General Convergence Results for Data Allocation in Online Unsupervised Learning Methods

V. Petridis
petridis@vergina.eng.auth.gr

Ath. Kehagias
kehagias@egnatia.ee.auth.gr
http://skiron.control.ee.auth.gr/ kehagias/index.htm

Automation and Robotics Lab
http://skiron.control.ee.auth.gr

Dept. of Electrical and Computer Engineering
Faculty of Engineering, Aristotle University
Thessaloniki, GR 54006 GREECE

General Convergence Results for Data Allocation in Online Unsupervised Learning Methods

V. Petridis and Ath. Kehagias
Dept. of Electrical and Computer Engineering
Aristotle University of Thessaloniki

1. Introduction

- Modular neural networks can be used for the solution of learning problems which involve data generated by several alternately activated sources.

- A well trained module models a particular source. A well trained modular neural network is characterized by a one-to-one association of sources to well trained modules (such that each module models a exactly one source).

- Training of a modular neural network can be decomposed in two components:

  – Module Competition for Data Allocation.

  – Training of the modules on the allocated data.

  Of these two components, competitive data allocation is the critical one.

- Competitive data allocation is based on the difference between source and module output.

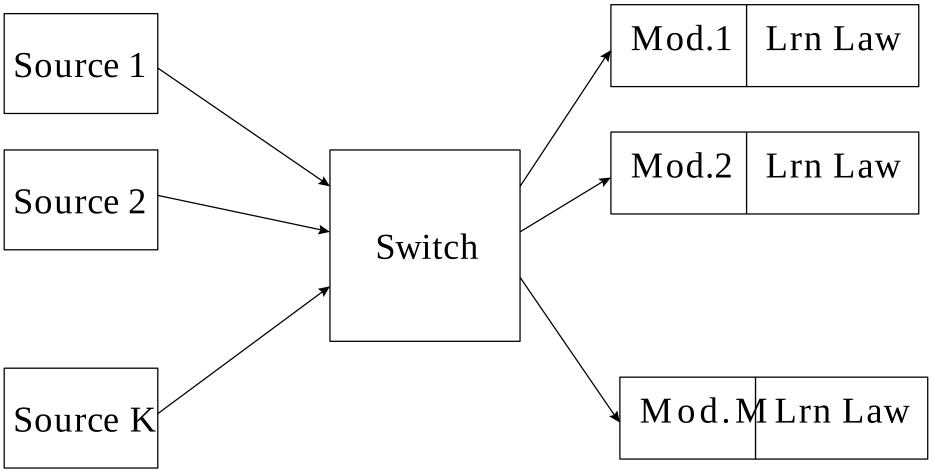
- Each module tends to collect data which correspond to a particular source behavior. Since each module is periodically retrained on its data set, the particular behavior is reinforced and more data of the same type are collected.

- It can be proved that under reasonable separability conditions, this process is self reinforcing, so that in the long run each module implements a constraint which characterizes a particular source. Hence a one-to-one correspondence of sources and modules is attained.

- The above framework can be applied to both static and dynamic data.

2. A General Framework for Unsupervised Competitive Learning

The type of problem we consider can be illustrated by the following figure.



The sources are described by models the form of which will be given separately for dynamic and static problems in the sequel.The learning system consists of a number of modules. Each module is equipped with a learning law. The relaying system can transform the data in a general way. However in this presentation we will restrict ourselves to relaying systems which at a specific instant of time simply select a source. In such a case we can talk of a "switching system". Hence data are presented serially to the learning system forming a time series Y_t, $t = 1; 2:::;$ taking values in R^N.

A module represents a family of models is parameterized by a parameter vector $µ$ taking values in $£_m$ ½ R^p (for $m = 1; 2; :::; M$ where M is the number of modules). It must be emphasized that the models need all be of the same form. At time t the m-th module has parameter vector $µ_t^m$. An error function is also defined (for $m = 1; 2; :::; M$) by

$$E_t^m = Q(Y_t; Y_{t¡1}:::; Y_{t¡L}; ; µ_t^m);$$

this is a measure of the extent at which Y_t satisfies the constraint implemented by module m.

At time t the allocation of data (i.e. which module or modules will use Y_t to adapt its parameters) is the result of competition among modules on the basis of the errors E_t^m (for $m = 1; 2; :::M$). A reasonable way of doing this is by allocating Y_t to the module with minimum E_t^m. The adaptation of module parameters is carried out by the learning law of the corresponding module. It is desirable that after the presentation of a sufficiently large number of data each module tends to specialize in one source; in other words the parameter vector $µ_t^m$ tends to a value $\bar{µ}^m$ such that $Q(Y_t; Y_{t¡1}:::; Y_{t¡L}; ; \bar{µ}^m)$ wins the competition whenever data from the m-th source appear. Hence $Q(Y_t; Y_{t¡1}:::; Y_{t¡L}; ; \bar{µ}^m)$ can be viewed as an approximate model for the m-th source.

The above formulation for unsupervised learning can be applied to either static or dynamic problems (i.e. problems involving time series or dynamic systems).

Dynamic Patterns

In this case the sources generate time series described by

$$Y_t = F_k(Y_{t-1}, Y_{t-2}, \ldots, Y_{t-L})$$

where $Y_t \in R^N$ and $F_k(\cdot)$ (for $k = 1, 2, \ldots, K$) are the source models. In general, Y_t^k may also depend on an input U_t, but we omit such a dependence for simplicity of presentation. The switching system generates a time series Z_t $(t = 1, 2, \ldots)$ taking values in a finite set $\{1, 2, \ldots, K\}$. Hence the output of the switching system is

$$Y_t = F_{Z_t}(Y_{t-1}, Y_{t-2}, \ldots, Y_{t-L}).$$

The task is to identify the different time series (or dynamic systems). In this case, the m-th module is defined by

$$Y_t^m = f_m(Y_{t-1}, Y_{t-2}, \ldots, Y_{t-L}, \theta_t^m)$$

For simplicity, assume that the number of modules is equal to the number of sources, i.e. that $K = M$. The error function is

$$E_t^m = |Y_t - Y_t^m|.$$

The models compete and the resulting winner adapts its parameter vector according to its learning law. Recall that θ_t^m is the estimate of the m-th parameter vector at time t. The goal is to adjust the θ's so that in the long run a correspondence

$$m(k) : \{1, 2, \ldots, K\} \to \{1, 2, \ldots, M\}$$

is achieved such that $f_{m(k)}(y_1, y_2, \ldots, y_L, \theta_t^{m(k)})$ is close to $F_k(y_1, y_2, \ldots, y_L)$. If $m(k)$ is also one-to-one, then we have obtained a well trained network.

Static Patterns

Assume that the k-th source represents data within a domain D^k ½ R^N with a given probability distribution P_k. A possible source model could be a function ($k = 1; 2; :::; K$)

$$v^k = \begin{cases} 1 & \text{for } Y_t \ 2 \ D^k \\ 0 & \text{for } Y_t \ \not 2 \ D^k: \end{cases}$$

Suppose now that the switching system activates a random generator r_t^k that at time t produces a datum Y_t from the k-th source. This means that Y_t belongs to D^k and is generated according to probability distribution P_k. As in the dynamic case, the switching system generates also a time series Z_t, $t = 1; 2; :::$ taking values in $f1; 2; :::; Kg$. Then the output of the switching system at time t is given by

$$Y_t = r_t^{Z_t} \text{ where } Y_t \ 2 \ [_{k=1}^K D^k: \tag{1}$$

A module is related to a set $\hat{D}_t^m = fy : Q_t^m(y; µ_t^m) < dg$;where d is a threshold, $Q_t^m(:; :)$ is a nonnegative function and $\hat{D}_t^k$ ½ R^N: In this case the module implements a function

$$\hat{v}_t^k = \begin{cases} 1 & \text{for } Y_t \ 2 \ \hat{D}_t^k \\ 0 & \text{for } Y_t \ \not 2 \ \hat{D}_t^k: \end{cases}$$

we have the error function is given by

$$E_t^m = Q_t^m(Y_t; µ_t^m):$$

It has been assumed here (for simplicity) that $K = M$, i.e. the number of modules is equal to the number of sources. The models compete and the resulting winner adapts its parameter vector according to its learning law. The goal of the learning process is that for a sufficiently large number of data the m-th module approximates the k-th source model in the sense that the data Y_t, for which the m-th module wins the competition, belong to $\hat{D}_t^m$ which is approximately equal to D^k . For example, if in the long run a correspondence

$$m(k) : f1; 2; :::; Kg \ ! \ f1; 2; :::; Mg$$

is achieved such that

$$¹ \left(D^k \ ¡ \ \hat{D}_t^{m(k)} \right) = 0;$$

where ¹(:) is a measure and m(k) is one-to-one, then it follows that

$$\hat{v}_t^k \ ! \ v_t$$

in some appropriate convergence sense and we have obtained a well trained network.

As an example, consider a simple competitive neural network learning scheme in which data are generated according to equation (1) and

$$E_t^m = jjW_t^m \ ¡ \ Y_tjj;$$

here W_t^m are the weights of the m-th neuron at time t and they play the role of the parameters $µ_t^m$. In this scheme the neuron (i.e. module) with the smallest value of E_t^m wins and its weights are updated; the cluster corresponding to the m-th module is represented by

$$\hat{D}_t^m = fy : jjW_t^m \ ¡ \ yjj < dg:$$

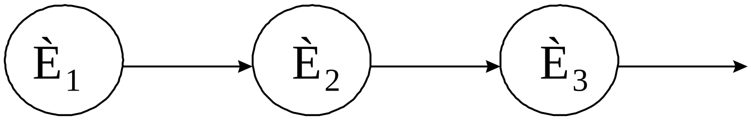
3. Module Competition

The competition is based on the errors E_t^m, $m = 1; 2; ::: $. In general two schemes of competition can be distinguished: serial and parallel.

Serial scheme

a. **Serial scheme with fixed number of modules.** In this scheme the number of modules, M , is fixed. The error of the 1st module , E_t^1 is compared with a threshold d ; if $E_t^m < d$, the incoming datum is added to this module's data set. Otherwise the error of the 2nd module is compared with d; if it is less than d module 2 wins. Otherwise the error is compared with the 3rd module and so on. The process continues until a module exhibits an error less than d. In case all E_t^m are greater than d, the last module is taken to be the winner.
b. **Serial scheme with open number of modules.** This scheme is the same as the previous one except that a new module is added whenever the condition $E_t^m < d$ is not satisfied for any of the existing modules.

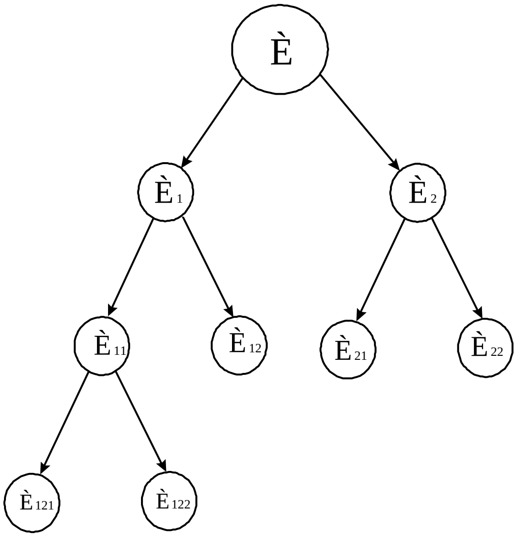
In both cases, predictors are added in a cascaded fashion (resulting in piecewise partition of the source space) as is illustrated in the figure.



Parallel scheme

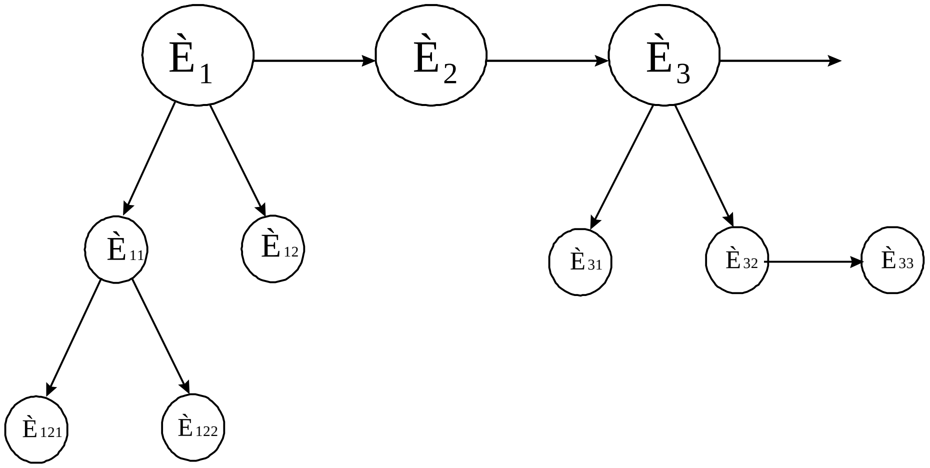
In this case we distinguish two types of competition as well: the parallel scheme with fixed number of modules and the parallel scheme with open number of modules.
a. **Parallel scheme with fixed number of modules.** In this scheme the error is computed for all modules ; the winner is the module with minimum error.
b. **Parallel scheme with open number of modules.** In this scheme the error is computed for all modules as well; the winner is the module with minimum error provided this is below a threshold . Otherwise a new module is added.

In both of the above schemes, predictors are added in a tree-like fashion (resulting in recursive partition of the source space) as is illustrated in the figure.



Hybrid Schemes

The first of the above schemes is based only on serial comparisons, while the second is based only on parallel comparisons. Hybrid schemes can be devised which use both serial and parallel comparisons. The following figure illustrates the mode of operation of hybrid schemes.



4. CONVERGENCE

Two Sources, Two Modules

In this section we will present convergence theorems concerning the serial and parallel schemes with two sources and modules. We will need the following variables.

- N_t^{ij}: is the number of data generated by the i-th source and allocated to the j-th predictor up to time t. $(i; j = 1,2)$.
- X_t: is the specialization variable, which is defined differently for the serial and parallel case.
  - Parallel Case: $X_t = [N_t^{11} - N_t^{21}] + [N_t^{22} - N_t^{12}]$. When X_t is large and positive, either module no.1 specializes in source no.1, or module no.2 specializes in source no.2, or both. When X_t is large and negative, either module no.1 specializes in source no.2, or module no.2 specializes in source no.1, or both.
  - Serial Case: $X_t = N_t^{11} - N_t^{11}$. When X_t is large and positive, module no.1 specializes in source no.1. When X_t is large and negative, module no.1 specializes in source no.2.
- Data Allocation probablities.

$$a_n \doteq \Pr(\text{Module no.1 accepts } Y_t | Z_t = 1; X_{t-1} = n);$$

$$b_n \doteq \Pr(\text{Module no.1 accepts } Y_t | Z_t = 2; X_{t-1} = n):$$

Note that even if the definition has the same form, a_n, b_n are actually different in the serial and parallel case, since X_t is defined differently in each case.

The following two assumptions are used both in the parallel and serial case; note however that in each case they refer to different variables.

A1 For all n, $a_n > 0$; and $\lim_{n \to +\infty} a_n = 1, \quad \lim_{n \to -\infty} a_n = 0.$

A2 For all n, $b_n > 0$, and $\lim_{n \to +\infty} b_n = 0, \quad \lim_{n \to -\infty} b_n = 1.$

Then it can be shown that X_t is Markovian and the following theorems hold (both for the parallel and serial case).

Theorem 1 If A1 and A2 hold, then

$$\text{For } m = \ldots, -1, 0, 1, \ldots \quad \Pr(X_t = m \quad \text{i.o.}) = 0, \tag{2}$$

$$\Pr\left(\lim_{t \to \infty} |X_t| = +\infty\right) = 1, \tag{3}$$

$$\Pr\left(\lim_{t \to \infty} X_t = +\infty\right) + \Pr\left(\lim_{t \to \infty} X_t = -\infty\right) = 1. \tag{4}$$

Theorem 2 If A1 and A2 hold, then

1. If $\Pr(\lim_{t \to \infty} X_t = +\infty) > 0$ then

$$\Pr\left(\lim_{t \to \infty} \frac{N_t^{21}}{N_t^{11}} = 0 \,\middle|\, \lim_{t \to \infty} X_t = +\infty\right) = 1, \tag{5}$$

$$\Pr\left(\lim_{t \to \infty} \frac{N_t^{12}}{N_t^{22}} = 0 \,\middle|\, \lim_{t \to \infty} X_t = +\infty\right) = 1. \tag{6}$$

2. If $\Pr(\lim_{t \to \infty} X_t = -\infty) > 0$ then

$$\Pr\left(\lim_{t \to \infty} \frac{N_t^{11}}{N_t^{21}} = 0 \,\middle|\, \lim_{t \to \infty} X_t = -\infty\right) = 1, \tag{7}$$

$$\Pr\left(\lim_{t \to \infty} \frac{N_t^{22}}{N_t^{12}} = 0 \,\middle|\, \lim_{t \to \infty} X_t = -\infty\right) = 1. \tag{8}$$

Theorem 2 states that with probability one both predictors will specialize, one to each source and in the “strong” ratio sense.

K sources, M modules

It can be shown (by an informal argument) that in this case specialization also takes place. Consider first the case of serial data allocation. Take a time series $y_1, y_2, \ldots, y_t, \ldots$ generated by K sources and start with two randomly initialized modules. In the initial phase of data allocation module no.1 may collect (perhaps in a random manner) a data set where one source is more heavily represented than the rest. This will result in a slight specialization in this source and, consequently, module no.1 will have a tendency to accept more data from the preferred source. It follows that module no.2 will collect more data from the remaining sources.

An error threshold is used to determine if and when a new module should be added. After a while, module no.1 will be very well specialized in some source. Since module no.2 receives a mixed data set it is unable to specialize; i.e. is still characterized by a large prediction error. In this case, after a while an additional module (module no.3) will be introduced and modules no.2 and 3 will receive all the data rejected by module no.1. Sooner or later module no.2 will also specialize in one source and module no.3 will receive incoming data from the remaining sources. Proceeding in this manner, modules will keep being added until to every active source will correspond one well specialized module. In short, serial data allocation can lead to succesful source identification.

Consider next the case of parallel data allocation. A time series $y_1, y_2, \dots, y_t, \dots$ generated by K sources is initially partitioned by using two randomly initialized modules. Now, if a data group is characterized by a preponderance of data from a particular source or group of sources, the respective module specializes in this source or group of sources, resulting in improved prediction accuracy for this type of data. The source groups can be refined by the gradual introduction of more modules in a top down manner. If after a large number of data have been collected and the average prediction error is above d, it may be assumed that in fact the data set corresponds to more than one sources; hence the algorithm attempts to split the data set into two subsets, by replacing the corresponding module by two identical copies of it. Initially both new modules may be expected to show a preference for data generated by the particular source group, but no particular specialization for any source belonging to this group. However, if most new incoming data have been generated by sources belonging to this group, it may be expected that a further partition of such data will be effected. In short, parallel data allocation can lead to succesful source identification.

5. Time Series Examples

We apply the above algorithms to the problem of allocating data from a time series generated by three chaotic sources. In particular, the sources are described by

1. Logistic: $Y_t = F_1(Y_{t ¡ 1})$, where $F_1(y) = 4 ¢ y ¢ (1 ¡ y)$.
2. Tent Map: $Y_t = F_2(Y_{t ¡ 1})$, where $F_2(y) = 2y$ when $y\ 2\ [0; 1=2]$, $F_2(y) = 1 ¡ 2y$ when $y\ 2\ (1=2; 1]$.
3. Double Logistic: $Y_t = F_3(Y_{t ¡ 1})$, where $F_3(y) = F_1(F_1(y))$.

The prediction modules used are sigmoid 1-5-1 neural networks. A sequence of experiments is conducted, where a time series Y_1, Y_2, ... , Y_{10000} is produced by periodic activation of the three sources (with a source activation period of 200 time steps). The time series is mixed with additive white noise, distributed uniformly in the range $[¡ A=2; A=2]$. Several data allocation eperiments are conducted for every noise level A.

Figure 1 presents a graph of data allocation for an experiment conducted at noise level $A = 0:04$. We plot allocation of ten-step data blocks to three modules (module no.4 corresponds to rejected data). Hence the x-axis corresponds to time and the y-axis corresponds to activated source (solid line) or module which receives the current data block (dashed and dotted line). It can be seen that the serial algorithm achieves faster data allocation, but the parallel algorithm is more accurate.

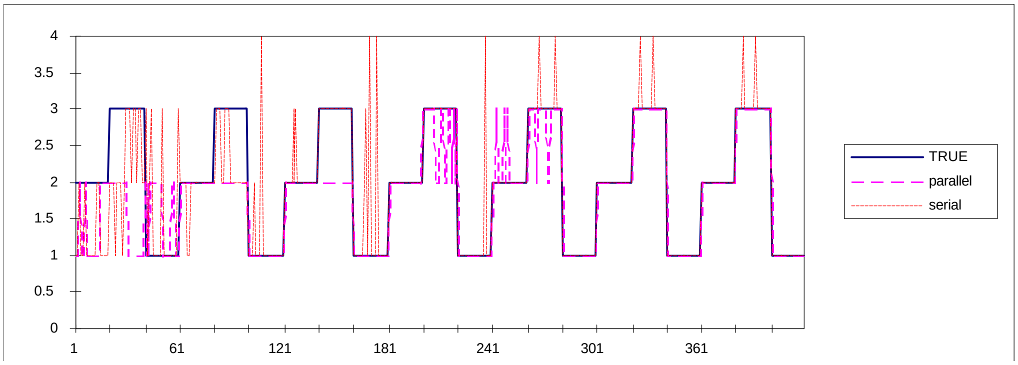


Figure 1

Figure 2 presents a graph of the classification accuracy, as measured by the variable c (corresponding to the x-axis and defined as number of correctly allocated data divided by total number of data) plotted against noise level A (corresponding to the x-axis). It can be seen that parallel data allocation is extremely robust to noise.

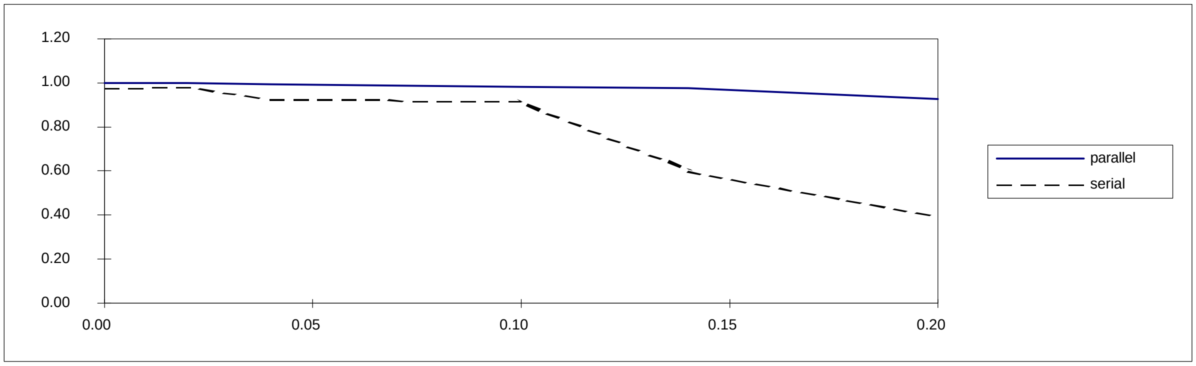


Figure 2

6. Discussion

The critical part in this process is the module competition for data allocation: if each module accepts data from a single source and rejects all other data, then a number of training algorithms are available to produce well specialized modules; if, on the other hand, a module receives a mixed set of data (i.e. data from several sources) then, no matter how efficient the learning algorithm is, specialization is unlikely to take place.

In our analysis two separability assumptions are crucial; namely, when a module accepts an additional datum from a particular source then: (A1) the probability of accepting more data from the same source increases and (A2) the probability of accepting data from other sources decreases. Under assumptions A1, A2, as the number of observed data goes to infinity, the fraction of misallocated data goes to zero with probability one. In this sense, asymptotically and with probability one, every source is associated with a distinct module and every module is associated with a distinct source.

We believe that the above conditions can be related to specific learning algorithms, network architectures and time series sources using the information theoretic concepts of source complexity and network capacity. A connection with PAC learnability is also possible. However, it must be noted that the analysis presented here is quite general and applies to a large variety of combinations of learning algorithms, network architectures and time series sources. Also, the analysis appplies equally well to static and time series data. Assumptions A1 and A2 are not specific to a particular competition scheme, hence the convergence analysis may be expected to hold for a large class of unsupervised learning algorithms which involve modular neural networks, such as mixtures of experts, regime decomposition, combined estimators, committee machines, predictive modular neural networks etc. In addition to modular neural networks, the analysis applies to situations where a single network of sufficiently complex neurons is employed, provided that each neuron specializes in a particular type of data; such cases include the well known k-means algorithm, Learning Vector Quantization (LVQ), Self Organizing Feature Maps (SOFM), Adaptive Resonance Theory (ART) systems etc. Our approach can be applied in case no alternative convergence analysis is available; if a case-specific convergence analysis is available, our approach still provides an attractive alternative. By appropriate modification of assumptions A and B, the analysis also applies to soft data sharing algorithms.

On a more applied level, we have presented two algorithms (parallel and serial) which implement the above idea and applied them to a problem of time series modelling. Both algorithms perform quite accurately; the parallel algorithm is more noise robust, while the serial one converges faster.

References

[1] G. A. Carpenter, S. Grossberg, and D. B. Rosen. 1991. Fuzzy ART: fast stable learning and categorization of analog patterns by an adaptive resonance system. Neural Networks, 4:759–771.

[2] G. A. Carpenter, S. Grossberg, and J. H. Reynolds. 1991. ARTMAP: supervised real-time learning and classification of nonstationary data by a self-organizing neural network. Neural Networks, 4:565–588.

[3] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. 1991. Adaptive mixtures of local experts. Neural Computation, 3:79–87.

[4] M. I. Jordan and R. A. Jacobs. 1994. Hierarchical mixtures of experts and the EM algorithm. Neural Computation, 6:181–214.

[5] A. Kehagias and V. Petridis. 1997. Time Series Segmentation using Predictive Modular Neural Networks. Neural Computation, 9:1691-1710.

[6] A. Kehagias and V. Petridis. 1997. Predictive Modular Neural Networks for Time Series Classification. Neural Networks, 10:31-49.

[7] T. Kohonen. Self-organized formation of topologically correct feature maps. Biological Cybernetics, 43:59–69, 1982.

[8] T. Kohonen. Learning vector quantization. Neural Networks, 1:303-315, 1988.

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

[10] V. Petridis and A. Kehagias. 1996. A Recurrent Network Implementation of Time Series Classification. Neural Computation, 8:357-372.

[11] V. Petridis and A. Kehagias. 1996. Modular Neural Networks for MAP Classification of Time Series and the Partition Algorithm. IEEE Trans. on Neural Networks, 7:73-86.