A Novel Fuzzy Classifier Using Fuzzy LVQ to Recognize Online Persian Handwriting

M. Soleymani Baghshah S. Bagheri Shouraki S. Kasaei
Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
soleyman@ce.sharif.edu sbagheri@ce.sharif.edu skasaei@sharif.edu

Abstract

Fuzzy logic is a powerful tool to represent imprecise and irregular patterns. This paper presents a novel fuzzy approach for recognizing online Persian (Farsi) handwriting. In this approach, a fuzzy classifier is introduced that uses a combination of the fuzzy LVQ learning model and the expert knowledge. This method applies an FLVQ network to distinguish between the similar tokens that appear at the end of the strokes. For other tokens, fuzzy linguistic terms are used to describe their features. The purposed method was run on a database of Persian isolated handwritten characters and achieved a high recognition rate compared to other available approaches.

Keywords: Persian Handwriting, Fuzzy Rule-Based, FLVQ, Recognition.

1. Introduction

The Persian character set (along with similar character sets) are used by more than 30% of the world's population and serve in the writing of many widespread languages such as Arabic and Aurdo [1]. In contrast to the advances in the online Latin and Chinese handwriting recognition, relatively few studies have been devoted to the Persian handwriting recognition. This is due, in part, to the cursive nature of the task.

In the last two decades a number of online handwriting recognition systems have been developed. Most of them employ stochastic methods (like hidden Markov models), connective learning-based methods (like neural networks), model matching, and structural/syntactical techniques. The major drawback of all previous methods is their high-dependability on small writing perturbations that tends to imprecise recognition results when working with very vast variety of writing styles.

Considering the limitation and drawbacks of the existing approaches, we define two major characteristics that a handwriting recognition system should possess: fast response and flexibility. To achieve these requirements, the knowledge base must be small but robust [2]. Changes in style or orientation should be handled by the flexible prototypes that contain widely valid descriptions of character information.

In this paper, in order to overcome the complexities of Persian handwriting and to achieve the above requirements, a practical fuzzy approach is proposed. In this approach, the online stroke is first preprocessed and segmented into tokens and the features of each token are extracted. A novel fuzzy classifier is then used to recognize the characters. In this classifier, the end token of the strokes are classified by using the *fuzzy learning vector quantization* (FLVQ) algorithm and the other tokens are characterized by using fuzzy linguistic terms that describe simple representative features. The knowledge base of this classifier is in the form of fuzzy rules that are extracted by an expert.

The rest of this paper is organized as follows: Section 2 presents a brief introduction to Persian handwriting and previous works. Section 3 explains the preprocessing and segmentation processes. Section 4 states the feature extraction process. In section 5 a novel fuzzy classifier is introduced and the recognition algorithm is presented. The experimental results are given in Section 6, and the last section presents the conclusions and future works.

2. Persian Handwriting and Previous Works

Persian characters are in cursive script. It comprises 32 main characters and is written from right to left. Characters have up to four different forms, depending on their position within a word (the beginning, middle, end and isolated forms). Figure 1 presents samples of different characters and their forms.

Also, the characters may be written in different forms based on personal handwriting styles. Persian characters have one main stroke and most of them have

also one to three secondary strokes. Usually the main stroke is written first and then secondary strokes are written. In Figure 2, strokes number 1, 3, and 6 are the main strokes. The secondary strokes may be in the form of dots, "sarkesh", and "daste".

There are some works devoted to recognition of online Persian/Arabic handwriting characters. Al-Emami used a structural analysis method for selecting features of Arabic characters and a decision tree for the classification [3]. Alimi introduced a Neuro-Fuzzy approach in [4] and a template matching and dynamic programming approach in [5]. In [6] combination of pruned Kohonen maps and in [7] combination of SOM and Perceptron is used to recognize online Arabic handwritten characters. A Persian handwriting recognition system is also introduced in [8] which is based on representing input data and character patterns using linguistic terms and comparisons of these terms. Recently we introduced a novel fuzzy approach in [9] that uses a fuzzy rule-based approach, where the rules are formed only by an expert. In this work, we have extended that approach to use the FLVQ learning model to improve the recognition results.

Since none of the above approaches were applied on the same data set, and many of them were not tested on independent and extensive test sets, it is not a fair match to compare how these approaches did against each other.

Letter Name	Isolated Form	Beginning Form	Middle Form	End Form
Alef	ا	ا	ا	ـا
Seen	س	سـ	ـسـ	ـس
Ein	ع	عـ	ـعـ	ـع
Dal	د	د	د	ـد

Figure 1. Sample of Persian printed characters and their forms in different positions within the word.

3. Preprocessing and Segmentation

In this section, we present the proposed preprocessing and segmentation methods. The input data are acquired in the form of a list of (x, y) coordinates. First, the input data are externally separated into strokes by the help of pen-ups. In Figure 2, some sample strokes are shown. Then, the noise reduction process is applied through smoothing and filtering steps. Smoothing averages the coordinates of a point using its neighboring pixels, and the filtering reduces the number of points by eliminating very close points. This filtering technique forces a minimum distance between consecutive points.

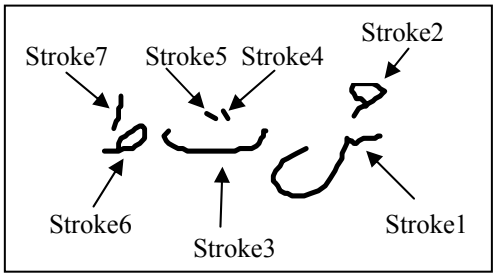

Figure 2. Some sample strokes.

After preprocessing our internal segmentation method is applied. It aims at separating each stroke into a set of small parts called tokens. In Figure 4, each star mark shows the end point of a token. These tokens can be lines, arcs or loops. Some features are used to describe the properties of each token.

To start the task, we first convert the list of (x, y) coordinates of input points into a set of vectors, each starting from one of the points and ending at the next point. Figure 3 shows vectors that link the input points.

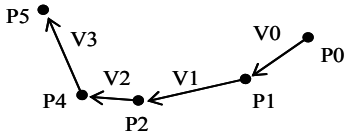

Figure 3. Vectors between points.

The angle between two consecutive vectors *Vi* and V_{i+1} is noted as A_i. The sign of A_i is considered as the curvature side at point P_i. For a sequence of points, the total curvature side is the sign of the sum of angles along the sequence.

$$Input: V_1..V_n,$$

$$A_i = \begin{cases} \angle V_{i+1} - \angle V_i + 180 & (\angle V_{i+1} - \angle V_i) <= -180 \\ \angle V_{i+1} - \angle V_i - 180 & (\angle V_{i+1} - \angle V_i) >= 180 \\ \angle V_{i+1} - \angle V_i & otherwise \end{cases} \quad (1)$$

At this stage, any point in which A_i exceeds a certain threshold, T_1, can be a candidate for the end of token. Also, when A_i exceeds another threshold T_2 (less than T_1) and curvature side at point *Pi* is opposed to the total curvature side, the P_i is a segmentation candidate. So, we save this point and start the process from the next point.

To finalize the segmentation task, a loop detection

method is used to find the loops and deletes additional candidate segmentation points that locate on them. It also defines new segmentation points. Two procedures are considered for this purpose; one that detects closed loops and the other that detects open loops that appear at the start of the strokes.

To detect closed loops, the trajectory of the pen movement is saved. If this trajectory intersects itself, and the points between the previous and current visit of the point construct a circle like curve, a closed loop is detected and two new segmentation points add at the start and end of this loop.

Open loops are always located at the start of the stroke. To find them, we start from the beginning of the stroke and for each point of it compute the distance between this point and the start point of that stroke and save these values in an array. The first local minima in this array, if exists any, is selected. If this value is small relative to the length of the curve up to corresponding point, and the curve is circle like, this point is considered as the end of the open loop. Otherwise, we continue the process from the next point.

As described above, in this work in contrast to the other works [2, 8, 9], a loop detection process is used to complete the segmentation process and to make the resulting tokens more meaningful. Figure 4 shows some segmentation samples.

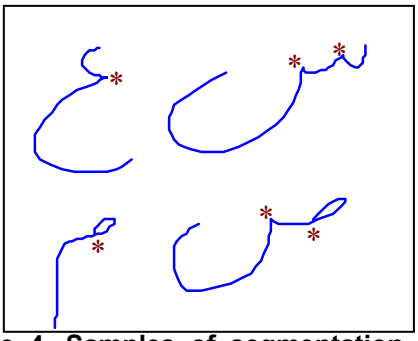

Figure 4. Samples of segmentation. The end point of each token is shown with a * mark.

4. Feature Extraction

After the input data is segmented, each token is described using a set of features. These are defined below.

1) Start2End_Direction: The direction of the straight line that starts from the first point of the token and end at the last point.

$$\Phi = \tan^{-1}((Y_{end} - Y_{statrt})/(X_{end} - X_{start})) \quad (2)$$

2) *Start2COG_Direction:* The direction of the straight line that starts from the first point of the token and ends at the center of gravity of the token.

3) *End2COG_Direction:* The direction of the straight line that starts from the last point of the token and ends at the center of gravity of the token.

4) Straightness: A new straightness measure that performs much better than the measure introduced in [10, 11]. This measure is the angle between the *Start2COG_Direction* and *End2COG_Direction*. Value of this measure for a direct line is 180 and for a circle is zero.

5) Horizontal_Motion: The relative horizontal motion in the written token profile:

$$HM = \frac{x_{\max} - x_{\min}}{\sum_{i=1}^{N-1} |x_{i+1} - x_i|} \quad (3)$$

6) Vertical_Motion: The relative vertical motion in the written token profile:

$$VM = \frac{y_{\max} - y_{\min}}{\sum_{i=1}^{N-1} |y_{i+1} - y_i|} \quad (4)$$

7) Curvature_Side: The direction of concavity computed by:

$$Curvature\ Side = Sign(\sum_{i=1}^{N-1} A_i) \quad (5)$$

where A_i is defined in (1). If this value is negative, the curvature side is assumed to be clockwise, and otherwise it is counter clockwise. These two terms are referred to as CWC and CCWC through the paper.

7) Aspect Ratio: Is computed by dividing the vertical size of the token by its horizontal size as:

$$Aspect\ \ Ratio = \frac{y_{\max} - y_{\min}}{x_{\max} - x_{\min}} \quad (6)$$

5. Fuzzy Classifier Using FLVQ

The next step after the feature extraction process is to create a classifier. This classifier must contain one or several rules for each character to identify the main stroke of it. In this section we first explain our fuzzy classifier and then the recognition algorithm is introduced to classify the inputs.

5.1. Fuzzy Classifier Formation

In this section, we split the tokens into two groups: The *end tokens* and the *non end tokens*. The tokens that appear at the end of the strokes are end tokens and the other tokens are non end tokens. If a stroke contains only one token, it is trivial that this token is an end token.

Since the classification of end tokens is a cumbersome task (as the expert has not already found

appropriate set of rules), in this paper, we use the FLVQ learning model to automatically classify the end tokens. The Non end tokens are characterized by the expert and for describing them, the expert uses linguistic terms. In this subsection, we first introduce the FLVQ algorithm and then we describe the proposed rule-based creation method.

5.1.1 FLVQ Algorithm

Here, we describe the FLVQ algorithm introduced in [12]. Let $U = \{\mu_{ki} \mid k = 1 \ldots N, i = 1 \ldots c\}$ be the fuzzy c-partition of training patterns $X = \{\overrightarrow{x_k} \mid k = 1 \ldots N\}$, and $V = \{\overrightarrow{m_i} \mid i = 1 \ldots c\}$ be the neuron's parametric vectors. First, assume that the number of competing neurons c is equal to the number of pattern classes. The goal of the FLVQ algorithm is to minimize the below objective function:

$$Q^m(U,V) = \sum_{k=1}^{N}\sum_{i=1}^{c}\left[t_{ki}{}^m - \mu_{ki}{}^m\right]D_{ki} \qquad (7)$$

Subjected to the constraints:

$$\sum_{i=1}^{c}\mu_{ki} = 1; \forall k \quad \text{and} \quad \mu_{ki} \in [0,1]; \forall k, i \qquad (8)$$

Where $D_{ki} = \left\|\overrightarrow{x_k} - \overrightarrow{m_i}\right\|^2$ is the distance between neuron i and training pattern k, m is a fuzziness parameter greater than 1, and $t_{ki} \in \{0,1\}$ is the target class membership value of neuron i for input pattern k.

The FLVQ learning law is:

$$m_i(t+1) = m_i(t) + \alpha(t)[t_{ki}{}^m - \mu_{ki}{}^m][\overrightarrow{x_k} - \overrightarrow{m_i(t)}]; \ \forall i \qquad (9)$$

And the membership updating rule is:

$$\mu_{ki} = \left[\sum_{l=1}^{c}\left(\frac{D_{ki}}{D_{kl}}\right)^{\frac{1}{m-1}}\right]^{-1} \qquad (10)$$

The extension of above mentioned learning technique is illustrated in Figure 5. It can be seen that by-passing the MIN layer results in the algorithm described above. Recall that the number of competing neurons has been assumed to be equal to the number of pattern classes. Such assumption can be relaxed by introducing the MIN layer to handle multiple neurons per pattern class network design. Consequently, the neuron underutilization problem in LVQ is resolved.

Corresponding to the general network architecture, the learning algorithm is modified as follows. First, the distance computation is redefined as:

$$D_{ki} = \min_{j \in S_i}\left\|\overrightarrow{x_k} - \overrightarrow{m_{ij}}\right\| \qquad (11)$$

Where S_i is the index set of the competing neurons for pattern class I, and m_{ij} is jth competing neuron's parametric vector for pattern class i. Thus, the FLVQ learning law will be:

$$m_i(t+1) = m_i(t) + \alpha(t)I_{ij}[t_{ki}{}^m - \mu_{ki}{}^m][\overrightarrow{x_k} - \overrightarrow{m_i(t)}]; \ \forall i \qquad (12)$$

Where:

$$I_{ij} = \begin{cases} 1 & if \left\|\overrightarrow{x_k} - \overrightarrow{m_{ij}}\right\| \le \left\|\overrightarrow{x_k} - \overrightarrow{m_{il}}\right\| \ \forall l \in S_i \\ 0 & otherwise \end{cases} \qquad (13)$$

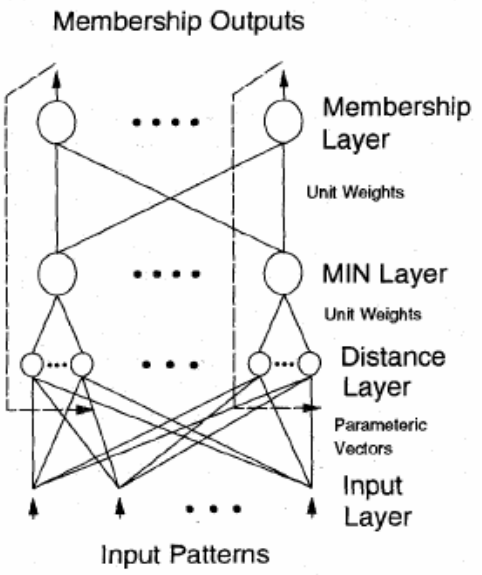

Figure 5.Typical FLVQ network [12].

Here, we use this FLVQ network to classify the end tokens. The input pattern to this network is a vector of features introduced in Section 4. The number of clusters is fixed to 10 (number of different types of end tokens). Figure 6 shows the members of different clusters. By using this network, we find memberships of the input to the clusters. The identifiers of these clusters are used in the final rules.

5.1.2 Fuzzy Rule-Based Creation

The knowledge base used in this paper, is in the form of fuzzy rules which found by an expert. To overcome the problems existing in multi-writer environments, the defined prototypes must maintain the syntax of the rules as short as possible to describe a multitude of character styles and simultaneously to process enough semantic information to distinguish the symbols easily

[11].

To define a rule, at first, some tokens are chosen to describe the main stroke of the character. The end token of the stroke is identified only by the identifier of the related cluster in the FLVQ network.

For non end tokens, some features are selected by the expert to describe them. The Expert uses *Straightness*, *End2COG_Direction,* and *Curvature_Side* features to describe circle-like curves and uses *Straightness*, *Start2End_Direction,* and *Curvature_Side* features, for other types of curves. Then, for each feature, an appropriate linguistic term is chosen. The memberships of some of these linguistic terms are shown in Figures 7 and 8. The linguistic terms in Figure 8 are the abbreviations of the eight directions: East, North East, North, North West, West, South West, South, South East, and East.

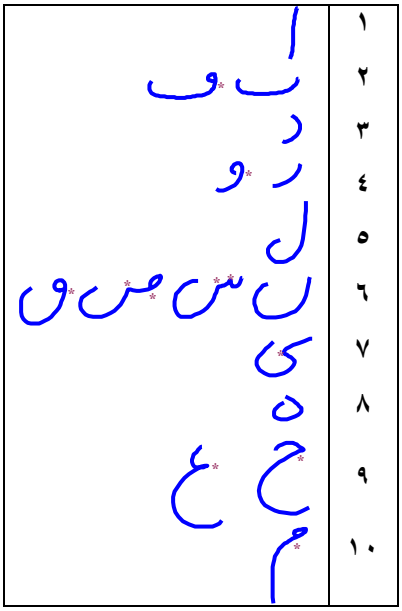

Figure 6. The members of different clusters. The end point of each non end token is shown with a * mark. For the characters that have more than one token, the end token is considered.

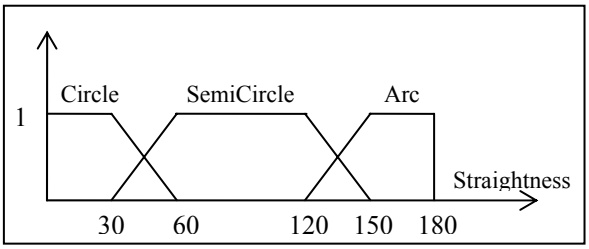

Figure 7. Fuzzy sets of the type variable.

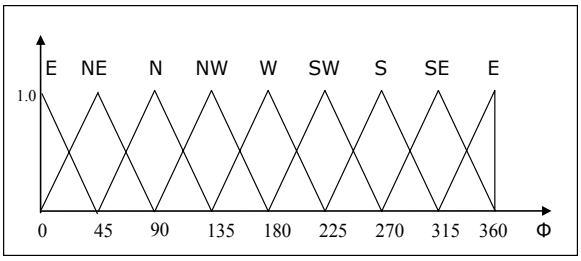

Figure 8. Fuzzy sets of the Direction variables.

Sometimes more than one linguistic term is needed to describe a feature of a non end token or several ways are exist to segment a stroke into tokens. In these cases more than one rule is used. Table 1 presents three sample rules which are used to define three characters shown in Figure 4. Each row of this table is related to one of the tokens of the character.

Table 1. Samples of rule definition.

	Cluster	Straightness	End2COG Direction	Start2COG Direction	Curvature Side
'Seen'					
1	-------	Semicircle	--------	W	CWC
2	-------	Semicircle	--------	W	CWC
3	6	--------	--------	--------	--------
'Eyn'					
1	-------	Semicircle	--------	S	CCWC
2	9	--------	--------	--------	--------
'Sad'					
1	-------	Circle	NE	---------	--------
2	-------	Arc	--------	W	--------
3	6	--------	--------	--------	--------

5.2. Proposed Recognition Algorithm

In Section 5.1, we described the rule-based formation. This subsection presents the purposed classification algorithm that determines the corresponding class using the extracted tokens.

The amount of similarity between the input token and a token of a rule is computed as follows.

- If a rule token has a cluster number, we use FLVQ network and find the membership of the input token to the corresponding cluster using (10); If this value is less than 0.1, we replace it by 0.
- Otherwise, we consider the fuzzy sets corresponding to the linguistic terms in the rule and find the minimum of the membership values of the input token features to the related fuzzy sets.

Since there might be extra short tokens in the input (which do not match with any tokens), to compute the

similarity between an input sequence of tokens with a rule, we find, if possible, the input tokens that are most similar to the rule tokens. The other of these tokens must also be corresponding to the order of the rule tokens. If this task is not possible, the total similarity value is set to zero, otherwise it is set to the minimum of the similarity of the achieved input tokens to the corresponding rule tokens.

Finally, the total similarity value is considered as a positive measure (PM) and the sum of the relative lengths of the unused input tokens is considered as a negative measure (NM). To recognize the class of the input, we find the rule that maximizes this equation:

$$O = PM - \alpha \times NM \qquad (14)$$

Where α is a constant coefficient.

6. Experimental Results

In this work, we use the relatively complete database introduced in [13] that contains the isolated characters written by 128 persons.

As there is no general bench mark for online handwriting recognition algorithm in Persian, we could not compare our result with other works quantitatively.

The recognition rate of the purposed method on this database is near 88% and improves to about 95% when tuning the parameters for a query writer. It must be noted that the performance of the proposed algorithm are more accurate when compared with the most recent results reported in [8, 9]. In addition, the computational cost of our approach is also much less than that of the approach introduced in [8]; and thus can be used in online environments. The number of the rules used in this test is 30.

7. Conclusion and Future Directions

There are very few available approaches for recognition of online Persian handwriting. In this paper, a novel method which is based on representation of input tokens with very simple features, using combination of a fuzzy rule-based and the FLVQ network is presented and a fuzzy inference is also introduced. As opposed to other available methods, an important advantage of the purposed method is its ability to segment the strokes into meaningful tokens. As presented in Section 6, this approach has been quite successful in accepting a wide range of variations for each letter and has shown promising results.

To follow this research, we are now working on recognition of the secondary strokes, to further improve the recognition results.

Acknowledgement

This work was in part supported by a grant from ITRC.

References

[1] I. S. I. Abuhaiba, M. J. J. Holt, S. Datta, “Recognition of Off-Line Cursive Handwriting”, Computer Vision and Image Processing, Vol. 71, No. 1, pp. 19-38, 1998.

[2] A. Malaviya, R. Klette, "A Fuzzy Syntactic Method for On-line Handwriting Recognition", Lecture notes in Computer Science 1121, Springer, Advances in Structural and Syntactical Pattern recognition, SSPR'96, pp. 381-392, 1996.

[3] Al-Emami, S. and Usher, M. “On-Line Recognition of Handwritten Arabic Characters”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 12, No. 7, July 1990. pp. 704-710.

[4] A. M. Alimi , “A Neuro-Fuzzy Approach to Recognize Arabic Handwritten Characters ” , IEEE International Conference on Neural Network, vol. 3, pp. 1397 - 1400, 1997.

[5] A. Alimi, O. Ghorbel, “The Analysis of Error in an On-Line Recognition System of Arabic Handwritten Characters”, Proceedings of ICDAR 1995, pp. 890-893, 14-16 August 1995, Montreal, Canada.

[6] N. Mezghani, M. Cheriet, "Combination of Pruned Kohenon Maps for On-line Arabic Character Recognition", Proceedings of the Seventh International Conf. on Document Analysis and Recognition (ICDAR'03), pp. 900-905 , Edinburgh, Scotland**,** August 2003.

[7] T. Klassen, "Towards Neural Network Recognition of Handwritten Arabic Letters", MS. Thesis, Dalhousie University, Halifax, Nova Scotia, 2001.

[8] R. Halavati, S. B. Souraki, M. Soleymani, “Persian On-line Handwriting Recognition Using Fuzzy Modeling”, to be published in IFSA’05, Beijing, China, July 2005.

[9] M. Soleymani Baghshah, S. Bagheri Souraki, S. Kasaei, “A Novel Fuzzy Approach to Recognition of Online Persian Handwriting”, to be published in ISDA’05, Wroclaw, Poland, September 2005.

[10] Romesh Ranawana, Vasile Palade, G.E.M.D.C. Bandara, "An Efficient Fuzzy Method for Handwritten Character Recognition", Proceedings of KES2004, pp.698-707, Wellington, New Zealand, September 2004.

[11] A. Malaviya, L. Peters, R. Camposano, "A Fuzzy Online Handwriting Recognition System : FOHRES", Second international conference on Fuzzy Theory and Technology, Durham, NC, Oct.13-16, 1993.

[12] F. L. Chung, T. Lee, "A Fuzzy Learning Model for Membership Function Estimation and Pattern Classification", IEEE International Conference on Fuzzy Systems, 1:426-431, 1994.

[13] S. M. Razavi, E. Kabir, "A Data base for Online Persian Handwritten recognition", 6th Conference on Intelligent Systems, Kerman, Iran, 2004.