2011 International Conference on Document Analysis and Recognition

Improvement of On-line Recognition Systems using a RBF-Neural Network based Writer Adaptation Module

Lobna Haddad, Tarek M. Hamdani, Monji Kherallah and Adel M. Alimi
REGIM: REsearch Group on Intelligent Machines
University of Sfax, National Engineering School of Sfax (ENIS)
BP 1173, Sfax, 3038, Tunisia
Email:{lobna.haddad, tarek.hamdani, monji.kherallah, adel.alimi}@ieee.org

***Abstract*—In this paper we designed an adaptation module (AM) with the objective to increase the performance of a recognition system for a new user or new writing style. The developed adaptation module is added after the recognition system, and its role is to examine the output of the independent system and produce a more correct output vector close to the desired response of the user. To achieve this end, we conceive an adaptation module based on Radial Basis Function Neural Network (RBF-NN) which is built using an incremental training algorithm. Two adaptation strategies are applied for adaptation module training: increase the number of new hidden units and adjust the parameters of the nearest unit (weights and location of center) using the standard descent gradient. This new architecture is evaluated by the adaptation of two recognition systems, one for digit recognition and one for alphanumeric character recognition. The results, reported according to the cumulative error, show that the adaptation module (AM) leads to decreasing the classification error and is capable of fast adaptation to the users handwriting. Moreover, results are compared with those carried out using the weights updating strategy of the nearest center apart from the addition of new units. In fact, the adaptation module decreases an average of 50% the error rate with standard recognition systems.**

***Keywords*-Writer Adaptation; Module Adaptation; Incremental learning of RBF-NN;**

I. Introduction

Nowadays, the appearance of new apparatuses such as PDA, Smart-phone,...etc make the communication man machine convivial and fast. Following these innovations, new needs emerged to make these apparatuses efficient. Among these needs, we mention the data acquisition, the treatment of these data and an on-line recognition system of the writing. This recognition proves very simple and intuitive but it is very complex to model. The most powerful solutions of recognition are based on neural networks, hidden Markov models, fuzzy inference systems, or combination of these various approaches [1], [2], [3]. The personal characteristic of these new apparatuses incited the researchers to generate a recognition system which adapts to a specific writer style. Thus, the training of the system is realized with an independent writer database. During the use of the apparatus, the system adapts and converges towards a dependent writer system faster and more effective.

There are two adaptation strategies depending on whether the adaptation is performed in off-line or on-line. In the case of off-line, the adaptation is carried out before the real use of the apparatus. So, the user is asked to write a small base containing some words or characters which is used to adapt the system. After that the system won't be modified. In the case of on-line, the system is adapted at each time the user reports a misclassification.

As part of on-line writer adaptation, the systems can be classified in three groups: systems reorganizing the prototypes of the database (addition, modification and deletion), systems updating the recognition systems parameters and systems incorporating an adaptation module without modifying the recognition systems.

Generally, the prototype based system can be adapted to a new writing style by reorganizing the standard prototype set or using also a new dependent user prototype set [4], [5], [6]. The second group includes most recognition systems which are adapted by modification of their specific parameters. Among these systems we mention those which are based on HMM [7], [8], [9]. Generally, the parameters' HMM were adapted using the expectation maximization (EM) retraining, the maximum a posteriori (MAP) adaptation and the maximum likelihood linear regression (MLLR) technique. We also find those which are based on support vector machines like the system described in [10] where an adaptation was realized by re-learning the different SVMs using virtual examples. Add to that, the system in [11] applied an SVM based multiple kernel learning where support vectors were adapted to better model the decision boundary of a specific writer. Also, we mention the system presented in [12], [13] which used a method based on incremental linear discriminant analysis where the writer adaptation is performed by updating the LDA transformation matrix and the classifier prototypes in the discriminative feature space. In the same way, [14] used an incremental learning of the Modified Quadratic Discriminant Function (MQDF) classifier.

Finally, there is the group of systems that associate an adaptation module which is the subject of modification. We mention the system [15] replacing the last layer of the Time Delay Neural Network with an Optimal Hyperplane.

For writer adaptation, the Optimal Hyperplane classifier was retrained to peculiar writing style. In this same context, we find [16] where an output adaptation module was added to the recognition system based on Multi Layer Perceptron. This module consists of a Radial Basis Function network (RBF-NN). So, the adaptation was focused on the sequential learning of the RBF-NN. Our work is based on the latter system using the adaptation module.

The organization of the paper is as follows. In Section II, we present the global architecture of the adaptive system. Besides we describe the different adaptation strategies applied by the adaptation module. In Section III, the performance of the adaptation module is evaluated by applying it on two recognition systems.

II. Proposed Approach

A. Objectives

In order to achieve a writer adaptation that can be applied to all systems independently of the implemented classifier's type, we opt to use a module to adapt a recognition system (RS). This technique is based on the consistency between the output of the RS and a specific style of characters being received, even in the case where the output is incorrect. Thus, the Adaptation Module (AM) learns to recognize these consistent incorrect output vectors and produces a more correct output vector [16]. To exploit this consistency, the (AM) is added after the RS. The architecture of the dependent recognition system is presented in Figure

The adaptation module is based on Radial Basis Function Neural Network (RBF-NN) and will benefit from the ownership of the network, to which each unit responds to a local region of input space. The training of the (AM) requires a sequential learning. Usually, the objective of training algorithm is to learn the relations between input and desired output from given training samples. In the writer adaptation context, the training algorithm is utilized in real time control with a small writer dependent database, which make the problem more complex.

The first suggestive incremental learning algorithm of RBF was that of Platt [17] named RAN (Resource Allocating Network). Subsequently, improvements were made on this algorithm and other algorithms were applied.

The objective of RAN is to learn the network easily and rapidly, leading to a good performance. In fact, this algorithm has been composed of two actions depending on how the network performs on a presented pattern. If the network performs poorly, a new unit was allocated satisfying some growth criteria. If the network performs well, the existing network parameters were updated using standard LMS gradient descent.

The RAN algorithm allows a sequential learning of the RBF-NN that initially contains no hidden nodes, and can add hidden units in the RBF-NN to extend the approximation ability when errors' classification are reported.

A simplified version of RAN [16] has been used in the context of writer adaptation. Applied as an output adaptation module, its basic principle was to map the response of the writer independent neural network into the correct user-dependent confidence vector.

B. Writer Adaptation Strategies

The goal of the adaptation Module (AM) is to produce an output dependent confidence close to the desired response. In this way, the (AM) adds to the output of the recognition system (O^{RS}) an adaptation vector (A) to produce a writer dependent output (AO), with $AO_i = O_i^{RS} + A_i$.

The following are the notations used in the algorithm presented below: I: Input Pattern which is the output of the recognition system, N: Number of units in hidden layer, σ: Width of RBF, z: Output hidden layer, i: Output layer, j: Hidden layer, C: RBF center, W: Weight between output and hidden neurons, D: Desired output. In our experiments, the target vector (D) is 1 for the neuron corresponding to the correct response and 0 for all other neurons. For a given pattern input (I, D), we calculate the RBF-NN output using the following equations:

$$z_j = exp(-\frac{(\sum_k C_{jk} - I_k)^2}{\sigma_j^2}) \tag{1}$$

$$A_i = \sum_j z_j \times W_{ji} \tag{2}$$

The learning algorithm of (AM) is divided into two phases: the growing and the updating of the RBF-NN.

- The growing of the RBF-NN: The RBF network begins with no hidden neurons. The misclassification reported by the user initiates new hidden neurons based on a growing criterion. The latter is the Euclidean distance between the input and the nearest unit. If this minimum distance exceeds a threshold (γ) then a new hidden unit will be allocated. In this case:
  - The input that is considered far from the existing units becomes the center of the new unit ($C_{N+1} = I$).
  - The weight values of connections between the new unit and the output layer correspond to the error realized by the system with a step size a. These weights are calculated using $W_{(N+1)i} = a(D_i - AO_i)$.
  - To avoid the overlap of different regions of RBF units, the width of the new unit is fixed to the distance (d_{min}) between the input and the unit which is the nearest to it ($\sigma_{N+1} = d_{min}$).
- Updating the Parameters of the Nearest Neuron: In order to do an adaptation transparent to the user, we need to decrease the learning speed of the RBF-NN which is in correlation with the size of the network

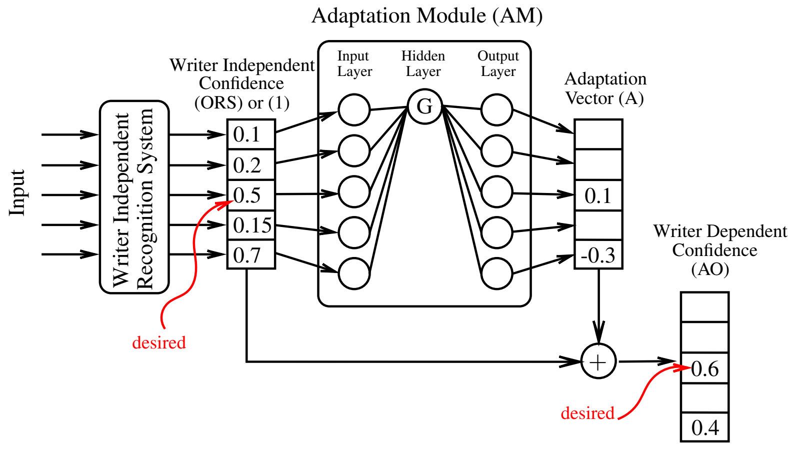


Figure 1: The architecture of the developed system including the writer adaptation module and the different recognition levels

and the number of units subject of update. For this reason, we will adjust only the parameters of the nearest neuron. The update can affect the three parameters of an RBF unit that are: center, width and weights. So, this phase is very important and delicate because the objective is to modify the behavior of the nearest neuron for the current input without causing the forgetting of what is already learned by this neuron. Based on this fact, and to bring the entry near to the nearest unit, we can be limited to the displacement of its center without modification of its width. After that, we modify its weights. The researche,s which were made in the field of sequential learning of RBF-NN, used either the standard LMS gradient descent or the Extended Kalman Filter (EKF) algorithm. Therefore, having an adaptation time and memory size constraints, we opt for the standard LMS gradient descent to decrease the error at each time no new unit is allocated. This is done using the following equations :

$$\Delta C_j = 2\frac{\alpha}{\sigma_j}(I_k - C_{jk})z_j[(\vec{D} - \vec{AO}) \cdot \vec{W_j}] \quad (3)$$

$$\Delta \vec{W_j} = \alpha[(\vec{D} - \vec{AO})]z_j \quad (4)$$

III. Experiments and Results

To test the performance of the Adaptation Module in the generation of the writer dependent recognition system, we connected it in the output of an independent recognition system. The lastter is developed using a generic toolkit (LipiTk) whose aim is to facilitate development of on-line handwriting recognition engines [18] available at http://lipitk.sourceforge.net.

We performed two writer independent recognition systems for numeral and alphanumeric characters. The IRONOFF handwriting database was used to train the two recognizers.

Table I: Recognition rate of the alphanumeric system (without adaptation)

Writer	Recognition rate	Writer	Recognition rate
’lo’	84%	’ta’	84.8%
’am’	85.6%	’fa’	82%
’om’	84.2%	’im’	86%
’ka’	87%	’ch’	85%
’ri’	90%	’bo’	84.2%

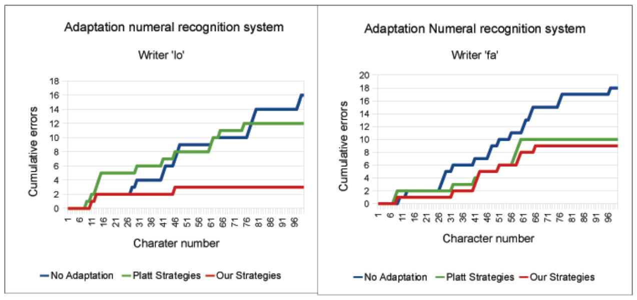
This implies that the output size of the (AM) depends on its input vectors resulting from the independent recognition system.

The evaluation was conducted using ten dependent databases that each one was collected by one writer apart from the training writers. The writers were asked to write on a tablet at least ten examples for each character class: digits [0–9] and lowercase letters [a–z]. The examples are taken randomly to be presented to the (RS) for recognition and adaptation. The recognition rates using the training (IRONOFF) and test databases are represented in table I.

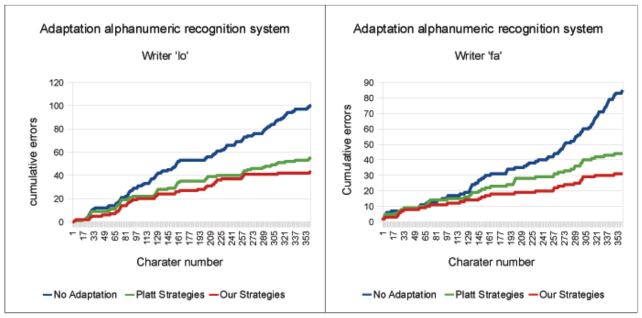
To show the effectiveness of the (AM) referring to the cumulative errors made during the interactive use of the apparatus, we chose databases of two writers ’lo’ and ’fa’ where their writing caused a great confusion of the recognition system. Figure 2b points out the results for three cases : without adaptation, with adaptation (strategy that updates only weights [16] and adds new centers) and with adaptation (strategy that justifies center and updates weights and adds new centers).

We trained the (AM) using the following parameter values that are similar for the two recognition tests : the step size a=0.25; the threshold γ=0.2 and the learning rate α=0.02.

Figure 2 shows the baseline cumulative error without adaptation. Also the total number of character errors from the time when the adaptation started is plotted to give



(a) Adaptation of numeral recognition system



(b) Adaptation of alphanumeric recognition system

Figure 2: The cumulative number of errors with and without adaptation

the estimate instantaneous error rate. Therefore, we note that the slopes for both writers, using the numeral or the alphanumeric recognition system, decrease dramatically by applying the (AM).

Moreover, this experiment is designed to examine the effect of different adaptation strategies (Platt [16] strategies or our strategies) on the recognition performance during the adaptation progress. Clearly, it is observed from figure 2b that our writer adaptation method using the weights update and center adjustment of the nearest unit can significantly reduce the cumulative character errors for both recognition systems. Quantitative results are shown in table II where we display cumulative errors obtained with and without adaptation module (AM) for both recognition systems and in the case of writers 'lo' and 'fa' test databases.

Furthermore, the performance comparison between adaptation strategies is given in table II in which we show the number of memories stored as well as the number of updates carried out during test. These results point out that our adaptation strategies reduce an average 11% the number of memories compared with the method of Platt.

Taking the two writers as examples, the recognition rate without and with adaptation together with the error rate reduction, are presented in table III. Clearly, it is observed from table III that the adaptation module applying our adaptation strategies can significantly reduce an average 50% the error rate for the two independent recognition systems.

Table III: Performance comparison of writer adaptation

Recognition system	Writer	Without adaptation	Our adaptation strategies	Error rate reduction	Elapsed time
Numeral	'lo'	72%	88%	57.14%	0.33s
	'fa'	76%	91%	62.5%	0.74s
Alphanumeric	'lo'	84%	97%	81.25%	1.22s
	'fa'	82%	91%	50%	0.95s

One of the most important properties of a writer adaptation is the transparency to the user. For this reason, we presented in the table III the elapsed time for the adaptation of the test dependent databases for both writers. We notice that at the most the elapsed time is 1.22 second to correct 57 errors made by the alphanumeric recognition system in the case of writer 'lo'. We conclude that the (AM) takes 0.033 second to correct one error. Moreover, the numeral recognition system take 0.74 second to be adapted to the 'fa' writing style. So, 0.08 second for one error. This is due to the increased number of the carried out updates, as already shown in table II.

For this reason, we have also tried to get the elapsed time for a parameters update of the nearest unit and an addition of new unit. Consequently, the obtained results confirm the rapidity of the (AM), then at the most, the addition takes 0.0078 second and the update 0.08 second.

IV. Conclusion

Our main idea, to perform a writer adaptation, is to conceive an adaptation module (AM) which can be added after any independent recognition system, and can learn and recognize the classification errors and produce a vector confidence close to the desired response mentioned by the user. The training of the (AM) which is based on the RBF neural network, is carried out by referring to the Resource Allocating Network (RAN) algorithm. Our adaptation of the (AM) consists of two strategies which are the addition of new units and the updating of nearest unit parameters (center and weights).

To test the performance of the Adaptation Module in the generation of the writer dependent recognition system, we connected it in the output of two recognition systems (numeral and alphanumeric). The evaluation was conducted using ten dependent databases. Taking two writers as examples, we show the result for both systems. Furthermore, we made a performance comparison of the (AM) based on the cumulative character errors realized since adaptation started.

The comparison was realized between the (AM) trained with our adaptation strategies and the adaptation strategies used by Platt [16]. We notice that the (AM) reduced an average 11% the number of memories stored during test compared to the (AM) built using Platt strategies.

Table II: Performance comparison of strategies on writer adaptation

Numeral recognition system						
Writer	Word written during test	Word errors (without AM)	Platt strategies		Our strategies	
			word errors (AM)	memories stored / number of updates	word errors (AM)	memories stored / number of updates
'lo'	100	16	12	10 / 4	3	9 / 2
'fa'	100	18	10	12 / 5	9	13 / 7

Alphanumeric recognition system						
Writer	Word written during test	word errors (without AM)	Platt strategies		Our strategies	
			word errors (AM)	memories stored / number of updates	word errors (AM)	memories stored / number of updates
'lo'	360	100	55	73 / 15	43	65 / 5
'fa'	360	85	44	62 / 5	31	57 / 5

ACKNOWLEDGMENT

The authors would like to acknowledge the financial support of this work by grants from General Direction of Scientific Research (DGRST), Tunisia, under the ARUB program.

REFERENCES

[1] A. M. Alimi, "An evolutionary neuro-fuzzy approach to recognize on-line Arabic handwriting," in *Proc. Inter. Conf. on Document Analysis and Recognition (ICDAR)*, 1997, pp. 382–386.

[2] ——, "Evolutionary computation for the recognition of on-line cursive handwriting," *IETE Journal of Research, Special Issue on Evolutionary Computation in Engineering Sciences*, vol. 48, no. 5, pp. 385–396, 2002.

[3] M. Kherallah, L. Haddad, A. M. Alimi, and A. Mitiche, "On-line Handwritten Digit Recognition based on Trajectory and Velocity Modeling," *Pattern Recognition Letters*, vol. 29, no. 5, pp. 580–594, 2008.

[4] L. Prevost and L. Oudot, "Self-Supervised Adaptation for On-line Script Text Recognition," *Electronic Letters on Computer Vision and Image Analysis*, vol. 5, no. 1, pp. 87–97, 2005.

[5] A. Nakamura, "A Method to Accelerate Writer Adaptation for On-Line Handwriting Recognition of a Large Character Set," in *Proc. Inter. Workshop on Frontiers in Handwriting Recognition (IWFHR)*, 2004, pp. 426–431.

[6] H. Mouchere, E. Anquetil, and N. Ragot, "Writer Style Adaptation in On-line Handwriting Recognizers by a Fuzzy Mechanism Approach : The ADAPT Method," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 21, no. 1, pp. 99–116, 2007.

[7] A. Brakensiek, A. Kosmala, and G. Rigoll, "Comparing Adaptation Techniques for On-line Handwriting Recognition," in *Proc. Inter. Conf. on Document Analysis and Recognition (ICDAR)*, 2001, pp. 486–490.

[8] M. Liwicki, A. Schlapbach, and H. Bunke, "Writer-Dependent Recognition of Handwritten Whiteboard Notes in Smart Meeting Room Environments," in *Proc. IAPR Inter. Workshop on Document Analysis Systems (DAS)*, 2008, pp. 151–157.

[9] S. D. Connell and A. K. Jain, "Writer Adaptation for Online Handwriting Recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 3, pp. 329–346, 2002.

[10] H. Miyao and M. Maruyama, "Writer Adaptation for Online Handwriting Recognition System Using Virtual Examples," in *Proc. Inter. Conf. on Document Analysis and Recognition (ICDAR)*, 2009, pp. 1156–1160.

[11] N. C. Tewari and A. M. Namboodiri, "Learning and Adaptation for Improving Handwritten Character Recognizers," in *Proc. Inter. Conf. on Document Analysis and Recognition (ICDAR)*, 2009, pp. 86–90.

[12] L. Jin, K. Ding, and Z. Huang, "Incremental Learning of LDA Model for Chinese Writer Adaptation," *Neurocomputing*, vol. 73, no. 10–12, pp. 1614–1623, 2010.

[13] Z. Huang, K. Ding, L. Jin, and X. Gao, "Writer Adaptive Online Handwriting Recognition Using Incremental Linear Discriminant Analysis," in *Proc. Inter. Conf. on Document Analysis and Recognition (ICDAR)*, 2009, pp. 91–95.

[14] K. Ding and L. Jin, "Incremental MQDF Learning for Writer Adaptive Handwriting Recognition," in *Proc. Inter. Conf. on Frontiers in Handwriting Recognition (ICFHR)*, 2010, pp. 559–564.

[15] N. Matie, I. Guyon, J. Denker, and V. Vapnik, "Writer-adaptation for On-line Handwritten Character Recognition," in *Proc. Inter. Conf. on Document Analysis and Recognition (ICDAR)*, 1993, pp. 187–191.

[16] J. C. Platt and N. P. Matic, "A Constructive RBF Network for Writer Adaptation," *Advances in Neural Information Processing Systems*, vol. 9, no. 1, pp. 765–771, 1997.

[17] J. Platt, "A Resource-Allocating Network for Function Interpolation," *Neural Computation*, vol. 3, no. 2, pp. 213–225, 1991.

[18] S. Madhvanath and D. V. T. M. Kadiresan, "LipiTk: A Generic Toolkit for Online Handwriting Recognition," in *in Proceedings of the ACM SIGGRAPH*, 2007.