Topology Based Fuzzy Clustering for Robust ANFIS Creation

Lord Kenneth PinPin
Daniel Fernando Tello Gamarra
Cecilia Laschi and
Paolo Dario
lordkenneth.pinpin@sssup.it
fgamarra@sssup.it
cecilia.laschi@arts.sssup.it
paolo.dario@sssup.it
Scuola Superiore Sant'Anna
ARTS Lab(Advanced Robotics Technology and Systems Laboratory)
Pisa-Italy

***Abstract*—This paper describes how the clustering topology of an input space data distribution is utilized to properly initialize an Adaptive Neuro-Fuzzy Inference System (ANFIS). We used a new unsupervised clustering algorithm called Topology based Fuzzy Clustering (TFC) that performs better than Growing Neural Gas (GNG) in extracting the input-space topology. The topology information in the form of number of nodes, node positions and node connectivity is used for the initialization of the ANFIS. Using two robotic modeling tasks as benchmarks, we demonstrate the improved performance of TFC-derived ANFIS when compared to the subclustering method found in the Fuzzy Logic Toolbox of Matlab.**

I. Introduction

Fritzke in [1], has shown how a new kind of neural network, the "growing neural gas" (GNG), is able to construct the topology in a data distribution. The topology is generated incrementally like a self organizing map (SOM) but overcomes the limitation of the SOM, in which the structure and dimension of the network must be defined a priori. Taking as motivation the growing nature of the GNG, other algorithms have been proposed in the literature such as [2] and [3], where topology is also grown. Topology information represents a reduction of dimensionality of the data distribution but still represents a rich source of information and its exploitation can help to solve many problems in different fields such as clustering, regression, data mining, and even control theory.

We tried a new algorithm based on GNG called Topology Based in Fuzzy Clustering (TFC), proposed by Abhishek Jaiantilal in [4], that exploits topological information to perform fuzzy clustering. The key innovation in TFC is the notion of an unknown fuzzy membership into a cluster. The following section shows this in more detail. We then used the topology obtained by TFC to initialize the structure of an ANFIS neural network by using node values and node connectivity. To test whether topology derived information is advantageous we used the ANFIS to solve the Inverse Kinematics problem in a two-link robot manipulator and the forward kinematics of a six-link manipulator. Both problems are employed as benchmarks to validate the proposed approach by comparing it with the subclustering algorithm [5] found in the fuzzy logic toolbox of Matlab. The use of the topology obtained by the TFC method in initializing an ANFIS is the main contribution of this paper.

The remainder of the paper is as follows. In the second section are described the algorithms used in this work; the third section describes the way in which the topology is used to initialize the ANFIS; in the fourth section the benchmark setup is detailed; in the fifth section results are presented; finally, conclusions and planned future works are described in the sixth section.

II. Topology Based Fuzzy Clustering(TFC)

The TFC algorithm proposed is a self-organizing neural network that takes the best properties of the GNG, i.e. its adaptivity, execution time and shape independence. However it is more effective than the GNG because it is just an one pass algorithm and its ability to find clusters and deal with outliers is superior to the GNG as reported in [4].

First is calculated the 'reference value' of every node (neuron) that is defined as a measure of the most exterior to the most interior node and are initialized with values between 0 and 1. The idea behind this was that the edge structure and relative distances between nodes can be used to find the central node(s) of the cluster. For each node a reference value is found, using the edge connection of the nodes and the relative distance between the nodes and neighbors and iterating the following equation:

$$\Phi'_p = \frac{\sum_{n=1}^{k} \Phi'_n * R_n}{\sum_{n=1}^{k} \Phi'_n * \sum_{n=1}^{k} R_n} \tag{1}$$

where Φ'_p is defined as the reference value of the node itself, n (from 1 to k), set of k neighbors of the node including itself, Φ'_n reference value of the nth node in the neighborhood, R_n, average distance between nodes and its neighbors. The initial reference values can be chosen at random. The iteration of this equation gives the converged reference values. The node with smallest reference value is considered the center of the

cluster. The proof of convergence and detailed examples can be found in [4].

Once the equation 1 has converged the fuzzy membership of each node in its corresponding cluster is calculated, according to:

$$fuzzy_{ic}(\Phi_{ic}) = \frac{min(reference_c)}{reference_{ic}} \tag{2}$$

Where $fuzzy_{ic}(\Phi_{ic})$ is the fuzzy membership of the ith node in cluster c, and $min(reference_c)$ is the minimum of the reference values in cluster c. This step normalizes each node's reference value with the least reference value within the cluster. For the four node example in Figure 1, equation 1 converges after 4 iterations:

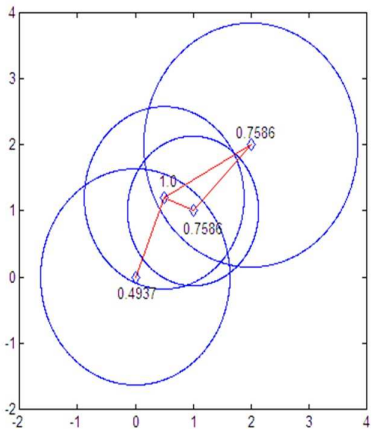

Fig. 1. Topology representation. Nodes are represented by blue diamonds, edges are in red. Average radius is drawn in blue circles. Values shown are the fuzzy membership

Iter1:referenceM= [0.3333 0.3333 0.3333 0.3333]
Iter2:referenceM= [0.5000 0.2500 0.3333 0.3333]
Iter3:referenceM= [0.5141 0.2539 0.3349 0.3349]
Iter4:referenceM= [0.5144 0.2540 0.3348 0.3348]
fuzzyM= [0.4937 1.000 0.7586 0.7586]

Modification of the learning rate of the nodes is done inversely proportional to the node's fuzzy membership into the cluster, thus the node having the highest membership value has the least learning rate and vice-verse for the node with the least membership. Thus

$$\Delta W_{S_1} = e_b(x - W_{S_1}) \tag{3}$$

for winner node S_1 is modified to

$$\Delta W_{S_1} = e_b(x - W_{S_1}) \frac{1}{FuzzyMembership(S_1)} \tag{4}$$

and for each neighbor S_i of S_1

$$\Delta W_i = e_n(x - W_i) \tag{5}$$

is modified to

$$\Delta W_i = e_n(x - W_i) \frac{1}{FuzzyMembership(S_i)} \tag{6}$$

e_b and e_n, are the learning rates for winner node and the neighbors respectively, the conditions for cluster creation and deletion are within the next cases:

1) Edge creation can cause cluster merging.
2) Edge deletion can cause cluster splitting.
3) Node deletion can cause cluster deletion.
4) Node insertion causes current cluster to increase in size.

After this phase, the algorithm passes through a phase of testing in which it tries to eliminate the outliers, the complete description of the algorithm and pseudocode is given in [4].

A. Adaptive Neuro Fuzzy Inference System (ANFIS)

Before showing how TFC is used to initialize the ANFIS, using a two input example, the ANFIS is reviewed briefly. Adaptive Neuro-Fuzzy Inference Systems are Fuzzy Sugeno models put in the framework of adaptive systems, a fuzzy Sugeno type is composed by rules of the type:

$$Rule\ 1 : if\ x_1\ is\ A_1\ \ and\ \ x_2\ is\ \ B_1,$$
$$then\ f_1\ \ = a_1x_1 + b_1x_1 + c_1$$

$$Rule\ \ 2 : if\ x_1\ is\ A_2\ \ and\ \ x_2\ is\ \ B_2,$$
$$then\ f_2\ \ = a_2x_1 + b_2x_2 + c_2$$

In Figure 2 is illustrated the architecture of the network. In the first layer the degree of the membership of the input is computed using a Gaussian membership function:

$$\mu_{A_i}(x) = \frac{1}{1 + [(\frac{x - c_i}{a_i})^2]^{b_i}}$$

where a_i, b_i, and c_i are the parameters of the gaussian function. Furthermore these parameters can be initialized by two approaches: grid partitioning and scatter partitioning. The

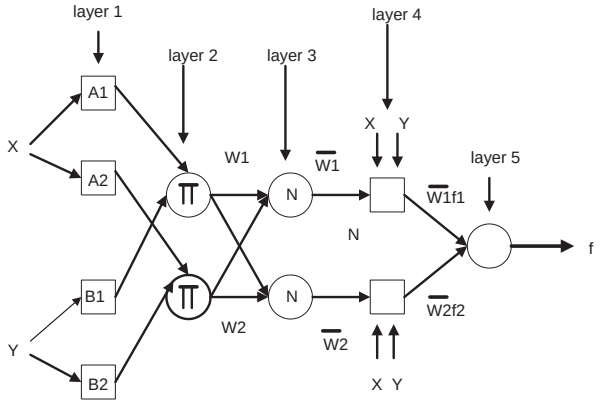

Fig. 2. ANFIS architecture

second layer calculates the firing strength (or weight) w_i of the ith rule,

$$w_i = \mu_{A_i}(x_1)\mu_{B_i}(x_2)$$

In the third layer the firing strengths are normalized with the sum of all rule's firing strengths:

$$\overline{w}_i = \frac{w_i}{w_1 + w_2}$$

In the fourth layer the output is calculated as the product of the normalized firing rate and the parameters set:

$$\overline{w}_i f_i = \overline{w}_i (p_i x + q_i y + r_i)$$

Finally in the fifth layer is calculated the overall output as the addition of all incoming signals,

$$\sum_i \overline{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i}$$

Training the network consists of finding suitable parameters for layer 1 and layer 4. Gradient descent methods are typically used for the non-linear parameters of layer 1 while batch or recursive least squares are used for the linear parameters of layer 4 or even a combination of both. See [6] for details.

B. Grid Partitioning Approach vs Scatter-Partitioning

Grid partitioning is an approach for initializing the structure in a fuzzy inference system. For example, if Gaussian membership functions are chosen, the centers of the Gaussians are confined to corners of a rectangular grid in the initialization of the Fuzzy inference system.

As defined in [7], the grid-partitioning approach to fuzzy systems has the serious disadvantage that the very regular partition of the input space may be unable to produce a rule set of acceptable size which is able to handle a given data set well. If, for example, the data contains regions with several small clusters of different classes, then small rule patches have to be created to classify the data in this region correctly. This problem becomes even more serious as the dimension of the input data increases.

To eliminate the problems associated with grid-partitioning, other ways of dividing the input space have been proposed. The approach in [7], known as scatter partitioning, is to allow the IF-parts of the fuzzy rules to be positioned at arbitrary locations in input space. This means that the centers of the Gaussians are not anymore confined to corners of a rectangular grid. Rather, they can be chosen freely, e.g., by a clustering algorithm working on the training data.

A problem to be solved with scatter partitions is to find a suitable number of rules and suitable positions and width of the rule patches in input space. As stated in [7], instead of first constructing a possibly poor neuro-fuzzy system and then improved later on, scatter partitioning has the advantage of immediately building a good system for the problem at hand.

The approach proposed in this paper uses TFC as the clustering tool for scatter partitioning the input space, and compare it with the subclustering algorithm which is also another scatter partitioning algorithm. The subclustering method is an extension of the mountain clustering method proposed by R. Yager and detailed in [5].

III. Using TFC Algorithm to Find the Data Topology and Initialize the ANFIS

The TFC is used to cluster the input training data. As shown in Figure 3 for the two-link inverse kinematics data, the topology involves a set of nodes interconnected by links. Each node belongs to a cluster (no multiple memberships) and depending on the distribution of the data the topology may have one or more clusters.

The ANFIS is constructed and initialized in this maner:

1) The number of rules (R) is equal to the number of nodes (N). The number of membership functions (M_x) for each input variable (x) is also equal to the number of nodes.
$$R = M_x = N;$$
2) The parameters of the first layer are taken from the nodes. For node (i), a Gaussian membership function is created for each input variable. So for the input variable x a Gaussian membership function is created G_{ix} with center c_{ix} equal to the x coordinate of node i.
3) The spread a of G_{ix} is taken from the x component of the average distance vector d of node i with its connected neighbors.
$$\vec{\mathbf{d}}_i = \frac{\sum_k^{Neighbors} \|(\mathbf{n}_i - \mathbf{n}_k)\|}{Neighbors}$$
4) After the ANFIS structure is created and layer 1 initialized, the parameters of layer 4 are computed by a batch least squares method that finds an estimate of the linear parameters of this layer based on the input and output training data. In [6] we can find a more detailed explanation of how the update is done.

At this stage, the ANFIS can be tested for initial performance with the training and test set.

IV. Benchmarks Setup

A. Inverse Kinematics for the Two Link Manipulator

In order to explore how topology can be used in robotics applications, It is used as a benchmark the known problem of the inverse kinematics in a robotic arm with two degrees of freedom. The inverse kinematics problem for a two link robot, as the one show in Figure 3, consists in finding the joint angle configuration (θ_1 and θ_2 angles for the first and second joint respectively) that can let us move the end effector to a desired position (X and Y) in the cartesian space. Inverse kinematics is a well known problem that has a simple mathematical solution, but becomes a cumbersome problem when the number of degrees of freedom of the robot is increased. The inverse kinematics solution for a two link manipulator will be our first benchmark to test the potential of our approach.

Matlab is used to simulate the robot of Figure 3. For the simulation the first link length has 10 units and the second link length is of 7 units. To generate the training data in Matlab θ_1 is defined as an angle that can be within 0 and $\frac{\pi}{2}$ radian degrees with an increment of 0.1, θ_2 as an angle within 0 and π radian degrees and with an increment of 0.1 degrees. The

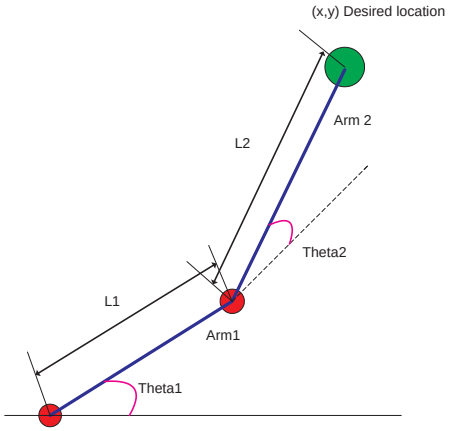

Fig. 3. Configuration of the two link manipulator

data for the training set is obtained from the mathematical solution of the forward kinematics:

$$X = L1 * cos(\theta_1) + L2 * cos(\theta_1 + \theta_2) \quad (7)$$

$$Y = L1 * sin(\theta_1) + L2 * sin(\theta_1 + \theta_2) \quad (8)$$

and for the testing set the solution of the inverse kinematics is used. For the coordinate X the data used is in a range within 0 and 2 with an increment of 0.1. For the coordinate Y the data is in range within 8 and 10 with an increment of 0.1 using the inverse kinematics equations:

$$cos(\theta_2) = (X^2 + Y^2 - L1^2 - L2^2)/(2 * L1 * L2)$$

$$sin(\theta_2) = sqrt(1 - cos(\theta_2)^2)$$

$$\theta_2 = atan2(sin(\theta_2), cos(\theta_2)); \quad (9)$$

$$k1 = L1 + L2 * cos(\theta_2)$$

$$k2 = L2 * cos(\theta_2)$$

$$\theta_1 = atan2(Y, X) - atan2(k2, k1); \quad (10)$$

The size of the training data is 512 samples while the testing data is 441 samples. Since the ANFIS is a sugeno fuzzy type inference system which has only one output, the two-link inverse kinematics problem is solved by two ANFIS systems- one for each output variable, θ_1 and θ_2.

B. Forward Model for the Six Link Manipulator

The second benchmark is to use the TFC-ANFIS to find the forward kinematic model of a six link manipulator. The motivation to try this test stems from one of the challenges in developmental robotics - how to find a sensory motor map by conducting self exploratory movements. This exploratory motor babbling is similar to how an infant creates his own sensory motor map that becomes an internal model for a reaching or grasping task. This approach of using motor babbling to create a forward model of an unknown robotic system is common in a developmental approach to robotic control as in [9].

Similarly an ANFIS neural network initialized with the TFC algorithm is used to find this forward model, that is, a relation between the joint angles of the manipulator and the position of the end effector in 3 dimensional space. The robotics toolbox of Peter Corke ([8]) for matlab is used as a simulation platform.

In order to generate the training data the robot passes through a phase of motor babbling, in which the workspace of the robot is divided into 4 regions and the joints are perturbed for a fixed number of cycles in each region. The end effector position is calculated using the forward kinematics of the robot. The total number of collected samples is split into 1000 and 400 for the training and testing set respectively. Three ANFIS systems are derived from the topology, one for each of the three output variables X, Y and Z.

C. TFC setup

TFC like GNG, has six parameters (e_b, e_n, α, β, A_{max} and λ) that control how the topology is created from the data. e_b and e_n control the adaptation rate of the winner and its topological neighbors as can be seen from equation 8 and 10. Node creation is controlled by λ, while edge deletion is controlled by A_{max} (note that nodes without any emanating edges are deleted as in the original GNG [1]). During node insertion, the error of the node with the maximum error (and its neighbor with the highest error) is adjusted using parameter α while the error of the other nodes are adjusting using parameter β.

Since, TFC is a single pass algorithm by fixing λ and A_{max}, the number of nodes can be fixed.

1) A_{max}, λ, manner of data presentation: We did preliminary runs to find values of λ that would result in fewer than 20 nodes for the two-link inverse kinematics problem. Also, we relaxed the one-pass option of TFC by duplicating each data sample, effectively doubling the training data, and randomly presenting it to the TFC algorithm in one pass. For the six-link forward model, the same data preparation and random presentation format (with A_{max} and λ values chosen to create less than 40 nodes).

2) e_b, e_n, α and β: These parameters were tuned by a genetic algorithm instead of manually tuning them. For the two-link, the fitness function was based on the Theta1 testing error of an ANFIS created from the TFC topology. Note, these 4 parameters were then used for the two-link inverse kinematics data the parameters in column 1 of Table I resulted in a topology with 17 nodes, while for the six-link forward kinematics data the parameters in column 2 resulted in a topology with 34 nodes.

The parameters of the TFC are summarized in Table I

V. Results

A. Two Link Manipulator Inverse Kinematics

Figure 4 shows the input data distribution for the two-link data. Comparing this with the topology obtained by TFC shown in Figure 5, we see that the input distribution is matched quite well. Following the method outlined in section *III* we

TABLE I
PARAMETERS OF THE TFC

Parameters	Two link	Six link
e_b	0.4421	0.4041
e_n	0.0129	0.0014
α	0.0289	0.0889
β	0.0069	0.0012
A_{max}	80	80
λ	65	60

initialize an ANFIS with 17 rules. The performance of the TFC-ANFIS with the training and test set is shown in Table II and III. Initial performance is already good and beats the subclustering algorithm of the Fuzzy Logic toolbox. Performance with the test set is shown in Figure 6. Performance is further improved with 100 epochs of the hybrid training method which is a combination of backpropagation with recursive least squares. The subclustering initialized ANFIS fails to surpass the TFC initialized ANFIS even after 100 epochs of training with the hybrid method.

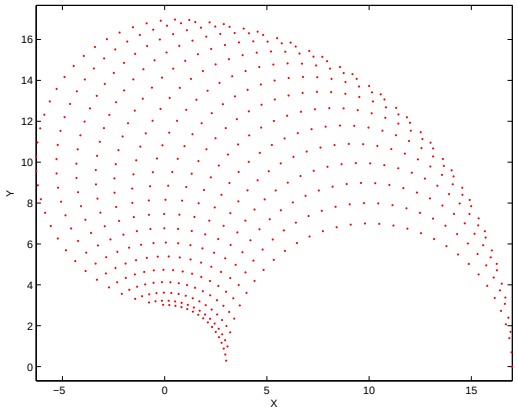

Fig. 4. X-Y coordinates generated for all theta1 and theta2 combinations using forward kinematics formulas used in the training set

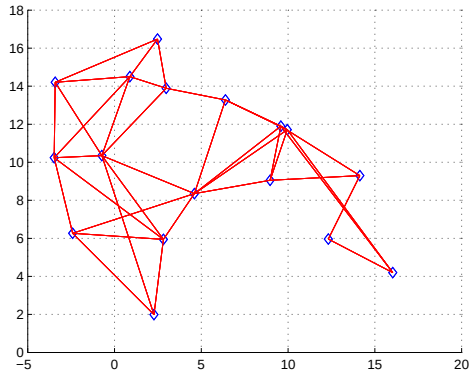

Fig. 5. Topology of the input Data

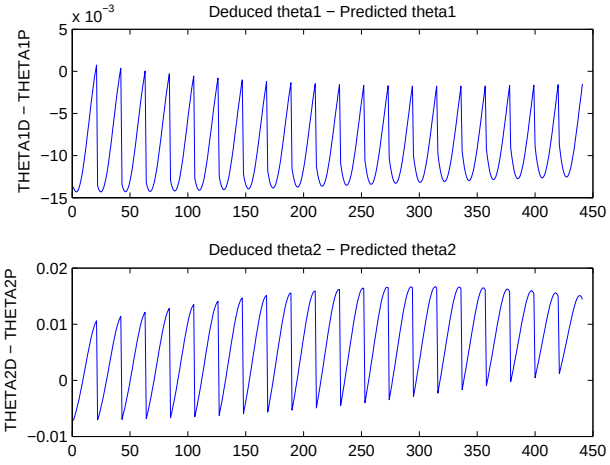

Fig. 6. Initial performance of TFC-ANFIS with the test set

TABLE II
INVERSE KINEMATICS OF THETA1

error	Subclustering	TFC
Training	0.030416	0.023636
Testing	0.0087433	0.010197
Training (100 epochs)	0.024802	0.021374
Testing (100 epochs)	0.005561	0.0058814

B. Six link manipulator forward kinematics

Figure 7 shows the topology obtained by TFC by plotting only θ_1, θ_2 and θ_3. It is evident that the four clusters found by TFC represent the four quadrants where motor/joint babbling was done. The performance of TFC derived ANFIS for each output variable X,Y,Z is shown in Tables IV, V, VI. With the exception of variable Z, using the topology from TFC to create the ANFIS results in a better performing ANFIS compared with using the subclustering tool of the toolbox. Performance is further improved by 100 epochs of hybrid training.

TABLE III
INVERSE KINEMATICS OF THETA2

error	Subclustering	TFC
Training	0.051657	0.02894
Testing	0.017418	0.0096486
Training (100 epochs)	0.040799	0.028608
Testing (100 epochs)	0.015167	0.0087282

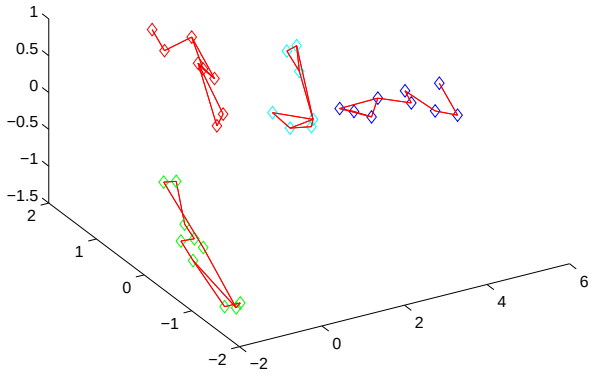

Fig. 7. Input space topology of the training data of the six-link manipulator- the $3D$ plot is a subset of the six variables made up of θ_1, θ_2 and θ_3

TABLE IV
FORWARD KINEMATICS OF X

error	Subclustering	TFC
Training	0.004324	0.0041086
Testing	0.005206	0.0048449
Training (100 epochs)	0.001801	0.0012637
Testing (100 epochs)	0.003572	0.0019591

TABLE V
FORWARD KINEMATICS OF Y

error	Subclustering	TFC
Training	0.005142	0.004343
Testing	0.00581	0.005895
Training (100 epochs)	0.002323	0.00137
Testing (100 epochs)	0.003342	0.003041

TABLE VI
FORWARD KINEMATICS OF Z

error	Subclustering	TFC
Training	0.0034549	0.0054177
Testing	0.0044379	0.0067046
Training (100 epochs)	0.0012921	0.0016889
Testing (100 epochs)	0.0020448	0.0024902

VI. CONCLUSION

It has been demonstrated in this paper how the topology of the input-space of the system can be useful in creating sugeno-type fuzzy inference systems. Also, we show that topology information creates a better ANFIS. This is the first time that the new unsupervised clustering algorithm, TFC, is used in a robotics application and demonstrated its advantages: fast-clustering (one-pass algorithm), and rich topological information. In terms of ease of use for the would-be ANFIS modeller, the TFC algorithm parameters lambda and Alpha allow independent control of the number of resulting nodes and hence the number of rules of the ANFIS.

ACKNOWLEDGMENT

The authors gratefully acknowledge the contribution of Abhishek Jaiantilal.

This work is supported by the ROBOTCUB project (IST-2004-004370), funded by the European Commission through Unit E5 "Cognitive SYSTEMS". Also we would like to thank the IIT (Italian Institute of Technology) for its financial support.

REFERENCES

[1] B. Fritzke, "A growing neural gas network learns topologies," *Advances in Neural Information Processing Systems*, vol. 7, 1995, MIT Press, Cambridge, Massachusetts, pp 625-632.

[2] Marsland, S., Shapiro J., and Nehmzow. U., "A self-organizing networks that grows when required," *Neural Networks*, vol. 15,pp 1041-1058, 2002.

[3] Alahakoon, D., Halgamuge, S.K. And Sirinivasan, B., "Dynamic Self Organizing Maps with Controlled Knowledge Growth for Knowledge Discovery" *IEEE Transactions on Neural Networks, Special Issue on Knowledge Discovery and Data Mining*, vol 11. pp 601-604, 2000.

[4] Abhishek Jaiantilal, *Online Clustering with Single-Pass Topology Based Fuzzy Clustering algorithm*, Master Thesis, Faculty of New Jersey Institute of Technology; Department of Electrical and Computer Engineering, 2006.

[5] Yager, R. and D. Filev, "Generation of Fuzzy Rules by Mountain Clustering," *Journal of Intelligent and Fuzzy Systems*, vol 2, No 3,pp 209-219, 1994.

[6] S. Roger Jang, "ANFIS: Adaptive–network– based fuzzy inference systems," *IEEE Trans. on System, Man and Cybernetics*, vol. 23, 1993, , No. 3, pp. 665-685.

[7] B. Fritzke, "Incremental neuro-fuzzy systems," *Proceedings of: Applications of Soft Computing, SPIE International Symposium on Optical Science, Engineering and Instrumentation*, 1997.

[8] Corke Peter, "A Robotics Toolbox for MATLAB" *IEEE Robotics and Automation Magazine, Magazine*, vol 3. pp 24-32, 1996.

[9] G. Sun and B. Scassellati, "A fast and efficient model for learning to reach" *International Journal of Humanoid Robotics*, vol 2., No.4, pp 24-32, 2005.