Online Handwritten Character Recognition of Devanagari and Telugu Characters using Support Vector Machines

H. Swethalakshmi, Anitha Jayaraman, V. Srinivasa Chakravarthy, C. Chandra Sekhar

Online Handwritten Character Recognition of Devanagari and Telugu Characters using Support Vector Machines

H. Swethalakshmi[1], *Anitha Jayaraman*[1], *V. Srinivasa Chakravarthy*[2], *C. Chandra Sekhar*[1]
[1]Department of Computer Science and Engineering, [2]Department of Biotechnology,
Indian Institute of Technology Madras, Chennai - 600 036, India.
{swetha, anitha}@lantana.tenet.res.in, schakra@ee.iitm.ac.in, chandra@cs.iitm.ernet.in

Abstract

A system for recognition of online handwritten characters has been presented for Indian writing systems. A handwritten character is represented as a sequence of strokes whose features are extracted and classified. Support vector machines have been used for constructing the stroke recognition engine. The results have been presented after testing the system on Devanagari and Telugu scripts.

Keywords: Online Handwritten Character Recognition, Stroke, Support Vector Machine.

1. Introduction

Handwritten Character Recognition(HCR) is the process of classifying written characters into appropriate classes based on the features extracted from each character. Handwritten character recognition can be performed either online or offline. A system has been developed for online HCR of Devanagari and Telugu writing systems using Support Vector Machines(SVMs). The character set of Indian languages is large and consists of more complex characters when compared to the Latin script. A handwriting or speech based interface to the computer[10] has become imperative for Indian languages because of complex keyboard mapping procedures for Indian language characters. As many Indian languages have a similar character set, developing a recognition engine for one Indian language serves as a framework for others as well.

Handwritten character recognition for any Indian writing system is rendered complex because of the presence of composite characters. Almost all the Indian scripts have been derived from the Brahmi script. The Brahmi script is a phonographic writing system, where the symbols are directly related to the phonemes of the language in which it is written. Another major characteristic of the Brahmi script which has been passed on to its descendants is that it is an abugida[11] i.e., it is a writing system where the vowels are written as diacritics on the consonants and a vowel is not explicitly written when it appears immediately after a consonant in a word. This combination of diacritics with consonants is called a composite character. A consonant can combine not only with each of the vowels of the writing system but also with other consonants to form ligatures.

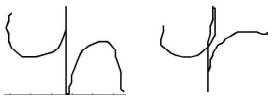
A few models that have been applied for the HCR system include motor models[13], structure-based models [1] [6], stochastic models[8] and learning-based models [9]. Learning-based models have received wide attention for pattern recognition problems. Neural network models have been reported to achieve better performance than other existing models in many recognition tasks. Support vector machines have also been observed to achieve reasonable generalization accuracy, especially in implementations of handwritten digit recognition[4] and character recognition in Roman[2], Thai[12] and Arabic[3] scripts. The present work is on the development of systems for online HCR of Devanagari and Telugu scripts using SVMs.

The Devanagari script, the most widely used Indian script, consists of 14 vowels and 34 consonants. It is used as the writing system for over 28 languages including Sanskrit, Hindi, Kashmiri, Marathi and Nepali. The Telugu script, the third most major script in India, is composed of 12 vowels and 35 consonants. Figures illustrating Devanagari and Telugu character sets are given in the Appendix.

The main challenge in online handwritten character recognition for Indian languages is to build a system that is able to distinguish between variation in writing the same stroke(when the same stroke is written by different writers or the same writer at different times) and minor variation in similar characters in the script. Some confusing characters in Devanagari script are given in Figure 1. A consonant stroke and its modifier that are similar to each other are shown in Figure 2. Other issues faced can be attributed to the large number of character and stroke classes. Some Indian scripts have character modifiers occurring in multiple non-overlapping horizontal units which are positioned on one or both sides of the main consonant. In such cases, we may also need to keep track of the sequence of horizontal units as they are written.

(a) dha (b) dya (c) gha

Figure 1. Similar characters in Devanagari script



(a) pha (b) half pha

Figure 2. A consonant and its modifier that look alike

The input to the recognition system consists of features of the strokes in each written character. An SVM-based stroke recognition module has been considered because of its generalization capability on large dimensional data. The recognized strokes, grouped based on proximity analysis, are mapped onto characters using information on valid stroke combinations for the script. The output of the recognition engine is the sequence of recognized characters.

The paper is organized as follows. Section 2 gives a brief overview of support vector machines. Section 3 describes the stroke recognition system. The experiments carried out and their results are presented in Section 4.

2. Support Vector Machines - An Overview

The objective of any machine capable of learning is to achieve good generalization performance, given a finite amount of training data, by striking a balance between the goodness of fit attained on a given training dataset and the ability of the machine to achieve error-free recognition on other datasets. With this concept as the basis, support vector machines have proved to achieve good generalization performance with no prior knowledge of the data.

The principle of an SVM is to map the input data onto a higher dimensional feature space nonlinearly related to the input space and determine a separating hyperplane with maximum margin between the two classes in the feature space[5]. This results in a nonlinear boundary in the input space. The optimal separating hyperplane can be determined without any computations in the higher dimensional feature space by using kernel functions in the input space. Commonly used kernels include:

1. Linear Kernel :

$$K(x, y) = x.y \tag{1}$$

2. Radial Basis Function (Gaussian) Kernel :

$$K(x, y) = exp(-||x - y||^2/2\sigma^2) \tag{2}$$

3. Polynomial Kernel :

$$K(x, y) = (x.y + 1)^d \tag{3}$$

An SVM in its elementary form can used for binary classification. It may, however, be extended to multiclass problems using the one-against-the-rest approach or by using the one-against-one approach.

3. Recognition of Handwritten Characters

3.1. Representation of the Stroke

Each character is represented as a combination of strokes. A stroke is defined as the trajectory traced by the pen from a pen-down event to a pen-up event and is represented using the data captured as the stroke is written. The number of points collected varies with the stroke and the speed of writing.

The representation of the stroke can be chosen to be of variable length or of fixed length. A variable length representation can be used in stroke recognition based on hidden Markov models, as described in Section 3.3.3. Fixed length format requires the feature vector representing the stroke to be of constant length. Since data should have a constant length representation in the SVM-based approach, fixed length feature vectors have been used to represent strokes.

3.2. Preprocessing of Stroke Data

The data of a stroke is preprocessed in 3 steps as explained in the following subsections.

3.2.1. Normalization

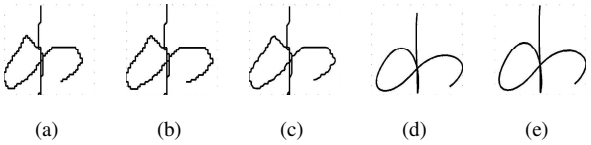
Some characters can be written using more than one stroke with some strokes extending above or below the main part of the character. The main part of a character gives a measure of the linespace and character size used by the writer. Generally, there will be at least one stroke in the main part of the character from which this information is obtained. This main stroke, taken to be the largest stroke that occurs in the main part of the character, is identified and the character is scaled down by its height value. In cases where it is not possible to identify the main stroke, the height of the character is used for normalization. The effect of normalization on the stroke given in Figure 3(a) is depicted in Figure 3(b). After normalization, the stroke is sampled uniformly along the length of its curve(Figure 3(c)).

3.2.2. Smoothing

The normalized stroke is still piecewise linear, with line segments connecting the points along its curve(Figure 3(c)). A Gaussian filter is used to smooth the strokes. A 1-dimensional Gaussian distribution is considered for the smoothing kernel which is convolved first along the x-axis and then along the y-axis. The smoothing achieved depends on the size of the window function and the smoothing kernel parameters used. The effect of smoothing can be seen in Figure 3(d).

3.2.3. Interpolation

The final step in preprocessing is where the smoothened stroke is interpolated to give a fixed number of points, equally spaced along the curve length. The number of points is chosen based on the average number of points per stroke in the given dataset. Figure 3(e) shows an interpolated stroke.



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

Figure 3. Different steps involved in preprocessing a Devanagari stroke for the character /ka/: (a) The raw stroke, (b) Normalized stroke, (c) Resampled uniformly along curve, (d) Smoothened stroke, (e) Interpolated and fully preprocessed stroke

3.3. Feature Extraction and Stroke Classification

Three approaches have been explored for stroke recognition. They are described in the following sub-sections.

3.3.1. Single Recognition Engine Approach

Each stroke is represented as an n-dimensional feature vector depending on the choice of the number of points for stroke representation. The features chosen to represent the curve are the co-ordinates of points in the preprocessed stroke. The stroke is then classified using SVMs. Kernel and associated parameters are experimentally determined. One-against-the-rest strategy is implemented for multiclass classification of strokes.

3.3.2. Multiple SVM Engines for Stroke Recognition

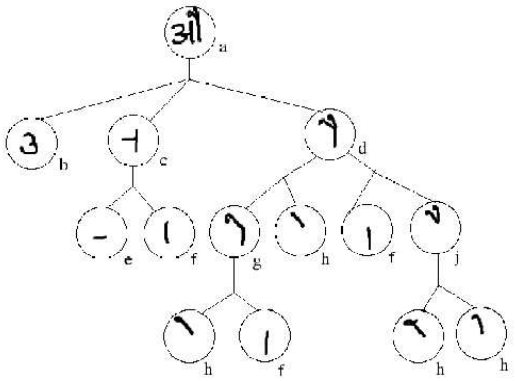
Multiple SVM engines have been explored for enhancing the recognition accuracy. Strokes are preclassified into two categories based on a threshold set for curve length. An SVM-based engine is constructed for each stroke category. Strokes with curvelength below the threshold are classified as small strokes. Small strokes are subjected to normalization and smoothing as mentioned in Sections 3.2.1 and 3.2.2, but they are interpolated to a significantly smaller number of points. All other strokes are preprocessed following the steps in Section 3.2. The choice of feature vector dimension is taken in a preclassification step prior to recognition. Strokes which have a curve length within a certain range around the threshold value are passed to both engines. Contention between the outputs of the two engines is resolved based on the value of the magnitude of the discriminant function[7].

3.3.3. Stroke Recognition using HMMs

A stroke recognition system based on HMMs has also been implemented for the dataset. The strokes are represented as variable-length sequences of frames. Each frame consists of a feature vector representing the features captured at the corresponding time instant. Here, we have used the co-ordinates of the point as features for a frame. Preprocessing involves size normalization, resampling and smoothening steps as mentioned in Sections 3.2.1 and 3.2.2. An HMM is built for each stroke class. The best set of parameters for each model is experimentally determined. A test stroke is given as input to each HMM and is assigned the label of the class whose model has the highest likelihood probability.

3.4. Character Identification from Strokes

A character consists of one or more strokes. All possible strokes in the script are identified from the dataset and the characters are expressed in terms of strokes. This results in the formulation of rules defining possible stroke combinations. There can be multiple rules for a character, which reflect the different ways in which it can be written. These rules are used to determine the character from a set of recognized strokes. The rules for writing the character /au/ in Devanagari script are illustrated in Figure 4. The set of strokes identified and the list of rules governing the stroke combinations are specific to each script.



(a)

Corresponding rules:

a : b + c + d	c : e + f	d : g + h
d : f + j	g : h + f	j : h + h

(b)

Figure 4. (a) The rules for writing the character /au/ in Devanagari script. The symbolic representation for the same is given in (b). Rules are applied where the character is written using more than one stroke.

4. Studies on HCR for Devanagari and Telugu Scripts

The main step in character recognition is the correct recognition of the constituent strokes, following which the character can be identified deterministically. Online handwritten character data used for the experiments has been collected as strokes, using the Superpen^{TM}, a product of UC Logic Inc. This section describes the experiments conducted on Devanagari and Telugu scripts and presents their results.

A value of ll for σ and a window size of 21 units have

been found optimal for the smoothing kernel. Linear and Gaussian kernels have been used in building the SVMs. The Gaussian kernel SVM is seen to converge faster and give a better performance.

4.1. Devanagari Script

The Devanagari script recognizer is trained on the stroke data collected from 90 users and tested on the data from 10 users. The number of examples used for training is 21780 and that for testing is 2420. The number of classes considered for Devanagari script is 91.

4.1.1. Single Recognition Engine Approach

Each stroke is represented using a 120-dimensional feature vector, where co-ordinates of the 60 points obtained after preprocessing are chosen as features. Gaussian kernel with a value of 30 for standard deviation has been found to give the best performance. A dot that occurs anywhere within the character is preclassified based on the size of its bounding box and its relative position within the stroke. The results for the experiments carried out using the Gaussian kernel SVMs are given in Tables 1 and 2.

Table 1. Performance with different values of standard deviation σ for Devanagari script strokes (Trade-off parameter C is chosen to be 100).

σ	Classification Accuracy(%)
2.5	42.85
5	78.26
10	89.83
20	91.53
25	86.53
30	91.12
40	89.26
50	87.44
100	78.35

Table 2. Performance for varying the C parameter value with standard deviation of 30 for Devanagari script strokes.

C	Classification Accuracy(in %)
10	85.66
50	89.5
100	91.12
150	91.16
180	91.28
200	90.99
250	91.07

4.1.2. Multiple Engines for Recognition

Since strokes with smaller average curve length formed the majority of the misclassified strokes, stroke classes were further categorized based on the curve length as described in Section 3.3.2, resulting in an SVM-based recognition engine constructed exclusively for small strokes. Instances of small strokes were represented with 30 points equispaced along the length of the curve. Other strokes were represented with 60 points. Table 3 presents the results obtained using two SVM engines, with 46 stroke classes identified as having smaller curve lengths, and 82 as having larger curve lengths. Gaussian kernel with a standard deviation of 30 has been used here. The results indicate that such a categorization can enhance the system performance. However, when the system was deployed for online recognition, it was observed that strokes were often wrongly preclassified. This is because the thresholds were determined and strokes were categorized using the training dataset, which does not include all possible variations in curve length. This system has to be improved with a more robust mechanism of preclassification.

Table 3. Performance obtained using two SVM engines on Devanagari stroke dataset.

Feature vector dimension	Number of classes	Classification Accuracy(%)
60	46	96.69
120	82	97.27

4.1.3. HMMs for Stroke Classification

The HMM-based stroke recognition system is built as described in Section 3.3.3. The optimal number of mixtures per state possible with the given dataset has been experimentally determined to be 2. The results of experiments conducted with the system are given in Table 4. As the number of states increases, it becomes feasible to build models only for those classes having larger curve length. When a stroke model could not be built at a particular level of complexity, the best performing simpler model is used in subsequent trials. The results clearly indicate the better generalization capability of SVM-based models over HMMs for the given dataset. Though the HMM-based models appear to handle the variability in the curve lengths, they have been observed to be slower and less accurate than the SVM-based models.

4.2. Telugu Script

The data necessary for training the SVMs has been collected from 82 users and data from 10 users is used for testing the system. There are 253 classes and a total of 33726 samples for training and 4091 samples for testing. As the Gaussian kernel was found to give a better performance for the Devanagari script, the same has been used in building a system for the Telugu script.

4.2.1. Single Recognition Engine Approach

Similar to the parameters used for the Devanagari script, a value of 30 for standard deviation of the Gaus-

Table 4. Performance of HMMs on Devanagari strokes.

Number of states	Classification Accuracy(%)
3	31.32
5	43.49
7	49.25
9	55.34
12	58.39
16	60.64
20	63.71
27	65.16
37	66.20
42	66.51
47	66.41

sian kernel was observed to give best classification accuracy for the strokes. A small increase in the accuracy was observed by changing the value of C. The results obtained after testing the 4091 test strokes are given in Tables 5 and 6.

Table 5. Performance with different values of standard deviation on Telugu script strokes. The value of C is 100.

σ	Classification Accuracy(in %)
120	73.30
100	78.30
80	80.30
60	81.86
40	82.50
30	82.96
20	82.82

Table 6. Results for varying C with standard deviation of 30 on Telugu script strokes.

C	Classification Accuracy(%)
100	82.96
60	83.04
50	83.08
40	82.94

4.2.2. Multiple Engine Approach

For obtaining further increase in the system recognition accuracy, the entire set of strokes was organized into 3 main categories based on relative position to the baseline on which a character is written. This resulted in:

1. Baseline strokes - Strokes touching/just above the baseline(Group 1)
2. Bottom strokes - Strokes below the baseline(Group 2)
3. Top strokes - Strokes well above the baseline(Group 3)

Baseline strokes(Group 1) have been further classified based on the presence of a loop with an x-intersection(Group 1a) or its absence(Group 1b). This loop feature within a stroke is illustrated in Figure 5(a). Some baseline strokes having the loop feature are shown in Figure 5.

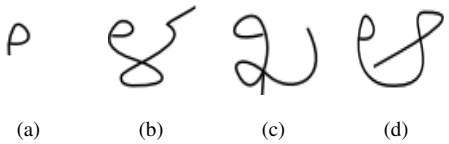


Figure 5. (a) The loop feature within a stroke. (b), (c) and (d) Strokes having the loop feature.

Groups 1a, 1b, 2 and 3 comprise of 64, 115, 44 and 30 classes respectively. This results in 4 stroke categories for the Telugu stroke set, for each of which an SVM-based recognition engine is built. Each engine is trained and tested using the best set of parameters on stroke data from the corresponding category. At least 82 samples per stroke class are used in the training phase. The system is tested with data from 10 instances of each stroke class. The results of the experiments conducted are listed in Table 7.

Table 7. Performance for different values of standard deviation on stroke categories in the Telugu dataset.

σ	Group1a	Group1b	Group2	Group3
20	74.62	99.84	82.18	88.17
30	75.99	99.91	81.83	88.54
50	76.76	95.87	80.22	88.54
60	75.38	84.41	79.36	87.80

The results show that SVMs can be used successfully for stroke recognition. The set of baseline strokes that do not contain loops have been found to give the highest recognition accuracy. When the confusion matrix was used to analyze the cause of comparatively lower performance of the other classes, it was found that confusion among strokes was due to the loss of positional information of the strokes. Some examples illustrating this are shown in Figure 6.

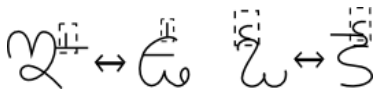


Figure 6. Strokes differentiated by position.

It was also found that the information regarding the presence of a cusp was lost during preprocessing of some strokes. The use of shape features is being considered to improve recognition performance of the engine.

5. Conclusion

The performance of character recognition is dependent on the accuracy of stroke recognition. The results

obtained for recognition of Devanagari and Telugu characters show that reliable classification is possible using SVMs. The results also indicate the scope for further improvement, especially in the case of Telugu character recognition. Future work is directed towards incorporating a database of words for spell-check at word level. The SVM-based methods described here for Devanagari and Telugu handwritten character recognition can be easily extended to other Indian scripts.

6. Appendix

Figure 7. Devanagari characters and their Unicode values.

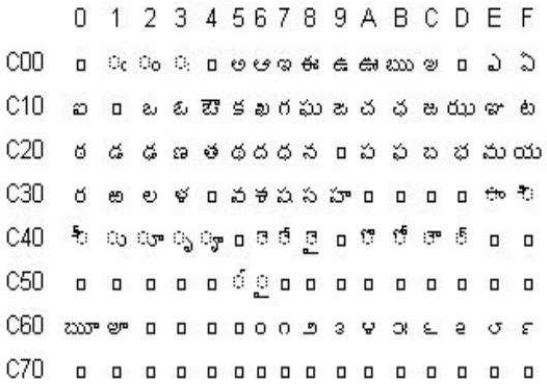


Figure 8. Telugu characters and their Unicode values.

References

[1] K. H. Aparna, Vidhya Subramanian, M. Kasirajan, G. Vijay Prakash, V. S. Chakravarthy, Sriganesh Madhvanath, "Online handwriting recognition for Tamil", *Proceedings of the Ninth International Workshop on Frontiers in Handwriting Recognition (IWFHR' 04)*, Tokyo, Japan, 2004, pp 438-443.

[2] C. Bahlmann, B. Haasdonk, H. Burkhardt, "Online handwriting recognition with support vector machines - a kernel approach", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 26, No. 3, pp 299-310, March 2004.

[3] H. Bentounsi, M. Batouche, "Incremental support vector machines for handwritten Arabic character recognition", *Proceedings of the International Conference on Information and Communication Technologies*, 2004, pp 1764-1767.

[4] Z. Bin, L. Yong, X. Shao-Wei, "Support vector machine and its application in handwritten numeral recognition", *Proceedings of the 15th International Conference on Pattern Recognition*, 2000, pp 720-723.

[5] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition", *Data Mining and Knowledge Discovery*, 1998, pp 121-167.

[6] K. F. Chan, D. Y. Yeung, "Elastic structural mapping for online handwritten alphanumeric character recognition", *Proceedings of 14th International Conference on Pattern Recognition*, Brisbane, Australia, August, pp 1508-1511, 1998.

[7] C. Chandra Sekhar, K. Takeda and F. Itakura, "Recognition of subword units of speech using support vector machines", *Recent Research Developments in Electronics and Communication, Transworld Research Network*, vol.1, November, 2002, pp 101-136.

[8] X. Li, R. Plamondon, M. Parizeau, "Model-based online handwritten digit recognition", *Proceedings of 14th International Conference On Pattern Recognition*, Brisbane, Australia, August, 1998, pp 1134-1136.

[9] S. Manke, U. Bodenhausen, "A connectionist recognizer for online cursive handwriting recognition", *Proceedings of ICASSP 94*, Vol. 2, 1994, pp 633-636.

[10] H. A. Murthy. C. Chandra Sekhar, C. S. Ramalingam, V. S. Chakravarthy, "A multimodal Indian language interface to the computer", *Conference on Sharing Capability in Localisation and Human Language Technologies - 2004*, Kathmandu, Nepal, Jan. 5-7, 2004.

[11] H. Rogers, *Writing Systems : A Linguistic Approach*, Blackwell Publishers, Australia, 2005.

[12] P. Sanguansat, W. Asdornwised, S. Jitapunkul, "Online Thai handwritten character recognition using hidden Markov models and support vector machines", *International Symposium on Communications and Information Technologies, 2004*, Japan, October 26-29, 2004, pp 492 497.

[13] L. R. B. Schomaker, H. L. Teulings, "A handwriting recognition system based on the properties and architectures of the human motor system", *Proceedings of the IWFHR, CENPARMI*, Concordia, Montreal, 1990, pp 195-211.