Fuzzy Model based recognition of handwritten Hindi characters

M. Hanmandlu, O.V. Ramana Murthy
Department of Electrical Eng., I.I.T. Delhi, India
mhmandlu@ee.iitd.in

Vamsi Krishna Madasu
School of Eng. Systems, QUT, Australia
v.madasu@qut.edu.au

Abstract

This paper presents the recognition of handwritten Hindi Characters based on the modified exponential membership function fitted to the fuzzy sets derived from features consisting of normalized distances obtained using the Box approach. The exponential membership function is modified by two structural parameters that are estimated by optimizing an objective function that includes the entropy and error function. A Reuse Policy that provides guidance from the past policies is utilized to improve the reinforcement learning. This relies on the past errors exploiting the past policies. The Reuse Policy improves the speed of convergence of the learning process over the strategies that learn without reuse and combined with the use of the reinforcement learning, there is a 25-fold improvement in training. Experimentation is carried out on a database of 4750 samples. The overall recognition rate is found to be 90.65%.

1. Introduction

Devanāgarī is an abugida script which forms the basis for several Indian languages, including Sanskrit, Hindi, Marathi, etc. It is written and read from left to right. Hindi characters based on the Devanāgarī script are distinguished by the presence of *matras* in addition to main characters. *Matras* are dependent vowels used for representing a vowel sound that is not inherent to the consonants. Therefore, algorithms developed for roman scripts cannot be applied to Indian scripts. Many OCRs for Indian scripts have been reported in [1,2,3,4,5]. However, very few of these have attempted the handwritten Hindi text consisting of composite characters that involve both the main characters and *matras*. In this paper, we present a recognition system specifically addressing the handwritten Hindi characters. However, the proposed recognition scheme is applicable to Hindi words as well after their decomposition into individual components.

Printed Devanāgarī character recognition is attempted based on Kohenen Neural Network (KNN) and Neural Networks [4, 1, 5]. These results are extended to Bangla [5], which also has the header line like Hindi. Structural features like concavities and inter-sections are used as features. A similar approach is tried for Gujarati in [2] with limited success. Reasonable results are reported for Gurumukhi script [4]. Preliminary results are also available in the literature on the recognition of two popular scripts in south India – Tamil and Kannada [2].

Unlike English and other Roman scripts, Hindi has a few, if any, commercial OCR renders; and the ones that have products provide only the custom enterprise solutions. Chaudhuri and Pal [5] have developed a Devanāgarī OCR system which being marketed as a custom solution is not yet available as an off the shelf product. The basic components of the system are described in the literature [5, 10]. After word and character segmentation, a feature based tree classifier is used to recognize the basic characters. Error detection and correction for the OCR based on the dictionary search has led to the recognition accuracy of 91.25% at the word level and 97.18% at the character level on clean images.

Bansal [4], has designed a Devanāgarī text recognition system by integrating knowledge sources, features of characters such as horizontal zero crossings, moments, aspect ratios, pixel density in nine zones, number, and position of vertex points, with structural descriptions of characters. These are used to classify characters and perform recognition. On printed Devanāgarī recognition rates of approximately 70% without any post-processing and 88% correct recognition with the help of a word dictionary are reported. Both of the above OCR systems require vast number of training samples to achieve an acceptable level of performance.

In [11], Hindi words are identified from bilingual or multilingual documents based on features of the Devanāgarī script using Support Vector Machines in the first step. Identified words are then segmented into

0-7695-3067-2/07 $25.00 © 2007 IEEE
DOI 10.1109/DICTA.2007.82

individual characters in the next step, where the composite characters are identified and further segmented based on the structural properties of the script and statistical information. Segmented characters are recognized using generalized Hausdorff image comparison (GHIC) and post processing is undertaken to improve the performance. The OCR system is applied on a complete Hindi–English bilingual dictionary and a set of ideal images is extracted from the Hindi documents in PDF format. The recognition accuracy has attained a figure of 88% for noisy images and 95% for ideal images.

Sinha and Mahabala [6] attempt to recognize Devanāgarī automatically according to their pattern analysis system. They choose 26 symbols and extract the structural information from the characters. However, their study is limited by the sample size and it couldn't achieve any quantitative recognition rate. The work of [12] is conceptually an extension by Sinha and Mahabala, In that they employ a more sophisticated thinning algorithm, a large set of characters, a more computer suitable feature extraction method, and an exhaustive experimental recognition test aiming at a practical level of automated Devanāgarī recognition.

In [13] simple structural features such as a full vertical bar, a horizontal line, diagonal lines in both the orientations, (e.g. in "च" and "त"), circles of varying radii, semicircles of varying radii and orientations are used. A simple feed-forward back propagation network with a single hidden layer is used. The network accepts 23 inputs, corresponding to 23 structural features in the feature vector. Using 11 hidden neurons, and 31 output neurons, where each output corresponds to a core /basic character in the subset of the Devanāgarī character set. A recognition rate of 76% is reported.

In [14] a combination of classifiers capturing both on-line and offline features is described yielding a classification accuracy of 86.5% with no rejects. The combination consists of hidden Markov model and nearest neighbor classifiers. The on-line features are *dx, dy, sin*(θ)*, cos*(θ) for each sample point; curvature and orientation for each critical point. The off-line features are stroke direction histogram for each box of 5x5 grid; *dx* and *dy* from beginning to end of each stroke; centre of gravity of each stroke.

Reinforcement learning [18] is a widely used tool to solve different tasks in different domains. By domain we mean the rules that define how the actions of the learning agent influence the environment, i.e. the state transition function. By task we mean the specific problem that the agent is trying to solve in the domain. The goal of this work is to study how action policies that are learned to solve a defined set of tasks can be used to solve a new and previously unseen task. In this work, we design a new Learning that implements the Reuse Policy ideas [19, 20] efficiently. This learning allows us to reuse the past errors to learn a new one, improving the results of learning from scratch. The improvement is achieved without prior knowledge.

We will now discuss the concept of Reuse Policy. A past policy provides a bias to guide the exploration of the environment and speed up the learning of a new action policy. The success of this bias depends on whether the past policy is "similar" to the actual policy or not. In this paper we make use of this concept in devising a new Reinforcement learning algorithm that reuses the past errors to bias the learning of a new one.

2. Pre-processing

The scanned image is first converted into binary image containing '0' and '1' pixels only. Pre-processing techniques like thinning [9], slant correction and smoothing are then applied [17]. After performing these techniques, there would be extra '0's on all four sides of a character. To standardize the characters, extra rows and columns containing only zeros are removed from all four sides of the character.

Depending on the Aspect Ratio (AR) a standard size is chosen. Aspect ratio is the ratio of height to width of the image. All the characters are therefore fitted in a standard window size of 42x32.

The recognition system has to be validated on the generated database as the standard database is not available at the moment. Hence, the need arises for a large database of handwritten Hindi Characters and matras. The database of totally unconstrained handwritten characters and matras is therefore created using the services of a large number of writers. Many different writing styles are present with different sizes and stroke widths. The database also includes some samples that are difficult to be recognized even by humans. The database is divided into two disjoint sets, one for training and the other for testing.

The training set captures as many variations and different styles of character / character classes as possible. In the training phase, we make use of the concept by which each feature when collected over several samples gives rise to a fuzzy set. We then construct a knowledge base (KB) which consists of means and variances of features of all fuzzy sets. The features extracted from the training set are stored in the knowledge base and at the recognition time, used as reference features for comparing with those of an unknown character or character.

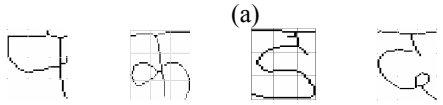
3. Coarse Classification of Characters

Devanāgarī characters can be classified into three major categories based on the presence of the vertical bar, namely, the end-bar characters; the middle-bar characters; and the characters without any bar line.

To determine the presence and position of a vertical bar the whole character is divided into 3x3 windows as shown in Fig. 1. To detect an end bar, the windows 1x3 and 2x3 are examined whether they contain more than 80% of the rows that have at least one black pixel. For detecting the presence of a middle bar, we similarly examine the windows 1x2 and 2x2. The rest of the characters are without a bar.

1x1	1x2	1x3
2x1	2x2	2x3
3x1	3x2	3x3

(a)



(b) (c) (d) (e)

Fig. 1: (a) 3x3 window, (b) End-bar (c) middle bar & (d) Without-bar handwritten characters

Sub Classification I-a

End bar characters are further classified into two categories based on whether the vertical bar and the rest of the character are connected or not to the bar.

- Not connected: **ग, श, ण**.
- Connected components: **म, ज, त, ल, न, च, स, ब, श्र, ज्ञ, य, प, अ, ध, थ, भ**.

(a) (b) (c)

Fig. 2: (a) original image of character (b) character image after removal of header line (c) contour image

To detect whether the components of a character are connected or not, the header line of the character is removed by whitening as shown in Fig. 2(b). The position of the first black pixel (from the top left corner) of the character without the header line is passed on to the contour tracing function. The function returns the first contour of the character as shown in Fig. 2(c). Then the total number of black pixels is calculated both in the contour image and the character image without the header line. If the total number of pixels of the character image is greater than the total number of pixels of the contour image then the components of the character under test are not connected as in Fig. 2(b).

Sub Classification I-b

The connected-components characters with end bar are further partitioned into two categories according to the height of the $(3/4)^{th}$ part of the characters.

- More than 80% black rows: **स, क्ष, ज्ञ, अ, झ, ख**.
- Less than 80% black rows: **ज, त, न, च, ल, ब, य, प, ध, थ, भ, म, व**.

First $(3/4)^{th}$ part of the character is tested to see whether it contains more than 80% of the rows having at least one black pixel as shown in Fig. 3(b). If so, it means that the characters belong to {स, क्ष, ज्ञ, अ, झ, ख}. otherwise to ज, त, न, च, ल, ब, य, प, ध, थ, भ, म, व.

(a) (b)

Fig. 3: (a) original character image '**स**' (b) Three-fourth portion of the character image

Classification II

Characters without vertical bars are further partitioned into five categories.

- Open to the right side: **ह, ट**
- Open to both the right and left sides: **ड**
- Closed or almost closed at the bottom: **ठ, ढ**
- Partially open to the right: **द**
- Remaining characters: **र, इ, छ, उ**

For further partitioning of without-bar characters, firstly, the whole character is divided into 3x3 windows as shown in Fig. 1. Characters ह and ट can be identified if the window 3x1 contains more than 80% of the rows with at least one black pixel. To detect the character "ड", we examine the window 3x3 for 80% of the rows that are black as in Fig. 1(d). For detecting the characters "ठ" and "ढ", we similarly check the windows 3x1 and 3x3. If the sum of pixels in the last row of the character image is less than three and if the window 3x3 has more than 80% of the rows with at least one black pixel, then the identity of the character is taken as "द" as shown in Fig. 1(e).

4. Feature Extraction

For extracting the features, the Box approach presented in [15, 16] is used here. This approach requires the spatial division of the character image. The major advantage of this approach stems from its robustness to small variations and ease of implementation.

Each character image is divided into 24 boxes so that the portions of a character will be in some of these boxes. The choice of the box size is discussed in [21].

There could be some boxes that are empty, as shown in Fig. 4 in which character ङ. is enclosed in the 6x4 grid for illustration. However, all boxes are considered for analysis in a sequential order. The choice of number of boxes is arrived at by experimentation. By considering the bottom left corner as the absolute origin (0,0), the coordinate distance (Vector Distance) for the k^{th} pixel in the b^{th} box at location (i,j) is computed as:

$$d_{kb} = (i^2 + j^2)^{1/2} \quad (1)$$

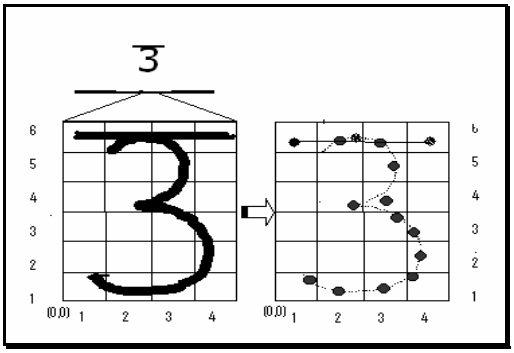


Fig. 4: Illustration of the box method on a Hindi character

By dividing the sum of distances of all black pixels present in a box with their total number, a Normalized Vector Distance (γ_b) for each box is obtained as:

$$\gamma_b = \frac{1}{N}\sum_{k=1}^{n_b} d_k^b, \; b=1,2,.....24 \quad (2)$$

where, N is total number of pixels in a box.

These vector distances constitute a set of features based on distances. Therefore, 24 γ_b's corresponding to 24 boxes will constitute a feature set. However, for empty boxes, the feature value will be zero.

5. Recognition Scheme

In order to recognize the unknown character set using the fuzzy logic, an exponential variant of fuzzy membership function is selected. The fuzzy membership function is constructed using the normalized vector distance.

The concept of a fuzzy set arising from a set of features is as follows. If there are '*n*' possible features for each character and '*m*' number of character samples then a particular feature from each of the samples forms a fuzzy set. The means and variances are computed for each of the 24 fuzzy sets and these constitute the knowledge base (KB). Here, we use the training dataset which contains reference characters for generating the KB.

5.1 Creation of KB & Membership Function

The means m_i and variance σ^2_i for each of the 24 fuzzy sets of KB are computed from the formulae:

$$m_i = \frac{1}{n}\sum_{j=1}^{N_i} f_{ij} \; ; \; \sigma_i^2 = \frac{1}{N_i}\sum_{j=1}^{N_i}(f_{ij} - m_i)^2 \quad (3)$$

where,

N_i is the number of samples in the i^{th} set

f_{ij} stands for the j^{th} feature value of reference character in the i^{th} fuzzy set and i = 1, 2…..24

For an unknown feature vector x, the 24 features are extracted using the Box method. The membership function is chosen as,

$$\mu_{x_i} = e^{\frac{-|x_i - m_i|}{\sigma_i^2}} \quad (4)$$

where, x_i is the i^{th} feature of the unknown character.

If all x_i's are close to m_i's which represent the known statistics of a reference character, then the unknown character is identified with this known character because all membership function values are close to 1 and hence the average membership function is almost 1. Let, $m_j(r)$, $\sigma^2_j(r)$ belong to the r^{th} reference character with r = 1, 2, 3 …36, we then calculate the average membership as,

$$\mu_{av}(r) = \frac{1}{c}\sum_{j=1}^{c} e^{\frac{-|x_j - m_j(r)|}{\sigma_j^2(r)}} \quad (5)$$

where, c denotes for the number of fuzzy sets.

Then $x \in r$ if $\mu_{av}(r)$ is the maximum for r=1, 2 …36. It is observed that some of the fuzzy sets have a very small variance and others, a large variance. This spurred the choice of a new membership function in [16] involving the structural parameters s & t given by,

$$\mu_{xi} = e^{-\Delta x_i' / \sigma_i^{2'}} \quad (6)$$

where,

$$\sigma_i^{2'} = (1+t) + t^2\sigma_i^2 \;\; \& \;\; \Delta x_i' = \left|(1-s) + s^2 \Delta x_i\right|, \quad \Delta x_i = \left|x_i - m_i\right|$$

The new mean and the new variance are functions of the mean and variance of the reference fuzzy set. Thus the structural parameters s, t models the variations in the mean and variance overall 24 boxes. The choice of these parameters has reasoning. That is, if s=1, $\Delta x_i' = \Delta x_i$. Thus, s would be perturbed around 1 to reflect changes in the means. Similarly, if t=-1, then $\sigma_i^{2'} = \sigma_i^2$ thus, t would reflect the changes in the variances.

The scanned image is enclosed in a 42x32 window which is divided into 24 boxes, each of size 6x4. The selected box size was determined after extensive experimentation with different box sizes (see [21]).

5.2 Estimation of Structural Parameters

An objective function needs to be defined for the estimation of these parameters by optimization. The average membership function of any character must be close to 1, when its features are fit into the statistics of a known character. We therefore define the error function as,

$$J_1 = [1 - J]^2 \quad (7)$$

where, $J = \frac{1}{C}\sum_{j=1}^{c} \mu_{x_j}$

In order to reduce the uncertainty in s and t, we define the entropy E as,

$$E = \sum -\left[\mu_{x_j} \ln \mu_{x_j} + (1-\mu_{x_j}) \ln(1-\mu_{x_j})\right] \quad (8)$$

The objective function to be minimized is therefore chosen as the product of (7) and (8),

$$G = E.J_1 \quad (9)$$

We now learn the parameters s and t, as follows

$$s^{new} = s^{old} - \varepsilon\, \partial G/\partial s$$
$$t^{new} = t^{old} - \varepsilon\, \partial G/\partial t \quad (10)$$

The learning factor ε (epsilon) is chosen as 0.01. The initial values of s and t are taken as 3 and 5 respectively.

5.3 Reinforcement Learning

In order to speed up the convergence of parameters using the gradient descent learning, we will use the reinforcement learning by which the past information of the objective function is utilized. However this requires a 're-use policy' of how to use the past information. For this we will define the re-use policy in the following way:

Definition: Reusing a defined past policy requires integrating the knowledge of the past policy into the current learning process.

Our approach is to bias the exploratory process of the new policy with the past one. In our previous work [16], ε was taken as a constant. In this work the policy reusage concept is used to derive new learning. Instead of taking ε as constant, we make it a variable depending on the past errors. We have used here the sigmoid function for ε in which the cumulative of the past errors is biased by the term k_2 and the slope or gain of the function is changed by the term k_1.

$$\varepsilon = \frac{1}{1 + e^{-(k_1 \sum err + k_2)}} \quad (11)$$

where, $err = G^{old} - G^{new} = \Delta G$.

In our work, the initial values of k_1 and k_2 are taken to be 0.5 which are then adjusted using the reuse policy. The steps in the algorithm are as follows:

Algorithm:

1. If $\sum err$ is increasing, then ε must decrease.
   So k_1 should increase.
2. If $\sum err$ is decreasing, then ε need not change.
   So k_2 should increase.
3. If $\sum err$ is constant, then ε should not change.
   So k_1 and k_2 are not changed.

The values of k_1 and k_2 have been changed with an increment of 0.1 while executing the algorithm.

6. Results

The overall recognition rate of all 36 Hindi characters considered together is found to be 69.78%. As the character set is very large as compared to character set a snapshot of this set is shown in Fig. 5, it is impossible to get correct classification with only one classifier in view of the fact that some of the characters have similar shapes.

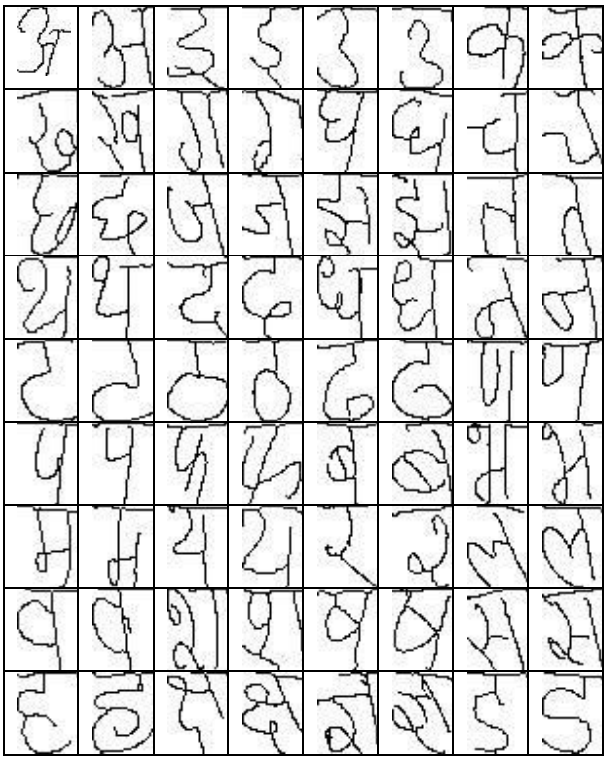


Fig. 5: Snapshot of Hindi character samples

There is a need for coarse classification based on structural information to improve the recognition results. This is accomplished in two steps: coarse classification followed by the final recognition. The three categories of course classification are:

1. Classification Based on Middle bar: **क, फ**
2. Classification based on the right vertical bar based on the following subdivisions -
   - Vertical bar and the rest of the character not connected to it: **ग, श, ण**
   - Characters with a horizontal middle bar: **झ, म, स, अ, भ**
   - Characters (no head line) with the upper top left open: **ज, त, ल, न, च, ब, ज्ञ**
   - Remaining characters: **श्र, य, प, ध, थ, ख, घ, ए**
3. Characters without any vertical bar consisting of following sub-classes:
   - Open to the right side: **ह, ट**
   - Open to both the right and left sides: **ड**
   - Closed or almost closed at the bottom: **ठ, ढ**
   - Partially open to the right: **द**
   - Remaining characters: **र, इ, छ, उ**

Coarse segmentation is initially made to obtain broad classes like characters having middle bar, end bar etc, as explained above. The characters within each broad class are then further classified into individual characters. Thus the recognition results are obtained by performing a coarse classification followed by fuzzy based classification.

We implemented the proposed recognition method with a variable learning factor ε determined from the reinforcement algorithm. The convergence of structural parameters s and t for constant ε is shown in Fig. 6 and due to variable ε is shown in Fig. 7. The convergence of E is shown in Fig.8. Clearly it can be seen that there is a 25 fold improvement in the speed of convergence during training of *s* and *t*.

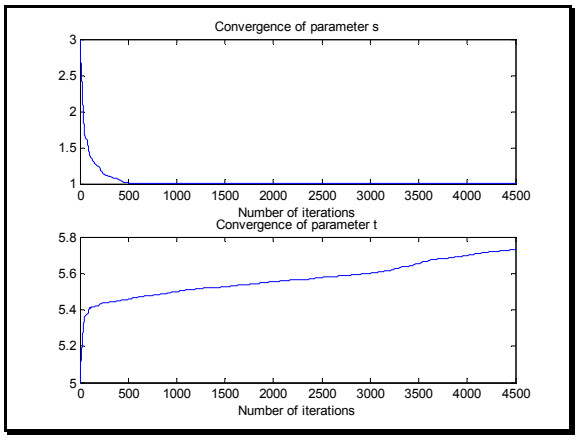


Fig. 6: The case where ε is constant

Table 1 shows learning parameters, structural parameters and recognition rates of Hindi characters. The overall recognition rate is increased to 90.65%. Barring two characters most of the characters have RR greater than 80%, whereas 12 characters have RR in between 80 to 90%. The rest have more than 90%.

Table 1: Recognition rates after coarse classification

Hindi Character	k_1	k_2	s	t	RR (%)
	22.9	8.1	1.1106	5.80	92%
	15.7	5.3	1.03	5.6168	92%
	7.9	3.1	1.0552	5.4982	85.19%
	26.5	8.5	1.101	5.8552	94.87%
	9.5	3.5	1.0524	5.5836	92.59%
	26.6	9.4	1.1195	5.9638	86.21%
	12.7	4.3	1.1267	5.5956	75.00%
	27.9	9.1	1.0334	5.659	81.82%
	25.6	9.4	1.0606	5.7144	93%
	-	-	-	-	100%
	21.4	7.6	1.0519	5.6637	91.67%
	21.6	7.4	1.0961	5.7739	92.31%
	25.3	11.7	1.0038	5.672	85.71%
	11.6	4.4	1.0874	5.5843	95%
	18.5	6.5	1.0984	5.7125	84%
	12.9	4.1	1.0678	5.5786	85%
	12.2	4.8	1.1106	5.6583	90%
	4.2	1.8	1.2152	5.4702	85.71%
	19	6	1.0745	5.8097	96.55%
	12.7	4.3	1.0836	5.5696	90.63%
	15.8	5.2	1.0807	5.7051	92%
	15.2	5.8	1.133	5.6246	84.62%
	8.7	3.3	1.0446	5.5188	95.83%
	-	-	-	-	100%
	6.5	2.5	1.0835	5.5496	94.44%
	25.8	9.2	1.0788	5.8175	100%
	9.2	3.8	1.0528	5.5234	76.92%
	21.4	7.6	1.0374	5.6341	88%
	13.6	4.4	1.0676	5.6185	90%
	12.3	4.7	1.07	5.5892	96.97%
	20.3	6.7	1.0994	5.7622	85.71%
	11.1	3.9	1.0106	5.5387	88%
	15.4	5.6	1.1008	5.6786	85%
	6.5	2.5	1.1144	5.5194	100%
	18.9	6.1	1.0577	5.668	94.44%
	15.4	5.6	1.0726	5.6814	94.87%
Overall Recognition Rate (RR)				**90.64%**	

Table 2: Recognition rates after Mismatch considerations

Hindi Character	k_1	k_2	s	t	RR (%)
	13.8	12.7	1.244	6.368	70.83%
	12.5	11	1.0376	6.4578	100%
	14.4	10.9	1.0651	6.4975	85.19%
	13.5	12.3	1.1369	6.5147	89.74%
	11.7	10.5	1.2314	6.491	92.59%
	13	13.7	1.2137	6.4222	100%
	14.6	11.3	1.1294	6.6211	60.71%
	13.5	11.3	1.0669	6.5004	78.79%
	14.1	13	1.0972	6.4738	96.67%
	-	-	-	-	100%
	12l.1	10.4	1.136	6.4517	88.89%
	13	10.8	1.2542	6.4098	92.31%
	11.1	11.6	1.0261	6.4193	85.71%
	13	12.8	1.1529	6.5148	95%
	11.8	12.8	1.1195	6.5199	84.62%
	11.7	10.1	1.0881	6.5029	100%
	12.5	14.9	1.1945	6.4783	83.78%
	11	12.3	1.115	6.4796	85.71%
	13	13.3	1.1949	6.4613	96.55%
	12	13.7	1.1849	6.5653	96.77%
	14.5	13.7	1.1799	6.5149	90%
	15.4	16.6	1.1289	6.5388	84.62%
	10.3	9.6	1.1082	6.4856	95.83%
	-	-	-	-	100%
	12.8	11.1	1.1067	6.5295	94.44%
	12.6	12.7	1.1615	6.4477	100%
	10.2	10.9	1.0952	6.4716	84.62%
	11.8	11.9	1.0593	6.4839	84%
	12.2	10.6	1.228	6.4132	93.33%
	12.5	11.8	1.0668	6.5913	96.97%
	15.1	14.7	1.1625	6.5009	100%
	12.3	9	1.0813	6.5267	91.67%
	11.9	12.3	1.2172	6.4206	83.78%
	13.1	12.1	1.117	6.5298	100%
	14.8	13.2	1.0539	6.486	100%
	14	11.9	1.1808	6.5651	96.88%
Overall Recognition Rate (RR)				**91.3172%**	

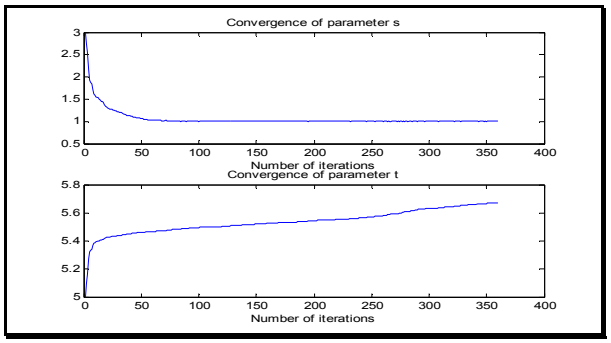


Fig 7: The case where ε is variable

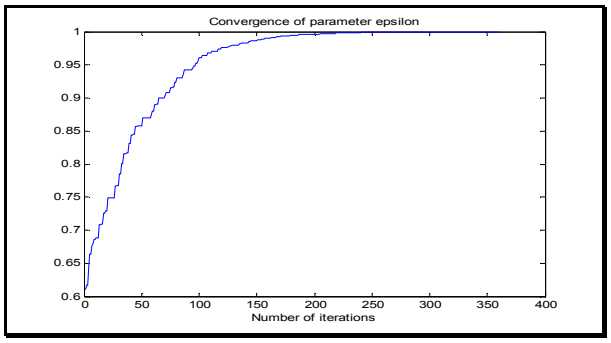


Fig. 8 Convergence of ε

Based on the tolerance sets obtained we now evaluate a second performance criterion to fine tune the recognition rate. The image is now divided into 7x4 instead of 6x4. We chose this particular size because of it being the second best next to 6x4. Now again the recognition rates are calculated this time comparing each character with only its mismatched characters. The overall recognition rate is increased to 91.3172%

7. Conclusions

As the recognition of Hindi characters is a daunting task, coarse classification is necessitated. The coarse classification of Hindi characters is undertaken by making use of structural features like the location of vertical bar, connectivity of character components, and which side the characters are open to etc. The normalized distance used as a feature is found to be effective. A modified membership function is used to represent the fuzzy sets arising out of features of samples.

The reinforcement learning was applied for training the structural parameters resulting in a 25-fold improvement in the speed of convergence. In document processing, where computing time is a major factor, this learning may be helpful. The overall recognition rate with coarse classification is found to be 90.65%.

Acknowledgement

The first two authors gratefully acknowledge the financial support of Department of Science & Technology, Government of India for this work.

8. References

[1] S. Khedekar, V. Ramanaprasad, S. Setlur, and V. Govindaraju, "Text - Image Separation in Devanāgarī Documents", Proc. *Seventh International Conference on Document Analysis and Recognition*, 2003, pp. 1265-1269.

[2] R. Bajaj, L. Dey, and S. Chaudhury, "Devnagari character recognition by combining decision of multiple connectionist classifiers", *Sadhana*, 27(1), 2002, pp. 59–72.

[3] S. Antanani and L. Agnihotri, "Gujarati character recognition", Proc. *Fifth International Conference on Document Analysis and Recognition*, 1999, pp. 418–421.

[4] V. Bansal and R. M. K. Sinha, "A Devanāgarī OCR and a brief review of OCR research for Indian scripts", Proc. *STRANS01*, 2001.

[5] B.B. Chaudhuri and U. Pal, "An OCR system to read two Indian language scripts: Bangla and Devanāgarī", Proc. *Fourth IEEE International Conference on Document Analysis and Recognition*, 1997, pp. 1011–1015.

[6] R.M.K. Sinha and H.N. Mahabala, "Machine recognition of Devanāgarī script", *IEEE Transactions on Systems, Man and Cybernetics,* 9(8), 1979, pp. 435-441.

[7] Y.B. Mahdy and M.T. El-Melegy, "Encoding patterns for efficient classification by Nearest Neighbor classifiers and Neural Networks with application to handwritten Hindi character recognition", Proc. *Third International Conference on Signal Processing*, pp. 1362-1365.

[8] H.Y.Y. Sanossian, "Feature Extraction Technique for Hindi Characters", Proc. *IEEE Workshop on Neural Networks for Signal Processing* VIII, 1998, pp. 524-530.

[9] Y. Suganuma, "Learning structures of visual patterns from single instances", *Artificial Intelligence,* 50(1), 1991, pp. 1–36.

[10] B. B. Chaudhuri and U. Pal, "Skew angle detection of digitized Indian script documents", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2), 1997, pp. 182–186.

[11] H. Ma and D. Doermann," Adaptive Hindi OCR using generalized Hausdorff image comparison", *ACM Transactions on Asian Language Information Processing*, 2(3), 2003, pp. 193–218.

[12] K. Jayanthi, A. Suzuki, H. Kanai, Y. Kawazoe, M. Kimura and K. Kido, "Devanāgarī Character Recognition Using Structure Analysis", Proc. *IEEE-TENCON*, 1989, pp. 363-366.

[13] P. Iyer, A. Singh and S. Sanyal, "Optical Character Recognition System for Noisy Images in Devanāgarī Script", Proc. *Workshop on OCR & DS-2005*, 2005.

[14] S.D. Connell, R.M.K. Sinha and A.K. Jain, "Recognition of Unconstrained On-line Devanāgarī Characters", Proc. *International Conference on Pattern Recognition*, 2000, pp. 368-371.

[15] M. Hanmandlu, K.R.M. Mohan, S. Chakraborty, S. Goyal and D. Roy Choudhury, "Unconstrained handwritten character recognition based on fuzzy logic", *Pattern Recognition*, 36(3), 2003, pp. 603-623.

[16] M. Hanmandlu, M.H.M. Yusof. And V.K. Madasu, "Off-line signature verification and forgery detection using fuzzy modeling", *Pattern Recognition*, 38(3), 2005, pp. 341-356.

[17] M. Hanmandlu and O. V. Ramana Murthy, "Fuzzy logic based handwritten Hindi character Recognition", Proc. *International Conference on Cognition and Recognition,* 2005.

[18] R. S. Sutton and A. G. Barto. **Reinforcement Learning: An Introduction**. MIT Press, Cambridge, Massachusetts, 1998.

[19] C. J. C. H. Watkins. Learning from Delayed Rewards, PhD Thesis, King's College, Cambridge, UK, 1989.

[20] F. Fern´andez and M. Veloso. Exploration and policy reuse. Technical Report CMU-CS-05-172, School of Computer Science, Carnegie Mellon University, 2005.

[21] M. Hanmandlu, O. V. Ramana Murthy, "Fuzzy Model based recognition of handwritten numerals", *Pattern Recognition,* 40(6), 2007, pp.1840-1854.