# An Adaptive Self-Organizing Algorithm for Clustering

**BONNY BANERJEE**

## I. INTRODUCTION

### *A. Motivation*

Visualization and hence perception of multidimensional data is a very difficult problem. Humans can readily perceive data values for two and three dimensions represented on spatial axes, but have difficulty at higher dimensions. There have been several strategies for circumventing this limitation like divide-and-conquer strategy [14], "emergent features" strategy [15], viewing the data in different formats or granularities, etc. However, the most common technique is to maintain the high dimensionality of representation but to code some data dimensions into non-spatial features, such as combinations of color, shape, size, brightness, etc.

### *B. Approach*

In this project, we propose a new algorithm for clustering multidimensional data sets in an unsupervised manner and hence, if necessary, interpret results from the clusters. Since no prior information about the data set is assumed, a generic clustering algorithm is required that can effectively handle any multidimensional data set. Thus, a lot of effort has been put to make sure that

the proposed algorithm works efficiently in most if not all cases. A lot of examples along with comparisons have been presented in the results, both with real and synthesized data sets.

The proposed algorithm maintains the high dimensionality of the given multidimensional data set all throughout and exploits the information content of each data point in order to extract information from the entire data set as a whole. It takes a top-down approach starting with a very few random seeds and gradually fragments into clusters increasing the number of seeds. For a given data set, the information content of each data point may be coded singly based on some property of that data point or may be coded into some weighted combination of elements of a multidimensional vector. The current work also endeavors to get rid of the common problem of seed initialization at null regions of many clustering algorithms by introducing weights based on the eigen values of the correlation matrix of the clusters. However, this introduction will not have any effect on the ultimate result if the boundary of each of the processors is not adaptively shrunk. Thus a new algorithm has been implemented, the results of which show marked improvement in feature extraction compared with some of the conventional algorithms.

The remaining paper is organized as follows. In the next section, the prevalent Topology Adaptive Self-Organizing Neural Network (TASONN) is described. Explanation of the drawbacks of the TASONN and the reasons for the emergence of the Information Extracting Self-Organizing Neural Network (IESONN) is described in section III. Finally, the essence of information extraction is presented with a comparison between the proposed and the traditional approaches. Also, the applications of IESONN to character recognition, to multidimensional data sets and for the design of microstrip patch antennas have been briefly illustrated. The article concludes with a general discussion and future research.

## II. The Topology-Adaptive Self-Organizing Neural Network

This section describes the conventional algorithm for the Topology Adaptive Self-Organizing Neural Network (TASONN). The TASONN is initialized with a very small number of non-interconnected processors, the weight corresponding to each of which assume random initial values. Each feature vector, presented to the TASONN, is associated with an input vector from the m-dimensional input space, and an output vector from the p-dimensional output space. The weight vectors of the processors, having exactly the same input/output dimensions as the features, are updated iteratively on the basis of the feature space S, $S=(P_1,P_2,...,P_N)$ being the set of feature vectors initially. On convergence, a set of processors whose weight vectors represent the features of the exhaustive input set is obtained.

In TASONN model, initially the topology is completely data driven. The net grows in size by means of a certain processor evolution mechanism, given by [10]

$$W_p(t+1) = W_p(t) + \alpha(t)[P_j - W_p(t)] \qquad \text{for } 0 < \alpha(t) < 1 \tag{1}$$

where $\alpha(t)$ is the gain term which decreases with t, $(\pi_1,\pi_2,...,\pi_n)$ denote the set of processors and $W_p(t)$ is the weight vector for the processor $\pi_p$ at the time t. At time instant t, $P_j$ is presented to the net. All the processors compete and two weight vectors $W_k(t)$ and $W_l(t)$ are selected such that

$$\|W_k(t) - P_j\| = \min_i \|W_i(t) - P_j\| \tag{2}$$

$$\|W_l(t) - P_j\| = \min_{(i \neq k)} \|W_i(t) - P_j\| \tag{3}$$

Hence $P_j$ modifies the weight vectors $W_k(t)$ and $W_l(t)$ according to (1). In this process the modification of the weights is continued, the weights tend to approximate the content of the feature set in an orderly fashion. The limiting weight vectors define the ordering. One presentation each of all the vectors makes one *sweep* consisting of N iterations. After one sweep is completed, the iterative process for the next sweep starts again from $P_1$ through $P_N$. Several sweeps make one *phase*. One phase is completed when the weight vectors of the current set of processors converge, that is, when

$$\|W_i(t) - W_i(t')\| < \varepsilon, \forall i \quad (4)$$

where t and t' are the iteration numbers at the end of two consecutive sweeps and ε is a predetermined small positive quantity that decreases with each phase exponentially.

After the completion of a phase, connectivity is assigned to the processors. For each pair of weight vectors, say $W_k$ and $W_l$, the number of feature vectors is calculated for which $W_k$ is the nearest and $W_l$ is the second nearest. Thus the entire space spanned by the feature vectors is partitioned with respect to the weight vectors and the cardinality of the partitions is stored in the form of a matrix, say N. If

$N(k,l)>\theta$, θ being a predefined threshold (5)

then a link between the processors $W_k$ and $W_l$ is established.

After the completion of a particular phase, a new phase starts with the introduction of a new processor. A new processor is inserted between the selected pair of processors $\pi_k$ and $\pi_l$ which has the weight vector as $\frac{W_k(t) + W_l(t)}{2}$. Thus a new set of processors is obtained and the object voxels are

presented again to this set of processors until the new weights converge. This process continues until the desired number of processors is reached.

## III. The Information-extracting Self-Organizing Feature Map

When implemented, the above algorithm tends to represent features of the multidimensional data set on the basis of density, which is not desired in many applications. In order to represent the information contained in the data set, the clusters have to be formed in such a way that each cluster represents an unique set of information and the number of clusters should be sufficient to hold the entire information contained in the given data set. When weights are introduced at the end of each phase, they should be introduced by comparing the information content of all the clusters and not on the basis of density. Further, after the weights are introduced in a region rich in information content, utmost care should be taken to see that the weight vector does not migrate to regions of lower information content. Weights will, in general, have a natural tendency to migrate to regions irrespective of the information content because they are selectively updated based on the Euclidean distance as a result of which the weights tend to settle for equilibrium based on density at convergence. In order to overcome this situation, the neighborhood of the weight vectors have to be dynamically determined such that they are influenced by feature vectors lying within that neighborhood only.

In order to compute the information content, the correlation matrix for each fragment of the given data set is computed. The correlation matrix provides a measure of the correlation among the different dimensions of all the elements in the cluster. It might be inferred that better the correlation is, lesser is the information content in the cluster. This logic, though might not seem to be so obvious in higher dimensions at the first instant, is pretty obvious to perceive in two dimensions. It is due to the same

logic that two points are enough to represent a straight line while two points are not enough to represent a parabola in any given dimension. This indicates that a parabola contains more information than a straight line. However, the information content is in no way related to the degree of equation depicting a curve as a circle and a parabola are both represented by second degree equations but a circle contains more information than a parabola.

The eigen value of each correlation matrix is computed. Each of the fragments are first normalized to have zero mean and unity variance before single value decomposition in order to ensure that the eigen values are sensitive only to the pattern of the fragments and not to their spatial position. As the variables become more correlated, the magnitude of the principle eigen value increases but the sum of the eigen values remains constant. Hence, the proposed algorithm tries to find the minimum of the principle (maximum) eigen values among all the clusters because lesser the principle eigen value is, less correlated the elements of that cluster are, and introduces the new processor in that cluster according to the equation

$$W_{new} = \frac{Q_{\max} + Q_{\min}}{2} \tag{6}$$

where $Q_{max}$ and $Q_{min}$ are the two extreme points of the selected cluster.

It is noteworthy that the feature vectors will attract the processors in each phase and hence their introduction in a particular cluster does not make considerable difference if the boundary of attraction is not intelligently determined at the beginning of each phase. As a result, for each weight vector, an adaptive nature of the boundary has been resorted to. The boundary is wide open when the algorithm starts with its initial list of processors, and any feature vector is free to attract any weight vector depending on the Euclidean distance. However, after a certain number of processors have been

introduced, a restriction to the boundary is adaptively imposed such that the newly introduced processor remains within that boundary at convergence. It might be noted that each time the radius of this boundary is chosen large enough not to hinder the significant influence of the feature vectors on the processor. Also, as the processor moves, it carries its boundary with itself, thus refraining from making the neighborhood topology of the net rigid. The equation for the radius R of the boundary at any instant may be given by

$$R = \frac{R_{initial}}{n_{weights}} \tag{7}$$

where $R_{initial}$ is the radius of the boundary at the initialization of the process, while $n_{weights}$ is the number of weight vectors at the instant under consideration. $R_{initial}$ is typically assigned a value such that the boundary at the initial phase of the algorithm consists of all the feature vectors in the entire data set. Thus the effect of introduction of the weights based on single value decomposition is kept intact.

**The Algorithm:**

**Step 1: Initialization**

Initialize weights to random values.

Set the initial radius of the neighborhood.

Obtain feature vectors in a random sequence.

Initialize sweep and phase to zero.

Set the desired number of processors at final convergence.

**Step 2:Sweep**

For all feature vectors, update the weight vectors as given by equations (2) & (3) if equation (7) is satisfied.

Increment number of sweeps by 1.

**Step 3:Check for convergence**

If equation (4) is not satisfied, go to Sep 2.

**Step 4:Phase**

Assign connectivity to the processors if equation (5) is satisfied.

If the desired number of processors hasn't yet been reached, find the single value decomposition of each cluster and insert a new processor according to equation (6), otherwise go to Step 5.

Set sweep equal to zero, increment phase by 1.

Randomize the feature vectors.

Go to Step 2.

**Step 5:Stop**

## IV. RESULTS

The results obtained by using the proposed algorithm have been shown in figures 1 through 6. Comparison of these results with the traditional TASONN and other prevalent algorithms like the K-means algorithm shows that introduction of weight vectors on the basis of information content not only helps in extracting more meaningful features but also helps to get rid of unnecessary information and thus helps assign more meaningful connectivity wherever relevant. Extraction of features based on

information content becomes particularly useful in exploration of huge data-structures (as shown in figure 4). Sometimes, it becomes necessary to infer from the extracted features of a huge data-structure in real-life problems [11],[12] whereby the information representation of the features becomes all the more important. In all the figures provided below (except figure 4), the bigger circles denote the weight vectors on convergence while the smaller ones denote the feature vectors specifying the dataset. The line segments joining the weight vectors denote the assigned connectivity wherever shown.

### *A. The Essence of Information Extraction: A comparison between the proposed and the traditional approaches*

In figure 1, the dataset has been intentionally generated to have four identical points at (10.0,10.0). Since those four points are identical, three of them are redundant, as they don't contain any more information than one of them. It is noteworthy that the conventional TASONN fails to recognize this redundancy and hence, the representation of the information content in the dataset is not achieved. But the proposed algorithm does recognize the redundancy and hence extracts all the meaningful information. This is evident from the position of the weight vectors on convergence.

In figure 2, a dataset has been synthesized to contain features along a straight line and a " Λ " shaped linear structure. It can be seen that the TASONN converges at spurious states while the proposed algorithm successfully extracts features based on the information content using the same number of weight vectors. The introduction of the weights is noteworthy as for the TASONN weights tend to get introduced based on density while for the proposed algorithm, weights get introduced based on information content. In this dataset, the density of points is almost the same everywhere as they are equidistant from each other but the information content of each point varies.

*B. Application to Character Recognition*

Figure 3 emphasizes on the accuracy of the proposed algorithm in representing a given structure. The connectivity of the weights depends on the placement of the weights on convergence. The fact that the proposed algorithm ends up representing the dataset better than the TASONN using the same number of weights due to better placement of the weights on convergence might be noted from the synthesized dataset in the figure.

In figure 4, the left and the right figures show the position of the weight vectors on convergence along with the connections using the traditional TASONN and the proposed algorithm respectively using the same number of weight vectors. It is noteworthy that the connectivity obtained using the proposed algorithm far outperforms the traditional TASONN due to the placement of the weight vectors on convergence as was also depicted in figure 3. For the proposed algorithm, connectivity is assigned in a similar way as it is in TASONN. This is an instance of application of the proposed algorithm for character recognition.

*C. Application to Multidimensional Data Sets*

Figure 6 shows the results of applying the proposed algorithm on a given flow pattern. The upper left figure is the given flow pattern while on the upper right is the three dimensional view of the magnitudes of the flow vectors at each pixel. On convergence, the weight vectors capture the flow patterns depending on the information content of each pixel. Thus, the critical points are adequately captured while coverage in the non-critical regions is also adequate not to lose any informative pattern. The lower left figure shows the position of the weight vectors with respect to the given dataset while the one on the lower right shows the independent position of the weight vectors both on convergence.

It is noteworthy that the entire information contained in the given $400 \times 400$ pixel map is successfully extracted by only 50 weight vectors.

*D. Application to the Design of Patch Antennas*

Recently, nonlinear neural optimization networks have been applied for analysis of arbitrarily shaped microstrip patch antennas with a very general bianisotropic grounded slab [20]. On the design side, reported works [21] are limited. In the design of square-patch antennas, the resonant frequency in the dominant mode is given. $\varepsilon_r$ and h are supplied, and l is to be calculated. Here, the proposed algorithm is used in place of conventional numerical techniques for the design.

*1) Data Generation and Preprocessing*

The resonant frequency of a square-patch antenna in its dominant mode is given by Wolf and Knoppik [23] as

$$f_r = \frac{c}{2l_{eff}\sqrt{\varepsilon_{dyn}}} \tag{8}$$

where c is the velocity of light in free space, $l_{eff}$ is the effective patch length (which a function of l and h), $\varepsilon_{dyn}$ is the dynamic dielectric constant considering the fringing field effects (which is again a function of l,h and $\varepsilon_{dyn}$. Determination of l from (8) is not direct. mathematical manipulations express it as an iterative equation, which is to be solved by standard numerical techniques. Application of the proposed algorithm reduces the numerical and computational complexities.

Equation (8) is used to generate data for $1.17 \le \varepsilon_r \le 15.0$, $0.0195cm \le h \le 0.3125cm$ and $0.2cm \le l \le 10.0cm$. The generated data was then used to train the processors of the proposed algorithm, each feature vector having three input fields namely, $\varepsilon_r$, h and l, and one output field $f_r$. The input fields $\varepsilon_r$ and l were normalized with respect to their highest values for transforming their range to [0,1] during training.

*2) Applying the Proposed Algorithm*

Here, the same proposed algorithm that had been used for all the illustrated applications is used to extract information from the multidimensional training set. Besides information extraction, this problem also demands the effective use of the knowledge stored as a result of the information extraction of the multidimensional feature input set generated from the following set of equations (9), (10), (11) & (12), and presented for training.

$$l + \Delta l = l_{eff} \tag{9}$$

$$\Delta l = 0.412h \frac{\left(\varepsilon_e + 0.3\right)\left(\frac{W}{h} + 0.264\right)}{\left(\varepsilon_e - 0.258\right)\left(\frac{W}{h} + 0.8\right)} \tag{10}$$

$$\varepsilon_e = \frac{\varepsilon_r + 1}{2} + \frac{1}{2}\frac{\varepsilon_r - 1}{\sqrt{1 + \frac{12h}{W}}} \quad \text{for} \quad \frac{W}{h} >> 1 \tag{11}$$

$$\varepsilon_{dyn} = \varepsilon_e \tag{12}$$

Out of 1290 vectors generated, 645 were used for training and the rest were used for testing of the trained neural net. The optimized values of the different parameters, which were obtained by trial and error for the training of the network, are as follows:

1. initial number of weights = 3,
2. final number of weights = 120,
3. learning rate, $\alpha(t) = \dfrac{0.1}{1 + \dfrac{t}{1000.0}}$, and
4. predetermined small positive constant or error, $\delta = 0.000001$.

*3) Results*

Network testing was done both for theoretical and experimental data. Table 1 demonstrates the comparison between theoretical patch length and that obtained using the proposed algorithm. A square patch antenna with l=4.14cm, $\varepsilon_r$=2.5, h=0.1524cm is experimentally found to resonate at 2.228GHz. When the proposed algorithm is used for the same antenna, the patch length is found to be 4.10cm. The equation used to derive the result may be given by

$$l = l_w \sqrt{\frac{\varepsilon_r^2 + h^2 + f_r^2}{\varepsilon_{rw}^2 + h_w^2 + f_{rw}^2}} \tag{13}$$

where $\{\varepsilon_{rw}, h_w, f_{rw}, l_w\}$ is the nearest weight vector to the feature vector $\{\varepsilon_r, h, f_r, l\}$ under consideration.

A distinct advantage of neurocomputing is that, after proper training, a neural network completely bypasses the repeated use of complex iterative processes for new cases presented to it. The proposed

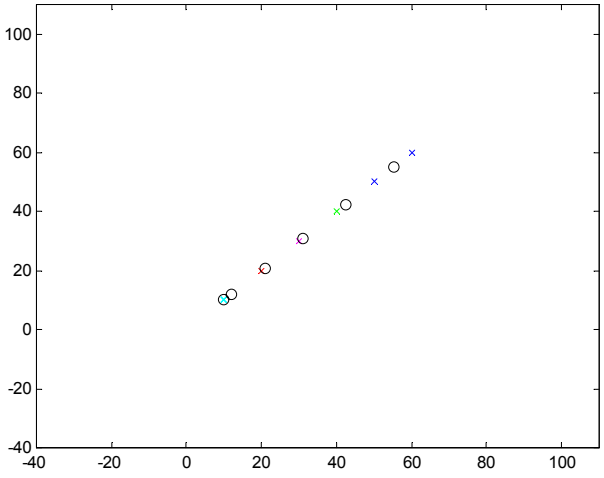
algorithm produces the length of the side of a square patch in a fraction of a second almost instantaneously for the other three specified parameters of the square patch antenna. The training time on a HP Workstation is just 12 minutes, which is to be carried out only once. Besides that, this single network structure can predict the results for a frequency range up to 15GHz provided the values for $\varepsilon_r$ and h are in the domain of the training set. Thus the proposed algorithm successfully designs a square patch antenna, the design being robust from the standpoint of computation time and accuracy. For a more detailed analysis for the problem of generalized design of patch antennas of any shape and size using the proposed algorithm in a much wider range, the reader is referred to [25].

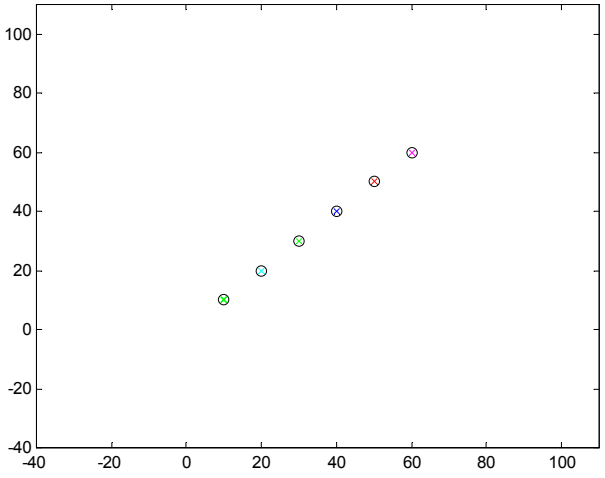
## V. CONCLUSION

From the given illustrations it is clear that the proposed algorithm successfully extracts "information" from a given dataset and hence is applicable to a much wider variety of real life problems compared to the traditional TASONN and many other prevalent clustering algorithms. The main emphasis in this article has been on the comparison between the proposed algorithm and TASONN because the TASONN has been known to extract features very efficiently and also gets rid of the rigidity of Kohonen's SOFM [6]. In the proposed algorithm, the nearest weight vectors relative to a feature is left to wander freely in the state space while the neighborhoods of the other weights have been adaptively shrunk to reduce the influence of the far off feature vectors. Thus the placement due to the introduction of the weights based on the information content of the clusters is kept significantly intact. As a result, on convergence, the final positions of the weights tend to extract meaningful information from the dataset under consideration rather than spurious features based on feature density.

The proposed algorithm has been experimented to successfully work with a lot of different practical problem areas like statistical pattern recognition (speech recognition, character recognition),

navigational planning (obstacle avoidance and path-planning) of mobile robots [12], adaptive design of various telecommunication devices [11], image compression, structure exploration, multidimensional optimization and classification problems. In fact, the proposed algorithm tries to "understand" a given dataset in the sense that it extracts only meaningful information with respect to certain properties while discards the redundant information. This notion of understanding a dataset is by itself a step forward in the field of machine intelligence as machines are limited to recognition only with almost no understanding. Understanding a dataset not only will help machines to act more like humans (as humans always extract features based on understanding and not blindly) but also will help solve many practical problems more efficiently a few of which has already been cited. Further research is underway for implementing a memory using the proposed algorithm that extracts context-based information from a given dataset.




| | | | | | |
|---|---|---|---|---|---|
| 10.0000 | 10.0000 | 10.0000 | 10.0000 | 10.0000 | 10.0000 |
| 10.0000 | 10.0000 | 11.9985 | 11.9985 | 20.0044 | 20.0044 |
| 10.0000 | 10.0000 | 20.8520 | 20.8520 | 29.9951 | 29.9951 |
| 10.0000 | 10.0000 | 30.9148 | 30.9148 | 39.9997 | 39.9997 |
| 20.0000 | 20.0000 | 42.4670 | 42.4670 | 50.0000 | 50.0000 |
| 30.0000 | 30.0000 | 55.1770 | 55.1770 | 59.9997 | 59.9997 |
| 40.0000 | 40.0000 | | | | |
| 50.0000 | 50.0000 | (b) | | (c) | |
| 60.0000 | 60.0000 | | | | |

. (a)

Fig 1: The pair of columns in (a), (b) and (c) shows the coordinates of points in a synthesized dataset, the coordinates of the six feature vectors on convergence using the conventional TASONN algorithm and the coordinates of the six feature vectors on convergence using the proposed algorithm respectively. The same is depicted in the figures presented above, where the synthesized data points are represented by “x” while the coordinates of the features at convergence by “o”, the left and the right figures are obtained by deploying the conventional TASONN and the proposed algorithm respectively.

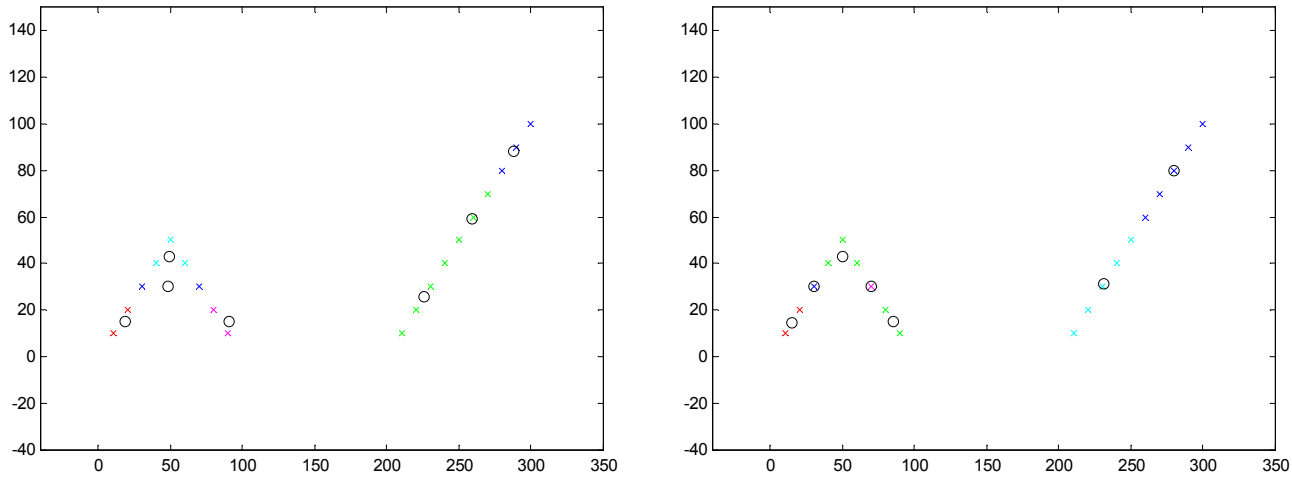



Fig 2: The left and the right figures show the position of the weight vectors on convergence using the traditional TASONN and the proposed algorithm respectively. The TASONN converges at spurious states while the proposed algorithm successfully extracts features based on the information content using the same number of weight vectors.

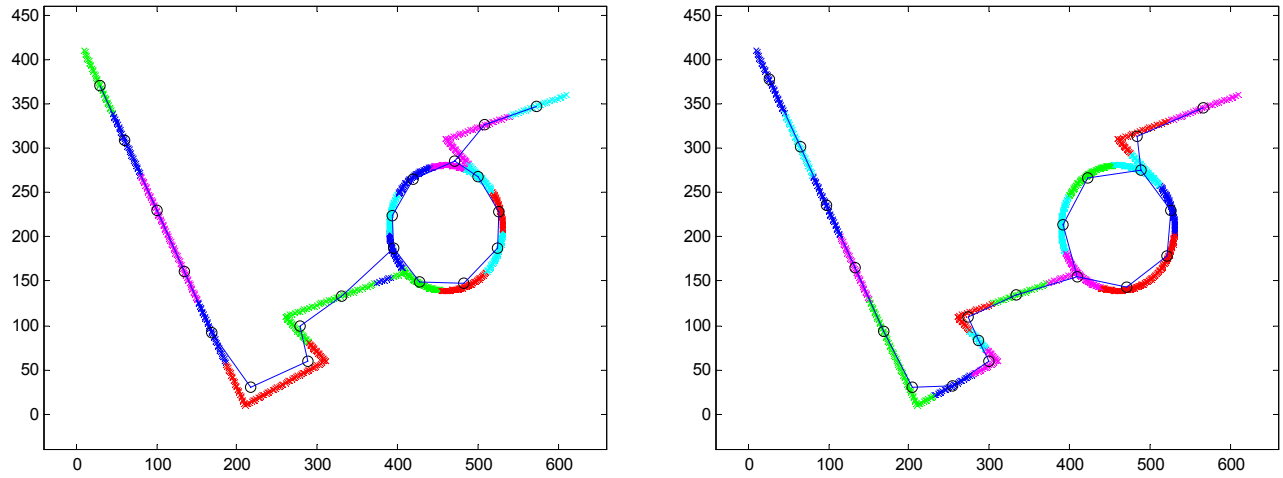



Fig 3: The left and the right figures show the position of the weight vectors on convergence along with the connections using the traditional TASONN and the proposed algorithm respectively. The proposed algorithm performs better than the traditional TASONN using the same number of weight vectors.

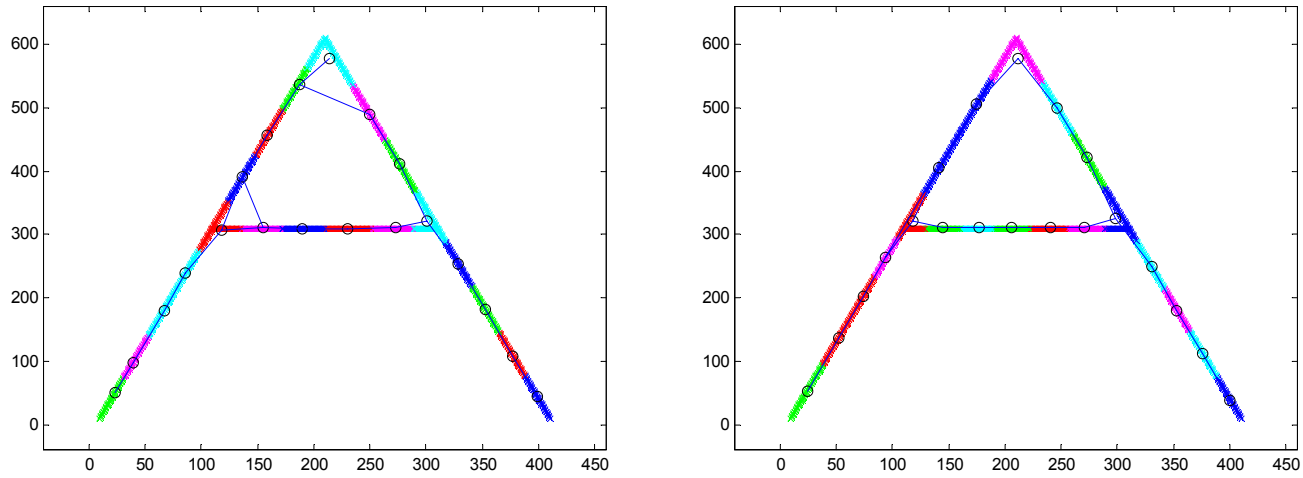



Fig 4: The left and the right figures show the position of the weight vectors on convergence along with the connections using the traditional TASONN and the proposed algorithm respectively. The connectivity obtained using the proposed algorithm far outperforms the traditional TASONN using the same number of weight vectors.

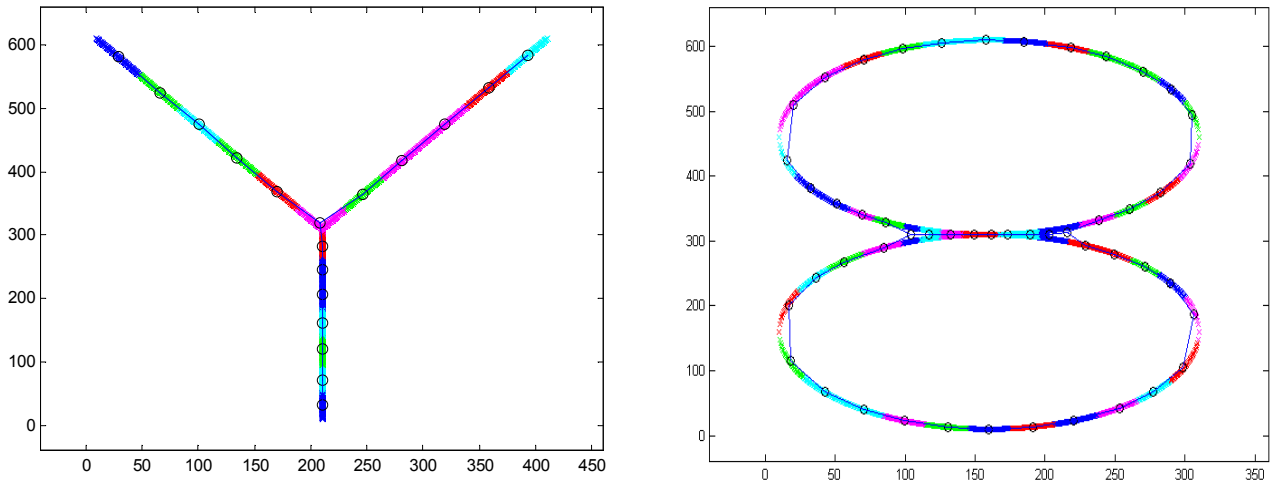



Fig 5: The left and the right figures show the position of the weight vectors on convergence along with the connections using the proposed algorithm. The results obtained by applying the proposed algorithm for the recognition of different alphabets and numerals depict the excellent scope of the algorithm in character recognition.

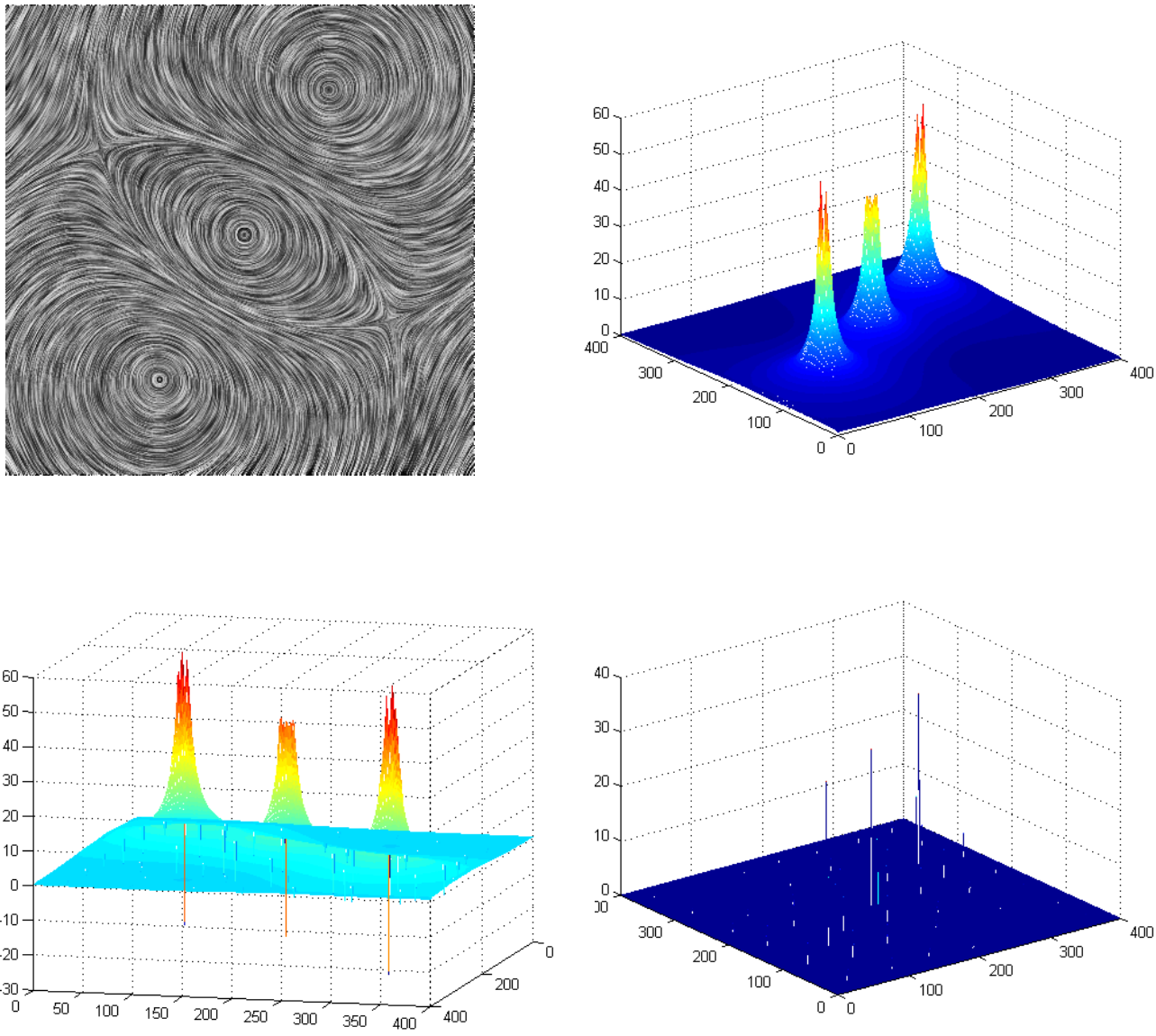



Fig 6: These figure shows the results of applying the proposed algorithm on a given flow pattern. The upper left figure is the given flow pattern while on the upper right is the three dimensional view of the magnitudes of the vectors at each pixel. The lower left and right figures show the position of the weight vectors on convergence using the proposed algorithm with respect to the given dataset and independently respectively.

TABLE 1: External inputs to the proposed algorithm after training and their outcomes, results are up to 99% accuracy when compared with [26].

| **INPUT** | | | **OUTPUT** |
|---|---|---|---|
| **Resonant Frequency, $f_r$ (MHz)** | **Relative Permittivity, $\varepsilon_r$** | **Substrate Thickness, h (cm)** | **Length, l (cm)** |
| 810 | 2.62 | 0.159 | 11.3 |
| 2228 | 2.50 | 0.1524 | 4.14 |
| 790 | 2.62 | 0.159 | 11.4 |

## REFERENCES


[1] A. Datta, T. Pal and S. K. Parui, "A modified self-organizing neural net for shape extraction", *Neurocomputing*, Vol 14, pp 3-14, 1997.

[2] A. Datta and S. K. Parui, "A skeleton generating neural network: a prerequisite for character recognition", *Proc. Int. Conf. on Computational Linguistics, Speech and Document Processing*, Indian Statistical Institute, Calcutta, 1998.

[3] T. Kohonen, *Self-Organization and Associative Memory*, Springer-Verlag, 1989

[4] A. Datta and S.K. Parui, "Shape extraction: a comparative study between neural network based and conventional techniques", *Neural Computing and Applications*, Vol 7, pp 343-355, 1998.

[5] J. A. Kangas, T. Kohonen and J. Laaksonen, "Variants of self-organizing maps", *IEEE Trans. Neural Networks*, Vol 1, pp 93-99, 1990.

[6] D. Choi and S. Park, "Self-creating and organizing neural networks", *IEEE Neural Networks*, Vol 5, pp 561-575, 1994.

[7] B. Fritzke, "Let it grow - self-organizing feature maps with problem dependent cell structure", In Kohonen T et al. (eds.), *Artificial Neural Networks*, Vol 1, North-Holland, 1991.

[8] M. Sabourin and A. Mitiche, "Modeling and classification of shape using a Kohonen associative memory with selective multiresolution", *Neural Networks*, Vol 6, pp 275-183, 1993.

[9] J. A. Anderson, *An Introduction to Neural Networks*, MIT Press, Cambridge, MA, USA, 1995.

[10] T. Kohonen, "The Self-Organizing Map", Proc. IEEE, Vol. 78, No. 9, pp. 1464-1480, 1990.

[11] B. Banerjee and I.S. Misra, "A neuro-computing model for the design of patch antennas", *Proc. Int. Conference on Communications, Computers and Devices (ICCCD)*, India, Dec 2000.

[12] B. Banerjee, A. Konar and S. Mukhopadhyay, “A Neuro-GA Approach for the Navigational Planning of a Mobile Robot”, *Proc. Int. Conference on Computers, Communications and Devices (ICCCD),* India, Dec. 2000.

[13] T. Kohonen, “Self-organized formation of topologically correct feature maps”, *Biological Cybernetics,* Vol 43, pp 59-69, 1982.

[14] W. Cleveland, *The elements of graphing data*, Wadsworth: Monterey, Ca, 1984.

[15] D. Montgomery and R. Sorkin, *Proceedings of the Human Factors and Ergonomics Society 37th Annual Meeting*, pp. 1325-1329, 1993.

[16] C. E. Shannon, ``A mathematical theory of communication," *Bell System Technical Journal,* vol. 27, pp. 379-423 and 623-656, July and October, 1948.

[17] T. Kohonen, *Self-Organization and Associative Memory*, Springer-Verlag, Berlin, 1984.

[18] G. A. Carpenter and S. Grossberg, “Neural Dynamics of Category Learning and Recognition: Attention, Memory Consolidation, and Amnesia”, in J. Davis, R. Newburgh and E. Wegman (Eds.) *Brain Structure, Learning and Memory*, AAAS Symposium Series, 1986.

[19] R. P. Lippmann, "An Introduction to Computing with Neural Nets", *IEEE ASSP Magazine*, April, pp. 4-22, 1987.

[20] L. Vegni and A. Toscano, “Analysis of microstrip antennas using neural networks”, IEEE Trans. Magn., Vol 33, pp. 1414-1419, march, 1997.

[21] R. K. Mishra and A. Patnaik, “Design of circular microstrip antenna using neural network”, Inst. Electron. Telecomm. (IETE) Eng. J. Res., Vol 44, nos 1-2, pp. 35-39, Jan.-Apr., 1998

[22] A. Patnaik, R. K. Mishra, G. K. Patra and S. K. Dash, “An artificial neural network model for effective dielectric constant of microstrip line”, IEEE Trans. Antennas Propagat., Vol 45, pp. 1697, Nov., 1997.

[23] I. Wolf and N. Knoppik, “Rectangular and circular microstrip disk capacitors and resonators”, IEEE Trans. Microwave Theory Tech., Vol MTT-22, pp. 857-864, Oct., 1974.

[24] R. K. Mishra and A. Patnaik, “Neural network-based CAD model for the design of square patch antennas”, IEEE Trans. Antennas Propagat., Vol 46, pp. 1890, Dec., 1998.

[25] B. Banerjee, “A self-organizing auto-associative network for the generalized physical design of microstrip of patches”, IEEE Trans. Antennas Propagat., [submitted for publication].

[26] Y. T. Lo et al, *Report on Study of Microstrip Antennas, Microstrip Phased Arrays and Microstrip Feed Network*, RADC-TR-77-406, EE Dept., University of Illinois, Urbana.