UNIVERSITÉ
LAVAL

A Comparison of Fuzzy ARTMAP and Multilayer Perceptron for Handwritten Digit Recognition

Martin Busque Marc Parizeau

October 31, 1997

Computer Vision and Systems Laboratory
Université Laval
Sainte-Foy (Québec), Canada
G1K 7P4
Tel: (418) 656-2131, ext. 7912
Fax: (418) 656-3159
Email: parizeau@gel.ulaval.ca

A Comparison of Fuzzy ARTMAP and Multilayer Perceptron for Handwritten Digit Recognition

Abstract

This paper explores the features of the Fuzzy ARTMAP neural network classifier in the context of on-line handwritten recognition. A Fuzzy ARTMAP classifier is compared with a multilayer Perceptron trained using the standard backpropagation algorithm. Results show that although the recognition performance of both networks is similar, the convergence of the Fuzzy ARTMAP is both much faster and incrementaly stable than that of the Perceptron. In fact, it is shown experimentally that a Fuzzy ARTMAP can be trained in a single pass with almost no performance penalty when using the fast learning mode. The recognition experiments were conducted on handwritten digits extracted from the Unipen training dataset.

Keywords: Neural Networks; Fuzzy ARTMAP; Multilayer Perceptron; Handwritten Digit Recognition; Unipen Dataset.

1 Introduction

There are many ways to model a system for handwriting recognition. One of them is to use a neural network. Still, there exist many different kinds of neural networks. Among them, the multilayer Perceptron and the Fuzzy ARTMAP.

The Perceptron is now a well-known architecture [1]. It is able to model a system, based on a set of inputs with the correponding desired outputs. It

appeared in the 1960's and has much evolved, especially since the development of the back-propagation algorithm in the 1980's. This supervised learning neural network has proven in practice to be quite efficient and has been used on a number of diverse problems.

What particularly interests us in this discussion is the Fuzzy ARTMAP. This network, introduced much more recently (1991), is a supervised network like the Perceptron. But it has some very interesting additional properties, such as a very fast convergence and a capacity for incremental learning. Our objective is to compare the Fuzzy ARTMAP with the Perceptron on handwritten digit recognition.

First, in Section 2, the Fuzzy ART will be presented. The Fuzzy ART is an unsupervised network which is the essential component of the Fuzzy ARTMAP. Then the Fuzzy ARTMAP itself will be explained in Section 3. We will discuss in Section 4 the experimental results obtained on handwritten digit recognition with both the Perceptron and the Fuzzy ARTMAP.

2 Fuzzy ART

The Fuzzy ART is a neural network introduced by Carpenter, Grossberg and Rosen in 1991 [2]. It is a modified version of the binary ART1 [3], which is notably able to accept analog fuzzy input patterns, i.e. vectors whose components are real numbers between 0 and 1. The Fuzzy ART is an unsupervised neural network capable of incremental learning, that is it can learn continouously without

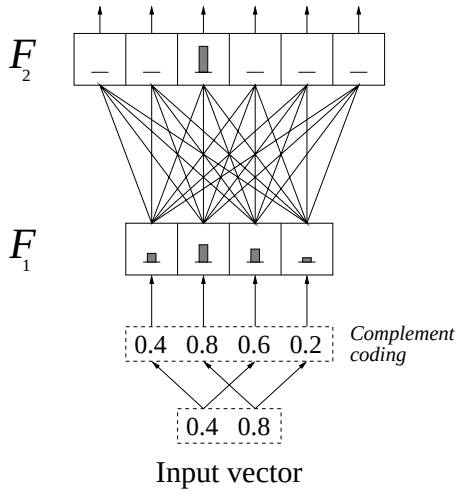

Figure 1: *Sample Fuzzy ART network.*

forgetting what it has previously learned.

A Fuzzy ART network is formed of two layers of neurons, the input layer F_1 and the output layer F_2, as illustrated in Figure 1. Both layers have an activity pattern, schematized on the figure with vertical bars of varying height. The layers are fully interconnected, each neuron being connected to every neuron on the other layer. Every connection is weighted by a number lying between 0 and 1. A neuron of F_2 represents one category formed by the network and is caracterized by its weight vector $\mathbf{w}_j$ (j is the index of the neuron) . The weight vector's size is equal to the dimension M of layer F_1. Initially all the weight vectors' components are fixed to 1. Until the weights of a neuron are modified, we say that it is *uncommited*. Inversely, once a neuron's weights have been modified, this neuron is said to be *commited*.

The network uses a form of normalization called complement coding. The operation consists on taking the input vector and concatenating it with its com-

plement. The resulting vector is presented to layer F_1. Therefore, the dimension M of layer F_1 is the double of the input vector's dimension. Complement coding can be deactivated, in this case layer F_1 will have the same dimension as the input vector. Unless specified otherwise, we will always suppose that complement coding is active.

The Fuzzy ART learns by placing hyperboxes in the $\frac{M}{2}$–dimensions hyperspace, M being the size of layer F_1. As said earlier, each neuron of layer F_2 represents a category formed by the network, and this category is defined by a box[1] [2, section 5]. The position of the box in the space is encoded in the weight vector of the neuron. Because of complement coding and of the learning process we will explain later, the first half of the weight vector memorizes one corner of the box (the closest to the origin) and the other half memorizes the opposite one.

Learning process

When an input vector is presented to the network, it is first preprocessed into complement coding form. The resulting vector, $\mathbf{I}$, represents a point in the space. The activity pattern on layer F_1 is set equal to $\mathbf{I}$. The **choice function** is then evaluated for each neuron of F_2. This function is defined by

$$T_j = \frac{|\mathbf{I} \wedge \mathbf{w}_j|}{\alpha + |\mathbf{w}_j|} \tag{1}$$

where $|\ |$ is the L_1 norm of the vector, i.e. the sum of its components ($|\mathbf{p}| \equiv \sum_{i=1}^{M} p_i$), $\wedge$ is the fuzzy AND operator ($(\mathbf{p} \wedge \mathbf{q})_i \equiv \min(p_i, q_i)$) and α is the choice

[1] To simplify the discussion, we will sometimes drop the "hyper" prefix and use the terms box and space to refer respectively to hyperbox and hyperspace.

parameter. The choice parameter α must be greater than zero. It is usually chosen close to zero for a good performance [4, p.701]. This parameter ensures that when a point is enclosed in more than one hyperbox, the smallest one is selected. This is in fact the objective of the choice function: to choose the smallest box in which the point is included. A point is said to be in a box when the weight vector $\mathbf{w}_j$ corresponding to this box is a fuzzy subset of the input vector $\mathbf{I}$. If no such box exist, either the one which needs to be the less expanded to enclose the point is selected or a new one is created, i.e. the first uncommited neuron is chosen. Note that we define the size of a box as the sum of its lengths in each dimension.

Once a neuron is selected (the *winning neuron*), a **vigilance criteron** is evaluated. The vigilance criterion causes the network to choose another box (neuron) if the one chosen is already too large. Thus the vigilance parameter ρ controls the maximum size of hyperboxes [2, section 5]. Mathematically, the vigilance criterion is

$$\frac{|\mathbf{I} \wedge \mathbf{w}_J|}{|\mathbf{I}|} \geq \rho \tag{2}$$

where J is the index of the winning neuron on F_2 and ρ is the vigilance parameter, which lies in the interval $[0, 1]$. If the criterion is respected, the Fuzzy ART learns the input vector, else it selects the next neuron with the highest choice function and reevaluates the vigilance criterion. These two steps are repeated until the vigilance criterion is respected. At that moment, it is said that the network is in *resonance*. All components of the activity pattern on F_2, which is also the output vector of the Fuzzy ART, are set to 0 except for the index of the winning neuron which is set to 1.

When the Fuzzy ART enters resonance, it proceeds with the **learning** of the input vector $\mathbf{I}$ by modifying the weight vector $\mathbf{w}_J$ of the neuron as follows

$$\mathbf{w}_J = \beta(\mathbf{I} \wedge \mathbf{w}_J) + (1 - \beta)\mathbf{w}_J \tag{3}$$

Here, J is the index of the winning neuron and $\beta \in [0, 1]$ is the learning rate. When $\beta = 1$, we are in *fast learning*. We may note that the weights are nonincreasing through time, which means that a hyperbox can never shrink but only expand itself. With fast learning, equation 3 means that the box associated with the neuron is enlarged just enough to include the point represented by the input vector. If β is smaller than 1, the box will expand towards the point, its enlargement being directly proportional to the size of β. When we set $\beta < 1$, we use what is called the *fast-commit slow-recode* option. This consists in setting $\beta = 1$ when the neuron is not commited and using the real value of β (< 1) after it is commited.

3 Fuzzy ARTMAP

The Fuzzy ARTMAP, introduced by Carpenter et al. in 1992 [4] is a supervised network which is composed of two Fuzzy ARTs. The Fuzzy ARTs are identified as ART_a and ART_b. The parameters of these networks are designated respectively by the subscripts a and b. The two Fuzzy ARTs are interconnected by a series of connections between the F_2 layers of ART_a and ART_b. The connections are weighted, i.e. a weight w_{ij} between 0 and 1 is associated with each one of them. These connections form what is called the *map field* F^{ab}. The map field has two

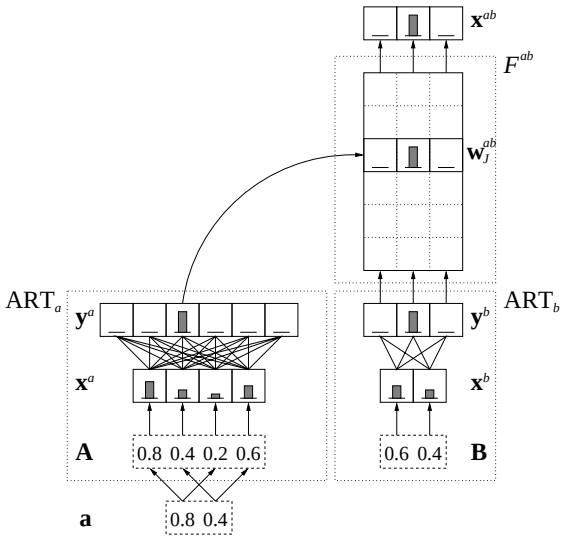

Figure 2: *Sample Fuzzy ARTMAP network.*

parameters (β_{ab} and ρ_{ab}) and an output vector $\mathbf{x}^{ab}$.

The input vector $\mathbf{a}$ of ART$_a$ is put in complement coding form, resulting in vector $\mathbf{A}$. Complement coding is not necessary in ART$_b$ so we present the input vector $\mathbf{B}$ directly to this network. Figure 2 presents the structure of the Fuzzy ARTMAP. The weights of the map field's connections are illustrated by a vertical bar with a height proportional to the size of the weight. The weights of the map field are all initialized to 1.

Training

In order to train the Fuzzy ARTMAP, we present to ART$_a$ a vector representing a data pattern and to ART$_b$ a vector which is the desired output corresponding to this pattern. The network uses a form of hypothesis testing. When it receives the first vector, it deduces to which category it should belong. With the second

vector, the Fuzzy ARTMAP can either confirm or reject the hypothesis, in which case the process is repeated with a new one.

During the training process, learning is deactivated in ART_a ($\beta_a = 0$) and is controlled by the Fuzzy ARTMAP. The vigilance parameter of ART_a, ρ_a, varies during learning. We identify the initial value of ρ_a as the *baseline vigilance* $\overline{\rho_a}$. The vigilance of ART_b, ρ_b, is set to 1 to perfectly distinguish the desired output vectors.

When vectors A and B are presented to ART_a and ART_b, both networks soon enter resonance. A vigilance criterion is then evaluated to verify if the winning neuron of ART_a corresponds to the desired output vector presented to ART_b. This criterion is

$$\frac{|\mathbf{y}^b \wedge \mathbf{w}_J^{ab}|}{|\mathbf{y}^b|} \geq \rho_{ab} \tag{4}$$

where $\mathbf{y}^b$ is the output vector of ART_b (the activity pattern on F_2^b), J is the index of the winning neuron on F_2^a, $\mathbf{w}_J^{ab}$ corresponds to the weights of the connections to the Jth neuron of F_2^a and $\rho_{ab} \in]0, 1]$ is the vigilance parameter of the map field. Note that if the map field uses fast learning, the value of this fraction will either be 0 or 1. If the criterion is not respected, the vigilance of ART_a is increased just enough to select another winning neuron ($\rho_a > |\mathbf{A} \wedge \mathbf{w}_J|/|\mathbf{A}|$, see equation 2) and the vector $\mathbf{A}$ is repropagated in ART_a.

When the vigilance criterion is respected, the map field learns the association between vectors $\mathbf{A}$ and $\mathbf{B}$ by modifying its weigths as follows

$$\mathbf{w}_J^{ab} = \beta_{ab}\mathbf{x}^{ab} + (1 - \beta_{ab})\mathbf{w}_J^{ab} \tag{5}$$

The weights in ART_a are also modified as in equation 3. The vigilance of ART_a is set back to its baseline value $\overline{\rho_a}$.

In terms of hypexboxes, the map field of the Fuzzy ARTMAP associates a category number to each box created by ART_a. This number is the index of the associated neuron of F_2^b. Consequently, each category learned by the Fuzzy ARTMAP is represented by many rectangles. In other words it learna to distinguish the data by mapping boxes into the space, enclosing each category with a certain number of these boxes.

Classifying

Once the Fuzzy ARTMAP is trained, it can be used as a classifier. In this case, the ART_b network is not used. We present a single input vector to ART_a which is propagated until resonance, with a (temporary) vigilance $\rho_a = 0$. Thus, the first category selected by the choice function is accepted. Learning in this network is also temporarily deactivated ($\beta_a = 0$).

The output vector of the map field is then set to

$$\mathbf{x}^{ab} = \mathbf{w}_J^{ab} \tag{6}$$

where J is the index of the winning node on F_2^a. If the map field used fast learning ($\beta_{ab} = 1$), the output vector contains only 0's except a 1. The index of this component is the number of the category in which the input vector $\mathbf{A}$ has been classified. The use of the map field is thus to associate a category number to each neuron of ART_a's F_2 layer (F_2^a), i.e. to each box in the hyperspace.

If fast learning was not used, one neuron of F_2^a can correspond to many categories with different degrees. One way to determine the category number could be to select the index of the maximal component. If needed, the desired output vector can be restored from the weights of the neuron of F_2^b whose index is the category number.

4 Experimental results

We used the multilayer Perceptron and the Fuzzy ARTMAP to recognize handwritten digits taken from the fifth release of the Unipen training set, section 1a[2]. All the samples in the database were identified with one of the following labels depending on their quality : Good, Ok and Bad. Good samples are perfectly recognizable, Ok samples are of a lower quality but should still be recognizable and Bad samples are either mis-labelled[3], mis-segmented[4], mis-scaled[5] or would be very difficult to recognize by a human.

The data set for our experiments is composed of a training set and a test set. We first selected randomly 300 Good and Ok samples of each digit (0-9) to form the 3000 samples training set. The test set consists of the rest of the database (3519 samples), approximately 11% of them being Bad samples.

To transform the data into a vectorial space, we used a fuzzy representation based on a segmentation of the digits into circular arcs and straight lines [5].

[2]For more information on the Unipen database, see `http://hwr.nici.kun.nl/unipen/`.

[3]For example a 1 labelled as a 0.

[4]A single digit is segmented into two different digits.

[5]Two different scales are applied on the X and Y axis.

Each vector representing a data sample has 49 components. Briefly, each digit is divided in 6 regions having 7 fuzzy characteristics related to the orientation and curvature of the segments in those regions. There are 7 more fuzzy variables related to the whole digit.

Multilayer Perceptron

We will now describe our results with the multilayered Perceptron using the described vectorial representation and data sets. We used a Perceptron with one hidden layer of 100 neurons. The ouput layer has 10 neurons, one neuron for each digit class. We trained this Perceptron using the standard back-propagation algorithm with a learning rate $\eta = 0.5$ and a momentum $\alpha = 0.5$ [1].

We first presented to the network the data set in a random order. After 200 epochs, an RMS error of 0.0613 was obtained and the training set was correctly classified at 99.97% (single error). The training took a CPU time of 1494 seconds (24.9 minutes) on a Sun Ultra-sparc 30. We then obtained a classification rate on the test set of 93.0%. We measured the CPU time to compare this result with the performance of the Fuzzy ARTMAP.

We also trained the Perceptron with the data in order, from digits 0 to 9. After 100 training epochs, an RMS error of 0.0212 was obtained. When we used the network to classify the test set, the rate of the data that was correctly classified was extremely low, 9.21%. This is not surprising because it is well known that the multilayer Perceptron is not capable of incremental learning. In fact it only remembers the few last data it was trained with, in this case the samples of nines.

Table 1: Experimental results with the Fuzzy ARTMAP trained until perfect classification of the training set when $\beta_a = 1$ and the training set is presented in order.

$\overline{\rho_a}$	*Epochs*	ART_a *categories*	*Classification rate (%)*		*Cumulated CPU time (s)*
			Training set	*Test set*	
0.60	1	88	98.2	86.5	7.3
	2	92	99.8	86.7	14
	3	93	100	86.8	21
0.65	1	137	99.0	88.2	12
	2	140	99.97	88.0	22
	3	141	100	88.0	32
0.70	1	219	99.6	90.6	26
	2	222	100	90.6	42
0.75	1	332	99.8	92.0	45
	2	333	100	92.0	69
0.80	1	502	99.9	91.8	79
	2	503	100	91.8	115
0.85	1	774	99.97	92.5	129
	2	775	100	92.5	185
0.90	1	1150	100	91.9	232

Indeed, all elements of the test set but two were effectively classified as nines.

Fuzzy ARTMAP

We now look at the work performed by a Fuzzy ARTMAP on the same data sets we used with the multilayer Perceptron. We fixed a number of parameters of the Fuzzy ARTMAP. For example, we always used fast learning ($\beta_{ab} = 1$) and a vigilance parameter $\rho_{ab} = 1$ in the map field. Other fixed parameters are $\alpha_a = \alpha_b = 0.01$, $\beta_b = 1$, $\rho_b = 1$. Thus the two varying parameters in our experiments are β_a and $\overline{\rho_a}$, the learning rate and baseline vigilance of the ART_a

module.

We trained the Fuzzy ARTMAP by presenting the training set in order, from the zeros to the nines. The learning rate of ART_a was set to $\beta_a = 1$. The network was trained until it perfectly classified the training set. Table 1 presents the results with different baseline vigilances $\overline{\rho_a}$. These results show that values of $\overline{\rho_a}$ between 0.75 and 0.90 are the ones yielding the best correct classification rate on the test set. We can note that increasing the baseline vigilance $\overline{\rho_a}$ causes the Fuzzy ARTMAP to form a higher number of ART_a categories. Because the maximal size of the boxes formed by a Fuzzy ART is smaller when the vigilance is higher, ART_a has to create a greater number of boxes, or categories.

A remarkable point in those results is that the number of training epochs and the corresponding CPU time are both much lower for the Fuzzy ARTMAP than for the multilayer Perceptron. Still, the best classification rate obtained with both networks are very close to one another (93.0% vs. 92.5%). Moreover, it is not needed to perfectly classify the training set to obtain good rates on the test set. In fact, the test set correct classification rate is almost always maximal in only one epoch.

Changing the order of the training set has a small but non negligible influence on the results. The test set classification rates may vary by about 1%. We were able to obtain, by randomly ordering the training set, a correct classification rate on the test set of 92.7% (with $\overline{\rho_a} = 0.85$) in one epoch (and a CPU time of 184 s).

Table 2 presents results obtained when varying the learning rate of ART_a, β_a. We used $\overline{\rho_a} = 0.85$, which yielded the best result with $\beta_a = 1$, and we presented the

Table 2: Experimental results with the Fuzzy ARTMAP trained until perfect classification of the training set with $\beta_a < 1$ when $\overline{\rho_a} = 0.85$ and the training set is presented in a random order.

β_a	*Epochs*	ART_a *categories*	*Classification rate (%)*		*Cumulated CPU time (s)*
			Training set	*Test set*	
0.8	4	1137	100	93.0	621
0.6	4	1077	100	92.1	570
0.4	11	963	100	93.1	967
0.2	11	958	100	93.0	905

data in a random order at each training epoch. The test set correct classification rate was increased up to 93.1%. However, this implies more training epochs and, since the number of ART_a categories is higher, a larger ART_a network.

Finally, to verify that Fuzzy ARTMAP can do incremental learning, we trained it in two separate steps, using fast learning and again $\overline{\rho_a} = 0.85$. First, we performed training with the samples of all digits except the nines. After two epochs, the training set was perfectly learned and the test set correct classification rate on digits 0 to 8 was 93.1%. Secondly, we trained the same network exclusively on the nines. It then correctly classified the whole test set at 92.4%. It is now clear that this network can learn new data without forgetting what it was previously trained with. In fact, only 7 more samples of digits between 0 and 8 were incorrectly classified (on a total of 3197 samples). Thus the Fuzzy ARTMAP is capable of incremental learning, which is not the case for the multilayer Perceptron.

5 Conclusion

We have seen the structure of the Fuzzy ARTMAP and its essential component, the Fuzzy ART. We performed handwritten digit recognition with the multilayer Perceptron and the Fuzzy ARTMAP and compared the performance of these two neural networks. We saw that they yield similar recognition rates, but Fuzzy ARTMAP learning process is a lot faster. Moreover, Fuzzy ARTMAP is capable of incrementaly stable learning.

We presented the best results we obtained so far but we believe the performance of the Fuzzy ARTMAP may still be improved. Nevertheless, it is clear that Fuzzy ARTMAP is a neural network that has attractive features for use in pattern recognition.

References

[1] Simon Haykin. *Neural Networks: A comprehensive foundation.* IEEE Computer Society Press, 1994.

[2] Gail A. Carpenter, Stephen Grossberg, and David B. Rosen. Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, 4:759–771, 1991.

[3] Gail A. Carpenter and Stephen Grossberg. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics and Image Processing*, 37:54–115, 1987.

[4] Gail A. Carpenter, Stephen Grossberg, Natalya Markuzon, John H. Reynolds, and David B. Rosen. Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks*, 3:698–713, 1992.

[5] Jean-François Hébert, Marc Parizeau, and Nadia Ghazzali. A new fuzzy geometric representation for isolated character recognition. Submitted to the 14th International Conference on Pattern Recognition (ICPR '98).