^k)$ is a decreasing function of the error (e.g. $F(e_t^k) = \frac{1}{|e_t^k|^2}$).

5.3 Additive Credit Function II

In this case the figure of merit p_t^k is given by

$$p_{t,m}^k = \frac{\sum_{s=t-m}^{t} 1_{\{e_t^k < e_t^l \ \ \forall l \neq k\}}}{m+1},$$

where 1_U is the *indicator function*, taking the value 1 when event U is true, and 0 otherwise. It should be noted that this update rule depends on an extra index m. The following convergence theorem holds.

Theorem 5 *Suppose that the following assumptions hold*

E1 $\{y_t\}_{t=1}^{\infty}$ *is ergodic.*

E2 $f(z_1, ..., z_N, \theta)$ *is a measurable function of* $z_1, ..., z_N$ *for every value of* $\theta \in \{\theta_1, ..., \theta_K\}$.

Then for all t *large enough*

$$\lim_{m\to\infty} p_{t,m}^k = Prob(e_t^k < e_t^l \ \ \forall l \neq k) \ \ with \ \ prob.\ 1.$$

The proof is omitted, since it is a trivial consequence of the Ergodic Theorem. Note that convergence depends on m, not t. In general, as in the previous case, no $p_{t,m}^k$ will tend to 1. However, the highest $p_{t,m}^k$ will be assigned to the model which has the highest probability of yielding minimum error.

6 The PREMONN Architecture

In summary, we have started with a Bayesian point of view to obtain a neural network (see Fig.1), the so-called PREMONN, that classifies time series. It can be seen from Fig.1 that PREMONN has a recursive, modular, hierarchical architecture. The bottom level consists of the prediction modules and the top level consists of the decision modules. The Bayesian version of PREMONN has been

our starting point, but it can be extended. Namely, the decision module can be modified by using nonprobabilistic credit functions, as explained in Section 5. Similarly, there are many choices for the prediction functions. They can be linear, sigmoid, radial basis functions, polynomial etc; also they can be implemented by feedforward or recursive neural networks. Finally, a PREMONN can be built such that every prediction module is of a different type, e.g. a linear predictor is used for one module, a sigmoid for another and so on.

Hence the PREMONN architecture can be considered as a general framework for the design of recursive, modular, hierarchical networks for time series classification. It should also be noted that the same architecture could be used for the classification of static patterns. We believe however, that the advantages of PREMONN become evident in time series problems, where there are correlations between observations.

7 Prediction

The original, probabilistic PREMONN can also be used for prediction, with slight modifications. The main idea is the following. If the true source θ_k were known, then y_t would be predicted with $y_t^k = f(y_{t-1}, ..., y_{t-N}, \theta_k)$. Since θ_k is not known, a *weighted predictor*

$$y_t^* = \sum_{k=1}^{K} p_{t-1}^k \cdot y_t^k \tag{18}$$

can be used. As it will be shown in Theorem 6, this predictor minimizes square error and is convergent. However, first the following concepts are needed.

The *conditional expectation* of y_t, conditioned on y_1, ..., y_{t-1} and $Z = \theta_k$ exists always (Breiman, 1968); it can be denoted as a function

$$f_t(y_{t-1}, ..., y_1; \theta_k) \doteq E(y_t \mid y_{t-1}, ..., y_1, Z = \theta_k). \tag{19}$$

In the following, we *assume* that f_t in fact depends only on the last N values y_{t-1}, ..., y_{t-N} and that our prediction modules express this dependence. In other words

$$E(y_t \mid y_{t-1}, ..., y_1, Z = \theta_k) = f(y_{t-1}, ..., y_{t-N}; \theta_k) = y_t^k. \tag{20}$$

Of course, the following results are correct to the extend that (20) approximates reality.

The *conditional expectation* of y_t, conditioned on y_1, ..., y_{t-1} also exists always (Breiman,1968) and it is given by

$$y_t^* = \sum_{k=1}^{K} Pr(Z = \theta_k \mid y_{t-1}, ..., y_1) \cdot E(y_t \mid y_{t-1}, ..., y_1, Z = \theta_k) = E(y_t \mid y_{t-1}, ..., y_1). \tag{21}$$

The properties of y_t^* are summarized in the following theorem.

Theorem 6 *Under the assumptions A1-A4 of Theorem 1*

$$E \mid y_t^* - y_t \mid^2 \leq E \mid \hat{y}_t - y_t \mid^2 \tag{22}$$

for all functions $\hat{y}_t$ that are $\sigma(y_1, ..., y_{t-1})$ measurable. Equality occurs only when $\hat{y}_t = y_t^*$ with prob. 1. Furthermore, if we assume

F1 *There is exactly one k^* such that $\frac{E|e_t^{k^*}|^2}{\sigma_{k^*}^2} < \frac{E|e_t^k|^2}{\sigma_k^2}$ for all $k \neq k^*$ and*

F2 *$\{y_t^k\}$ is bounded for $k = 1, 2, ..., K$*

then

$$\lim_{t \to \infty} E \mid y_t^* - y_t^{k^*} \mid = 0, \tag{23}$$

$$\lim_{t \to \infty} \mid y_t^* - y_t^{k^*} \mid = 0 \ \ with \ \ prob. \ 1. \tag{24}$$

The proof of the Theorem is given in the Appendix. [3]

8 Implementation Issues

In this section we discuss various implementation issues which do not lend themselves to a rigorous analysis, but are important in practice.

8.1 Deterministic Time Series

We have implicitly assumed that the time series considered is stochastic. This is not necessary. Suppose that y_{t-N} ,..., y_{t-1} completely determine y_t:

$$y_t = f(y_{t-1}, ..., y_{t-N}). \tag{25}$$

In practice, prediction error will still be nonzero because the exact form of f in (25) and/or the value of N are not known, or, finally, because the parametric form of the predictors cannot reproduce (25) exactly. The prediction error is deterministic, but since we know nothing about it, we can simply assume that it has Gaussian probability given by (7). Or, we can take p_t^k to be figures of merit, without a probabilistic significance. This is all a matter of interpretation, which does not affect the performance and convergence of the network.

[3] There is an apparent contradiction here. It appears that we have proven y_t^* to have mean square error smaller or equal to that of any linear combination of y_t^k, $k = 1, ..., K$. So in particular, y_t^* has smaller mean square error than, say y_t^1. Now supposing that the true source is $Z = 1$, would we not expect y_t^1 to be better than y_t^*? Actually this is not a problem, because the expectation in (22) is conditioned on $y_1, ..., y_t$ *and* Z. There are two possibilities. Either $Prob(Z = \theta_1) < 1$, or $Prob(Z = \theta_1) = 1$. In the first case y_t^1 is better than y_t^* only with probability $Prob(Z = \theta_1)$, but it is worse with probability $1 - Prob(Z = \theta_1)$ and, taking expected values, y_t^* is overall better. On the other hand, if $Prob(Z = \theta_1) = p_0^1 = 1$, then $p_0^k = 0$ for $k = 2, 3, ..., K$ and so, by (4) $p_t^k = 0$ for $k = 2, ..., K$, $t = 0, 1, ...$ In that case $p_t^1 = 1$ for $t = 0, 1, ...$ and by (18) $y_t^* = y_t^1$ for $t = 0, 1, ...$ The same analysis applies for every $k = 1, ..., K$, in case $Z = k$.

8.2 Variance and Threshold

Error variance σ_k^2 affects the performance of the PREMONN. The convergence rate of the posterior probabilities depends (see Appendix) on $d_{k,l} = \frac{exp(-E|e_t^k|^2/\sigma_k^2)}{exp(-E|e_t^l|^2/\sigma_l^2)}$, $k, l = 1, \ldots, K$. If predictor k has large mean square error and predictor l has small mean square error, then $d_{k,l}$ is close to 0, and convergence is fast. However, one can see that when the σ's are large, $d_{k,l}$ is small and convergence is slow. Intuitively, larger variance means that less information is generated per observation, so more observations must be collected to reach a certain level of confidence. σ_k^2 can be estimated during the predictor training phase, but we found by computer experimentation that sometimes it is advantageous to modify this estimate. For faster classification, the variance must be decreased.

A *probability threshold* is also introduced, so that p_t^k never becomes zero. This is necessary because of *source switching*. So far we have assumed that the whole time series is produced by a single source but in many cases this is not true. For instance, in a speech time series each phoneme is produced by a different source and different sources are active over different time intervals. We refer to this phenomenon as source switching.

If a source, say $S(\theta_1)$, has been active for a long time, p_t^1 is close to 1 and $p_t^2, \ldots, p_t^K$ are close to 0; in fact, due to numerical underflow they may become equal to 0. Then, referring to (4) we observe that they will remain 0 even in case of a source switching; this will result in false classification. For this problem to be resolved, whenever p_t^k falls below a specified threshold h, it is reset to h. In essence, this thresholding is equivalent to introducing a forgetting factor. No matter how unlikely a sequence of observations is, a source model always retains a small posterior probability, equal to h; thus it is never removed from consideration. If the model performs well at predicting a later sequence of observations, its posterior probability can recover a high value. An alternative way of looking at the use of threshold h is that whenever one or more p_t^k's fall below h (because the corresponding source models perform poorly) we restart the algorithm using new prior probabilities, obtained from resetting the corresponding p_t^k's to h. Under this interpretation, our prior belief in any of the source models never goes below h.

Both variance and threshold are related to a speed/accuracy trade-off. Large variance slows the network down and assigns little importance to individual errors; small variance speeds the network up, but also assigns more importance to instantaneous fluctuations and makes the network more prone to instantaneous classification errors. Similarly, a low threshold (or absence of threshold) incurs on the posterior probabilities a large recovery time between source switchings. On the other hand, a high threshold tends to obliterate the significance of past performance and makes the response of the network to source switchings faster, at the cost of spuriously interspersed false classifications.

Analogous considerations apply to nonprobabilistic credit functions.

8.3 Parallelism and Scaling

The modularity of the network introduces parallelism naturally. Prediction modules can execute in parallel and send the results to the decision module. Hence execution time is independent of the number of classes in the classification problem.

It is well known that lumped neural networks scale badly. As the number of parameters increases

training may take exponentially long time and/or fail to achieve a good solution. Hence, modular networks, where fixed size modules are trained piecewise, are highly desirable. PREMONNs fall in this category. Clearly, both network size and training time scale linearly with the number of sources (categories) that must be learned. Also, PREMONNs perform well even when the individual prediction modules have poor performance, as long as they are clearly separated in the parameter space: the decision module will simply pick the "least bad" prediction module. In particular, PREMONN is immune to a high level of noise in the data (see also Section 9).

9 Experiments

We include a number of classification experiments to corroborate our theoretical results and to illustrate the implementation issues of the previous section. In particular we want to show that convergence to the true source takes place, to investigate the effect of the variance and threshold parameters to convergence rate, to compare performance of various credit functions and to show that the algorithm is immune to a high level of noise in the data.

9.1 Logistic Detection

Our first set of experiments deals with the problem of separating a logistic time series from white noise. We have chosen these data because the logistic resembles noise, both visually and statistically. Hence separating the logistic from the noise is an interesting classification task. The logistic time series is produced by the relationship $y_t = \alpha \cdot y_{t-1} \cdot (1 - y_{t-1})$. For values of α higher than 3.67..., $\{y_t\}_{t=1}^{\infty}$ is a chaotic time series, (see Figs.2, 3 and 4). All networks used are of size 18-5-1. The 18 inputs are the values y_{t-1}, ... , y_{t-18} and the target output is y_t. For $\alpha = 4.0$ a logistic time series is generated and used to train a sigmoid neural network predictor; it attains mean square prediction error 0.15. Also a neural network predictor is trained on a Gaussian white noise time series, with mean $\mu_w = 0.50$ and standard deviation σ_w =0.25. In this case the mean square error of the predictor is 0.3.[4]

In the first experiment 200 time steps of a logistic time series have been generated (again with α = 4.0, but with different initial condition) and a PREMONN consisting of two prediction modules has been used. One module has been trained on the logistic training data and the other on the white noise training data. We have used $\sigma_1 = \sigma_2 = 0.15$, $h = 0.01$ and the Bayesian credit function. The results of this experiment are presented in Fig. 2. It can be seen that classification to the logistic is very fast.

In the second experiment a composite time series has been used. The first half of it has been 100 time steps of the logistic time series of the previous experiment and the second half of it has been 100 samples of Gaussian white noise with mean $\mu_w = 0.5$ and standard deviation σ_w =.25. In this case also, the PREMONN used consisted of the same two predictor modules as in the previous experiment. We have used $\sigma_1 = \sigma_2 = 0.20$, $h = 0.01$ and Bayesian credit function. The results of this experiment are presented in Fig.3. In the beginning of the time series, classification to the logistic is almost instantaneous. Then at the switching point t_s=82, a very quick switch to noise classification is observed. In the third

[4]Note that 0.25, the noise variance, is the minimum theoretically attainable MSE, which can be attained by using the constant predictor $\hat{y}_t$ =0.5.

experiment *ten* predictor modules have been trained (18-5-1 sigmoid neural networks) on logistics with α= 3.0, 3.1, ... , 3.9. Then 200 time steps of a test logistic have been generated with α=3.8 and a new initial condition. The PREMONN consisted of the previously mentioned ten prediction modules. We have used $\sigma_1 = ... = \sigma_{10} = 0.15$, $h = 0.01$ and the Bayesian credit function. The results of this experiment are presented in Fig.4. It can be seen that classification to the true logistic is very fast. This demonstrates PREMONN's ability to deal with a large number of source classes.

We have performed many additional experiments, which for economy of space we present in tabular form. Dataset can be either A (a 200 time steps logistic time series with α= 4.0) or B (a 100 time steps logistic followed by 100 time steps of Gaussian white noise with μ_w=0.50 and σ_w= 0.25). Classification performance is measured by $c_{0.6}$ and $c_{0.9}$, evaluated at two levels of confidence: 0.60 and 0.90. Namely, $c_{0.6}$ (respectively $c_{0.9}$) is computed by dividing the number of time instants where the posterior probability of the "true" module is above 0.6 (respectively 0.9) by the total number of data points. In Table 1 we experiment with σ_k values, in Table 2 with h values, in Table 3 with noise levels (i.e. we add to the data Gaussian, zero-mean white noise with various values of standard deviation σ_w) and in Table 4 with credit functions. It can be seen that the overall performance is very good.

9.2 McKey-Glass Time Series Discrimination

In the second set of experiments the problem is discrimination of two Mackey-Glass time series. The Mackey-Glass time series is produced by the relationship

$$\frac{d\psi}{ds} = \frac{0.2 \cdot \psi(s-\tau)}{1+\psi^{10}(s-\tau)} - 0.1 \cdot \psi(s). \tag{26}$$

Here we use $\tau = 10$ and $\tau = 17$ to obtain two chaotic time series which must be identified. For each value of τ we integrate eq.(26) and then sample at times s=5, 10, 15, ... secs to obtain a time series y_1, y_2, ..., where $y_t = \psi(5 \cdot t)$, t=1, 2, ... We train two sigmoid networks, both of size 5-5-1, to predict y_t using as inputs y_{t-1}, y_{t-2}, ... , y_{t-5}. The mean square prediction error is in both cases approximately 0.04.

In the first experiment a 200 time steps Mackey-Glass time series has been generated (with $\tau = 17$, but with different initial condition) and a PREMONN consisting of two prediction modules has been used. One module has been trained on $\tau = 17$ and the other on $\tau = 10$. We used $\sigma_1 = \sigma_2 = 0.040$, h = 0.01 and the Bayesian credit function. The results of this experiment are presented in Fig. 5. It can be seen that classification is very fast.

In the second experiment a composite Mackey-Glass time series has been used. The first 200 time steps have $\tau = 17$ and the last 200 time steps have $\tau = 10$. Also, Gaussian, zero-mean white noise with standard deviation $\sigma_w = 0.3$ has been added to the data. The PREMONN consisted of the same two predictor modules as in the previous experiment. We used $\sigma_1 = \sigma_2 = 0.04$, $h = 0.01$ and Bayesian credit function. The results of this experiment are presented in Fig. 6. In the beginning of the time series, classification to the logistic is almost instantaneous. Then at the switching point t_s=200, a very quick switch to noise classification is observed.

We have performed many additional experiments, which we present in tabular form. Dataset can be either A (200 time steps Mackey-Glass time series with $\tau = 17$) or B (200 time steps Mackey Glass

time series with $\tau = 10$ followed by 200 time steps with $\tau = 17$). Classification performance is measured by $c_{0.6}$ and $c_{0.9}$, evaluated at two levels of confidence: 0.60 and 0.90, in the same manner as in the case of logistic experiments. In Table 5 we experiment with σ_k values, in Table 6 with h values, in Table 7 with noise levels and in Table 8 with credit functions. It can be seen that the overall performance is very good.

9.3 Enzyme Classification

This experiment involves real world data. It considers classification of a set of enzymes, called β-lactamases. The data and problem are described in Papanicolaou & Medeiros (1990); here we give a short overview. β-lactamases are enzymes which determine the resistance of many bacteria to β-lactam antibiotics. The problem is classification of β-lactamases. To solve this problem biomedical researchers have developed several methods. However conventional methods present many difficulties and amino-acid sequencing is time consuming (Papanicolaou & Medeiros, 1990). A new method to overcome some of these difficulties is presented in Papanicolaou & Medeiros (1990); it uses an "inhibition" experiment where the hydrolysis rate of a chemical called nitrocefin, in the presence of a β-lactamase / β-lactam pair, is measured. The β-lactamase enzyme (henceforth to be called simply the *enzyme*) causes hydrolysis of nitrocefin, and the β-lactam (henceforth to be called simply the *inhibitor*) slows hydrolysis down by inhibiting the action of the enzyme. Nitrocefin concentration is measured optically and recorded over a 40-minute interval. In this way, for every enzyme/inhibitor pair a characteristic "inhibition profile" (as illustrated in Fig.7) is obtained. Ideally, the inhibition profile (for a given inhibitor) characterizes the enzyme. This method has a high classification success. However, there are problems: the properties of enzymes and inhibitors depend strongly on the conditions under which they were prepared, and this may result in different inhibition profiles for two different preparations of the same enzyme/inhibitor pair. However, the dynamic properties of the profile appear to remain invariant; it is reported in Papanicolaou & Medeiros (1990) that enzyme classification depended on the slope of the inhibition profile at various times during the duration of the experiment, as well as on the final concentration of nitrocefin. This information was used by a human operator who combined various characteristics of the inhibition profile to classify the enzyme.

We automate the enzyme classification process, using the inhibition profiles as input time series to a PREMONN. Eight enzymes are classified using inhibition profiles for a given inhibitor. The data set of inhibition profiles is separated into a test set and a training set[5]. For each enzyme we train a fifth order linear predictor using the corresponding inhibition profile (40 min time series) from the training set. Mean square prediction error is approximately 0.1 for all enzymes. Next, we choose an inhibition profile from the test set and proceed to classify it, i.e. determine the enzyme it corresponds to. The PREMONN classifies correctly all eight inhibition profiles in the test set; in Fig.8 we present the posterior probability evolution for a particular enzyme inhibition profile. Note that in this task classification is performed using *only the final* values p^1_{40}, p^2_{40}, ... , p^8_{40}.

We present a number of experiments in tabular form. We use two data sets, consisting of inhibition profiles for two different inhibitors (and all eight enzymes). Classification performance is measured by

[5]We want to thank G.A. Papanicolaou for kindly permitting us to use the inhibition profile data.

c, the number of correctly classified enzymes (at time t=40 mins) divided by eight, the total number of enzymes. We experiment with values of σ_k (Table 9) and h (Table 10) and with different types of credit functions (Table 11). For inhibitor # 1 classification is 100% correct for a very wide range of choices of these parameters; for inhibitor # 2 classification is generally 100% correct, with some exceptions for extreme choices of parameter values.

9.4 Classification of phonemes

In this final experiment we also use real world data: two utterances of the word "one". They have been sampled at 2 KHz. The time series is plotted in Fig. 9. It contains three phonemes ([oo], [ah] and [nn]) which can be visually distinguished in the figure.

Three linear prediction modules of the form $\hat{y}_t^k = w_1^k \cdot y_{t-1} + ... + w_{18}^k \cdot y_{t-18}$, $k = 1, 2, 3$ have been trained, using data from the three phonemes. We combine the predictors into a Bayesian PREMONN and perform several classification experiments using the second utterance of "one".

The results of a classification experiment, which uses the data of the [ah] to [nn] transition, are presented in Fig. 10. We have used $\sigma_1 = \sigma_2 = \sigma_3 = .01$, $h = 0.01$ and the Bayesian credit function. The same setup has been used for classification experiments at various noise levels. The results are presented in Table 12.

10 Conclusions

We have defined Predictive Modular Neural Networks (PREMONNs) for time series classification and investigated their properties. PREMONNs have a hierarchical organization: at the bottom level we have a bank of prediction modules and at the top level a decision module that processes the prediction errors of the bottom level and adaptively updates their Bayesian posterior probabilities or figures of merit. Our basic conclusions are:

1. PREMONNs have a probabilistic motivation and interpretation, but this is not necessary for their succesful operation. They can be applied just as well to the classification of stochastic or deterministic time series.

2. Convergence to correct classification with prob. 1 has been mathematically proven for both Bayesian and nonprobabilistic PREMONNs.

3. PREMONNs can be used for prediction as well as for classification. The PREMONN prediction has minimum mean square prediction error among all (linear and nonlinear) predictions that are functions of the past observations.

4. PREMONNs can be parallelized in an obvious way (one processor per module) and the learning time scales linearly with the number of sources that have to be learned.

5. PREMONNs can operate online.

While the variance σ_k^2 and threshold h parameters can be used to optimize the tradeoff between speed and accuracy of classification, experimental results show that classification success does not depend crucially on σ_k and h as long as they do not take extreme values. Experiments also show that PREMONNs are robust to prediction error; that is, they classify correctly even if the prediction modules perform poorly. In particular, they are robust to noise in the data. While the Bayesian credit function performs better than the additive ones, the latter perform satisfactorily and are simpler to implement. Finally, it should be noted that in experiments involving source switching, $c_{0.6}$ and $c_{0.9}$ are smaller than those in the corresponding non-switching experiments. This is due to the transient stage of the p_t^k's around the switching point.

In the following paragraphs we list some directions in which the investigations of PREMONNs can be continued.

1. **Source switching.** We have not modelled source switchings, instead we have treated them via thresholding. An alternative solution is to assume Markovian switching; essentially this brings PREMONNs in the hidden Markov models framework. Our task then is to extend the posterior update formula so that it takes into account the Markovian switching. A second task would then be to prove some convergence property of the modified PREMONN.

2. **Adaptive partition.** In many problems the time series sources are known in advance and there is sufficient data for the offline training of the predictor modules. Typical problems of this category are phoneme recognition, radar and sonar signal classification and so on. However, there are other problems where the sources are not known in advance. For such problems, we want to try a version of PREMONNs in which modules are adaptively generated. In this version, an initial prediction module is trained and the next incoming datum is used for the computation of the module's posterior probability. If this is high, the datum is used to update the module parameter estimates. Otherwise, the datum is placed in a separate data pool; after a while this data pool is used for the training of a second module. The incoming data are now tested against both modules, and if they do not fit either one, they are set aside and eventually used for the training of a third module etc. There is a maximum number of modules that can be active at a given time, so modules with a low posterior probability are deactivated .

3. **Applications to system identification.** This idea relates to the original use of the partition algorithm in the control context (Lainiotis, 1971). In many control problems we must estimate the parameters of a nonlinear system. In the simplest case, these parameters can only take a small number of values. Then a PREMONN can be applied quite readily for the computation of the most likely parameter values. A difficulty with this idea is the "curse of dimensionality": the number of possible modules increases exponentially with the parameters of the system. This leads us to the next idea.

4. **Parameter search schemes.** In case the parameters can take a large number of values, a method must be found to search efficiently the parameter space. One parameter could be searched at a time, possibly cycling through all the parameters several times. Or orthogonal grids of progressively finer resolution could be used. Or, the parameters could be changed in the direction

of steepest ascent of the posterior probability. Finally, genetic algorithms could be used for the parameter search as well.

5. **Consistency.** For PREMONNs with fixed prediction modules it has been proven here that $\hat{Z}_t \to Z$. Will this be true for the adaptive and refined versions of paragraphs (2) and (4) above? In fact we expect that, when all the data comes from a fixed source, the following convergence result will be true: *if the adaptation and/or refinements occur at a slow enough rate (that is, if enough data is accumulated before the prediction modules are changed) then* $\hat{Z}_t$ *converges to* Z. This is what statisticians call a *consistency* result, and adheres to the philosophy of Grenander's *method of sieves* (Grenander, 1981).

A Appendix

Here we present the proofs of Theorems 1, 2 and 6.

Proof of Theorem 1: For every $k \in \mathcal{K}_1$ and $l \in \mathcal{K}_0$ we have

$$\frac{p_t^k}{p_t^l} = \frac{p_0^k}{p_0^l} \cdot \frac{exp(-\frac{\sum_{s=1}^t |e_s^k|^2}{2\sigma_k^2})}{exp(-\frac{\sum_{s=1}^t |e_s^l|^2}{2\sigma_l^2})} \Rightarrow \sqrt[t]{\frac{p_t^k}{p_t^l}} = \sqrt[t]{\frac{p_0^k}{p_0^l}} \cdot \frac{exp(-\frac{\sum_{s=1}^t |e_s^k|^2}{2\sigma_k^2 \cdot t})}{exp(-\frac{\sum_{s=1}^t |e_s^l|^2}{2\sigma_l^2 \cdot t})} \tag{27}$$

Because f is measurable as a function of the y's, y_t^k is ergodic (see (Breiman, 1968)) and the same holds for $e_t^k = y_t - y_t^k$ and $|e_t^k|^2$. Since y_t and y_t^k are both square integrable, $|e_t^k|$ is also square integrable: $E|e_t^k|^2 < \infty$. (Incidentally, this shows that d is well defined and finite, and $\mathcal{K}_0$ not empty.) Since $E|e_t^k|^2 < \infty$ and e_t^k ergodic, by the continuity of the $exp(\cdot)$ function, the Ergodic Theorem (Breiman,1968) and (27) we see that for all $\epsilon > 0$ and almost all y_1, y_2, ..., there is a t_ϵ (depending on y_1, y_2, ...) such that for all $t \geq t_\epsilon$

$$\sqrt[t]{\frac{p_t^k}{p_t^l}} \leq \sqrt[t]{\frac{p_0^k}{p_0^l}} \cdot \left(\frac{exp(-\frac{E|e_t^k|^2}{2\sigma_k^2})}{exp(-\frac{E|e_t^l|^2}{2\sigma_l^2})} + \epsilon \right) \tag{28}$$

Define $d_{k,l} \doteq exp(-\frac{E|e_t^k|^2}{2\sigma_k^2})/exp(-\frac{E|e_t^l|^2}{2\sigma_l^2})$. By assumption $\frac{E|e_t^k|^2}{2\sigma_k^2} > \frac{E|e_t^l|^2}{2\sigma_l^2}$, so we can choose ϵ such that $d_{k,l} + \epsilon$ is less than 1. Then, raising (28) to the t power, we have that for all $t \geq t_\epsilon$ and almost all y_1, y_2, ...

$$0 \leq \frac{p_t^k}{p_t^l} \leq \frac{p_0^k}{p_0^l} \cdot \left(\frac{exp(-\frac{E|e_t^k|^2}{2\sigma_k^2})}{exp(-\frac{E|e_t^l|^2}{2\sigma_l^2})} + \epsilon \right)^t \Rightarrow \lim_{t\to\infty} \frac{p_t^k}{p_t^l} = 0 \ \ with\ prob.\ 1.$$

This proves (10). Note that the term p_0^k/p_0^l does *not* affect convergence, as long as neither p_0^k nor p_0^l are zero. Hence the priors are not crucial to the convergence of the algorithm, as long as they are not zero. Now, by the continuity of the max function we also have $\lim_{t\to\infty} \max_{k\in\mathcal{K}_1, l\in\mathcal{K}_0}(\frac{p_t^k}{p_t^l}) = 0$ with

prob. 1. Define $m \doteq |\mathcal{K}_1|$, $M \doteq |\mathcal{K}_0|$ and observe that

$$0 \leq \frac{\sum_{k\in\mathcal{K}_1} p_t^k}{\sum_{l\in\mathcal{K}_0} p_t^l} \leq \frac{m \max_{k\in\mathcal{K}_1} p_t^k}{M \min_{l\in\mathcal{K}_0} p_t^l} \leq \frac{m}{M} \max_{k\in\mathcal{K}_1, l\in\mathcal{K}_0} \frac{p_t^k}{p_t^l} \to 0 \ \ with\ prob\ 1.$$

Hence $\sum_{k\in\mathcal{K}_1} p_t^k / \sum_{l\in\mathcal{K}_0} p_t^l$ tends to 0 with probability 1. From this and the fact that $\sum_{k\in\mathcal{K}_1} p_t^k + \sum_{l\in\mathcal{K}_0} p_t^l = 1$, (11) and (12) follow immediately. •

Proof of Theorem 2: It can be proven (see (Grenander & Szego,1984)) that under condition **B1**, $\{y_t\}_{t=1}^{\infty}$ is Gaussian, ergodic and square integrable. (So in particular, we have that the covariance matrix R exists and is finite.) Hence Y_t is also ergodic and square integrable. Taking the same steps as in the previous proof, for every $k \in \mathcal{K}_1$, $l \in \mathcal{K}_0$

$$\frac{p_t^k}{p_t^l} = \frac{p_0^k}{p_0^l} \cdot \frac{exp(-\frac{\sum_{s=1}^t Y_s'(A-A_k)(A'-A_k')Y_s + 2w_s'(A'-A_k')Y_s + w_s^2}{2\sigma_k^2})}{exp(-\frac{\sum_{s=1}^t Y_s'(A-A_l)(A'-A_l')Y_s + 2w_s'(A'-A_l')Y_s + w_s^2}{2\sigma_l^2})} \tag{29}$$

By the Ergodic Theorem and the continuity of the $exp(\cdot)$ function, (29) implies that for all $\epsilon > 0$, and almost all y_1, y_2, ..., there is a t_ϵ (depending on y_1, y_2, ...) such that for all $t \geq t_\epsilon$

$$\sqrt[t]{\frac{p_t^k}{p_t^l}} \leq \sqrt[t]{\frac{p_0^k}{p_0^l}} \cdot \left(\frac{exp(-\frac{E(Y_t'(A-A_k)(A'-A_k')Y_t + w_t^2)}{2\sigma_k^2})}{exp(-\frac{E(Y_t'(A-A_l)(A'-A_l')Y_t + w_t^2)}{2\sigma_l^2})} + \epsilon \right) \tag{30}$$

It is a well known fact that $E(Y_t'(A-A_k)(A'-A_k')Y_t)) = (A'-A_k')R(A-A_k) = |A-A_k|_R^2$. Similarly, $E(Y_t(A-A_l)(A'-A_l')Y_t)) = (A'-A_l')R(A-A_l) = |A-A_l|_R^2$. Hence (30) becomes

$$\sqrt[t]{\frac{p_t^k}{p_t^l}} \leq \sqrt[t]{\frac{p_0^k}{p_0^l}} \cdot \left(\frac{exp(-\frac{|A-A_k|_R^2}{2\sigma_k^2})}{exp(-\frac{|A-A_l|_R^2}{2\sigma_l^2})} + \epsilon \right). \tag{31}$$

Recall that we have assumed $\sigma_k = \sigma$ for all k; then define $d_{k,l} \doteq exp(-\frac{|A-A_k|_R^2}{2\sigma^2}) / exp(-\frac{|A-A_l|_R^2}{2\sigma^2})$. By assumption $\frac{|A-A_k|_R^2}{2\sigma^2} > \frac{|A-A_l|_R^2}{2\sigma^2}$, so we can choose ϵ such that $d_{k,l} + \epsilon$ is less than 1. The rest of the proof proceeds like the proof of Theorem 1. Note that again the term p_0^k/p_0^l does *not* affect convergence, as long as neither p_0^k nor p_0^l are zero. Hence the priors are not crucial to the convergence of the algorithm, as long as they are not zero. •

Proof of Theorem 6: A remarkable, but easy to prove (Doob,1953) property of the conditional expectation is the following. Take a σ-field $\mathcal{F}$, a random variable X, its conditional expectation with respect to $\mathcal{F}$, $E(x|\mathcal{F})$, and a random variable Y that is $\mathcal{F}$-measurable. We have

$$E(Y(X - E(X|\mathcal{F}))) = 0. \tag{32}$$

Now take any σ $(y_1, ..., y_{t-1})$ measurable predictor of y_t, call it $\hat{y}_t$. Since y_t^* is a function of y_{t-1}, ..., y_{t-N}, it is also σ $(y_1, ..., y_{t-1})$ measurable. Since y_t^* is the conditional expectation of y_t with respect to

$\sigma\ (y_1, ..., y_{t-1})$, we have

$$E|\hat{y}_t - y_t|^2 = E|\hat{y}_t - y_t^* + y_t^* - y_t|^2 = E|\hat{y}_t - y_t^*|^2 + E|y_t^* - y_t|^2 + 2E((\hat{y}_t - y_t^*)(y_t^* - y_t)). \tag{33}$$

Now $\hat{y}_t - y_t^*$ is $\sigma\ (y_1, ..., y_{t-1})$ measurable, so we can use (32) with $\mathcal{F} = \sigma(y_1, ..., y_{t-1})$, $Y = \hat{y}_t - y_t^*$, $X = y_t$, $E(X|\mathcal{F}) = y_t^*$. Then (33) becomes

$$E|\hat{y}_t - y_t|^2 = E|\hat{y}_t - y_t^*|^2 + E|y_t^* - y_t|^2. \tag{34}$$

From (34) it becomes obvious that $E|\hat{y}_t - y_t|^2 \geq E|y_t^* - y_t|^2$, with equality occurring only when $\hat{y}_t = y_t^*$ with probability 1. This proves (22). To prove (23) and (24), note that, by Assumption **F1**, $\mathcal{K}_\mathbf{0}$ has only one member, namely k^*. Then, by Theorem 1, with probability 1, $p_t^{k^*} \to 1$ and $p_t^k \to 0$ for all $k \neq k^*$. On the other hand, by the definition of y_t^* and the triangle inequality, we have

$$0 \leq |y_t^* - y_t^{k^*}| \leq |1 - p_t^{k^*}| \cdot |y_t^{k^*}| + \sum_{k \neq k^*} p_t^k |y_t^k|. \tag{35}$$

Integrate (35) and use the Bounded Convergence Theorem to prove (23). To prove (24), define $M \doteq \max_{1 \leq k \leq K, t=1,2,...} |y_t^k|$; this exists and is finite by assumption **F2**. Then (35) implies

$$0 \leq |y_t^* - y_t^{k^*}| \leq M \cdot (|1 - p_t^{k^*}| + \sum_{k \neq k^*} p_t^k). \tag{36}$$

Taking the limit as $t \to \infty$ in (36) we obtain (24) and the proof is complete. •

References

[ ] Ayestaran, H.E. & Prager, R.W. (1993). The logical gates growing network. Tech. Report, Cambridge Un. Engineering Dept., TR CUED F-INFENG TR 137.

[ ] Baxt, W.G. (1992). Improving the accuracy of an artificial neural network using multiple differently trained networks. *Neural Computation*, **4**, 135-144.

[ ] Beyer, U. & Smieja, F. (1993). Learning from examples, agent teams and the concept of reflection, *Preprint.*

[ ] Breiman, L. (1968). *Probability.* New York: Addison - Wesley.

[ ] Doob, J.L. (1953). *Stochastic Processes.* New York: Wiley.

[ ] Farmer, J.D. & Sidorowich, J.S. (1988). Exploiting chaos to predict the future and reduce noise. Los Alamos Nat. Laboratory, TR LA UR 88 901.

[ ] Gorman, R.P. & Sejnowski, T.J. (1988). Analysis of hidden units in a layered network trained to classify sonar targets. *Neural Networks*, **1**, 75-89.

[ ] Grenander, U. (1981). *Abstract Inference.* New York: Wiley.

[ ] Grenander, U. & Szego, G. (1984). *Toeplitz Forms*. New York: Chelsea.

[ ] Haykin, S. & Cong, D. (1991). Classification of radar clutter using neural networks. *IEEE Trans. on Neural Networks*, **2**, 589-600.

[ ] Hilborn, C.G. & Lainiotis, D.G. (1969). Optimal estimation inthe presence of unknown parameters. *IEEE Trans. on Systems Science and Cybernetics*, **5**, 38-43.

[ ] Jacobs, R.A. et al. (1991). Adaptive mixtures of local experts. *Neural Computation*, **3**, 79-87.

[ ] Jordan, M.I. & Jacobs, R.A. (1992). Hierarchies of adaptive experts. In *Proc. of NIPS 4*, eds. J. Moody, S. Hansen & R. Lippman. San Mateo, CA: Morgan Kauffman.

[ ] Kadirkamanathan, V. & Niranjan, M. (1992). Application of an architecurally dynamic neural network for speech pattern classification. *Proc. of the Inst. of Acoustics*, **14**, 343-350.

[ ] Lainiotis, D.G. (1971). Optimal adaptive estimation: structure and parameter adaptation. *IEEE Trans. on Automatic Control*, **10**, 160-170.

[ ] Lainiotis, D.G. , Katsikas, S.K & Likothanasis, S.D. (1988). Adaptive deconvolution of seismic signals: performance, computational analysis, parallelism. *IEEE Trans. Acoustics, Speech and Signal Processing*, **36**, 1715-1734.

[ ] Moody, J. (1989). Fast learning in multi-resolution hierarchies. Tech. Report, Dept. of Computer Sc., Yale Un., TR YALEU DCS RR 681.

[ ] Neal, R.M. (1991). Bayesian mixture modelling by Monte Carlo simulation. Tech. Report, Dept. of Computer Science, Un. of Toronto, TR CRG-TR-91-2.

[ ] Nowlan, S.J. (1990). Maximum likelihood competition in RBF networks. Tech. Report, Dept. of Computer Science, Un. of Toronto, TR CRG-TR-90-2.

[ ] Nowlan, S.J. (1990). Competing experts: an experimental investigation of associative mixture models. Tech. Report, Dept. of Computer Science, Un. of Toronto, TR CRG-TR-90-5.

[ ] Nowlan, S.J. (1990). Maximum likelihood competitive learning. In *Proc. of NIPS 2*, ed. D. Touretzky. San Mateo, CA: Morgan Kauffman.

[ ] Papanicolaou, G.A. & Medeiros, A.A. (1990) Discrimination of Extended-Spectrum β-Lactamases by a Novel Nitrocefin Competition Assay. *Antimicrobial Agents and Chemotherapy*, **34**, 2184-2192.

[ ] Perrone, M.P. & Cooper, L.N. (1993). When networks disagree: ensemble methods for hybrid neural networks. In *Neural Networks for Speech and Image Processing*, ed. R.J. Mammone. New York: Chapman-Hall.

[ ] Petridis, V. & Kehagias, A. (1993). Modular neural networks for Bayesian classification of time series and the partition algorithm. *Submitted*.

[ ] Shadafan, R.S. & Niranjan, M. (1993). A dynamic neural network architecture by sequential partitioning of the input space. Tech. Report, Cambridge Un. Engineering Dept., TR CUED F-INFENG TR 23.

[ ] Schwarze, H. & Hertz, J. (1992). Generalization in fully connected commmittee machines. *Preprint.*

[ ] Schwarze, H. & Hertz, J. (1992). Generalization in a large commmittee machine. *Preprint.*

[ ] Waibel, A. et al. (1989). Modularity and scaling in large phonemic neural networks. *IEEE Trans. on Acoustics Speech and Signal Processing*, **37**, 1888-1898.

[ ] Zetterberg, L.H. et al. (1981). Computer analysis of EEG signals with parametric models. *Proc. of the IEEE* **69**, 451-461.

[ ] Zhu, K. et al. (1989). Training neural networks for ECG feature Recognition. In *Proceedings of the IJCNN*, ed. M. Caudill. Washington, DC: Morgan Kauffman.