Data Classification for Unsupervised Learning of Multiple Models: Convergence Results

V. Petridis and Ath. Kehagias

Dep. Of Electrical and Computer Engineering
Aristotle University
Thessaloniki GR 54006, Greece

ABSTRACT

In this paper we examine a problem which arises in connection with the application of the Lainiotis Partition Algorithm to tasks of signal classification, prediction and parameter estimation. We are particularly interested in tasks which involve composite systems, comprising of a finite number of *switched sub-systems*. The problem we consider arises in situations of *unsupervised, online* classification and modeling and can be characterized as a problem of *data allocation*, i.e. how to partition observed data into separate training sets and use the members of each set for training the model of a particular sub-system. We propose an algorithm that effects unsupervised, online data allocation and prove that under mild separability conditions the algorithm converges to the "correct" solution. The proposed algorithm is also tested by numerical experiments.

Keywords: Partition Algorithms, Classification, Prediction, Parameter Estimation, Multiple Models.

1. INTRODUCTION

The Lainiotis partition algorithm [4,10] is a powerful tool to be used in classification, prediction and parameter estimation problems involving *switched systems,* i.e. composite systems which comprise of alternatively activated sub-systems (for examples of such applications see [1,2,3,5,6,7,8]). Formally, we have in mind the following situation:

(1) $x(t) = f(x(t-1), u(t-1) ; z(t))$
(2) $y(t) = x(t) + v(t)$

where u(t) is the *control* input (taking values in $\mathbf{R}^m$), x(t) is the *state vector* process, y(t) is the *observation* process and v(t) is a *white noise* process (all taking values in $\mathbf{R}^n$) and z(t) is the *switching* process taking values in {1,2,…,K}. In particular, eqs.(1), (2) imply that the system comprises of K sub-systems which are described by the following equations (for k = 1, 2, …, K)

(3) $\chi(t) = f(\chi(t-1), u(t-1) ; k)$
(4) $\psi(t) = \chi(t) + v(t)$

In other words, the "master" system of eqs.(1), (2) consists of a collection of sub-systems which evolve in parallel; at time t the "master" system behaves in accordance to the equations of the z(t)-th sub-system. Another interpretation is that at time t the z(t)-th sub-system is *switched on*, to generate the next state of the "master" system.

The Lainiotis Partition Algorithm is a powerful tool which can be utilized to perform classification, prediction and parameter estimation tasks involving systems of the form of eqs.(1)-(4). However, the application of the Partition Algorithm requires that either the functions f(. , . , k) or adequate approximations thereof are available. In case exact models are not available, approximations $f^{(k)}(. , .)$ can be obtained from labeled training data. Obtaining such models from labeled data is a problem of *supervised* learning.

In this paper we are interested in applying the Lainiotis Partition Algorithm to unsupervised learning situations. In other words neither approximate models $f^{(k)}(. , .)$ nor labeled data are initially available. In other words, the problem we are interested in is as follows: a system is observed, in the sense that pairs {u(1), y(1)}, {u(2), y(2)}, … are available and it is known that the data have been generated by a switched system of the form of eqs.(1)-(4). No other information (e.g. the number K of sub-systems, the switching process z(t) etc.) is available. The task is to find K and obtain accurate models $f^{(k)}(. , .)$ for k = 1, 2, … , K.

To solve the above problem we propose the algorithm of Section 2.

2. THE DATA ALLOCATION ALGORITHM

We introduce an online algorithm which allocates data to a number of models and iteratively trains each model on the data allocated to it. Data allocation is performed on the basis of *predictive error*. Specifically, the basic ideas involved in the operation of the algorithm are as follows.

1. K_1 models are randomly initialized.
2. At times t = 1, 2, … observations of the true system (i.e. {u(t), y(t) } pairs) are collected and used to obtain K_1 estimates $y^{(k)}(t)$ (k = 1,2, … , K_1); the respective estimation errors $e^{(k)}(t)$ (k = 1,2, … , K_1) are computed.
3. When a *block* of T_{alloc} observations becomes available, it is allocated to the data pool of the model which has minimum estimation error for the respective period of time. This is expressed in terms of the *data allocation variable* z*(t), which takes the value k when the respective data block is allocated to the k-th model.
4. If, as a result of the allocation, the data pool of a model contains more than T_{store} data pairs, the oldest T_{alloc} data pairs are discarded.
5. Every T_{train} time steps all models are retrained.

The algorithm can be described in pseudo-code as follows.

Data Allocation Algorithm

Input: a sequence of inputs and observations {u(1),y(1)}, {u(2),y(2)}, … ; a sequence of randomly initialized models $f^{(0)}(.,.;k)$, k=1,2 … , K_1.
Parameters: K_1 (number of models / predictors) , T_{alloc} (size of data block), T_{train} (retraining period), T_{store} (size of data kept in memory).
Output: At times $n\cdot T_{train}$, n=1,2, … a sequence of *trained* models $f^{(n)}(.,.;k)$, k=1,2 … , K_1.

Initialization:
$n_{alloc} = 1$;
$n_{train} = 1$;
$f^{(1)}(.,.,;k)$ for k = 1, 2, … , K_1 are randomly initialized; the *data pools* of the k models (k = 1, 2, … , K_1) are filled with T_{alloc} random data pairs.

Main:
```
For t = 1, 2, …
    Read u(t), y(t).
    For k = 1, 2, … , $K_1$
        $x^{(k)}(t) = f^{(k)}(x(t-1),u(t-1))$
        $e^{(k)}(t) = y(t) - y^{(k)}(t)$
    Next k
    If $t = (n_{alloc}+1)\cdot T_{alloc}$ Then
        $n_{alloc} \leftarrow n_{alloc} +1$
        For k = 1, 2, … , $K_1$
            $E^{(k)} = \Sigma_{s=t-Talloc}^{t}|e^{(k)}(s)|$
        Next k
        $k^* = \arg\min E^{(k)}$
        Add $\{u(t-T_{alloc}),y(t-T_{alloc})\}$, … , {u(t),y(t)}
          to the data pool of model k*.
        If the data pool of model k* has more than $T_{store}$ data
          pairs $\{u(\tau), y(\tau)\}$ delete the earliest $T_{alloc}$ data pairs.
    End If
    If $t = (n_{train}+1)\cdot T_{train}$ Then
        $n_{train} \leftarrow n_{train} +1$
        For k = 1, 2, … , $K_1$
            Retrain the k-th model to obtain $f^{(n)}(.,.;k)$
        Next k
    End If
Next t
```

As will be seen in Section 4, when this algorithm is applied to the identification of a switched system, consisting of K sub-systems, it usually produces K highly accurate models $f^{(k)}(.,.,)$ of the sub-system functions f(.,.;k); the remaining K_1-K models are irrelevant. To be more precise, the algorithm produces a mapping $\phi:\{1,2,...,K\} \rightarrow \{1,2,...,K_1\}$ such that (for k = 1,2, .. , K) the k-th sub-system is accurately represented by the $\phi(k)$-th model. An explanation of the effectiveness of the algorithm is presented in the next section.

3. CONVERGENCE

The algorithm presented above is based on a *self-reinforcement* idea. Let us explain this idea informally for a problem involving only two sub-systems and two models Suppose then that initially the two models are randomly initialized; it may be expected that the first model will be slightly better in approximating one of the two sub-systems (say, sub-system 1). As a result, it may be expected that the prediction error $y^{(1)}(t)$ will be somewhat smaller than $y^{(2)}(t)$ for times t where the first sub-system is activated. Hence, generally speaking, the first model will have a tendency to collect more data blocks which contain sub-system 1 data than the second model. At the next retraining time, the data pool of model 1 will contain more sub-system 1 data than sub-system 2 data; hence after retraining model 1 will be even better at modeling the behavior of sub-system 1. This will reinforce the tendency of model 1 to collect more sub-system 1 data, hence the data pool of this model will contain such data at an even higher proportion. It turns out that, under suitable conditions, this process is reinforced to the extent that, asymptotically, the data pool of model 1 will contain exclusively sub-system 1 data. Correspondingly, the data pool of model 2 will contain exclusively sub-system 2 data. Of course, it may turn out that model 1 is mapped to sub-system 2, rather than to sub-system 1. However, it turns out that, *with probability* 1, each model will be mapped to one sub-system, in the sense that each data pool will contain data belonging exclusively to one sub-system.

The above informal analysis can be stated and proved rigorously, in the form a theorem. To state the theorem, we need to define the following quantities:

$N_{ij}(t)$ = Number of data pairs generated by sub-system i and assigned to model j (i,j = 1,2) up to time t.

$$X(t) = N_{11}(t) - N_{21}(t) + N_{22}(t)-N_{12}(t).$$

X(t) signifies the surplus of assignments from either sub-system to the first model, plus the surplus of assignments from either sub-system to the second model. Hence, if X(t) goes to either plus infinity or minus infinity, it follows that at least one predictor has a surplus of assignments of data blocks generated by a particular sub-system.

Now we can state the data allocation convergence theorem for the case of two sub-systems and two models. The proof of the theorem appears in [9]. Conditions **A1, A2**, which are mentioned in the theorem are separability conditions and can also be found in [9].

Theorem. If conditions **A1, A2** hold, then

(i) $\mathrm{Prob}\left(\lim_{t\to\infty} X(t) = +\infty\right)+\mathrm{Prob}\left(\lim_{t\to\infty} X(t) = -\infty\right) = 1.$

(ii) $\mathrm{Prob}\left(\lim_{t\to\infty} N_{21}(t)/N_{11}(t) = 0 \,\middle|\, \lim_{t\to\infty} X(t) = +\infty\right) = 1,$
$\mathrm{Prob}\left(\lim_{t\to\infty} N_{12}(t)/N_{22}(t) = 0 \,\middle|\, \lim_{t\to\infty} X(t) = +\infty\right) = 1,$
$\mathrm{Prob}\left(\lim_{t\to\infty} N_{11}(t)/N_{21}(t) = 0 \,\middle|\, \lim_{t\to\infty} X(t) = -\infty\right) = 1,$
$\mathrm{Prob}\left(\lim_{t\to\infty} N_{22}(t)/N_{12}(t) = 0 \,\middle|\, \lim_{t\to\infty} X(t) = -\infty\right) = 1.$

The above theorem refers to the case of two sub-systems and two models. We have not been able to prove a corresponding theorem for the case of K sub-systems and K_1 models, but certain ehuristic arguments (presented in [???]) indicate that in this case too convergence to correct data allocation will take place.

The above theoretical and heuristic analysis is in agreement with the experimental results which we present in the next section.

4. EXPERIMENTS

We have performed several numerical experiments to test the performance of our data allocation algorithm. In this section we present the results of two groups of experiments, based on data generated by a *switched* system, composed of three linear subsystems.

The System
The composite, switched system consists of the combination of three linear, periodically activated, systems. More precisely, the composite system is described by the following equations.

(5) $x(t) = A(z(t)) \cdot x(t-1) + B(z(t)) \cdot u(t-1)$
(6) $y(t) = x(t) + v(t)$.

Here z(t) is a periodic function: z(t) = 1 for times t = 1,2,…, 50, 151, 152, … , 200, … ; z(t) = 2 for times t = 51,52,…, 100, 201, 202, … , 250, … ; z(t) = 3 for times t = 101, 102,…, 150, 251, 252, … , 300, … . In other words, the composite system consists of three linear systems, which have the form (k=1, 2, 3).

(7) $\chi(t) = A(k) \cdot \chi(t-1) + B(k) \cdot u(t-1)$
(8) $\psi(t) = \chi(t) + v(t)$.

The system of eqs.(5), (6) satisfies $x(t) \in R^3$ and $u(t) \in R^2$; the sub-systems of eqs.(7), (8) satisfy $\chi(t) \in R^3$ and $u(t) \in R^2$ (i.e. $u(t) = [u_1(t)\ u_2(t)]^T$. Input $u_1(t)$ is taken to be a sinusoid and $u_2(t)$ a constant input.

We assume full but noisy state observation. More precisely, v(t) is a noise term, with a structure to be discussed in later sections. The raw data used in all experiments are 2400 time steps-long observation sequences: y(1), y(2), … , y(2400). Two experiment groups have been performed, which differ with respect to the characteristics of the observation noise. All algorithmic parameters are the same for both experiment groups; namely we have used K_1=5 (5 models), T_{alloc} = 10, T_{train} = 30, T_{store}=300.

Experiment Group A: Additive Noise
In the first group of experiments we use additive observation noise. In other words, the sequence v(t) is white noise, with zero mean and variance equal to σ.

In Figs. 1 -- 9 we present various aspects of the data allocation performance. In particular, in Fig.1 we present classification accuracy (c), in Fig.6 prediction error (e) and in Fig.9 parameter estimation error (q). The details regarding the computation of c, e and q will be discussed presently.

In all Figs. 1, 6 and 9 the horizontal axis denotes signal-to-noise ration (S/N) which is computed by the following formula

$$S/N = \sqrt{\left[\Sigma_{\tau=1}^{2400} (y(t))^2\right] / \sigma^2 \cdot 2400}.$$

The value of $\sqrt{\Sigma_{\tau=1}^{2400} (y(t))^2}$ has been computed by averaging a large number of y(t) sequences and has been found to be very close to 150. The value of $\sqrt{\sigma^2 \cdot 2400}$ for σ =1 is 15. Hence, for unit variance white noise, we have S/N = 10. We have repeated the experiment at the following levels of noise, expressed by the signal to noise ratio: S/N = ∞ (noise free), 200, 100, 50, 20, 10, 5, 3, 2.

Classification Accuracy: This is denoted by c(t), in other words is a function of time. It is computed over a sliding window of length equal to 50 time steps. More specifically, if the data allocation variable is denoted by z*(t), then at time t classification accuracy is given by

$$c(t) = \left[\Sigma_{\tau=0}^{49} \mathbf{1}(z(t-\tau) = \phi(z^*(t-\tau)))\right] / 50,$$

where $\mathbf{1}(\ldots)$ is the indicator function and $\phi(.)$ is the previously referred to mapping between sub-systems and models. In short, c(t) counts the proportion of instances, over the last 50 time steps, where the system activation variable equals the *transformed* data allocation variable.

In Fig.1 we present *final* classification accuracy results. In other words, we plot c(2400) vs. S/N ratio. It is worth noting that final classification accuracy is 1 for quite high noise levels (for S/N ratio up to 10). Even when S/N ratio reaches the value 3, classification accuracy stays at 0.8; slow degradation sets in afterwards.

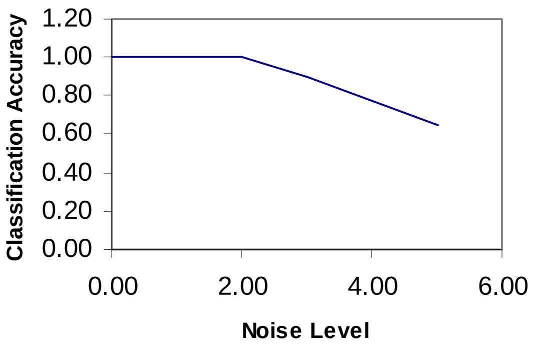

Fig.1
Final classification accuracy plotted against S/N ratio.

It is also instructive to observe the evolution in time of c(t), and of the data allocation variable z*(t). Profiles of these functions are presented for two representative experiments: (a) at S/N = ∞ (noise-free case) in Figs. 2 and 3 and (b) at S/N= 5 in Figs.4 and 5.

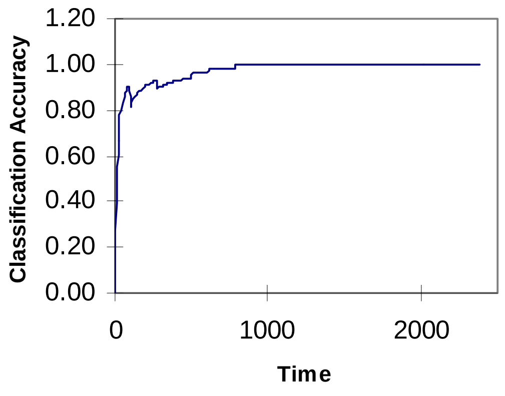

Fig.2
Evolution of classification accuracy in time. This figure corresponds to S/N = ∞.

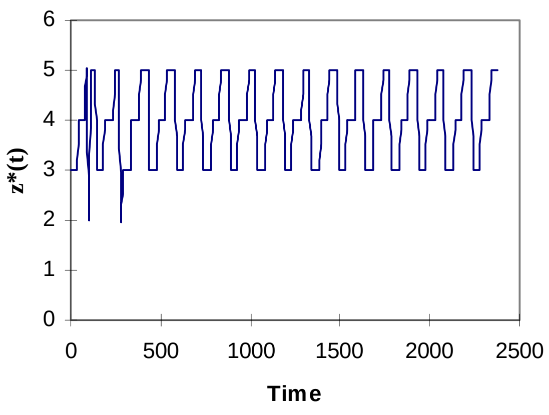

Fig.3
Evolution of z*(t) in time. This figure corresponds to S/N = ∞.

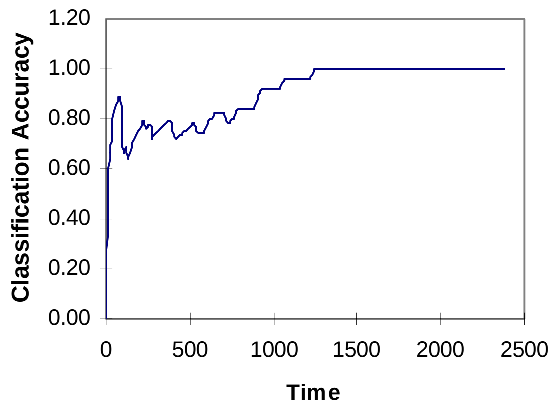

Fig.4
Evolution of classification accuracy in time. This figure corresponds to S/N ratio 5.

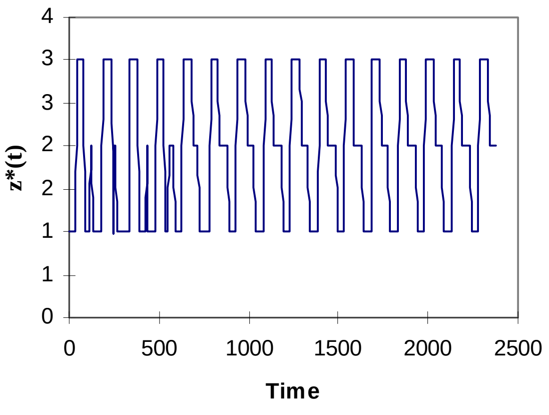

Fig.5
Evolution of z*(t) in time. This figure corresponds to S/N ratio 5.

It can be seen that in the initial stages of the training process, classification accuracy is low (Figs. 2, 4) and a relatively high number of misclassifications take place (Figs. 3, 5). However, after a certain point in time (approximately: t = 1000 for the noise-free case and t = 1400 for the noisy case) an appropriate 1-to-1 correspondence φ(.) is established between the sub-systems and some models; for instance in Fig.3 we have φ(1)=3, φ(2)=4, φ(3)=5. From that point on classification is highly accurate; this is also reflected in the classification accuracy diagrams.

Prediction Error: This is denoted by e(t), in other words is a function of time, computed over a sliding window of length equal to 50 time steps. More specifically, if the optimal prediction is denoted by y*(t), then at time t prediction error is given by

$$e(t) = \sqrt{\left[\Sigma_{\tau=0}^{49}|y(t-\tau) - y^*(t-\tau)|^2\right]}/50.$$

In Fig.6 we present *final* prediction error results. In other words, we plot e(2400) vs. S/N ratio. We see a steady increase of prediction error in relation to S/N ratio. One important point to keep in mind that the relatively large size of the prediction error is not due to an weakness of our data allocation method, but to the intrinsically low information content of the noise-contaminated data. In fact, given the nearly perfect classification results we have presented above, it becomes obvious that the prediction error obtained here is close to the theoretically optimum (minimum total square error).

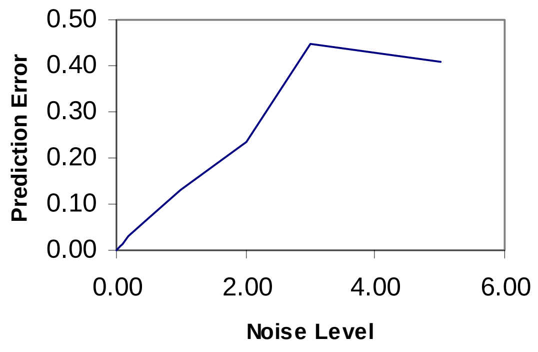

Fig.6
Final prediction error plotted against S/N ratio.

It is also instructive to observe the evolution in time of e(t). Profiles of this function are presented for two representative experiments: (a) at S/N = ∞ (noise-free case) in Fig. 7 and (b) at S/N= 5 in Fig.8. It can be seen that for the noise free case prediction error reduces to practically zero after t= 1300 or thereabout.

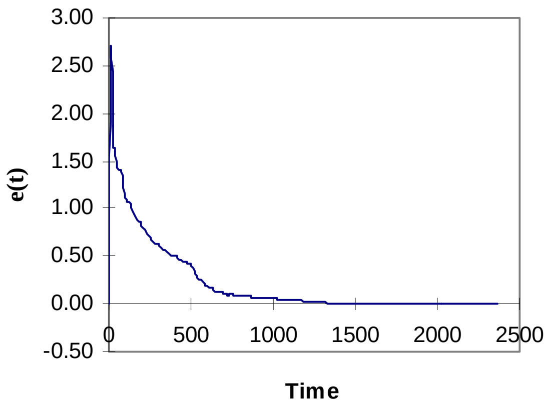

Fig.7
Evolution of prediction error in time. This figure corresponds to S/N = ∞.

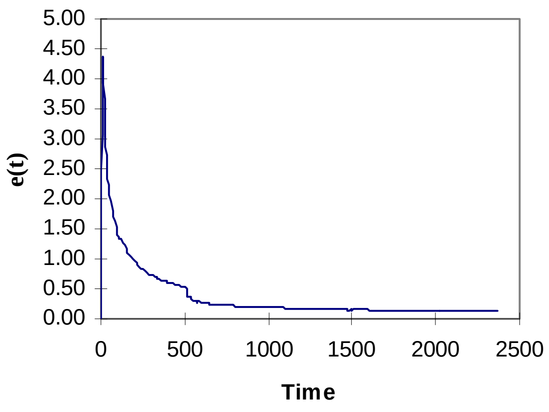

Fig.8
Evolution of prediction error in time. This figure corresponds to S/N = 5.

Parameter Estimation Relative Error: Finally, we present the relative error in parameter estimation, as computed at the *final* step of the data allocation process. This is denoted by q and is computed as follows.

$$q = \Sigma_{k=1,=1,j=1}^{3,3,3} | A_{i,j}(k)-A_{i,j}^{*}(\phi(k))| / | A_{i,j}(k)|+$$
$$\Sigma_{k=1,=1,j=1}^{3,3,2} | B_{i,j}(k)-B_{i,j}^{*}(\phi(k))| / | B_{i,j}(k)|$$

In other words q is computed by averaging relative error over all sub-systems and corresponding models (the appropriate correspondence is denoted by $\phi(k)$ -- recall that $\phi(k)$ is the function which maps the k-th sub-system to the $\phi(k)$-th model.), and over all components of the transition and input matrices.

In Fig.9 we plot q against S/N ratio. Once again we see that parameter estimation deteriorates rather rapidly as S/N ratio decreases – this is a weakness of the training data. However, note that with relatively clean data we obtain a practically perfect estimate of the parameters.

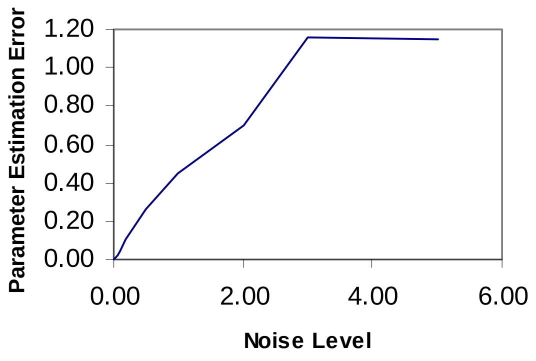

Fig.9
Parameter Estimation error plotted against S/N.

Experiment Group B: Multiplicative Noise
Experiment group B follows closely experiment group A. All algorithmic parameters are kept at the same values. The only difference is that we now use multiplicative observation noise. In other words, the sequence $v(t) = \alpha \cdot w(t) \cdot x(t)$, where w(t) is white noise, of zero mean and unit variance. The parameter α determines the “strength” of the noise and is related to the S/N ratio by the following formula

$$S/N = 10/ \alpha.$$

We have chosen α so that the experiment is repated at noise levels S/N= ∞ (noise free), 200, 100, 65, 50, 35, 20, 12.5, 10, 8, 6.5.

The results presented below are in complete correspondence to the ones of the previous section (additive noise experiments).

In Figs. 10 -- 18 we present various aspects of the data allocation performance. In particular, in Fig. 10 we present classification accuracy (c), in Fig.15 prediction error (e) and in Fig.18 parameter estimation error (q). In all these figures, the horizontal axis denotes signal-to-noise ration (S/N)

Classification Accuracy: In Fig.10 we present *final* classification accuracy results. In other words, we plot c(2400) vs. S/N ratio. It is worth noting that final classification accuracy is 1 for very high noise levels (for S/N ratio up to 10). Even when S/N ratio reaches the value 6.5, classification accuracy stays at 0.85.

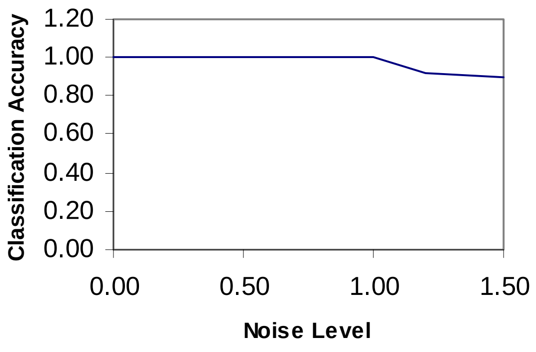

Fig.10
Final classification accuracy plotted against S/N ratio.

In Figs. 11 and 12 we present the temporal evolution of classification accuracy c(t) and data allocation z*(t) for a noise free experiment. The same quantities are presented in Figs.13 and 14 for an experiment with S/N = 10.

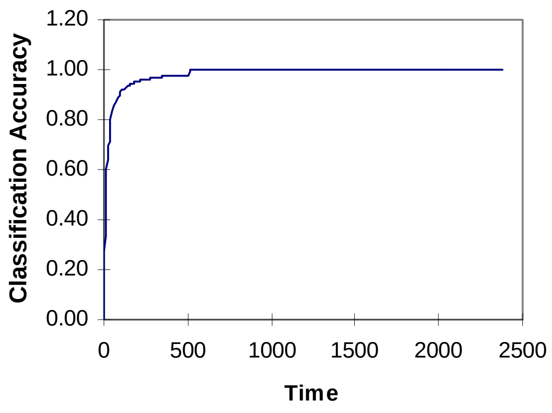

Fig.11
Evolution of classification accuracy in time. This figure corresponds to S/N ratio ∞.

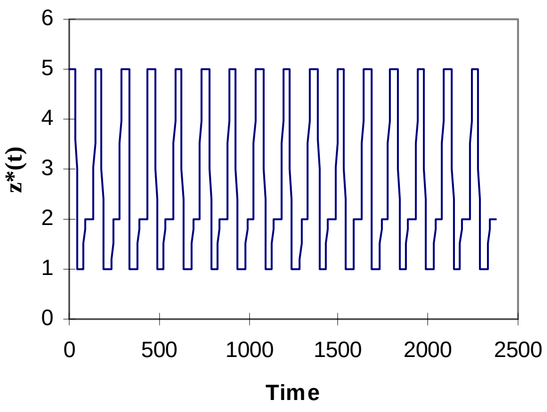

Fig.12
Evolution of z*(t) in time. This figure corresponds to S/N ratio ∞.

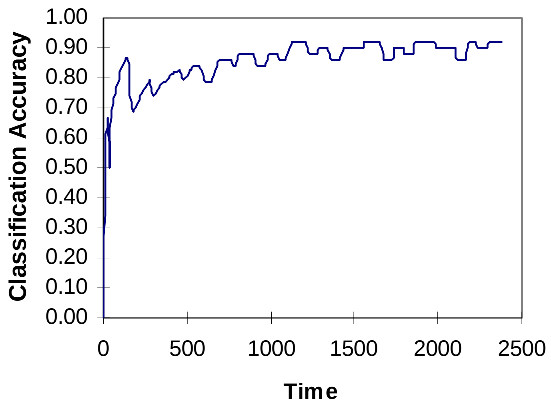

Fig.13
Evolution of classification accuracy in time. This figure corresponds to S/N = 10.

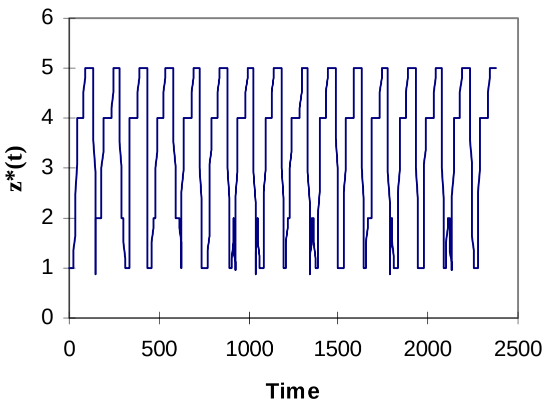

Fig.14
Evolution of z*(t) in time. This figure corresponds to S/N = 10.

Prediction Error: In Fig.15 we present *final* prediction error results. In other words, we plot e(2400) vs. S/N ratio.

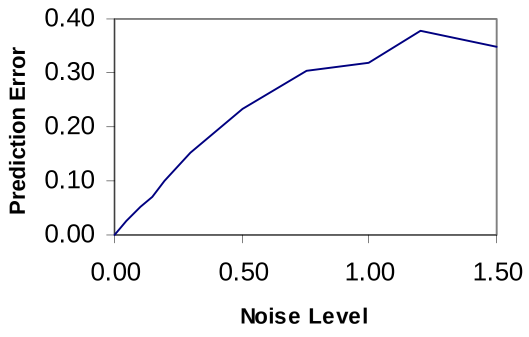

Fig.15
Final prediction error plotted against S/N ratio.

It is also instructive to observe the evolution in time of e(t). Profiles of this function are presented for two representative experiments: (a) at S/N = ∞ (noise-free case) in Fig. 16 and (b) at S/N= 10 in Fig.17.

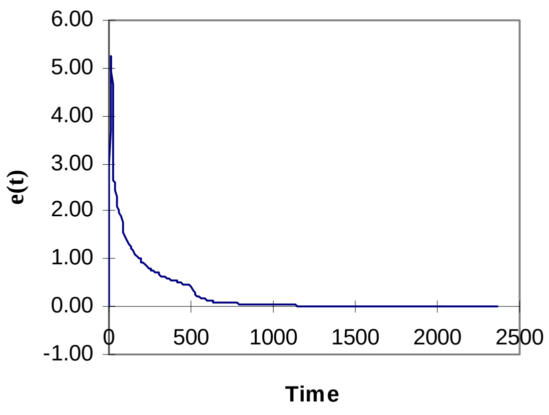

Fig.16
Evolution of prediction error in time. This figure corresponds to S/N = ∞.

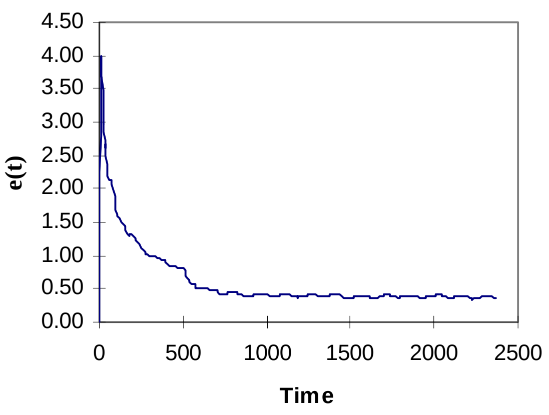

Fig.17
Evolution of prediction error in time. This figure corresponds to S/N = 10.

Parameter Estimation Relative Error: Finally, in Fig.18 we present the relative error in parameter estimation, as computed at the *final* step of the data allocation process.

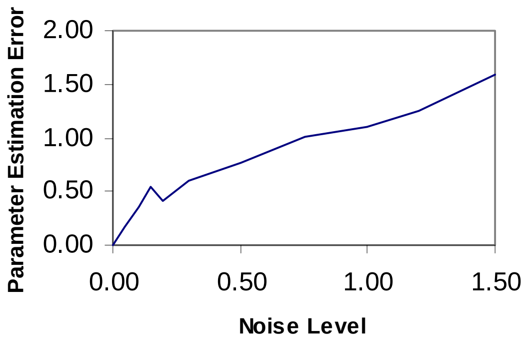

Fig.18
Parameter Estimation error plotted against noise level.

5. CONCLUSION

We have presented an algorithm which can be used to develop models of the sub-systems comprising a switched system. Our algorithm operates online and is appropriate for *unsupervised* problems, where no initial models or labeled training data are available. The algorithm provides accurate allocation of observation data to several training data pools, one data pool corresponding to each sub-system. Hence the allocated data can be utilized to train one model per sub-system and provide well-trained models to be used as components of a Lainiotis partition algorithm. The algorithm we propose is highly robust to observation noise (as evidenced by numerical experiments) and there is strong theoretical evidence to justify its very good performance.

6. REFERENCES

[1] Ath. Kehagias , “Convergence properties of theLainiotis partition algorithm”, Control and Computers}, vol.19, 1991, pp.1-6.

[2] Ath. Kehagias and V. Petridis, “Predictive modular neural networks for time series classification”, Neural Networks, 1996.

[3] D.G. Lainiotis, S.K Katsikas and S.D. Likothanasis, “Adaptive deconvolution of seismic signals: performance, computational analysis, parallelism”, IEEE Trans. Acoustics, Speech and Signal Processing, Vol.36, 1988, pp. 1715-1734.

[4] D.G. Lainiotis, ``Optimal adaptive estimation: structure and parameter adaptation", \textit{IEEE Trans. on Automatic Control}, vol.16, pp. 160-170, April 1971.

[5] V. Petridis, “A Method for Bearings-Only Velocity and Position Estimation”, IEEE Trans. on Automatic Control, Vol. 26, 1981, pp. 488-493.

[6] V. Petridis and Ath. Kehagias, “Recurrent neural networks for parameter estimation”, Proc. of EANN 1996, 1996.

[7] V. Petridis and Ath. Kehagias, “Modular neural networks for Bayesian classification of time series and the partition algorithm”, IEEE Trans. on Neural Networks, 1996, vol.7, pp.73-86.

[8] V. Petridis and Ath. Kehagias, “A recurrent network implementation of time series classification”, Neural Computation, 1996, vol.8, pp.357-372.

[9] V. Petridis and Ath. Kehagias. *Predictive Modular Neural Networks: Applications to Time Series*. Kluwer Academic Publishers, 1998.

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