Journal of Computer Science 4 (8): 631-637, 2008
ISSN 1549-3636

Self-Generation ART-1 Neural Network with Gradient-Descent Method Aid for Latin Alphabet Recognition

Mbaïtiga Zacharie
Department of Media Information Engineering, Okinawa National College of Technology, 905 Henoko, Nago, 905-2192, Okinawa Prefecture, Japan

Abstract: Problem statement: In this study a self-generation ART-1 neural network that is an efficient algorithm that emulates the self-organizing pattern recognition developed to avoid the stability-plasticity dilemma in competitive networks learning, is presented for Latin alphabet recognition to use in a vision system for road sings recognition. **Approach:** The first step of our approach deals with the training process where a set of input vectors are presented sequentially to the preprocessor to specify the inputs for the networks. Secondly the value of the mean squared error was used to measure the candidate for the output in the recognition phase. Thirdly to move down the large error-surface created by delta rule during the search phase the gradient-descent is used by changing each value of the weights by an amount that is proportional to the negative of the sigmoid function slope. **Results:** In the simulation test our system can self organize in real time producing stable recognition while getting inputs pattern beyond those originally stored. It can preserve its previously learned knowledge while keeping its ability to learn new patterns. **Conclusions:** The result suggests that the proposed system is pertinent to be put in practical use.

Keywords: ART-1, binary input vector, character recognition, Latin alphabet

INTRODUCTION

The human brain performs the formidable task of storing flood of sensory information received from the environment. From a deluge of trivia, it must exact vital information, act upon it, and perhaps file it away in long-term memory. Understanding the human memorization presents serious problems (it may take a century to pierce this memorization secret or may be not); news memories are stored in such a fashion that existing ones are not forgotten, modified or destroy. This creates a dilemma: how can the brain remain plastic, able to record new memories as they arrive, and yet retain the stability needed to ensure that existing memories are not erased or corrupted in the process?

Conventional artificial neural networks have failed to solve the stability-plasticity dilemma. Too often, learning a new pattern erases or modifies previous training. In some cases, this is unimportant. If there is only a fixed set of training vectors, the network can be cycled through these repeatedly and may eventually learn them all. In a back-propagation network, for example, the training vectors are applied sequentially until the network has learned the entire set. If, however, a fully trained network must learn a new training vector, it may disrupt the weights so badly that complete retraining is required. In a real world case, the network will be exposed to a constantly changing environment it may never see the same training vectors twice. Under such circumstances a back-propagation will often learn nothing; it will continuously modify its weights to no avail, never arriving at satisfactory setting, even worse. Carpenter and Grossberg[1] have shown example of a network in which only four training patterns, presented cyclically, will cause network to change continuously never converging. This instability is one of the main factors that led Grossberg and his associates to explore radically different configurations, hence the birth of the adaptive resonance theory or ART. One of the excellent points of the ART is that it can maintain the plasticity required to learn new patterns, while preventing the modification of patterns that have been previously learned. This potential advantage has created considerable interest in the possibilities for applying ART neural networks in engineering; and has resulted in great deal of research over the last 25 years. Some of the claimed advantages are exaggerated, but others are certainly proven, and ART neural networks are becoming a standard technology for many engineering. But many people have found the theory difficult to understand. The mathematics behind ART is

complicated, but the fundamental ideas and the implementation are not. ART is divided into two paradigms: ART-1 and ART-2; each defined by the form of the input data and its processing. ART-1 is designed to accept only binary input vectors, whereas ART-2 can classify both binary and continuous inputs. In this paper, a self-generation of ART-1 neural network with gradient-descent method aid for Latin alphabet recognition is presented. We first constructed a preprocessor by forming the letters A, B, C, D, E, F, G, H and I on a 3x3 square grid that we denoted as pixel for consistency with generally accepted terminology. These letters are then changed into binary input vectors by converting the grid of pixels (1 = Yes, 0 = No) into a sequence of numbers for the preprocessor to specify the inputs for the networks. Next the gradient-descent method is used to move down the large error-surface created by delta rule by changing each value of the variables by an amount that is proportional to the negative of the sigmoid function slope.

MATERIALS AND METHODS

Art-1 architecture: Figure 1 illustrates the main features of a typical ART-1 network. Rectangles represent fields where STM patterns are stored. Triangles represent adaptive filter pathways and arrows represent paths which are not adaptive. Filled squares represent gain control nuclei, which sum input signals. Their output paths are nonspecific in the sense that at any given time a uniform signal is sent to all nodes in a receptor field. Gain control at F_1 and F_2 coordinates STM processing with input presentation rate.

In Fig. 1, two successive stages, F_1 and F_2 of the attentional subsystem encode patterns of activation in short-term memory (STM). Each bottom-up or top-down pathway between F_1 and F_2 contains an adaptive long-term memory (LTM) trace that multiplies the signal in its pathway. The rest of the circuit modulates these STM and LTM processes. Modulation by gain1 enables F_1 to distinguish between a bottom-up input pattern and a top-down priming or template pattern, as well as to match these bottom-up patterns.

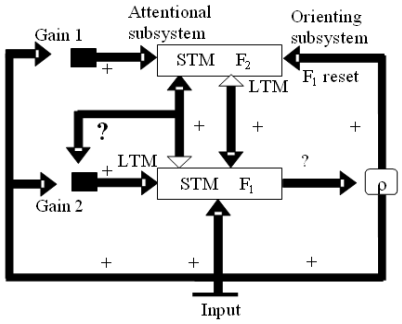

Fig.1: Simplified ART-1 Architecture

In particular, bottom-up input can supraliminally active F_2; top-down expectations in the absence of bottom-up inputs can subliminally sensitize, or prime F_1; and a combination of bottom-up and top-down input is matched according to a 2/3 Rule which actives the nodes within the inter-section of the bottom-up and top-down patterns.

Thus, within the context of self-organizing ART architecture intentionally implies a special matching rule. Carpenter and Grossberg[1] prove that 2/3 Rule matching is necessary for self-stabilization of learning within ART-1 in response to arbitrary sequences of binary input patterns. The orienting subsystem generates a reset wave to F_2 when the bottom-up input pattern and top-down template pattern mismatch at F_2 according to the vigilance criterion. The reset wave selectively and enduringly inhibits active F_2 cells until the current input is shut off. Offset of the input pattern terminates its processing at F_1 and triggers offset of gain2. Gain 2 offset causes rapid decay of STM at F_2 and thereby prepares F_2 to encode the next input pattern without bias. The criterion for an adequate match between an input pattern and a chosen category template is adjustable in ART-1 architecture. The matching criterion is determined by a vigilance parameter that controls activation of the orienting subsystem. All other things being equal, higher vigilance imposes a stricter matching criterion, which in turn partitions the input set into finer categories. Lower vigilance tolerates greater top-down/bottom-up mismatches at F_1, leading in turn to coarser categories. In addition, at every vigilance level, the matching criterion is self-scaling: a small mismatch may be tolerated if the input pattern is complex, while the same featural mismatch would trigger reset if the input represented only a few features.

Proposed self-generation Art-1 neural network: The fundamental feature of ART-1 is that it is composed of a large number of interconnected processing units. These units are relatively simple, and the network gets it computational power from the many units being connected, with outputs from the units being inputs to others. So it is essential that the input data are in the right form for a network to operate on, and they must contain sufficient information for the classification to be made for a relevant output. It is in this sense that, for the proposed application, before starting the network training process, all weights vector B_j and T_j as well as the vigilance parameter ρ are set to initial values. The weights of the bottom-up vectors B_j are all initialized to the same low values. That is

$$bij > L\,(L-1+m) \text{ for all } i, j \quad (1)$$

Where:
m = The number of components in the input vector
L = A constant >1 (for precision L=2).

This value is critical; if it is too large the network will allocate all recognition-layer neurons to a single input vector. The weights of the top-down vectors T_j are initialized to 1, that is to say

$$T_{ij} = 1 \text{ for all } j, i \quad (2)$$

This value also is critical; Carpenter and Grossberg[1] prove that top-down weights that are too small will result in no matches at the comparison-layer and no training. The vigilance parameter ρ is set in the range from 0 to 1 depending upon the degree of mismatch that is to be accepted between the stored parameter input vectors.

Training process: The training is the process in which a set of input vectors are presented sequentially to the input of the network, and the network weights are so adjusted that a similar vector activate the same recognition layer neuron. We began our training phase by constructing the following preprocessor.

Step1: Form 9 samples of Latin alphabet A, B, C, D, E, F, G, H and I each on a 3×3 square grid as (Fig. 2).

Step2: We all know that ART-1 is designed to accept only a binary input, and to satisfy this configuration, we converted the grid of pixel containing letters into a sequence of binary numbers(1= Yes, 0 = No). But in the program code, we trained the network not to ignore totally 0 instead replace it by dot (.)

Step3: Let the preprocessor as shown in Fig. 3 specifies the input for the network and the letters assigned to each class give th**e** information required by the post-processor to make a classification.

The preprocessor converts the data available to the system into a form that can be input to ART-1 neural network, i.e., it encodes the input data as a list of numbers. The network then does the essential classification work to give the desired outputs. The outputs are presented as a list of numbers which is to be decoded by the post-processor.

Delta rule: search phase: The search space for this proposed application is multidimensional, with the number of dimensions corresponding to the weights. The value used to measure the candidate is the mean squared error Ë. This error is the difference between the desired output *d* and the actual output *y*. This value, *e*, is squared and the average value found for all letters in the training, which consists of all the known input-output pairs. To be sure as we do not know precisely what is going on inside the network, we supposed that if there are *p* letters in the training for a single processing unit the mean squared error is

$$\ddot{E} = \frac{1}{p}\sum_{p=1}^{p} e_p^2 \quad (3)$$

Where $e_p = d_p - y_p$ for each letter. The weighted sum is

$$S = \sum_{i=0}^{1} w_i x_i = w_0 x_0 + w_1 x_1, \text{ since } x_0 = 1 \quad (4)$$

$$S = w_0 + w_1 x_1 \quad (5)$$

then the squared error is

$$\ddot{E} = \frac{1}{2}\sum_{p=1}^{2}(d_p - y_p)^2 \quad (6)$$

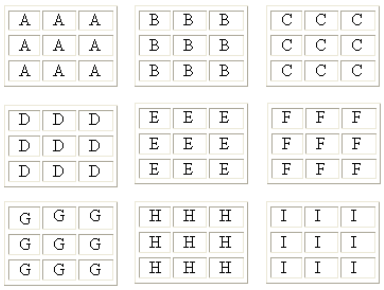

Fig. 2: Formation of the sample letters

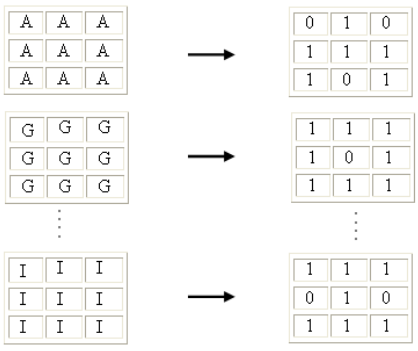

Fig. 3: ART-1 training session

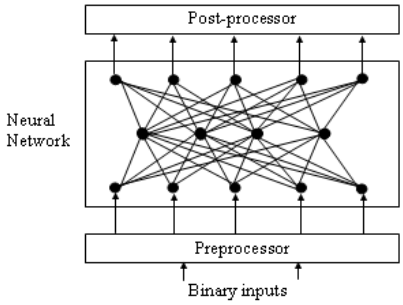

Fig. 4: ART-1 Neural network sandwiches the network between a preprocessor and post-processor.

In order to show how $\ddot{E}$ changes with the value of w_0 and w_1 we used a small sleight of hand to move the hard-limiting from the output and compared the desired value with the weight sum since the are only two possible binary inputs (1 and 0). The mean squared error has been calculated as follows:

$$\ddot{E} = \frac{1}{2}\sum_{p=1}^{2}(d_p - y_p)^2 = [\,(d_1 - s_1)^2 + (d_2 - s_2)^2 \qquad (7)$$

When $x_1 = 0$, the desired output is $d_1 = 1$ and the output is $x_0 = 1$. Similarly, when $x_1 = 1$, the desired output is $d_1 = 0$ and the actual output is

$$S = w_0 + w_1 \qquad (8)$$

The total error is:

$$\ddot{E} = \frac{(1 - w_0)^2 + (0 - w_0 - w_1)}{2} \qquad (9)$$

At this stage the search phase is stimulated but it fails because a valley exists at the point created by the surface error where w_0=0 and w_1= -1 as shown in Fig. 4. So another neuron is assigned in the recognition layer, and the weight are set to equal the corresponding component of the input vector.

In order to move down the surface error to minimum, we used the gradient descent

Gradient descent: This is achieved by changing the each value of the variables (weights) by an amount that is proportional to the negative of the slope of the sigmoid function. That is:

$$\Delta w_i = -\alpha\frac{\partial \ddot{E}}{\partial w_i} \qquad (10)$$

Where:
α = A constant,

$\ddot{E}$ = The mean squared error.

The symbol Δ is the delta and the notation ∂w_i means ″ the change to w_i ″. The derivative of the mean squared error with respect to a weight w_i is

$$\frac{\partial \ddot{E}}{\partial w_i} = \frac{1}{p}\sum_{p=1}^{p}\frac{\partial e^2}{\partial w_i} = \frac{1}{p}\sum_{P=1}^{P}\frac{\partial}{\partial w_i}(d - y)^2 \qquad (11)$$

By the chain rule of calculus,

$$\frac{\partial \ddot{E}}{\partial w_i} = \frac{1}{p}\sum_{P=1}^{P}\frac{\partial}{\partial w_i}(d - y)^2 \times \frac{\partial y}{\partial w_i} = \frac{1}{p}\sum_{P=1}^{P}\frac{\partial}{\partial w_i} - 2(d - y)\,\frac{\partial y}{\partial w_i} = -\frac{2}{p}\sum_{P=1}^{P} e\frac{\partial y}{\partial w_i} \qquad (12)$$

This shows that in order to find the derivative of mean squared error with respect to the weights, the derivative of output y with respect to the weights is needed. This is why the hard-limiter function won't do as it is not a difference function. In order to overcome this problem a difference output function is applied to the summed weights. This has to be differentiable and monotonic which mean that for every value of weighed sum, there is only one value of output. A commonly used function is the sigmoid function as shown in Fig. 6 and sigmoid function has been used.

The equation for this sigmoid output is:

$$y = \frac{1}{1 + e^{-s}} \qquad (13)$$

Where, $S = \sum_{i=0}^{n} w_i x_i$ and e is the base of natural algorithms.

In Fig. 5, when the weighted sum is greater than 0, the value of the output rises to 1 as the value of the weighted sum increases and similarly when the weighted sum is less than 0 the output falls to 0 as the value of the weighted sum decreases. When the weighted sum is 0 the output is 0.5.

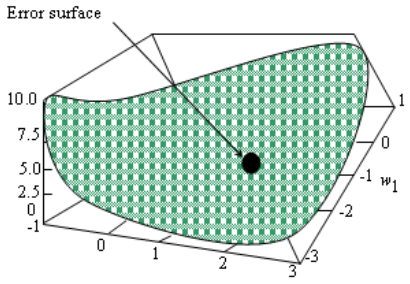

Fig.5: Error surface to be moved down

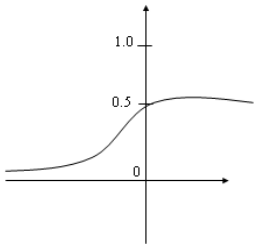

Fig.6: Sigmoid function

The derivative of the sigmoid function with respect to the weighted sum, *s*, is:

$$\frac{dy}{ds}=\frac{e^{-s}}{(1+e^{-s})^2}=y(1-y) \tag{14}$$

This give us enough information to be able to evaluate the changes that have to be made to the weights to reduce the error and consequently to find the solution.

The derivative of the output with respect to a weight has been found using the chain following rule.

$$\frac{\partial y}{\partial w_i}=\frac{\partial y}{\partial s}\times\frac{\partial s}{\partial w_i},\ \text{where}\ \ S=\sum_{i=0}^{n} w_i x_i \tag{15}$$

For a sigmoid function, $y=\frac{1}{1+e^{-s}}$, we have

$$\frac{dy}{ds}=\frac{e^{-s}}{(1+e^{-s})^2}=y(1-y) \tag{16}$$

Also

$$\frac{\partial s}{\partial w_i}=\frac{\partial}{\partial w_i}\sum_{j=0}^{n} w_j x_j=\sum_{j=0}^{n}\frac{\partial wj}{\partial wi}x_j=x_i \tag{17}$$

Substituting these, we get

$$\frac{\partial \ddot{E}}{\partial w_i}=-\frac{2}{p}\sum_{P=1}^{P} e\frac{\partial y}{\partial w_i}=-\frac{2}{p}\sum_{P=1}^{P} e\frac{\partial y}{\partial s}\times\frac{\partial s}{\partial w_i}=$$
$$-\frac{2}{p}\sum_{P=1}^{P} ey(1-y)x_i\ ,$$

Which can be written as:

$$\frac{\partial \ddot{E}}{\partial w_i}=-\frac{2}{p}\sum_{P=1}^{P}\delta x_i$$

where $\delta=ey(1-y)=y(1-y)(d-y)$

Therefore, from (10)

$$\Delta w_i=-\alpha\frac{\partial \ddot{E}}{\partial w_i}=\frac{2\delta}{p}\sum_{p=1}^{p}\delta x_i \tag{20}$$

which can be written by the Greek symbol η (eta) as

$$\Delta w_i=\eta\sum_{p=1}^{p}\delta x_i \tag{21}$$

Where $\Delta w_i=\eta=\frac{2\alpha}{p}$ is a constant

This states that the adjustment to the weighted w_i is the sum of $\eta\delta x_i$ taken over all of the letters in the training. It is common practice to simply this procedure by simply changing the value of w_i by an amount $\eta\delta x_i$ after each letter in the training. The result is delta rule formula:

$$\Delta w_i=\eta\delta x_i\ \text{where}\ \delta=y(1-y)(d-y) \tag{22}$$

and η is a small positive constant usually between 0 and 1, which we called the learning rate. From this formula the change to a weight is positive when η and x_i *are* both positive and negative. If x_i is 0.3 and y is 0.2 when the desired output is 1, then

$$\delta=0.2(1-0.2)(1-0.2)=0.128 \tag{23}$$

$$\Delta wi=\eta\times0.128\times0.3=0.0384\eta \tag{24}$$

We assumed that $\eta = 0.5$ and w_i is initially -0.8.The value of Δw_i is added to the old value of the weight, w_i to produce the new value. The new value for w_i is:

$$\begin{aligned} w_i(\text{new}) &= w_i(\text{old}) + \Delta w_i \\ &= -0.8 + (0.0384 \times 0.5) = -0.7807 \end{aligned} \quad (25)$$

Hence the value of the weight gets more positive so that the value of the weighted sum is increases, and consequently the output is very closer to the desired value.

RESULTS

During the simulation test the network starts by clamping the input at F_1 because the output of F_2 is zero. G_1 and G_2 are both on and the pattern bottom-up weights was initialized to 0.16. Initialization of the bottom-up weights to low values is essential to the correct functioning of the ART-1 system and to ensure that an uncommitted neuron will not ″overpower″ a trained recognition-layer neuron. If they are too high, the input vector that has already been learned will activate an uncommitted recognition-layer rather than the one that has been previously trained. The vigilance value is first set to 0.62.The number of nodes F_2 layer is set to 14.

Then, the letter A is presented to the newly initialized system the search phase fails. The reason for this failure is because there is not stored pattern that matches it within the vigilance limit, a new neuron is assigned in the recognition layer and weights T_j are set to equal the corresponding components of the input vector, with weights B_j becoming a scaled version. Next, the letter B is presented, this also fails in the research phase and another new neuron is assigned. The letter C is presented again to the system where it fails.

To deal with the failure search phase, we used the gradient descent and moved the surface error created by delta rule where $w_0 = 1$ and $w_1 = -1$ by changing each value of the weights by an amount of that is proportional to the negative of the sigmoid function slope, and then increase the vigilance value to 0.96 in order to produce highly detailed memories. The letter A is presented for the second time to the network; where a message comes out on the computer screen: The following Latin alphabet ″ A ″ has been recognized. Next, the letter B is presented, the network again recognizes ″ B″ . This process is repeated for ″ C″, ″D″ as well as for all other letters where we realized how important the gradient-descent was, when the delta rule created an error surface. We continued our simulation by presenting for this time the letters ″ A″, ″ B ″,″D ″ ″F ″and ″ I ″ at the same time to the network and within few seconds the following message pop-up on the computer screen. Five of nine letters has been recognized and they are: ″ A″, ″ B″, ″D″, F ″and ″ I″.

This phase is repeated 8 times of which no unsuccessful result has been obtained. To initiate the search for more than three letters, we reset the signal temporarily that disables the neuron in the recognition layer for the duration of the search, G_1 goes to one, and a different recognition layer neuron wins the competition. Its pattern is then tested for similarity and the process is repeated until either a recognition neuron wins the competition with similarity greater that the vigilance. The search simulation for a combination of more than three letters is initialized by presenting the letters ″ C″, ″ F″, ″B ″ ″D ″ and ″I ″ to the network system and unlike for the for previous test after the recognition the system rearranged them in alphabetic order . In the simulation test our system can self organize in real time producing stable recognition while getting inputs pattern beyond those originally stored. It can also preserve its previously learned knowledge while keeping its ability to learn new pattern. A parameter called the attentional vigilance parameter determines how fine the categories will be.

DISCUSSION

Adaptive Resonance Theory-1 (ART-1) is a competitive neural network learning method the most used for letters, characters recognition due to its ability to process information almost the same way the human brain processes information. The connections are distinguished by a weight, which represents the data that will be used to perform the task assigned. An interesting aspect of the ART-1 neural network is that there is no need for the patterns to be inputted in any order. The connections of the units in the layers are arranged in such a way that the input layer is connected to the interface layer, each unit in the input and interface layer are connected to the reset unit, which in turn is connected to every cluster layer by two pathways forming a cycle[2]. In general the algorithm of the ART-1 begins by initializing a binary input vector which is then presented to the input layer, and this data is sent to the corresponding interface layer.

The interface layer then send signals to the cluster layer over connections pathways. Each cluster unit computes its net input and units compete for right to be active. This is way ART-1 is called a competitive

learning method. As the ART-1 can maintain the plasticity to learn new patterns, it becomes a standard technology for many engineering of which many research have been done in the field of pattern recognition, such characters (including Japanese, Chinese characters) and letters recognition. Some of the claimed advantages are exaggerated, but others are certainly proven.

In [3] the author used the ART for letter recognition such as L, R, B, F and S representing Left, Right, Backward, Forward and Stop for mobile vehicle control when moving on a road vehicle, precisely to read the information written on the traffic signs. The target letters are written by a red color and pasted to a yellow board as background and presented to the ART network. The output result was very good as the ART can recognize clearly the Letter L without missing some part of it. But unfortunately compare to our model, when more than three letters are written without red color and yellow background are presented to the author model, the ART cannot recognize all of them but only one. Hence this model cannot be used to control the mobile vehicle on a road vehicle for traffic signs recognition. Because the information written on traffic signs are the combination of more than three letters such as TURN LEFT, which is composed of 8 letters.

Whereas our model is capable of recognizing more than four letters written in any color sequentially and can preserve its previously learned knowledge while keeping its ability to learn new input patterns that can be saved in such a fashion that the stored patterns cannot be destroyed or forgotten. It can also prevent the modification of the letters that have been previously learned.

In [4] the author used the ART for on-line Chinese character recognition where the value of vigilance parameter used in the author system is the same value we have used in our model. The simulation result was excellent but compare to our model, the correct classification rate deteriorates when the value of the vigilance parameter goes beyond 0.96. In other word the recognition performance deteriorates around 0.96 which is the opposite in our model. In our system the ART-1 performs well when the value of the vigilance parameter is increasing. One of the weakness of the author model is that the ART cannot as in [3] recognize the combination of more than three characters while our model does.

CONCLUSION

The self-generation ART-1 neural network with gradient descent method aid for Latin alphabet recognition was modeled using Microsoft Foundation Class /C++ tools. In the proposed work, the ART-1 can recognize a combination of more than three letters within a short period of time without encounter any error due to the gradient-descent that moves down the error surface created by the delta rule. The training session consists of all the known input-output pairs. The ART-1 self organize in real time producing stable recognition while getting inputs pattern beyond those originally stored. It can also preserve the modification of the previous learned. This model has been compared to other research already in the open literature.

REFERENCES

1. Gail Carpenter, A. and S. Grossberg, 1987. A massively parallel architecture for self-organi-zing neural pattern recognition machine. Comput. Vision Graphics Image Process., 37: 54-115. http://portal.acm.org/citation.cfm?id=28157
2. Faussett, Laurene, 1994. Fundamentals of Neural Networks, Architectures Algorithms and Appplitions. US Edn., Prentice-Hall, Inc., Upper Saddle River, NJ, USA., p: 461. ISBN: 0-13-334186-0
3. Massanori Sugisaka and Fengzhi Dai, 2001. Letter recognition and obstacle avoidance by art neural network. Reports Faculty Eng., 44: 31-34. http://sciencelinks.jp/j-east/article/200202/000020020201A1039282.php
4. Hang Joon Kim, Jong Wha Jung and Sang Kyoon Kim, 1996. On-line character recognition using art-based stroke classification. J. Elsevier Sci., 17: 1311-1322. DOI: 10.1016/0167-8655(96)00078-5
5. Massanori Sugisaka and Fengzhi Dai, 2003. Pattern recognition and control by adaptive methods for an intelligent mobile vehicle. J. Artificial Life Robotics, 7: 164-169. DOI: 10.1007/BF02471200
6. Mbaïtiga Zacharie, 2007. Adaptive Resonance Theory1 (ART1) neural network based horizontal and vertical classification of 0-9 digits recognition. J. Comput. Sci., 3: 869-873. http://www.scipub.org/fulltext/jcs/jcs311869-873.pdf.
7. Gail A. Carpenter and Stephen Grossberg, 1987. A massively parallel architecture for a self-organizing neural network recognition machine. Comput. Vision, Graphics, Image Process., 37: 54-115. http://portal.acm.org/citation.cfm?id=28154.28157
8. Carpenter, G.A., Grossberg, S., Markuzon, N., Reynolds, J.H., and Rosen, D.B, 1992. Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps. IEEE Trans. Neural Networks, 3: 698-713. DOI: 10.1109/72.159059