TEMPORAL SELF-ORGANIZATION THROUGH COMPETITIVE PREDICTION

Craig L. Fancourt and Jose C. Principe

Computational NeuroEngineering Lab
Dept. of Electrical Engineering, University of Florida, Gainesville, FL 32611
fancourt@cnel.ufl.edu

ABSTRACT

Two self-organizing principles for the competitive identification and segmentation of piecewise stationary time series are described. In the first, a neighborhood map of one step predictors competes for the data during training. The winner is granted the largest parameter update, while other predictors are allowed smaller updates, decreasing with distance from the winner on the neighborhood map. In addition to performing piecewise segmentation and identification, the technique maps similar segments of the time series as neighbors on the neighborhood map. In the second, we propose a new cost function for competitive prediction that imbeds memory in the error metric and couples the memory with the degree of competition. Performing gradient descent on the cost function yields a self-annealing system that can also perform piecewise segmentation and identification of a time series.

1. INTRODUCTION

Many real world signals are created by systems that have time varying parameters (i.e. speech). In such cases an important engineering problem is to find regions where the signal properties are reasonably constant before attempting further block oriented signal processing (such as spectral analysis). Conventional signal processing schemes for segmentation have not produced good results. Pawelzik *et. al.* [7] have recently proposed an architecture for segmentation called the Annealed Competition of Experts (ACE). An alternative is based on the Mixture of Experts [2], called Non-linear Gated Experts [8]. These two algorithms both employ a set of expert predictors, but differ in their implementation of the gate, which assesses which expert is valid at each time step. Here we describe two algorithms for identification and segmentation where the gate is dependent on both the input and output, through the error.

2. NEIGHBORHOOD MAP OF PREDICTORS

Our implementation [1] is shown in Figure 1. A set of predictors works in parallel in the input time series. The winning predictor is the one with the smallest composite squared error:

$$winner(t) = \arg\min_i [\varepsilon_i(t)] \tag{1}$$

where the memory of the squared error is computed using an exponentially decaying window,

$$\varepsilon_i(t) = \lambda e_i^2(t) + (1-\lambda)\varepsilon_i(t-1), \tag{2}$$

where e_i is the instantaneous squared error of the ith predictor and λ is the memory term ($0<\lambda<1$). The effective memory depth, in number of samples, can be shown to be λ^{-1}. We will show later how to adapt the memory depth, but for the results presented, the optimal memory depth was experimentally determined from the time series. The gating function, which moderates the learning rate of the predictors, is determined by the distance from the winning predictor to the other predictors:

$$g_i(t) = e^{\left(-\frac{d^2_{i,j}(t)}{2\sigma^2(t)}\right)} \tag{3}$$

where i is the predictor to be updated, j is the winning predictor, $d_{i,j}$ is the neighborhood distance from predictor i to predictor j, and σ is an annealing parameter which controls the neighborhood width. This results in the model shown in Figure 1. 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:

$$\sigma(t) = \sigma_0 e^{\frac{-t}{\tau}} \text{ and } \mu(t) = \mu_0 e^{\frac{-t}{\tau}} \tag{4}$$

where τ is the annealing rate. The overall learning rate for the ith predictor is thus given by:

$$\eta_i(t) = \mu(t) \cdot g_i(t) \tag{5}$$

2.1 Simulation

In order to better understand the algorithm, we used linear predictors trained with the normalized LMS algorithm [1] for the following simulation. For viewing convenience, we used a one dimensional continuous map, where the last

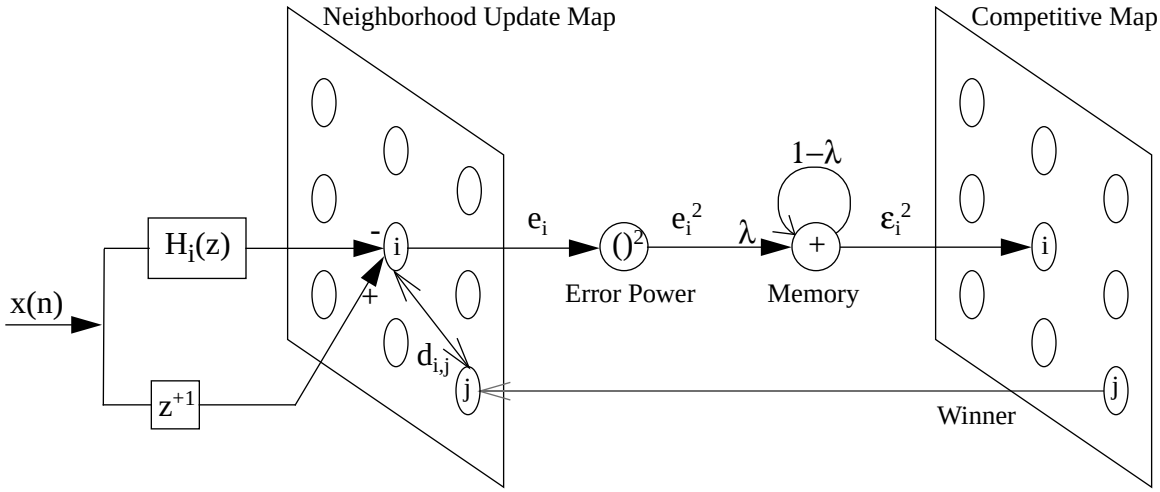


Figure 1: Neighborhood map of competitive predictors.

node is mapped as a neighbor to the first node. Training was conducted for 50 passes through the entire time series, using an annealing rate of $\tau = 5$ passes.

The example is a switching FIR process. The time series consisted of zero mean Gaussian noise of unit variance filtered by a 6th order normalized FIR filter, whose coefficients are random and change every 200 samples. The time series, which consists of a total of 8 stationary regions, is shown in Fig. 2a.

For the predictors, we used eight 8th order predictors. The memory depth of the error was 25 samples. The initial neighborhood and learning rate were 16 and 1, respectively. The winning predictors after training are shown in Fig. 2b.

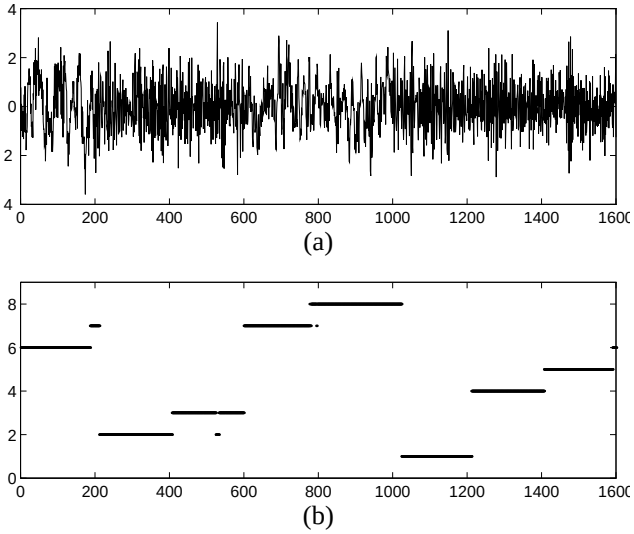


Figure 2: Switching FIR process: a) time series, b) winning predictors after training.

The time series was fairly successfully segmented, except for a longer than expected winning run for predictor 8, and minor switching errors near samples 200 and 475. Those latter errors correlate well with outliers in the time series, which temporarily degrades the performance of the winning predictor. However, in both cases, recovery occurred rapidly.

From the FIR coefficients that produced the series, we can define a similarity coefficient, which is tabulated in Table 1:

$$|\cos\theta_{i,j}| = |W_i \cdot W_j| . \qquad (6)$$

Table 1: Distance between the eight stationary regions

Region	I	II	III	IV	V	VI	VII	VIII
I	1							
II	.38	1						
III	.36	.61	1					
IV	.86	.51	.40	1				
V	.75	.23	.17	.79	1			
VI	.07	.38	.03	.40	.59	1		
VII	.25	.46	.63	.25	.38	.42	1	
VIII	.65	.59	.69	.43	.39	.01	.57	1

Region I refers to samples 1-200, region II refers to samples 201-400, and so on. Using this metric, the two regions with the highest similarity (0.86) were sample regions 1-200 and 601-800. From the neighborhood map, we indeed find that neighboring predictors 6 and 7 won those regions. Likewise, the two regions with the second highest similarity (0.79) were sample regions 601-800 and 801-1000,

won by neighboring predictors 7 and 8. Alternatively, we find that the two regions the most dissimilar (0.01) were sample regions 1001-1200 and 1401-1600. Indeed, the winners of those two regions are predictors 1 and 5, as far apart as the map will allow. Overall, there is a very high correlation between the similarity of two regions and their distance on the neighborhood map.

3. SELF-ANNEALING COMPETITIVE PREDICTION

In the ACE algorithm [3], memory is used in the gating function only. Here we propose a cost function that imbeds memory in the error measure:

$$E(t) = \sum_{i=1}^{K} g_i(t)\,\varepsilon_i(t) \tag{7}$$

where ε_i is the recursive estimate of the mean square error of the ith predictor from (2). Furthermore, in the ACE algorithm, the degree of competition, 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 [6] 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 noninformative prior distribution, Niedzwiecki's formula for the probability of the ith expert is given by

$$g_i(t) = \frac{\varphi_i(t)}{\sum_{j=1}^{K} \varphi_j(t)}, \tag{8}$$

$$\varphi_i(t) = [\varepsilon_i(t)]^{\frac{-M}{2}}, \tag{9}$$

$$\varepsilon_i(t) = \frac{1}{M}\sum_{m=0}^{M-1} e_i^2(t-m)\,. \tag{10}$$

M is the memory depth in samples. Equation (9) embodies the coupling between memory depth and degree of competition. We have borrowed Niedzwiecki's result and incorporated it into our gating function. We use (8) as the gating function, except that we replace the moving average calculation of the mean square average in (10) by our recursive calculation in (2). Likewise, we replace the memory depth, M, in (9) by the recursive parameter λ^{-1}.

$$\varphi_i(t) = [\varepsilon_i(t)]^{\frac{-1}{2\lambda}}. \tag{11}$$

We then do gradient descent on *both* the calculated gate and the mean square errors of (7). Taking the partial derivative with respect to any free parameter, w_k, in the kth system, results in the following sets of equations

$$\Delta w_k = -\eta\frac{\partial}{\partial w_k}E(t) = -\eta g_k\left[\frac{E(t)}{\varepsilon_k(t)} + 2\lambda - 1\right]z_k(t)\,, \tag{12}$$

$$z_k(t) = e_k(t)\frac{\partial}{\partial w_k}y_k(t) + (1-\lambda)\,z_k(t-1)\,, \tag{13}$$

and with respect to the λ parameter:

$$\Delta\lambda = \frac{-\eta}{2\lambda}\sum_{i=1}^{K} g_i\left[\left(\frac{E}{\varepsilon_i} + 2\lambda - 1\right)v_i(t) - \frac{[E-\varepsilon_i]\log\varepsilon_i(t)}{\lambda}\right] \tag{14}$$

$$v_i(t) = \frac{\varepsilon_i(t) - \varepsilon_i(t-1)}{\lambda} + (1-\lambda)\,v_i(t-1)\,. \tag{15}$$

Equation (13) 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 (7) instead of the instantaneous error.

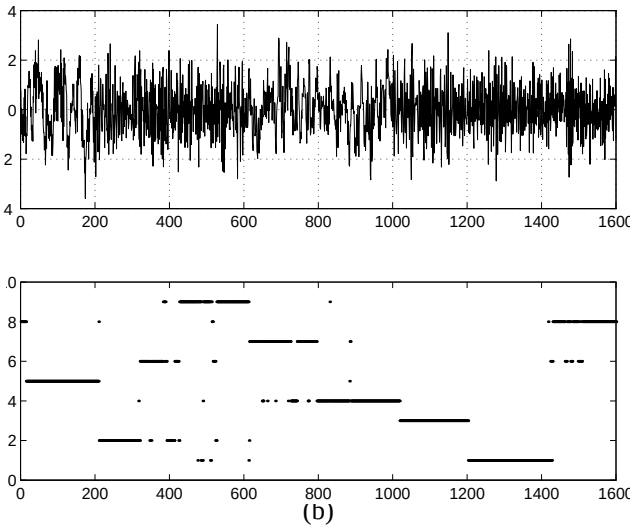
The competitive nature of the algorithm is evident in (12), where the total weight update is given by the product of z_k with the probability of the kth expert, g_k, and another term which depends on the inverse of the mean square error of the kth expert. Thus, the expert with the smallest mean square error will have the largest weight update. Furthermore, (12) 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.

3.1 Simulation

To test the algorithm, we used the same switching FIR process as in the previous simulation. We also used the same number and type of predictors. The initial value of λ was 1, and after 30 iterations, it had evolved to a value of 0.6, corresponding to a memory depth of ~1.6 samples. The winning predictors after training are shown in Fig. 3b.

The time series was partially segmented, except there is more spurious switching than there was for the Neighborhood Map of Predictors. This is because λ did not adapt to a small enough value to invoke hard competition, whereas in the previous algorithm, it was experimentally set to a value which gave a stable segmentation. Still, the

results are encouraging.



(b)

Figure 3: Switching FIR process: a) time series, b) winning predictors after training.

4. CONCLUSIONS

We have taken the concept of self-organization in space and extended to time series through a front end of dynamic models trained as (linear or nonlinear) predictors. The individual predictors compete for the input data on the basis of their error performance in the recent past. The winner is granted the largest parameter update, and others are granted smaller updates on the basis of their distance from the winner. Annealing of the map is achieved in exact analogy to Kohonen learning. After training, observation of the winning predictors segments the time series according to stationary regions, and "similar" stationary regions are mapped as neighbors in the predictor map. Using this algorithm, we successfully segmented and identified different time series.

We then proposed a new cost function that imbeds the mean square error which, when computed recursively, leads naturally to momentum learning. We also coupled the memory depth of the mean square error with the degree of competition such that the system can autonomously adapt the memory depth for the problem at hand. Simulation revealed that the model does indeed self-organize, but the segmentation was not as hard as when the memory depth was determined experimentally.

Experiments are being conducted to validate the models for speech recognition and time series analysis.

Acknowledgments: This work was partially supported by ARPA/ONR grant N00014-94-1-0858 and NSF grant ECS-9510715.

References

[1] Fancourt C., and Principe J., A Neighborhood Map of Competing One Step Predictors for Piecewise Segmentation and Identification of Time Series, *ICNN 96* , vol. 4, pp. 1906-1911, 1996.

[2] 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.

[3] Kohlmorgen J., Muller K.-R., and 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

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

[5] Muller K.R., Kohlmorgen J., and Pawelzik K., Analysis of Switching Dynamics with Competing Neural Networks, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, in press, 1995

[6] Niedzwiecki M., Multiple Model Approach to Finite Memory Adaptive Filtering, *IEEE Transactions on Signal Processing*, vol. 40, no. 2, pp. 470-473, 1992.

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

[8] Weigend A.S., Mangeas M., and Srivastava A.N., Nonlinear gated experts for time series: discovering regimes and avoiding overfitting, International Journal of Neural Systems, Vol. 6, No. 4, 1995.

[9] Widrow B., and Stearns S.D., Adaptive Signal Processing, Englewood Cliffs, NJ: Prentice-Hall, 1985.