# Flexible Architecture of Self Organizing Maps for Changing Environments*

Rodrigo Salas[1,2], Héctor Allende[2,3],
Sebastián Moreno[2], and Carolina Saavedra[2]

[1] Universidad de Valparaíso; Departamento de Computación; Valparaíso-Chile
{rodrigo.salas}@uv.cl
[2] Universidad Técnica Federico Santa María; Dept. de Informática,
Casilla 110-V; Valparaíso-Chile
{hallende, smoreno, saavedra}@inf.utfsm.cl
[3] Universidad Adolfo Ibañez; Facultad de Ciencia y Tecnología

**Abstract.** Catastrophic Interference is a well known problem of Artificial Neural Networks (ANN) learning algorithms where the ANN forget useful knowledge while learning from new data. Furthermore the structure of most neural models must be chosen in advance.

In this paper we introduce a hybrid algorithm called Flexible Architecture of Self Organizing Maps (*FASOM*) that overcomes the Catastrophic Interference and preserves the topology of Clustered data in changing environments. The model consists in $K$ receptive fields of self organizing maps. Each Receptive Field projects high-dimensional data of the input space onto a neuron position in a low-dimensional output space grid by dynamically adapting its structure to a specific region of the input space.

Furthermore the *FASOM* model automatically finds the number of maps and prototypes needed to successfully adapt to the data. The model has the capability of both growing its structure when novel clusters appears and gradually forgets when the data volume is reduced in its receptive fields.

Finally we show the capabilities of our model with experimental results using synthetic sequential data sets and real world data.

**Keywords:** Catastrophic Interference, Artificial Neural Networks, Self Organizing Maps, Pattern Recognition.

## 1 Introduction

During this decade a huge amount of real data with highly dimensional samples have been stored for some sufficiently large period of time. Models were constructed to learn this data, but due to the changing nature of the input space, the neural networks catastrophically forgets the previously learned patterns [4].

* This work was supported in part by Research Grant Fondecyt 1040365 and 7050205, DGIP-UTFSM, BMBF-CHL 03-Z13 from German Ministry of Education, DIPUV-22/2004 and CID-04/2003.

M. Lazo and A. Sanfeliu (Eds.): CIARP 2005, LNCS 3773, pp. 642–653, 2005.

In addition, the neural designer has the difficulty to decide in advance the architecture of the model, and if the environment change, the neural network will not obtain a good performance under this new situation. To overcome the architectural design problem several algorithms with adaptive structure have been proposed (See [2], [5], [12] and [13]).

In this paper we propose a hybrid problem-dependent model based on the Kohonen's Self Organizing Maps [8] with the Bauer et al. growing variant of the *SOM* [2], the $K$-means [11], the Single Linkage clustering algorithm [7] and the addition of new capabilities of gradually forgetting and contracting the net. We call this algorithm Flexible Architecture of Self Organizing Maps (*FASOM*). The *FASOM* is a hybrid model that adapts $K$ receptive fields of dynamical self organizing maps and learn the topology of partitioned spaces [13]. It has the capability of detecting novel data or clusters and creates new maps to learn this patterns avoiding that other receptive fields catastrophically forget. Furthermore the receptive fields with decreasing volume of data can gradually forget by reducing their size and contracting their grid lattice.

The remainder or this paper is organized as follows. The next section we briefly discuss the ANN Catastrophic Interference problem. Then we review the models where our hybrid model is based. In the fourth section, our proposal of the *FASOM* model is stated. Simulation results on synthetic and real data sets are provided in the fifth section. Conclusions and further work are given in the last section.

## 2 The ANN Catastrophic Interference Problem

Artificial neural networks with highly distributed memory forget catastrophically when faced with sequential learning tasks, i.e., the new learned information most often erases the one previously learned. This major weakness is not only cognitively implausible, as human gradually forget, but disastrous for most practical applications. (See [4] and [10] for a review)

Catastrophic interference is a radical manifestation of a more general problem for connectionist models of memory, the so-called *stability-plasticity* problem [6]. The problem is how to design a system that is simultaneously sensitive to, but not radically disrupted by, new input. A number of ways have been proposed to avoid the problem of catastrophic interference in connectionist networks (see [1], [4]).

## 3 Review of Unsupervised Clustering and Topology Preserving Algorithms

### 3.1 Unsupervised Clustering

Clustering can be considered as one of the most important unsupervised learning problem. A cluster is a collection of "similar" objects and they should be

"dissimilar" to the objects belonging to other clusters. Unsupervised clustering tries to discover the natural groups inside a data set.

The purpose of any clustering technique [9] is to evolve a $K \times N$ partition matrix $U(\mathcal{X})$ of the data set $\mathcal{X} = \{\underline{x}_1, ..., \underline{x}_N\}$, $\underline{x}_j \in \mathbb{R}^n$, representing its partitioning into a number, say $K$, of clusters $\mathcal{C}_1, ..., \mathcal{C}_K$. Each element $u_{kj}$, $k = 1..K$ and $j = 1..n$ of the matrix $U(\mathcal{X})$ indicates the membership of pattern $\underline{x}_j$ to the cluster $\mathcal{C}_k$. In crisp partitioning of the data, the following condition holds: $u_{kj} = 1$ if $\underline{x}_j \in \mathcal{C}_k$; otherwise, $u_{kj} = 0$.

There are several clustering techniques classified as partitional and hierarchical [7]. In this paper we based our model in the following algorithms.

**K-means**

The $K$-means method introduced by McQueen [11] is one of the most widely applied partitional clustering technique. This method basically consists on the following steps. First, $K$ randomly chosen points from the data are selected as seed points for the centroids $\overline{z}_k$, $k = 1..K$, of the clusters. Second, assign each data to the cluster with the nearest centroid based on some distance criterion, for example, $\underline{x}_j$ belongs to the cluster $\mathcal{C}_k$ if the distance $d(\underline{x}_j, \overline{z}_k) = \left\|\underline{x}_j - \overline{z}_k\right\|$ is the minimum for $k = 1..K$. Third, the centroids of the clusters are updated to the "center" of the points belonging to them, for example, $\overline{z}_k = \frac{1}{N_k} \sum_{\underline{x}_j \in \mathcal{C}_k} \underline{x}_j$, where $N_k$ is the number of data belonging to the cluster $k$. Finally, repeat the procedure until either the clusters centroids do not change or some optimal criterion is met.

This algorithm is iteratively repeated for $K = 1, 2, 3, ...$ until some validity measure indicates that partition $U_{K_{opt}}$ is a better partition than $U_K$, $K < K_{opt}$ (see [9] for some validity indices). In this work we used the $F$-test to specified the number $K$ of clusters. The $F$-test measures the variability reduction by comparing the sum of square distance of the data to their centroids $E_K = \sum_{j=1}^{N} \sum_{k=1}^{K} u_{kj} \left\|\underline{x}_j - \overline{z}_k\right\|^2$ of $K$ and $K+1$ groups. The test statistic is $F = \frac{E_K - E_{K+1}}{E_{K+1}/(n-K-1)}$ and is compared with the $F$ statistical distribution with $p$ and $p(n-K-1)$ degrees of freedom.

**Single Linkage**

The Single Linkage clustering scheme, also known as the nearest neighbor method, is usually regarded as a graph theoretical model [7]. It stars by considering each point as cluster of its own. The single linkage algorithm computes the distance between two clusters $\mathcal{C}_k$ and $\mathcal{C}_l$ as $\delta_{SL}(\mathcal{C}_k, \mathcal{C}_l) = \min_{\underline{x} \in \mathcal{C}_k, \underline{y} \in \mathcal{C}_l} \{d(\underline{x}, \underline{y})\}$. If the distance between both clusters is less than some threshold $\theta$ then they are merged into one cluster. The process continues until the distance between all the clusters are greater than the threshold $\theta$.

This algorithm is very sensitive to the determination of the parameter $\theta$, for this reason we compute its value proportional to the average distance between the points belonging to the same clusters, i.e., $\theta(\mathcal{X}) \propto \frac{1}{N_k} \sum_{\underline{x}_i, \underline{x}_j \in \mathcal{C}_k} d(\underline{x}_i, \underline{x}_j)$. At the beginning $\theta$ can be set as a fraction (bigger than one) of the minimum

distance of the two closest points. When the algorithm is done, clusters consisting of less than $l$ data are merged to the nearest cluster.

### 3.2 Topology Preserving Neural Models: The Self Organizing Maps

It is interesting to explore the topological structure of the clusters. The Kohonen's Self Organizing Map [8] and their variants are useful for this task. The self-organizing maps (*SOM*) neural model is an iterative procedure capable of representing the topological structure of the input space (discrete or continuous) by a discrete set of prototypes (*weight vectors*) which are associated to neurons of the network.

The map is generated by establishing a correspondence between the input signals $\underline{x} \in \mathcal{X} \subseteq \mathbb{R}^n$, $\underline{x} = [x_1, ..., x_n]^T$, and neurons located on a discrete lattice. The correspondence is obtained by a competitive learning algorithm consisting on a sequence of training steps that iteratively modifies the weight vector $\underline{m}_k \in \mathbb{R}^n$, $\underline{m}_k = (m_1^k, ..., m_n^k)$, where $k$ is the location of the prototype in the lattice.

When a new signal $\underline{x}$ arrives every neuron competes to represent it. The best matching unit (*bmu*) is the neuron that wins the competition and with its neighbors on the lattice they are allowed to learn the signal. The *bmu* is the reference vector $c$ that is nearest to the input $\underline{x}$, i.e., $c = \arg\min_i\{\|\underline{x} - \underline{m}_i\|\}$.

During the learning process the reference vectors are changed iteratively according to the following adjusting rule,

$$\underline{m}_j(t+1) = \underline{m}_j(t) + \alpha(t)h_c(j,t)[\underline{x} - \underline{m}_j(t)] \qquad j = 1..M$$

where $M$ is the number of prototypes that must be adjusted. The learning parameter $\alpha(t) \in [0,1]$ is a monotonically decreasing real valued sequence. The amount that the units learnt will be governed by a neighborhood kernel $h_c(j,t)$, that is a decreasing function of the distance between the unit $j$ and the *bmu* $c$ on the map lattice at time $t$. The neighborhood kernel is usually given by a Gaussian function:

$$h_c(j,t) = \exp\left(\frac{-\left\|\underline{r}_j - \underline{r}_c\right\|^2}{\sigma(t)^2}\right) \tag{1}$$

where $\underline{r}_j$ and $\underline{r}_c$ denote the coordinates of the neurons $j$ and $c$ in the lattice. In practice the neighborhood kernel controlled by the parameter $\sigma(t)$ and is chosen wide enough in the beginning of the learning process to guarantee global ordering of the map, and both its width and height decrease slowly during the learning process. More details and properties of the *SOM* can be found in [3] and [8].

## 4 The Flexible Architecture of Self Organizing Maps

The *FASOM* is a hybrid model that adapts $K$ receptive fields of dynamical self organizing maps and learn the topology of partitioned spaces. It has the capability of detecting novel data or clusters and creates new maps to learn this patterns

avoiding that other receptive fields catastrophically forget. Furthermore the receptive fields with decreasing volume of data can gradually forget by reducing their size and contracting their grid lattice.

The learning process has two parts. The first part occurs when the model is created an learn the data for the first time. The second part of the learning process occurs when a new pattern is presented to the trained model. The description of the algorithm follows.

### 4.1 First Part: Topological Learning Algorithm

**First Step: Clustering the data**
The purpose of this step is to find the number of clusters presented in the data. To this purpose, first we execute the $K$-means algorithm, presented in section 3.1, with a very low threshold in order to find more clusters than they really are. Then, the Single Linkage algorithm is executed to merge cluster that are closer and to obtain, hopefully, the optimal number of clusters.

When the number of clusters $K$ and their respective centroids $\overline{z}_k$, $k = 1..K$, are obtained then we proceed to create a grid of size $2 \times 2$ for each cluster.

**Second Step: Topological Learning**
During the learning process when an input data $\underline{x}$ is presented to the model at time $t$ the best matching map ($bmm$) is found as follows. Let $\mathcal{M}_k$ be the set of prototypes that belong to the map $\mathcal{C}_k$. The best matching units ($bmu$) $\underline{m}_{c_k}^{[k]}$, $k = 1..K$, of the sample $\underline{x}$ for each of the $K$ maps are detected. The map that contains the closest $bmu$ to the data will be the $bmm$ whose index is given by

$$\eta = \arg \min_{k=1..K} \{ \left\| \underline{x} - \underline{m}_{c_k}^{[k]} \right\|, \underline{m}_{c_k}^{[k]} \in \mathcal{M}_k \} \tag{2}$$

Then all the units that belong to the $bmm$ will be updated iteratively according to the following rule:

$$\underline{m}_j^{[\eta]}(t+1) = \underline{m}_j^{[\eta]}(t) + \alpha(t) h_{c_\eta}^{[\eta]}(j,t)[\underline{x} - \underline{m}_j^{[\eta]}(t)] \qquad j = 1..M_\eta$$

where the neighborhood kernel $h_{c_\eta}^{[\eta]}(j,t)$ is given by equation (1) and the learning parameter $\alpha(t)$ is a monotonically decreasing function through time. For example this functions could be linear $\alpha(t) = \alpha_0 + (\alpha_f - \alpha_0)t/t_\alpha$ or exponential $\alpha(t) = \alpha_0(\alpha_f/\alpha_0)^{t/t_\alpha}$, where $\alpha_0$ and $\alpha_f$ are the initial and final learning rate respectively, and $t_\alpha$ is the maximum number of iteration steps to arrive $\alpha_f$

**Third Step: Growing the Maps Lattices**
In this part we introduce the variant proposed by Bauer et. al. of growing the SOM [2]. If the topological representations of the input space partitions are not good enough, the maps will grow by increasing the number of their prototypes. The quality of the topological representation of each map $\mathcal{C}_k$ of the *FASOM* model is measured in terms of the deviation between the units. At the beginning

we compute the quantization error $qe_0^{[k]}$ over the whole data belonging to the cluster $\mathcal{C}_k$.

All units must represent their respective Voronoi polygons of data at a quantization error smaller than a fraction $\tau$ of $qe_0^{[k]}$, i.e., $qe_j^{[k]} < \tau \cdot qe_0^{[k]}$, where $qe_j^{[k]} = \sum_{\underline{x}_i \in \mathcal{C}_j^{[k]}} \left\| \underline{x}_i - \underline{m}_j^{[k]} \right\|$, and $\mathcal{C}_j^{[k]} \neq \phi$ are the set of input vectors belonging to the Voronoi polygon of the unit $j$ in the map lattice $\mathcal{C}_k$. The units that not satisfy this criterion require a more detailed data representation.

When the map lattice $\mathcal{C}_k$ is chosen to grow, we compute the value of $qe_j^{[k]}$ for all the units belonging to the map. The unit with the highest $qe_j^{[k]}$, called error unit $e$ , and its most dissimilar neighbor $d$ are detected. To accomplish this the value of $e$ and $d$ are computed by $e = \arg\max_j \left\{ \sum_{\underline{x}_i \in \mathcal{C}_j^{[k]}} \left\| \underline{x}_i - \underline{m}_j^{[k]} \right\| \right\}$ and $d = \arg\max_j \left( \left\| \underline{m}_e^{[k]} - \underline{m}_j^{[k]} \right\| \right)$ respectively, where $\mathcal{C}_j^{[k]} \neq \phi, \underline{m}_j^{[k]} \in \mathcal{N}_e$ and $\mathcal{N}_e$ is the set of neighboring units of the error unit $e$. A row or column of units is inserted between $e$ and $d$ and their model vector are initialized as the means of their respective neighbors. After insertions, the map is trained again by executing the second step.

### 4.2 Second Part: Adapting to Changing Environments

The behavior of the input space could change through time, for example, clusters of data could be created, moved, and even vanished. The model should be able to adjust its architecture to the new environment.

Let $FASOM_T$ be the model and $\mathcal{X}_T$ the training dataset, both considered at the training stage $T$. The adaptation is done as follows.

**First step: Detection of strikingly new data**
The samples that do not have a good topological representation with the actual model $FASOM_{T-1}$ are identified by computing the influence of the sample $\underline{x}$ to the model $FASOM_{T-1}$ as $\rho(\underline{x}, FASOM_{T-1}) = \left\| \underline{x} - \underline{m}_{c_\eta}^{[\eta]}(t) \right\|$, where $\underline{m}_{c_\eta}^{[\eta]}(t)$ is the *bmu* of the *bmm* $\eta$ to the data $\underline{x}$. Let $\mathcal{X}_T^{[new]}$ be the set of all the strikingly new data whose influence function are bigger than some threshold $\theta$, i.e., $\rho(\underline{x}, FASOM_{T-1}) > \theta$.

**Second Step: Creating Maps for the new data**
A new model $FASOM_T^{[new]}$ based on the samples $\mathcal{X}_T^{[new]}$ extracted from the previous step is created. This is accomplished by applying the first step of the previous part (Clustering the data).

**Third step: Integration of the maps**
Both models are integrated in an unique updated version, i.e.,

$$FASOM_T = FASOM_{T-1} \bigcup FASOM_T^{[new]}$$

**Fourth step: Learning the samples**
The model $FASOM_T$ learns the whole dataset $\mathcal{X}_T$. This is accomplished by applying second step of the previous part.

**Fifth step: Gradually forgetting the old data**
Due to the environmental of the input space is not stationary, the clusters behavior change through time, and, for example, either their variance or their data volume can be reduced, or even more, the clusters could be vanished. For this reason, and to keep the complexity of the model rather low we let the maps gradually forget the clusters by shrinking their lattices towards their respective centroids and reducing the number of their prototypes.

Centroid neurons $\overline{m}^{[k]}$ representing the cluster $k$ modelled by the map $\mathcal{C}_k$ are computed as the mean value of the prototypes belonging to the map lattice, i.e., $\overline{m}^{[k]} = \frac{1}{M_k}\sum_{j=1}^{M_k} m_j^{[k]}$, where $m_j^{[k]}$ is the $j-$th prototype of the grid $\mathcal{C}_k$ and $M_k$ is the number of neurons of that grid.

The map $\kappa$ will forget the old data by applying once the forgetting rule:

$$\underline{m}_j^{[\kappa]}(T+1) = \underline{m}_j^{[\kappa]}(T) + \gamma[\underline{x} - \overline{m}^{[\kappa]}] \qquad j = 1..M_\kappa$$

where $\gamma$ is the forgetting rate. $\underline{m}_j^{[\kappa]}(T)$ and $\underline{m}_j^{[\kappa]}(T+1)$ are the values of the prototypes at the end of the stage $T$ and the beginning of stage $T+1$ respectively. The objective is to shrink the maps toward their centroids neurons.

Then, if the units of the map $\kappa$ are very close, the map is contracted. If the map lattice is a rectangular grid, then we search two rows or columns whose neurons are very close. To accomplish this the value $\nu$ is computed by

$$\nu = \underset{e=1..N_r-1;d=1..N_c-1}{\arg\min} \left( \frac{1}{N_c}\sum_{j=1}^{N_c} \left\| \underline{m}_{(e,j)}^{[\kappa]} - \underline{m}_{(e+1,j)}^{[\kappa]} \right\|, \frac{1}{N_r}\sum_{i=1}^{N_r} \left\| \underline{m}_{(i,d)}^{[\kappa]} - \underline{m}_{(i,d+1)}^{[\kappa]} \right\| \right)$$

where $\underline{m}_{(i,j)}^{[\kappa]}$ is the unit located at the position $(i,j)$ in the map lattice $\kappa$. $N_r$ and $N_c$ are the number of rows and columns of the map lattice respectively. If the criterion $\nu < \beta$ is met then a row (or a column) of units are inserted between $\nu$ and $\nu+1$ and their model vector are initialized as the mean of their respective neighbors. Then the rows (or columns) $\nu$ and $\nu+1$ of prototypes are both eliminated. The map is contracted iteratively until no other row or column satisfies the criterion. After contraction, the map is trained again by executing the fourth step.

### 4.3 Clustering the Data and Evaluation of the Model

To classify the data $\underline{x}_j$ to one of the cluster $k = 1..K$, we find the best matching map given by equation (2) and the data will receive the label of this map $\eta$, i.e., for the data $\underline{x}_j$ we set $u_{j\eta} = 1$ and $u_{jk} = 0$ for $k \neq \eta$.

To evaluate the clustering performance we compute the percentage of right classification given by:

$$PC = \frac{1}{N} \sum_{\underline{x}_j, j=1..N} u_{jk} \qquad k = \text{True class of } \underline{x}_j \tag{3}$$

**Evaluation of the adaptation quality**
To evaluate the quality of the partitions topological representation, a common measure to compare the algorithms is needed. The following metric called the mean square quantization error is used:

$$MSQE = \frac{1}{N} \sum_{\mathcal{M}_k, k=1..K} \sum_{\underline{m}_j^{[k]} \in \mathcal{M}_k} \sum_{\underline{x}_i \in \mathcal{C}_j^{[k]}} \left\| \underline{x}_i - \underline{m}_j^{[k]} \right\|^2 \tag{4}$$

## 5 Simulation Results

To validate the *FASOM* model we apply first the algorithm to computer generated data and then to *El Niño* real data. The models used to compare the results were the K-Dynamical Self Organizing Map *KDSOM* [13] and our model proposal *FASOM* and $FASOM_\gamma$ where the last gradually forgets.

To execute the simulations and to compute the metrics, all the dimensions of the training data were scaled to the unit interval. The test sets were scaled using the same scale applied to the training data (Notice that with this scaling the test data will not necessarily fall in the unit interval).

### 5.1 Experiment #1: Computer Generated Data

For the synthetic experiment we create gaussian clusters of two-dimensional distribution $\underline{X}_k \sim \mathcal{N}(\mu_k, \Sigma_k)$, $k = 1, ..., K$, where $K$ is the number of clusters, and, $\mu_k$ and $\Sigma_k$ are the mean vector and the covariance matrix respectively of the cluster $\mathcal{C}_k$. The behavior of the clusters change through the several training stages.

The experiment has 5 stages where in the first four we re-train the model. The information about the clusters in the several stages are given in table 1. As can be noted seven clusters were created. The cluster 1 vanishes in the second stage, the cluster 2 decreases the number of data, cluster 3 appears in the second stage, cluster 4 moves from $(0.1, 0.8)$ to $(0.42, 0.56)$, cluster 5 increases its variance and finally the clusters 6 and 7 appear in the third and fourth stage respectively.

The summary of the results is given in table 2. To evaluate the ability of the net to remember past learned information, the model was tested with data of the previous stages, p.e., the model trained after stage four was tested with the data of stage three, two and one. As can be noted, the *FASOM*'s models outperform the *KDSOM*. The $FASOM_\gamma$ shows lower MSQE with less number of prototypes.

**Table 1.** Summary of the clusters generated for the Synthetic Experiment in the several training stages

| Cluster | 1 | 2 | 3 | 4 | 5 | 6 | 7 | Total |
|---|---|---|---|---|---|---|---|---|
| $N_{trainT1}$ | 250 | 250 | 0 | 250 | 250 | 0 | 0 | 1000 |
| $N_{testT1}$ | 250 | 250 | 0 | 250 | 250 | 0 | 0 | 1000 |
| $\mu_{T1}$ | $[0.9, 0.01]^T$ | $[0.8, 0.5]^T$ | $[0.6, 0.8]^T$ | $[0.1, 0.8]^T$ | $[0.2, 0.1]^T$ | $[0.5, 0.5]^T$ | $[0.01, 0.5]^T$ | |
| $\Sigma_{T1}$ | $0.05^2 * I_2$ | $0.05^2 * I_2$ | $0.05^2 * I_2$ | $0.05^2 * I_2$ | $0.05^2 * I_2$ | $0.05^2 * I_2$ | $0.05^2 * I_2$ | |
| $N_{trainT2}$ | 0 | 250 | 250 | 250 | 250 | 0 | 0 | 1000 |
| $N_{testT2}$ | 0 | 250 | 250 | 250 | 250 | 0 | 0 | 1000 |
| $\mu_{T2}$ | $[0.9, 0.01]^T$ | $[0.8, 0.5]^T$ | $[0.6, 0.8]^T$ | $[0.18, 0.74]^T$ | $[0.2, 0.1]^T$ | $[0.5, 0.5]^T$ | $[0.01, 0.5]^T$ | |
| $\Sigma_{T2}$ | $0.05^2 * I_2$ | $0.05^2 * I_2$ | $0.05^2 * I_2$ | $0.05^2 * I_2$ | $0.054^2 * I_2$ | $0.05^2 * I_2$ | $0.05^2 * I_2$ | |
| $N_{trainT3}$ | 0 | 188 | 250 | 250 | 250 | 750 | 0 | 1688 |
| $N_{testT3}$ | 0 | 188 | 250 | 250 | 250 | 750 | 0 | 1688 |
| $\mu_{T3}$ | $[0.9, 0.01]^T$ | $[0.8, 0.5]^T$ | $[0.6, 0.8]^T$ | $[0.26, 0.68]^T$ | $[0.2, 0.1]^T$ | $[0.5, 0.5]^T$ | $[0.01, 0.5]^T$ | |
| $\Sigma_{T3}$ | $0.05^2 * I_2$ | $0.05^2 * I_2$ | $0.05^2 * I_2$ | $0.05^2 * I_2$ | $0.0583^2 * I_2$ | $0.05^2 * I_2$ | $0.05^2 * I_2$ | |
| $N_{trainT4}$ | 0 | 125 | 250 | 250 | 250 | 750 | 250 | 1875 |
| $N_{testT4}$ | 0 | 125 | 250 | 250 | 250 | 750 | 250 | 1875 |
| $\mu_{T4}$ | $[0.9, 0.01]^T$ | $[0.8, 0.5]^T$ | $[0.6, 0.8]^T$ | $[0.34, 0.62]^T$ | $[0.2, 0.1]^T$ | $[0.5, 0.5]^T$ | $[0.01, 0.5]^T$ | |
| $\Sigma_{T4}$ | $0.05^2 * I_2$ | $0.05^2 * I_2$ | $0.05^2 * I_2$ | $0.05^2 * I_2$ | $0.063^2 * I_2$ | $0.05^2 * I_2$ | $0.05^2 * I_2$ | |
| $N_{trainT5}$ | 25 | 62 | 250 | 250 | 250 | 750 | 250 | 1837 |
| $N_{testT5}$ | 25 | 62 | 250 | 250 | 250 | 750 | 250 | 1837 |
| $\mu_{T5}$ | $[0.9, 0.01]^T$ | $[0.8, 0.5]^T$ | $[0.6, 0.8]^T$ | $[0.42, 0.56]^T$ | $[0.2, 0.1]^T$ | $[0.5, 0.5]^T$ | $[0.01, 0.5]^T$ | |
| $\Sigma_{T5}$ | $0.05^2 * I_2$ | $0.05^2 * I_2$ | $0.05^2 * I_2$ | $0.05^2 * I_2$ | $0.068^2 * I_2$ | $0.05^2 * I_2$ | $0.05^2 * I_2$ | |

In figure 1 each row of the graphs array corresponds to one model, the first is the *KDSOM*, the second is the $FASOM_\gamma$ and the last is the *FASOM*. Each column of graphs array corresponds to a different training stage. In the figure is easy to note how the *KDSOM* model catastrophically forgets different cluster and tries to model two or three different cluster with the same grid. Instead, the *FASOM* and $FASOM_\gamma$ models learn new cluster in different stages, furthermore, the $FASOM_\gamma$ forgets the cluster with no data as cluster 1, as a consequence the model has a lower complexity.

In figure 2, the models obtained after the fifth stage were tested with data of all previous stages. The graphs show how the models forget previously learned patterns, the *KDSOM* was the most affected model while the *FASOM*'s models obtain comparable results.

### 5.2 Experiment #2: Real Datasets

In the real dataset experiment we test the algorithm with the *El Niño Data*. The data can be obtained from *http://kdd.ics.uci.edu/databases/el_nino/el_nino.html*. The *El Niño Data* is expected to aid in the understanding and prediction of El Niño Southern Oscillation (ENSO) cycles and was collected by the Pacific Marine Environmental Laboratory National Oceanic and Atmospheric Administration. The data set contains oceanographic and surface meteorological readings taken from a several buoys positioned throughout the equatorial Pacific.

The data consists in the following variables: date, latitude, longitude, zonal winds ($west < 0$, $east > 0$), meridional winds ($south < 0$, $north > 0$), relative humidity, air temperature, sea surface temperature and subsurface temperatures down to a depth of 500 meters. Data taken from the buoys are as early as 1980 for some locations.

The data set was modified by discarding those data with missing values. Finally we obtains 130454 instances of 4 dimensions (meridional winds, relative

**Table 2.** Summary of the results obtained for the synthetic experiments. The first column **S** indicates the training stage of the experiment, **M** is the neural model where **KD** is the KDSOM, $\mathbf{FS}_\gamma$ and **FS** are the FASOM with and without ($\gamma = 0$) forgetting factor respectively. The column **PC** shows the percentage of the correct classification and **MSQE Test** $X$ is the MSQE error with test data of the stage $X$ but using the trained model of the current stage.

| **S** | **M** | **Neurons** | **Grids** | *MSQE* **Train** | *PC* **Test** | *MSQE* **Test1** | *MSQE* **Test2** | *MSQE* **Test3** | *MSQE* **Test4** | *MSQE* **Test5** |
|---|---|---|---|---|---|---|---|---|---|---|
| S1 | KD | 48 | 4.0 | 1.87 | 100 | 2.02 | – | – | – | – |
| | $FS_\gamma$ | 47 | 4.0 | **1.51** | 100 | **1.62** | – | – | – | – |
| | FS | 48 | 4.0 | 1.83 | 100 | 1.96 | – | – | – | – |
| S2 | KD | 49 | 4.0 | 2.49 | 75.70 | 4.13 | 2.70 | – | – | – |
| | $FS_\gamma$ | 69 | 5.9 | **1.62** | 100 | **3.32** | **1.77** | – | – | – |
| | FS | 69 | 5.8 | 1.95 | 100 | 3.62 | 2.11 | – | – | – |
| S3 | KD | 49 | 4.0 | 5.88 | 73.34 | 14.16 | 6.74 | 6.31 | – | – |
| | $FS_\gamma$ | 78 | 7.1 | **2.39** | **95.92** | **10.82** | **3.44** | **2.65** | – | – |
| | FS | 87 | 7.6 | 2.81 | 95.66 | 11.80 | 4.03 | 3.05 | – | – |
| S4 | KD | 49 | 4.0 | 7.13 | 66.01 | 24.54 | 12.99 | 8.69 | 7.32 | 7.89 |
| | $FS_\gamma$ | 91 | 8.3 | **2.48** | **95.67** | **18.92** | **8.78** | **4.10** | **2.85** | **3.36** |
| | FS | 101 | 8.8 | 2.72 | 95.52 | 19.58 | 9.72 | 4.55 | 3.09 | 3.56 |

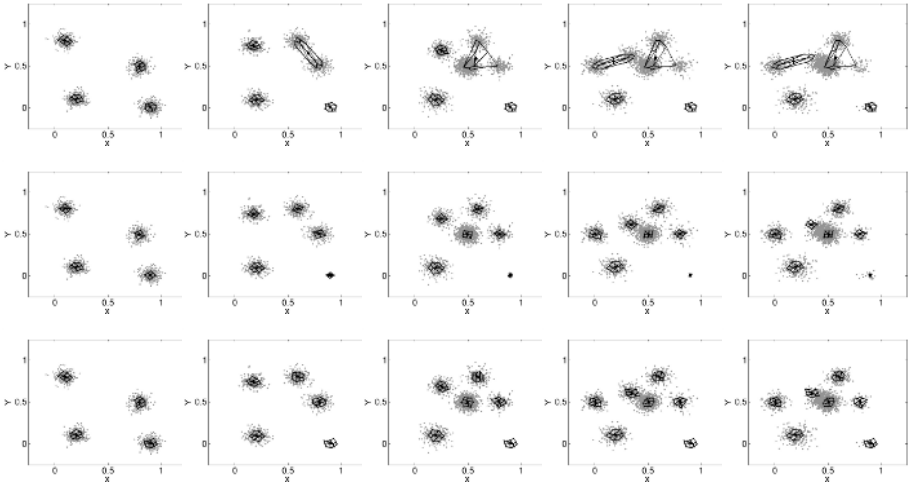

**Fig. 1.** Synthetic data results: Each column correspond to the training stage. The first row is the KDSOM model, the second is the $\text{FASOM}_\gamma$ with forgetting factor and the last is the FASOM without forgetting.

humidity, sea surface temperature and subsurface temperatures). We divided the dataset according to the years into 19 training sets and one test set with size of 1000 at most for each clusters.

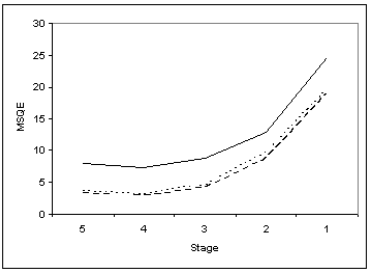


**Fig. 2.** Synthetic data results: Results obtained with the KDSOM (continuous line), FASOM$_\gamma$ (segmented line) and FASOM (points) models. In the horizontal axis, the stage and in the vertical axis the MSQE of the model trained at stage 4 but evaluated with data of previous stages.

The summary of the results are shown graphically in figure 3. The *FASOM*'s models increase considerably the number of prototypes when new cluster of data are found in stage 4. The *KDSOM* model has the greater MSQE error in train and test set, although it has the lower complexity, the model is not able to adapt when the input space change and forgets the previously learned patterns. It is important to mention that the *FASOM*$_\gamma$ obtain better performance with less complexity and at the same time it adapts to changing environment.

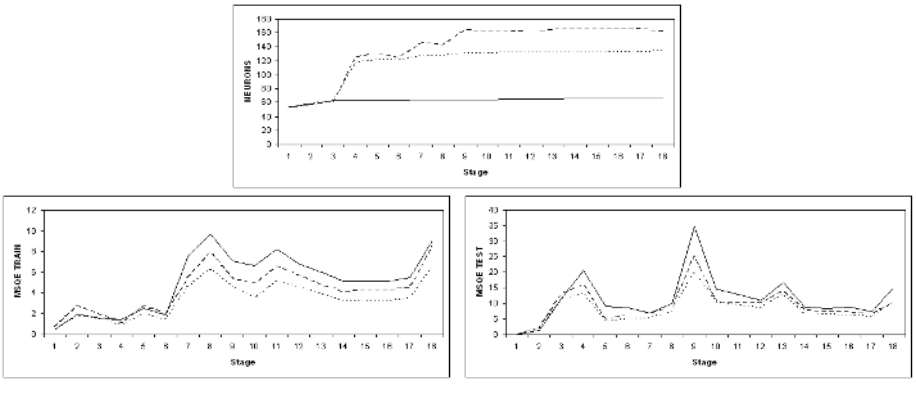


**Fig. 3.** Real Experiment: Results obtained with the KDSOM (continuous line), FASOM$_\gamma$ (points) and FASOM (segmented line) models. (Up) Number of neurons of the models in each stage. (down-left) MSQE of training for each stage. (down-right) MSQE of the model trained at stage $T$ but evaluated with data of stage $T-1$.

## 6 Concluding Remarks

In this paper we have introduced the Flexible Arquitecture of Self Organizing Maps (*FASOM*). The *FASOM* is a hybrid model that adapts $K$ receptive fields

of dynamical self organizing maps and learn the topology of partitioned spaces. It has the capability of detecting new data or clusters by creating new maps and avoids that other receptive fields catastrophically forget. In addition, receptive fields with few data can gradually forget by reducing their size and contracting their grid lattice.

The performance of our algorithm shows better results in the simulation study in both the synthetic and real data sets. In the real case, we investigated *El Niño* data. The comparative study with the *KDSOM* and *FASOM* without forgetting factor shows that our model the $FASOM_\gamma$ with forgetting outperforms the alternative models while the complexity of our model stays rather low. The *FASOM*s models were able to find the possible number of cluster and learn the topological representation of the partitions in the several training stages.

Further studies are needed in order to analyze the convergence and ordering properties of the maps.

## References


1. B. Ans, *Sequential learning in distributed neural networks without catastrophic forgetting: A single and realistic self-refreshing memory can do it*, Neural Information Processing - Letters and Reviews **4** (2004), no. 2, 27–37.
2. H. Bauer and T. Villmann, *Growing a hypercubical output space in a self-organizing feature map*, IEEE Trans. on Neural Networks **8** (1997), no. 2, 226–233.
3. E. Erwin, K. Obermayer, and K. Schulten, *Self-organizing maps: ordering, convergence properties and energy functions*, Biological Cybernetics **67** (1992), 47–55.
4. R. French, *Catastrophic forgetting in connectionist networks*, Trends in Cognitive Sciences **3** (1999), 128–135.
5. B. Fritzke, *Growing cell structures - a self-organizing network for unsupervised and supervised learning*, Neural Networks **7** (1994), no. 9, 1441–1460.
6. S. Grossberg, *Studies of mind and brain: Neural principles of learning, perception, development, cognition, and motor control*, Reidel Press., 1982.
7. A.K. Jain and R.C. Dubes, *Algorithms for clustering data*, Prentice Hall, 1988.
8. T. Kohonen, *Self-Organizing Maps*, Springer Series in Information Sciences, vol. 30, Springer Verlag, Berlin, Heidelberg, 2001, Third Extended Edition 2001.
9. U. Maulik and S. Bandyopadhyay, *Performance evaluation*, IEEE. Trans. on Pattern Analysis and Machine Intelligence **24** (2002), no. 12, 1650–1654.
10. M. McCloskey and N. Cohen, *Catastrophic interference in connectionist networks: The sequential learning problem*, The psychology of Learning and Motivation **24** (1989), 109–164.
11. J. McQueen, *Some methods for classification and analysis of multivariate observations*, In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and probability, vol. 1, 1967, pp. 281–297.
12. S. Moreno, H. Allende, C. Rogel, and R. Salas, *Robust growing hierarchical self organizing map*, IWANN 2005. LNCS **3512** (2005), 341–348.
13. C. Saavedra, H. Allende, S. Moreno, and R. Salas, *K-dynamical self organizing maps*, To appear in Lecture Notes in Computer Science, Nov. 2005.